O`ZBEKISTON RESPUBLIKASI
OLIY VA O`RTA MAXSUS TA'LIM VAZIRLIGI

TERMIZ DAVLAT UNIVERSITETI

A.X.TOYIROV

PYTHONDA MATEMATIK HISOBLASHLARNI DASTURLASH

(Uslubiy qo'llanma)

Termiz-2022

A.X. Toyirov. Pythonda matematik hisoblashlarni dasturlash. – Termiz. “Bekshox print servis” nashriyoti, 2022. 160 bet.

Ushbu uslubiy qo‘llanma Python dasturlash tilida matematik hisoblash ishlarini dasturlashga bag‘ishlangan. Bunda asosiy e’tibor massiv va matritsalar bilan ishlash, ular ustida bajariladigan amallar va funksiyalarni qo‘llash, ilmiy grafiklarni yaratish funksiyalari imkoniyatlari bilan tanishish, SciPy paketi funksiyalari imkoniyatlarini o‘rganish qaratilgan. Uslubiy qo‘llanmada Pythonda matematik hisoblashlarni dasturlash usullari yoritilgan bo`lib, undan “Amaliy matematika(sohalar bo’yicha)” MD-70540201 mutaxassisligi magistrantlariga hamda 60540201-“Amaliy matematika” yo`nalishi talabalari, “Fizika”, “Matematika”, “Matematik modellashtirish” sohalari bo’yicha ilmiy-tadqiqot faoliyatini olib borayotgan tayanch doktorantlar, stajyor tadqiqotchilar va mustaqil izlanuvchilar ham foydalanishlari mumkin.

Taqrizchilar:

Z.R. Raxmonov – O`zbekiston Milliy universiteti “Amaliy matematika va intellektual texnologiyalar” fakulteti dekani, f.-m.f.d.

Ch.B. Normurodov – Termiz davlat universiteti “Amaliy matematika va informatika” kafedrasi mudiri, f.-m.f.d., professor

Uslubiy qo‘llanma Termiz davlat universiteti o‘quv-metodik kengashining 2022 yil 25 iyundagi 11-sonli qaroriga asosan chop etildi.

KIRISH

Tabiatda uchraydigan ko‘plab jarayonlarning holatini o‘rganish, kelgusi harakatini bashoratlash uchun matematik modellar tuziladi, natijada turli matematik masalalarga kelinadi. Ba’zida bu masalalar nochiziqli harakterga ega bo‘lishi, ularning yechimlarini topishni qiyinlashtiradi. Modelda jarayonga ta’sir qiluvchi kattaliklar qanchalik ko‘p bo‘lsa, masala yechimini topishda dasturiy ta’minotga murojaat qilish zarurati tug‘iladi.

Amaliy matematik modellashtirish matematik modellarni sonli tadqiq etish asosida amalga oshiriladi. Matematik modellarga differensial va algebraik tenglamalar sistemalari bilan bog‘langan hususiy hosilali differensial tenglamalarning chiziqli bo‘lmagan sistemalari kiradi. Ularning taqribiy yechimi sonli tahlil algoritmlariga asoslangan bo‘lib, ular orasida chiziqli va chiziqli bo‘lmagan tenglamalar sistemalarini va approksimatsiyalash algoritmlarini yechish usullarini qayd etish mumkin. Python paketlari orasida hisoblash matematikasi masalalarini hal qilishga qaratilgan ko‘plab vositalar mavjud. Ilmiy hisoblash uchun mo‘ljallangan Python paketlaridan SciPy ni qayd etish mumkin. SciPy paketi matematika, tabiiy fanlar va muhandislik uchun ochiq kodli dasturiy ta’minotdir. SciPy N o‘lchamli NumPy massivlari bilan qulay va tez ishlashni ta'minlaydi. Bundan tashqari, SciPy paketi raqamli integratsiya va optimallashtirish dasturlari kabi ko‘plab samarali hisoblangan sonli tahlil protseduralarini taqdim etadi.

O‘zbekiston Respublikasi Prezidentining 2017 yil 7 fevraldagi PQ-4947-son «O‘zbekiston Respublikasini yanada rivojlantirish bo‘yicha harakatlar strategiyasi to‘g‘risida»gi, 2017 yil 17 fevraldagi PQ-2789-son «Fanlar akademiyasi faoliyati, ilmiy-tadqiqot ishlarini tashkil etish, boshqarish va moliyalashtirishni yanada takomillashtirish chora-tadbirlari to‘g‘risida»gi, 2017 yil 20 apreldagi PQ-2909-son «Oliy ta’lim tizimini yanada rivojlantirish chora-tadbirlari to‘g‘risida»gi, 2018 yil 27 apreldagi PQ-3682-son «Innovatsion g‘oyalar, texnologiyalar va loyihalarni amaliyotga joriy qilish tizimini yanada takomillashtirish chora-tadbirlari to‘g‘risida»gi, 2020 yil 7 maydagi PQ-4708-son «Matematika sohasidagi ta’lim

sifatini oshirish va ilmiy-tadqiqotlarni rivojlantirish chora-tadbirlari to‘g‘risida»gi va 2020 yil 6 oktyabrdagi PQ-4851-son «Axborot texnologiyalari sohasida ta’lim tizimini yanada takomillashtirish, ilmiy tadqiqotlarni rivojlantirish va ularni IT-industriya bilan integratsiya qilish chora-tadbirlari to‘g‘risida»gi qarorlari hamda mazkur faoliyatga tegishli boshqa normativ-huquqiy xujjatlarda belgilangan vazifalarni amalga oshirishda ushbu uslubiy qo‘llanma muayyan darajada xizmat qiladi.

Uslubiy qo‘llanma 7 bobdan iborat bo‘lib, unda Python tilida ishlash bo‘yicha asosiy tushunchalar, massivlar va ular ustida bajariladigan funksiyalar, matematik hisoblashlar asosida olingan natijalar asosida oddiy va murakkab grafiklarni tasvirlash, ilmiy hisoblashning kuchli dasturiy ta’minoti hisoblangan SciPy paketi imkoniyatlari bayon etilgan.

I BOB. PYTHON DASTURLASH TILI DASTURIY TA’MINOTI.

Ushbu bobda dasturiy ta’minotni Windows operatsion tizimida ishlovchi kompyuterga o‘rnatish hamda uni sozlash qaraladi. Bunda Python ni kompyuterga o‘rnatish va uning paketlari qisqacha muhokama qilinadi. Python dasturlash tilida buyruqlarni kiritishni qulaylashtirish hamda olingan natijalarning ko‘rgazmaliligini ta’minlash maqsadida PyCharm muhitini o‘rnatish, u bilan ishlash bayon etilgan.

1.1-§. Python dasturlash tili ilovalari

Python dasturlash tili interpretatorining asosiy uch turdagi ilovasi mavjud: CPython, Jython va IronPython.

CPython – portativ ANSI C tilida yozilgan standart dastur bo‘lib, ushbu qo‘llanmada interpretatorning mana shu ilovasi bilan ishlanadi.

CPython C tilida yozilgan Pythonning asosiy ilovasi. U Python kodini oraliq bayt-kodga kompilyatsiya qiladi va keyinchalik virtual mashina tomonidan interpretatsiya qilinadi. CPython C da yozilgan kengaytmali modullarga ega Python paketlari uchun eng yuqori darajadagi muvofiqlikni ta'minlaydi.

Agar siz ochiq kodli dastur yozayotgan va iloji boricha kengroq auditoriyani qamrab olmoqchi bo‘lsangiz, CPython dan foydalaning. C da yozilgan kengaytmalarga bog‘liq bo‘lgan paketlardan foydalanganda, CPython sizning yagona dastur variantingizdir.

Python tilining barcha versiyalari C tilida yozilgan, chunki CPython asosiy dastur hisoblanadi.

Jython – Java dasturlash tili bilan integratsiyani ta’minlaydi. Jython ilovasi ish muhiti sifatida Java (JVM, Java Virtual Machine) virtual mashinasidan foydalangan holda Python kodini Java bayt kodiga kompilyatsiya qiluvchi Java sinflaridan iborat.

Jython Python interpretatori ilovasi bo‘lib, Python kodini Java bayt-kodiga kompilyatsiya qiladi, keyin esa JVM tomonidan bajariladi. Bundan tashqari, u har qanday Java sinfini Python moduli sifatida import qilishi va ishlatishi mumkin.

Jython hozirda Python 2.7 gacha bo‘lgan Python versiyalarini qo‘llab-quvvatlaydi.

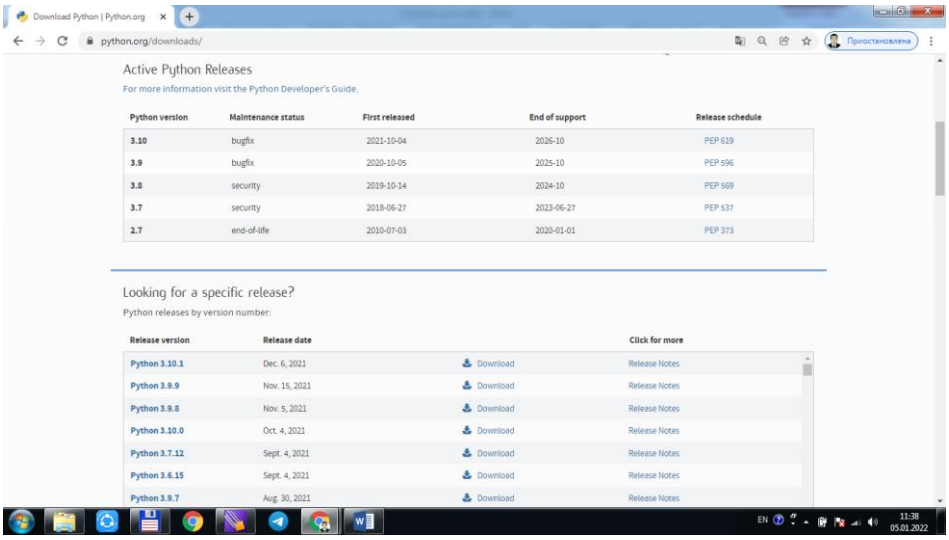
Iron Python – Python dasturlarini Microsoft .NET Framework (Windows operatsion tizimi va Linux da Mono) da ishlash uchun tuzilgan ilovalar bilan integratsiyasini ta’minlash uchun mo‘ljallangan. Python dasturlarini MSIL (Microsoft Intermediate Language) oraliq tiligi kompilyatsiya qiladi.

1.2-§. Python paketini yuklab olish va o‘rnatish.

Kompyuteringizda allaqachon Python o‘rnatilgan bo‘lsa, uni ishga tushirish uchun Пуск (Start) menyusidan Python x.x ni qidiring. Agar kerak bo‘lsa, versiyangizni yangilang.

Hozirda ikkita naql qo‘llab-quvvatlanadi: Python 2.x va Python 3.x. Python 3.x naqli oldingi Python 2.x naqliga mos kelmaydi.

Python-ni www.python.org rasmiy veb-saytidan yuklab olish mumkin. Yuklab olish (download) sahifasidan operatsion tizimingizga mos keluvchi Python naqlini tanlang.



1.1-rasm. Downloads (yuklab olish) sahifasi

Windows operatsion tizimi uchun Python paketi .msi kengaytmali standart o‘rnatish fayli sifatida tarqatiladi. C:\Python2x (yangi versiyalar uchun C:\Python3x) standart o‘rnatish katalogi bo‘lib hisoblanadi.

Python paketi kompyuterga o‘rnatilgandan so‘ng Пуск (Start) | Все программы (All Programs) menyusida Python2x (Python3x) guruhi paydo bo‘lib,

u quyidagilardan iborat bo‘ladi: IDLE ni ishga tushirish (IDLE Python GUI), hujjatlarga kirish (Module Docs), interaktiv buyruq satrini ishga tushirish (Python (command line)), qo‘llanmalarga kirish (Python manuals) va Python dasturini kompyuterdan o‘chirish. Python interpretatori .py kengaytmali fayllar bilan ish olib boradi.

Python interpretatorini istalgan katalogdan ishga tushirish uchun bajariladigan python.exe fayliga manzil path yoki PATH muhitni o‘zgartirish sozlamasida ko‘rsatilgan bo‘lishi kerak. Ushbu manzillar Python interpretatorini dastlabki o‘rnatish vaqtida standart katalogga avtomatik ravishda yoziladi. Python ning bir nechta versiyalarini parallel ravishda o‘rnatish kerak bo‘lsa, u holda bu manzillarni tizimga moslab qayta ko‘rsatish kerak.

Muhitni o‘zgartirish sozlamasiga *Tizim xususiyatlari (Свойства системы /System Properties)* dialog oynasini chaqirish orqali kirish mumkin: *Boshqarish paneli (Панель управления/Control Panel) | Tizim (Система/System). Qo‘shimcha (Дополнительно/Advanced)* bandida *Muhitni o‘zgartirish (Переменные среды/ Environment Variables)* ni tanlang.

PYTHONPATH ikkinchi muhitni o‘zgartirish sozlamasi Python interpretatori modullarini qidirish uchun kataloglar ro‘yxatini belgilaydi. Odatda, interpretator bajariladigan fayl bilan bir katalogda joylashgan modullarni qidiradi.

Paketlarni o‘rnatish. Python uchun juda ko‘plab dasturiy ta’minot ishlab chiqilgan bo‘lib, ulardan turli masalalarni hal qilishda foydalanish juda qulay. Dasturiy ta’minot modullar ko‘rinishida ishlab chiqilgan bo‘lib, ular o‘z navbatida paketlarga yig‘ilishi mumkin. Modul alohida fayl sifatida tuzilgan, paket esa alohida katalog shaklida bo‘ladi.

Paketlarni o‘rnatishning bir nechta usullari mavjud: Windows uchun standart o‘rnatuvchi, quyidagi buyruq yordamida

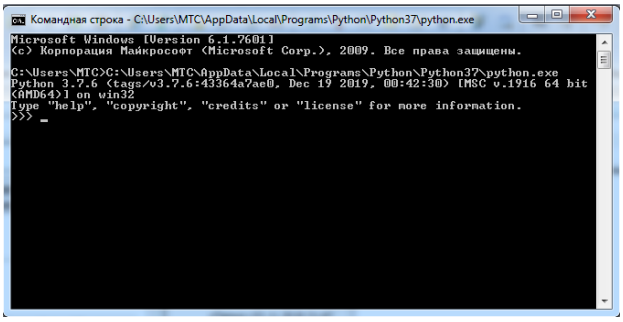
python setup.py install

va Python paketlari uchun bog‘liqlik mexanizmidan foydalanish imkonini beruvchi *Python eggs* orqali. Dastlabki ikkita usulda paketning ishlashi uchun qo‘shimcha ravishda nimani o‘rnatishingiz kerakligini diqqat bilan kuzatib borishingiz kerak.

Eng oddiy yondashuv Python versiyasi uchun tuzilgan standart oʻrnatish faylidan foydalanishdir. Oʻrnatish fayllarini Python ning rasmiy web saytidan yuklab olish mumkin.

1.3-§. Buyruqlar satrida ishlash

Python tili interpretatori bilan interaktiv rejimda ishlash uchun buyruqlar satridan foydalaniladi. Buyruqlar satri quyidagicha ishga tushiriladi: Пуск (Start) | Все программы (All Programs) | Стандартные (Accessories) | Командная строка (Command Prompt). Buyruqlar satriga *python* buyrugʻi kiritgandan soʻng interpretator versiyasi haqidagi ma'lumotlar va >>> buyruq kirish taklifidan iborat boʻlgan oyna ochiladi. Interaktiv rejimda ishlaganda, kiritilgan har bir buyruq matn kiritilgandan soʻng darhol bajariladi.



1.2-rasm. Buyruqlar satri

Masalan, a=2**8 ($a=2^8$) ni hisoblaymiz va keyin *a* ning qiymatini *print(a)* buyrugʻi bilan chop qilamiz. Natija quyidagicha koʻrinadi:

```
>>> a = 2 ** 8
>>> print(a)
256
```

Interpretator uchun koʻp satrli dasturlar buyruqlari modullar deb ataladigan fayllarga yoziladi. Masalan, yuqoridagi koʻrsatmalar matn muharriridan foydalanib test.py fayliga yoziladi:

```
a = 2 ** 10
print(a)
```

Bu modul quyidagi buyruq orqali ishga tushiriladi

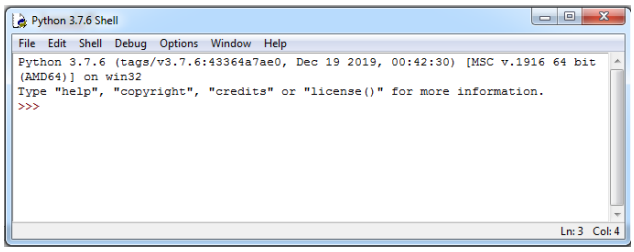
```
python test.py
```

Windows operatsion tizimi sozlamalariga qarab buyruqlar satrida test.py fayli joylashgan katalogga yo‘l to‘liq ko‘rsatilishi yoki ko‘rsatilmasligi mumkin

1.4-§. Standart integratsiyalashgan ish muhiti.

Integratsiyalashgan ish muhitlari (IDE) dasturchiga yanada qulayroq ishlash muhitini taqdim etadi. Ular grafik foydalanuvchi interfeysi (GUI, Graphical user interface) bilan bir qobiqda turli xil vazifalarni (modul tayyorlash, kompilyatsiya va hisoblash) hal qilish imkonini beradi.

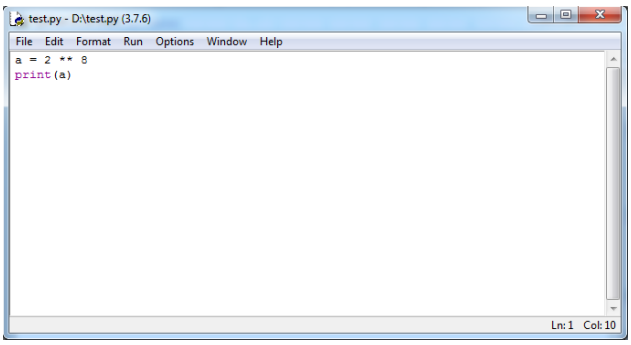
IDLE Python uchun eng oddiy integrallashgan muhit bo‘lib, u Python paketining standart va erkin tarqatiluvchi qismidir. IDLE bosh menyudagi Python x.x bo‘limida IDLE yorlig‘i (Python GUI) yordamida ishga tushiriladi. Bundan tashqari Python moduli (.py kengaytmali fayl) yorlig‘ida sichqonchaning o‘ng tugmasini chertish orqali hosil bo‘lgan kontekst menyu yordamida ham tezlikda ishga tushirishni ta'minlash mumkin.



1.3-rasm. IDLE bosh oynasi

IDLE ning bosh oynasi Python bilan interaktiv rejimda ishlashni ta’minlaydi. IDLE standart matn muharriri xususiyatlariga ega. Xususan, kod sintaksisini ajratib ko‘rsatish ta'minlangan bo‘lib, bu asosiy elementlarni vizual tarzda ajratib ko‘rsatish imkonini beradi.

Modullar (.py kengaytmali matnli fayllar) bilan ishlash to‘liq qo‘llab-quvvatlanadi. IDLE-da dastur kodi yozilgan faylni ochish va tahrirlash uchun matn muharriri oynasi ochiladi, buning uchun Python Shell dasturining File menyusidan Open (Ctrl+O) buyrug‘ini tanlash kerak. Natijada quyidagi oyna ochiladi.



1.4-rasm. IDLE muharriri

IDLE muharriridan modulni ishga tushirish uchun Run menyusidan Run Module buyrug‘ini tanlang. Modul ishining natijalari, xususan, xatolik haqidagi xabarlar Python Shell asosiy oynasida namoyon bo‘ladi. Yangi modul yaratish uchun, yoki muharrirda yoki Python Shell qobig‘ida, File | New Window buyrug‘idan foydalaniladi.

IDLE ning asosiy xususiyatlari orasida xatoliklarni tuzatish imkoniyatini qayd etish lozim. Tahrirlash oynasidagi xatoliklarni tuzatish modulida kontekst menyu yordamida tanlangan kod satrlarida to‘xtash nuqtalari o‘rnatiladi. O‘rnatilgan tuzatuvchi asosiy oynadagi Debug | Debugger buyrug‘i yordamida chaqiriladi. Debug Control (xatoliklarni tuzatish) oynasi paydo bo‘lgandan so‘ng, xatoliklarni tuzatish moduli ishga tushiriladi (muharrirdagi Run | Run Module buyrug‘i) va o‘zgaruvchilarning joriy qiymatlarini ko‘rish orqali dasturni qadamma-qadam ishlatish mumkin bo‘ladi.

1.5-§. Python uchun NetBeans IDE muhiti

IDLE dan tashqari, Python da ishlashning qulay muhitini yaratuvchi ko‘plab dasturiy mahsulotlar (muharrirlar, integratsiyalashgan ish muhitlari) mavjud. Buyruqlarni sintaktik ajratish (kalit so‘zlar, satrlar, sharhlarni ajratib ko‘rsatish), matnni tahrirlash, xatoliklarni tekshirish kabi odatiy funksiyalarga qo‘shimcha ravishda, ular loyiha fayllari bilan ishlashga imkon beradi, manba kodi versiyasini boshqarish tizimlari bilan integratsiyani ta'minlaydi va hokazo.

Python uchun IDE va turli muharrirlar, ularning asosiy imkoniyatlarining tavsifi, o‘zaro farqlarini Python ning rasmiy web saytining Python Editors sahifasida topish mumkin. Bepul taqdim etiladigan mahsulotlar orasida eric4 IDE ni ajratib

koʻrsatish mumkin. Windows operatsion tizimida Pythonda ishlash uchun PyScripter dasturi tavsiya qilinadi.

Bundan tashqari, Eclipse - koʻplab dasturlash tillarida ishlashni taʼminlovchi, platformalararo integratsiyalashgan ish muhitli dasturiy taʼminotni ta'kidlash mumkin. PyDev kengaytmali moduldan foydalanib, Pythonda dasturlar bilan ishlash imkoniyati yaratiladi. Shuningdek, ushbu dasturiy mahsulotlar sinfiga Java, JavaFX, Ruby, Python, PHP, JavaScript, C++ va boshqa dasturlash tillaridagi ilovalar uchun bepul integratsiyalashgan ish muhiti boʻlgan NetBeans IDE ham kiradi.

NetBeans IDE loyihasi Sun Microsystems tomonidan qoʻllab-quvvatlansa ham, NetBeans ishlab chiquvchilar hamjamiyati (NetBeans Community) hamda NetBeans Org tomonidan mustaqil ravishda ishlab chiqilgan. NetBeans IDE dasturi Microsoft Windows, GNU / Linux, FreeBSD, Mac OS X, Solaris kabi asosiy platformalar uchun oldindan tuzilgan distributivlar (kompilyatsiya qilingan binar fayllar) sifatida mavjud. Python uchun NetBeansni Netbeansning rasmiy sayti www.netbeans.org dan yuklab olish mumkin.

NetBeans muhitida dasturni yaratish, muvaffaqiyatli kompilyatsiya qilish NetBeans ning oʻzini ishlatish uchun Sun JDK (Java Development Kit), Sun firmasidan bepul taqdim qilinuvchi Java tilida ishlovchi dasturlar toʻplami oldindan oʻrnatilgan boʻlishi kerak.

NetBeans ni Windows operatsion tizimida ishga tushirish Пуск(Start) menyusining Все программы (All programs) boʻlimidagi NetBeans guruhidan amalga oshiriladi. Dastur oʻrnatilgandan soʻng, NetBeans IDE kompyuteringizda oʻrnatilgan Python versiyalarini taniydi. Python interpretatorining Jython ilovasi NetBeans ga kiritilgan. Ushbu oynadan siz NetBeans IDE uchun mavjud boʻlgan Python versiyalarini qoʻshishingiz yoki oʻchirishingiz mumkin. Python platformasi dispetcheri yordamida Python interpretatori jildlari manzillarini ham sozlashingiz mumkin. Konfiguratsiya jarayonining bir qismi sifatida siz Python modullari manzillarini shunday oʻrnatishingiz mumkinki, shunda ular Pythonning har bir alohida versiyasi uchun NetBeansni har safar ishga tushirganingizda mavjud boʻladi.

1.6-§. Python uchun PyCharm IDE muhiti

PyCharm— Python dasturlash tili uchun mo‘ljallangan integrallashgan ishlab chiqarish muhiti (*inglizcha IDE — integrated development environment*) hisoblanadi. Ushbu dastur JetBrains kompaniyasi tomonidan Intellij IDEA dasturlash muhiti asosida ishlab chiqarilgan. PyCharm orqali dastur kodlarini analiz qilish, loyihani testdan o‘tkazish, shuningdek, Django web-freymvorki bilan ishlash imkoniyatlari mavjud. Shuningdek, dastur Linux operatsion tizimida terminalni ham qo‘llab quvvatlaydi, ya’ni foydalanuvchi yoki dasturchi, PyCharm ilovasining o‘zidayoq terminal bilan ishlay oladi.

Hozirgi kunda PyCharm dasturining uch xil toifasi mavjud, bular:

- PyCharm Edu — dasturlashni o‘rganuvchilar uchun mo‘ljallangan;
- PyCharm Community — dasturchilar uchun mo‘ljallangan;
- PyCharm Pro — ancha mukammallashgan versiya bo‘lib, ba’zi xizmatlari pulli ekanligi bilan ajralib turadi.

Umuman olganda, PyCharm — bepul dasturiy paket bo‘lib, Windows, Linux, MacOS operatsion tizimlarida ishlay oladi.

PyCharm uchta nashrda mavjud:

- **Hamjamiyat** (bepul va ochiq manbali): Pythonni realizatsiya qilish uchun, shu jumladan kod yordami, qayta ishlash, vizual disk raskadrovka va versiyalarni boshqarish integratsiyasi.
- **Professional** (pullik): professional Python, web va ma’lumotlar bazasini rivojlantirish uchun, shu jumladan kod yordami, qayta ishlash, vizual disk raskadrovka, versiyani boshqarish integratsiyasi, masofaviy konfiguratsiyalar, tarqatish, Django va Flask kabi mashhur web-frameworklarni qo‘llab-quvvatlash, ilmiy vositalar (shu jumladan, Jupyter-ni qo‘llab-quvvatlash), katta ma’lumotlar vositalari.
- **Edu** (bepul va ochiq manbali): dasturlashtirilgan tillarni va shu bilan bog‘liq texnologiyalarni integratsiyalashgan ta'lim vositalari bilan o‘rganish uchun.

PyCharm bilan Python-da ishlashni boshlash uchun platformangizga qarab Python-ni python.org saytidan yuklab olishingiz va o‘rnatishingiz kerak.

PyCharm Python-ning quyidagi versiyalarini qo‘llab-quvvatlaydi:

• Python 2: versiya 2.7

• Python 3: 3.5 versiyasidan 3.9 versiyasiga qadar

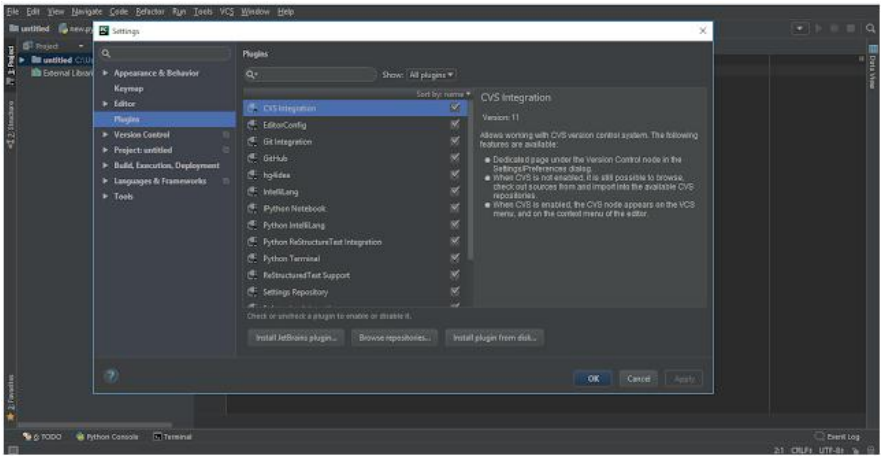
Shuningdek, Professional nashrida Django, Flask va Piramida dasturlarini ishlab chiqish mumkin. Bundan tashqari, u HTML (shu jumladan HTML5), CSS, JavaScript va XML-ni to‘liq qo‘llab-quvvatlaydi: ushbu tillar IDE-ga plaginlar orqali to‘planadi va sukunat bo‘yicha siz uchun yoqiladi. Boshqa tillar va frameworklar uchun yordamchi plaginlar orqali ham qo‘shish mumkin.

PyCharm dasturchilar uchun quyidagi imkoniyatlarni taqdim etadi:

- Sintaksis va xatolarni ajratib ko‘rsatish
- Kodni to‘ldirish
- Navigatsiya
- Blok nomlari uchun to‘ldirish
- Maxsus teglar va filtrlar uchun to‘ldirish
- Teglar va filtrlar uchun tezkor hujjatlar
- Ularni disk raskadrovka qilish imkoniyati

1.7-§. PyCharm IDE muhitida loyiha yaratish

PyCharm yordamida siz Pythonda ilovalar ishlab chiqishingiz mumkin. Bundan tashqari, Professional Edition bilan Django, Flask va Pyramid ilovalarini ishlab chiqishingiz mumkin. Shuningdek, u HTML (jumladan, HTML5), CSS, JavaScript va XML-ni to‘liq qo‘llab-quvvatlaydi: bu tillar plaginlar orqali IDE-ga kiritilgan va sukunat bo‘yicha siz uchun yoqilgan. Boshqa tillar va freymworklar uchun qo‘llab-quvvatlash plaginlar orqali ham qo‘shilishi mumkin (**Settings | Plugins** yoki MacOS foydalanuvchilari uchun **PyCharm | Preferences | Plugins** qo‘shimcha ma’lumot olish yoki IDE-ning birinchi ishga tushirilishida ularni o‘rnatish uchun plaginlar).



1.5-rasm. Sozlamalar oynasi

PyCharm - bu Windows, MacOS va Linuxda ishlaydigan platformalararo ishlab chiqish muhiti. Agar sizga PyCharmni o‘rnatishda yordam kerak bo‘lsa, Linux, macOS va Windows uchun o‘rnatish yo‘riqnomalariga qarang.

Loyiha yaratish uchun quyidagi qadamlarni bajarish kerak:

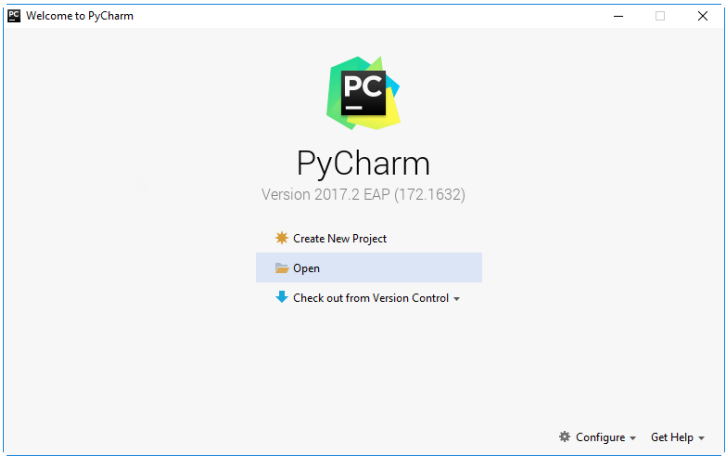
1-qadam: PyCharmda loyihani ochish/yaratish

PyCharmda qilgan barcha ishlaringiz loyiha kontekstida amalga oshiriladi. U kodlashni qo‘llab-quvvatlash, kodlash uslubi izchilligi va boshqalar uchun asos bo‘lib xizmat qiladi.

IDE ichida loyiha ustida ishlashni boshlash uchun sizda uchta variant mavjud:

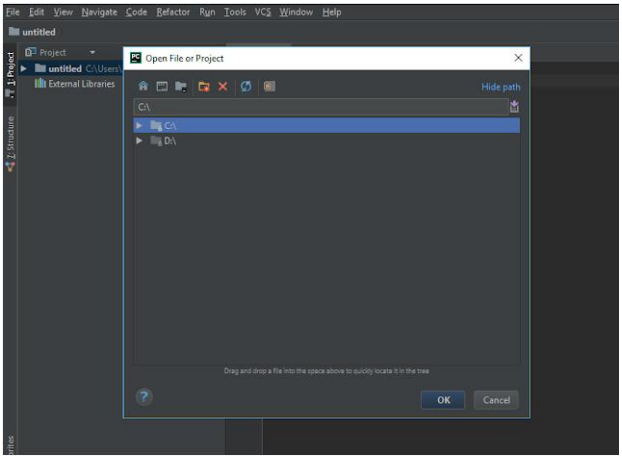
1. Mavjud loyihani ochish

Kompyuteringizda saqlangan mavjud loyihalardan birini ochish bilan boshlang. Buni muhit ishga tushirilganda ochiladigan dastlabki oynada (yoki **File | Open) ochish (Open**) tugmasini bosish orqali qilishingiz mumkin:



1.6-rasm. Muhitning ish boshlash oynasi

Yoki **File** menyusidan **Open** buyrug‘ini tanlang va manbalaringiz joylashgan katalogni belgilang:

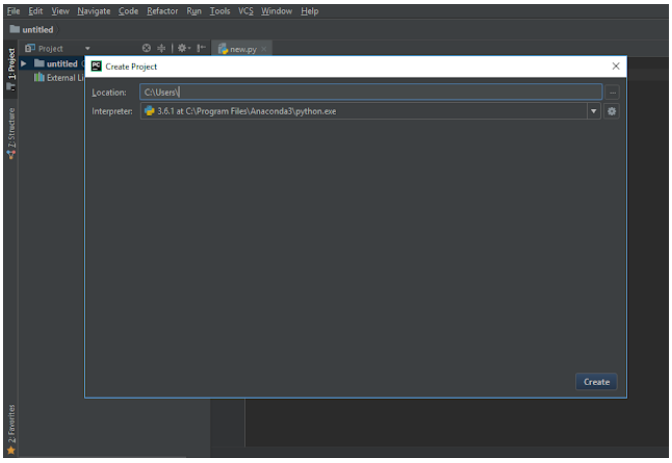


1.7-rasm. File|Open muloqot oynasi

Keyin PyCharm sizning manbalaringizdan siz uchun loyiha yaratadi. Shuningdek, mavjud manba kodidan loyihani import qilish ham mumkin.

2. Loyihani noldan yaratish.

Agar siz loyiha yaratishni noldan boshlashni afzal ko'rsangiz, New Project tugmasini bosing va ochilgan ekranda dialog oynasiga loyiha nomini kiriting va Python loyihasi yaratiladi.

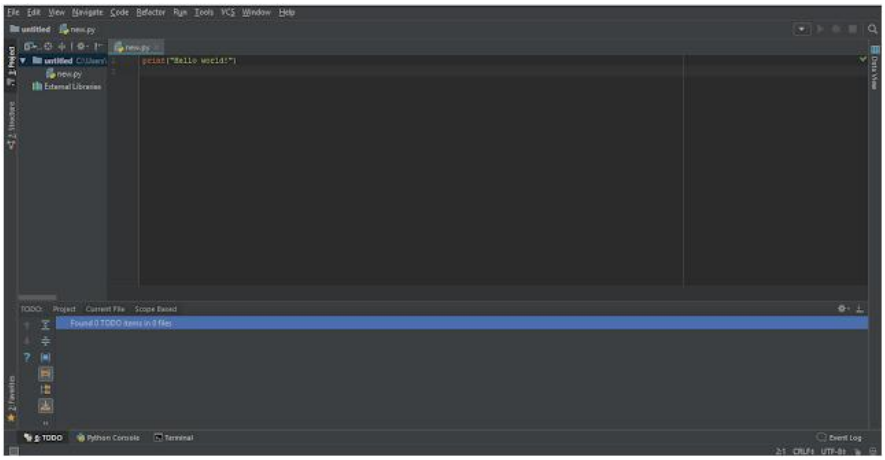


1.8-rasm. Yangi loyiha yaratish oynasi

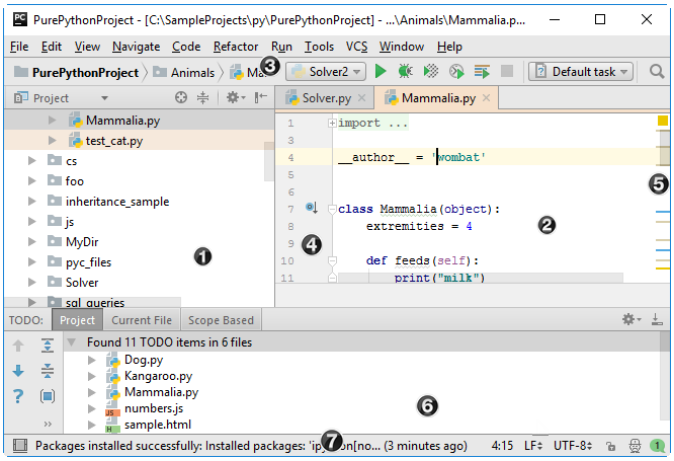
2-qadam. Interfeys bilan tanishish

PyCharmni birinchi marta ishga tushirganingizda yoki ochiq loyihalar bo'lmaganda, siz xush kelibsiz ekranni ko‘rasiz. U sizga IDE ga asosiy kirish nuqtalarini taklif qiladi: loyihani yaratish yoki ochish, versiyani boshqarish bilan loyihani tekshirish, hujjatlarni ko‘rish va IDE ni sozlash.

Loyiha ochilganda siz bir nechta mantiqiy sohalarga bo‘lingan asosiy oynani ko‘rasiz. PyCharm oynasi qoramtir hamda yorqin interfeysga ega bo‘lishi mumkin. Quyida asosiy oyna elementlarini ko‘rib chiqiladi:



1.9-rasm. PyCharm muhitining qoramtir interfeysi



1.10-rasm. PyCharm muhitining yorqin interfeysi

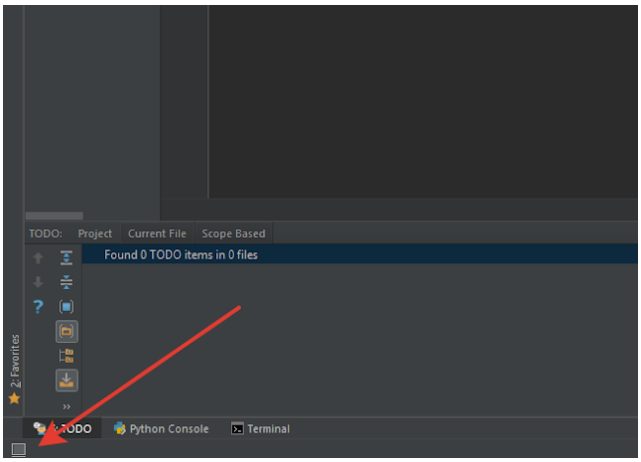
Muhit interfeysi quyidagi qismlardan iborat:

1. **Project Tool Window.** *Loyiha asboblar paneli*. Ushbu oynada loyiha fayllaringiz ko‘rsatiladi.
2. **PyCharm Editor.** *PyCharm muharriri*. O‘ng tomonda joylashgan, kod yoziladigan soha. Unda ochiq fayllar orasida oson harakatlanish uchun yorliqlar mavjud.
3. **Navigation Bar.** *Navigatsiya paneli.* PyCharm muharriri tepasida joylashgan bo‘lib, u ilovangizni tezda ishga tushirish va generatsiya qilish, shuningdek, VCS versiyasini boshqarish protseduralarini bajarish imkonini beradi.
4. **Left gutter.** Chap ustun, muharrir yonidagi vertikal satr, to‘xtash nuqtalarini ko‘rsatadi va kod ierarxiyasida harakat qilishning qulay usulini taqdim etadi. Shuningdek, u qator raqamlari va VCS tarixini ko‘rsatadi.
5. **Right gutter.** O‘ng ustun, tahrirlovchining o‘ng tomonida joylashgan. PyCharm sizning kodingiz sifatini doimiy ravishda kuzatib boradi va doimiy ravishda o‘ng ustunda tekshirish natijalarini ko‘rsatadi: xatolar, ogohlantirishlar va boshqalar. Yuqori o‘ng burchakdagi ko‘rsatkich butun fayl uchun umumiy kodni ko‘rib chiqish holatini ko‘rsatadi.
6. **PyCharm Tool Windows**. *PyCharm uskunalar paneli*. Bular ish maydonining pastki va yon tomonlariga biriktirilgan maxsus oynalar bo‘lib, ular loyihani boshqarish, manba kodini qidirish va navigatsiya, versiyalarni

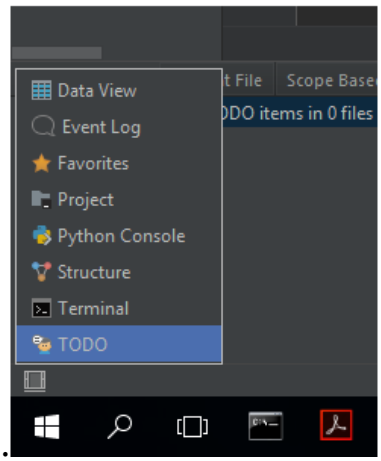
boshqarish integratsiyasi va boshqalar kabi umumiy vazifalarga kirishni ta‘minlaydi.

7. **Status Bar.** *Holat paneli.* Loyihangizning holatini ko‘rsatadi va turli xil ogohlantirish va ma’lumot xabarlarini ko‘rsatadi.

Bundan tashqari, PyCharm oynasining pastki chap burchagida holat satrida tugma mavjud. Ushbu tugma asboblar paneli ko‘rinishini almashtiradi. Agar sichqonchani ushbu tugma ustiga olib kelsangiz, hozirda mavjud panellar ro‘yxati paydo bo‘ladi:



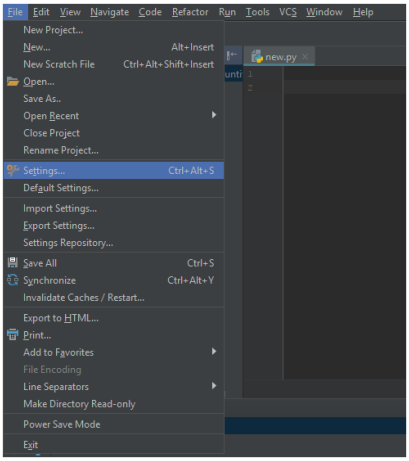
1.11-rasm. Asboblar paneli ko‘rinishini almashtirish tugmasi



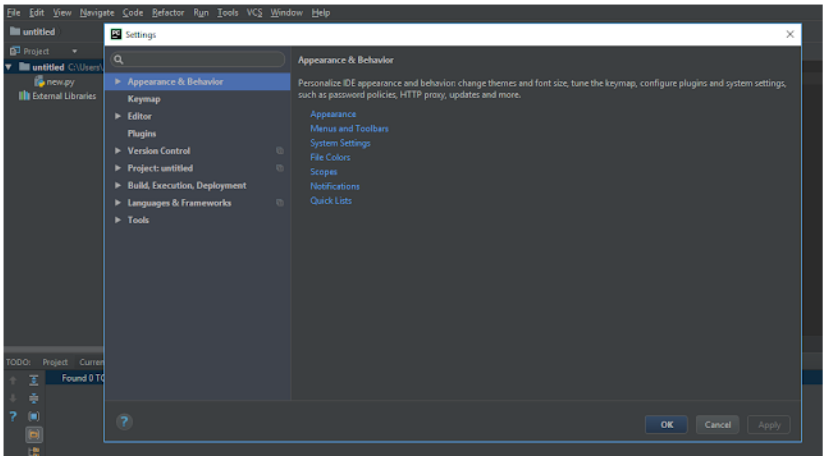
Ushbu tugma asboblar paneli ko'rinishini almashtiradi. Panellar o‘rtasida almashish

3-qadam. Muhitni sozlash

IDEni ehtiyojlaringizga to‘liq mos keladigan va siz uchun qulay bo‘lishi uchun sozlashingiz mumkin. Mavjud sozlamalar variantlari ro‘yxatini ko‘rish uchun **File/Settings** menyusiga o‘ting

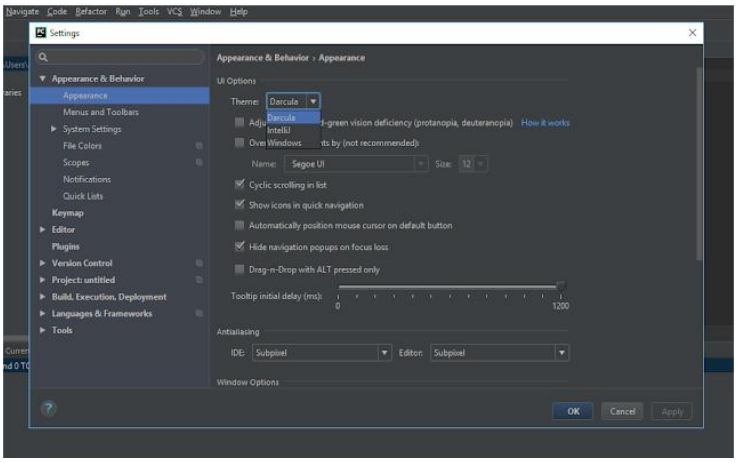


1.13-rasm. File|Settings oynasini ochish



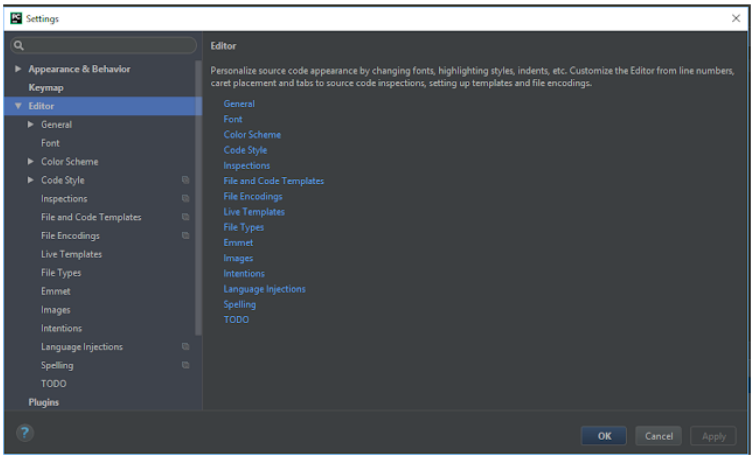
1.14-rasm. Sozlamalar oynasi

Tanlanadigan birinchi narsa - bu umumiy "tashqi ko'rinish". IDE dizaynini tanlash uchun **File/Settings/Appearance and Behavior/Appearance** menyusiga o‘ting: IDE quyidagi dizaynni taklif etadi: standart dizayn yoki qoramtir dizaynni yoqtirsangiz Dracula dizaynini taqdim etadi:



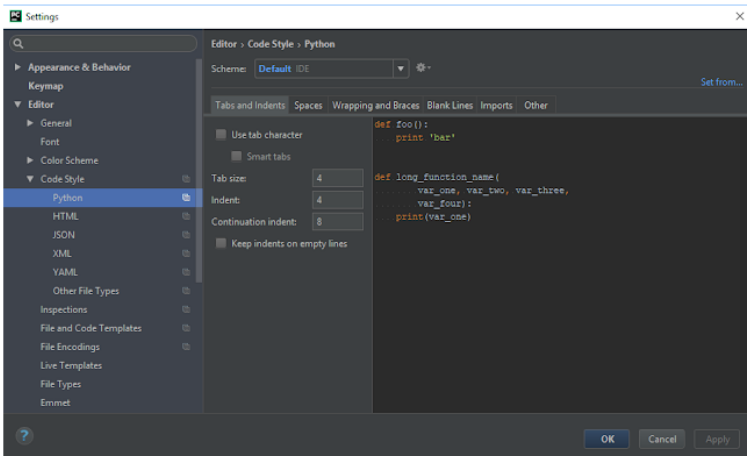
1.15-rasm. Tashqi ko‘rinish va xatti-harakatlar oynasi

File / Settings / Editor (Muharrir) menyusida mavjud bo‘lgan ko‘plab yorliqlar muharrir xatti-harakatlarining har bir jihatini moslashtirishga yordam beradi. Bu yerda umumiy sozlamalardan boshlab ko‘plab variantlar mavjud (masalan, Drag'n'Drop funksiyasidan foydalanish, aylantirish konfiguratsiyasi va boshqalar). Rangni har bir mavjud til uchun sozlash va foydalanish holatlari, yorliqlar va kodlarni toifalash sozlamalari, kodni bajarish harakati va hokazolar uchun.



1.16-rasm. Muharrirni sozlash oynasi

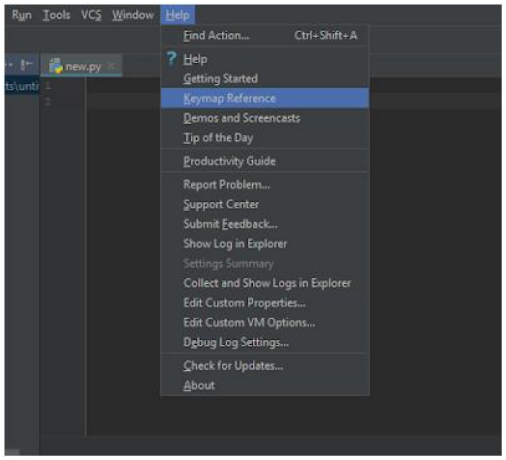
Kod stili har bir til uchun aniqlanishi mumkin **File / Settings / Editor / Code Style**. Shuningdek, siz o‘zingizning kod uslubingizni yaratishingiz va saqlashingiz mumkin.



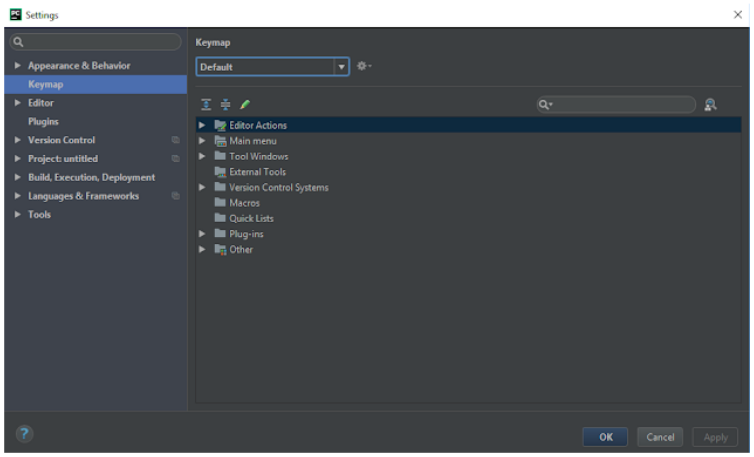
1.17-rasm. Kod stili bilan ishlash oynasi

PyCharm klaviaturaga yo‘naltirilgan yondashuvni qo‘llaydi, ya’ni IDE xaritasida mavjud bo‘lgan deyarli barcha amallar klaviatura yorliqlariga birlashtiriladi. Siz ishlaydigan tezkor tugmalar sizning shaxsiy odatlaringizdan biridir - barmoqlaringiz ma’lum tugmalar birikmalarini "eslab qoladi" va bu odatlarni o‘zgartirish juda qiyin.

PyCharm sizga standart tugmalar xaritasini taqdim etadi (asosiy menyudan **Help / Keymap Reference** buyrug‘i), bu kod yozishni haqiqatan ham samarali va qulay qiladi. Biroq, uni har doim **File / Settings / Keymap** bo‘limidan o‘zgartirishingiz mumkin

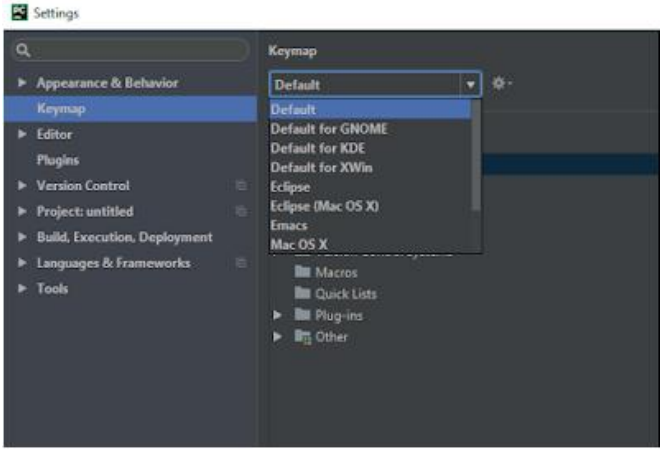


1.18-rasm. Klaviatura ma'lumotnomasi oynasini ochish



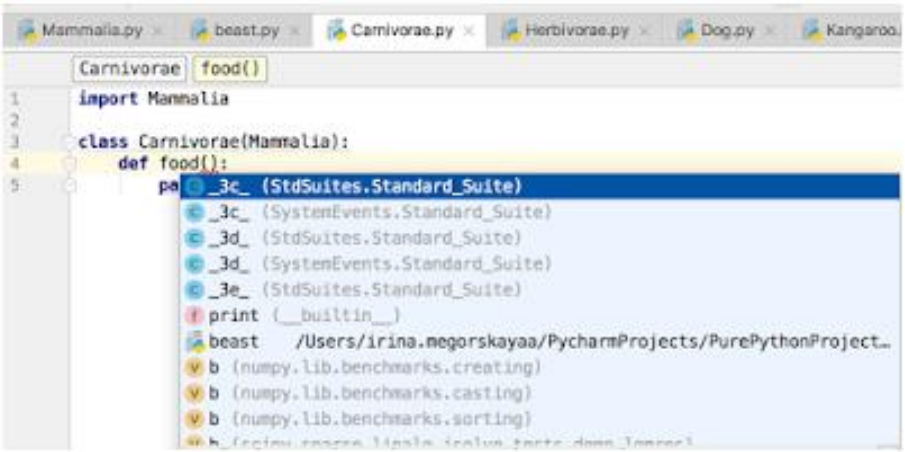
1.19-rasm. Klaviatura ma’lumotnomasi oynasi

Bundan tashqari, ba’zi oldindan belgilangan klaviatura maketlari (masalan, Emacs, Visual Studio, Eclipse, NetBeans va boshqalar) mavjud va siz mavjud bo‘lganiga asoslanib o‘z maketingizni ham yaratishingiz mumkin.



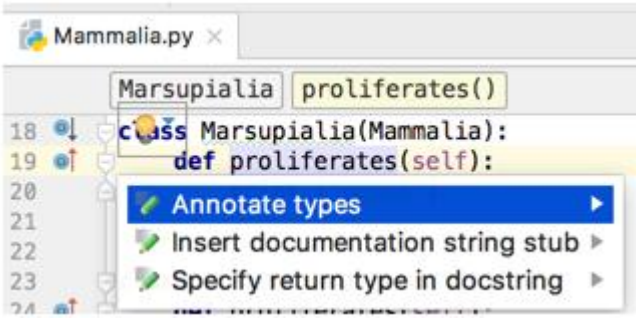
1.20-rasm. Ba’zi oldindan belgilangan klaviatura tartiblari

Qaysi turdagi fayl bilan ishlayotganingizdan qat’i nazar, avtomatik to‘ldirish kodi vaqtni tejash imkonini beradi. To‘ldirish siz yozganingizda ishlaydi va istalgan nomni bir zumda to‘ldiradi. Bashoratli kiritish hozir ishlayotgan kontekstni tahlil qiladi va shu tahlil asosida aniqroq takliflarni taklif qiladi.



1.21-rasm. PyCharm kodini avtomatik to‘ldirish

PyCharm hozir nima qilayotganingizni kuzatib boradi va **Intention Actions (harakatlar bashorati)** deb nomlangan aqlli takliflarni beradi. Sariq chiroq bilan ko‘rsatilganda, niyat harakatlari kodga avtomatik o‘zgarishlarni qo‘llash imkonini beradi.



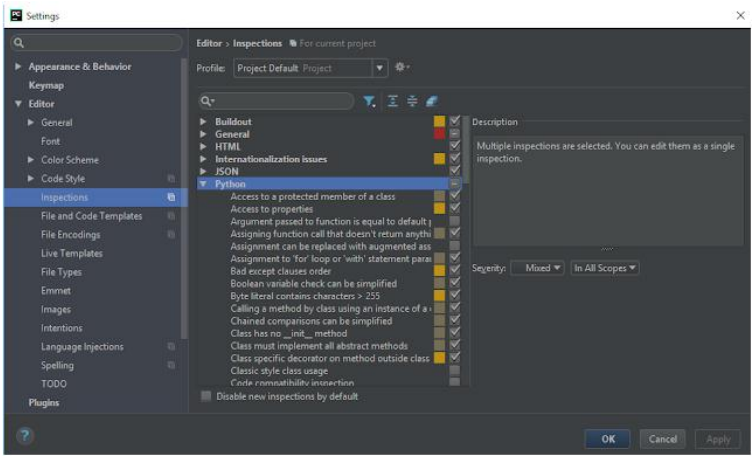
1.22-rasm. Harakatlar bashorati oynasi

Mavjud harakatlar bashoratlarining to‘liq ro‘yxatini **File / Settings / Editor / Intentions** bo‘limida topish mumkin

4-qadam: Kodni tartibli yozish

PyCharm sizning kodingizni boshqaradi hamda uni aniq va tartibli saqlashga harakat qiladi. U mumkin bo‘lgan xato va muammolarni aniqlaydi va ularni tezda tuzatishni taklif qiladi. IDE har safar foydalanilmagan kodni, cheksiz siklni va e’tiboringizni talab qilishi mumkin bo‘lgan boshqa narsalarni topsa, siz sariq chiroqni ko‘rasiz. Tuzatishni qo‘llash uchun uning ustiga sichqonchani yoki

Alt+Enter tugmalarini bosing. Mavjud tekshiruvlarning to‘liq ro‘yxatini **File / Settings / Editor / Inspections** menyu bo‘limida topish mumkin. Ulardan ba’zilarini o‘chiring yoki boshqalarni yoqing va har bir tekshiruvning jiddiylik darajasini belgilang. Bu xato yoki shunchaki ogohlantirish deb hisoblanishi kerakmi, o‘zingiz qaror qilasiz.



1.23-rasm. Koddagi xatoliklarni tekshirishni sozlash oynasi

5-qadam Kodni yaratish

PyCharmda mavjud kod yaratish opsiyalaridan foydalansangiz, kod yozish ancha oson va tezroq bo‘lishi mumkin. **The Code / Generate (Alt+Insert)** sizga belgilarni yaratishda yordam beradi va shuningdek, ayrim funksiyalarni bekor qilish/amalga oshirishni taklif qiladi.

II BOB. PYTHON DASTURLASH TILI ELEMENTLARI.

Ushbu bob Python tili bo‘yicha to‘liq va keng qamrovli qo‘llanma bo‘lib xizmat qilmasada, Python bilan ishlashni boshlash uchun yetarli bo‘lgan minimal ma’lumotni taqdim etadi. Agar siz boshqa algoritmik tillar bilan tanish bo‘lsangiz, bu tilni o‘rganish qiyin bo‘lmaydi.

2.1-§. Python tilining umumiy xususiyatlari

Python - faol rivojlanuvchi, kuchli, portativ, foydalanish uchun qulay va erkin foydalanish mumkin bo‘lgan dasturlash tili. U ishlab chiquvchilarning yuqori mahsuldorligiga va maksimal kodni o‘qishga qaratilgan. Inqilobiy xususiyatlar va innovatsiyalarni taklif qilmasdan, Python tili turli xil dasturlash tillarining asosiy eng yaxshi xususiyatlarini birlashtiradi. Python tilini yaratish ishlari 1980-yillarning oxirida gollandiyalik matematik Gvido van Rossum (Guido van Rossum) tomonidan boshlangan va tilni rivojlantirish foydalanuvchilar hamjamiyati tomonidan faol qo‘llab-quvvatlanmoqda.

Python tilining afzalligi - bu dasturlarning soddaligi va o‘qish uchun qulayligidir. Bunga qat'iy kodlash qoidalari orqali erishiladi, bu esa tushunishni osonlashtiradi. Ko‘p bo‘lmagan sondagi aniq asosiy tushunchalar tilni o‘rganish va undan foydalanishni osonlashtiradi. Python interpretatorining interaktiv ish rejimi imkoniyati tilni o‘rganish vaqtini qisqartiradi hamda qo‘yilgan vazifalarni hal qilishni osonlashtiradi.

Python tezkor dasturlarni ishlab chiqishga qaratilgan (RAD, Rapid Application Development). Dasturlarni kamroq vaqt ichida yaratishga o‘rnatilgan yuqori darajadagi ma’lumotlar tuzilmalari, dinamik toifalar va oddiy sintaksis yordamida erishiladi. Pythonda dasturlash C, C++ va Java kabi an'anaviy dasturlash tillariga nisbatan ishlab chiquvchilar samaradorligini sezilarli darajada oshiradi. Pythonda dastur kodining kichik hajmdan iborat bo‘lishi dasturiy mahsulotni yozish, qayta ishlash va texnik xizmat ko‘rsatish uchun vaqtni sezilarli darajada tejash imkonini beradi.

Python ochiq kodli dasturiy ta’minot toifasiga kiradi. Siz to‘liq Python dastur manbalarini olishingiz va undan hech qanday cheklovlarsiz foydalanishingiz

mumkin (nusxalash, tarqatish, boshqa mahsulotlarga joylashtirish). Bunday qo‘llab-quvvatlash Internet orqali butun dunyo bo‘ylab malakali mutaxassislar tomonidan taqdim etiladi.

Python ko‘plab platformalar uchun ishlab chiqiladi va unda yozilgan dasturlar hech qanday o‘zgarishsiz platformalar bo‘ylab ko‘chma hisoblanadi. Python tilining standart ilovasi portativ ANSI C da yozilgan bo‘lib, u deyarli barcha asosiy platformalarda kompilyatsiya qilinadi va ishlaydi. Asosiy til xususiyatlaridan va standart kutubxonalardan foydalanadigan Python dasturlari portativ bayt-kodga kompilyatsiya qilinadi hamda Windows va Linuxda, shuningdek Python o‘rnatilgan boshqa har qanday operatsion tizimda bir xil ishlaydi.

Python standart kutubxonasi amaliy dasturlarda qo‘llaniladigan keng imkoniyatlarni taqdim etadi. U matn, multimediya formatlari, arxivlar, tarmoq protokollari va Internet formatlari bilan ishlash modullarini o‘z ichiga oladi, modulli testlashni qo‘llab-quvvatlaydi va hokazo. Pythonda turli xil vazifalarni (web ilovalar, ma’lumotlar bazalari, grafik kutubxonalar va boshqalar) hal qilish uchun ko‘plab ilovalar kutubxonalari mavjud. Ularning ichida biz uchun eng qiziqarlisi raqamli usullar kutubxonalari bo‘lib, ular funksionalligi jihatidan MATLABni almashtirish imkonini beradi.

Ishlab chiqilgan dasturiy mahsulotlarning funksionalligi alohida komponentlarning integratsiyasi hisobiga kengaytiriladi. Siz Python interpretatorini boshqa tildagi dasturga joylashtirishingiz (embedding) yoki aksincha, Python uchun modullarni boshqa tillarda yozishingiz mumkin (extending). Standart kutubxona Python dasturlariga C tilida yozilgan dinamik bog‘langan kutubxonalarga (DLL, Dynamic-link library) to‘g‘ridan-to‘g‘ri kirish imkonini beradi. C/C++ kodini to‘g‘ridan-to‘g‘ri Python manba fayllariga joylashtirish imkonini beruvchi modullar mavjud, bu o‘z modullaringizni boshqa tillarda yozish uchun dasturlash interfeysidir. Ushbu kengaytma modullari C/C++ kodining samaradorligini Python interpretatorining qulayligi va moslashuvchanligi bilan birlashtirish imkonini beradi. Ushbu usulda Python tilining kamchiliklari to‘ldiriladi, ular birinchi navbatda dasturni bajarish tezligi bilan bog‘liq bo‘lib, ular har doim ham C yoki C++

kabi quyi darajadagi dasturlash tillarida yozilgan dasturlar kabi yuqori bo‘lmasligi mumkin.

2.2-§. Python ob’ektlari

Ob’ektga yo‘naltirilgan dasturlashning barcha hususiyatlaridan foydalangan holda, Python tilidagi barcha ma’lumotlar muayyan harakatlar amalga oshiriladigan ob’ektlardir. Alohida ob’ekt qiymatlari va ular bilan bog‘liq operatsiyalar to‘plami xotira maydoni bilan bog‘langan.

Python tilidagi dasturni modullar, ko‘rsatmalar, ifodalar va ob’ektlar kabi asosiy komponentlarga ajratish mumkin. Ierarxik tuzilma quyidagilarga asoslanadi:

• dasturlar modullardan iborat;

• modullar ko‘rsatmalardan iborat;

• ko‘rsatmala obyektlarni yaratuvchi va qayta ishlovchi ifodalardan iborat.

Ob’ektlar, shuningdek, boshqa vositalar yordamida yaratilgan ob’ektlar, masalan, C tilida yozilgan kengaytmali kutubxonalar, Python tili taqdim etilganda o‘rnatilgan bo‘lishi mumkin. Ob’ektlar o‘zgarmas va o‘zgaruvchan bo‘lishi mumkin. Misol uchun, Pythonda satrlar o‘zgarmasdir va shuning uchun satrlar ustida bajarilgan operatsiyalar yangi satrlarni yaratadi.

Python ko‘plab umumiy dasturlash masalalarini hal qiladigan tilga o‘rnatilgan ob’ektlar toifasining kuchli to‘plamini taqdim etadi. To‘plamlar (ro‘yxatlar) va qidirish jadvallari (lug‘atlar) kabi o‘rnatilgan ob’ektlardan bevosita foydalanish mumkin. Pythonning o‘rnatilgan turlariga raqamlar (butun sonlar, haqiqiq sonlar, kompleks sonlar), satrlar, ro‘yxatlar, lug‘atlar, kortejlar va fayllar asosiy turlar sifatida kiradi.

Python tili dinamik tuzilmaga ega dasturlash tillari sinfiga kiradi, bunda o‘zgaruvchi statik tuzilmada bo‘lgani kabi o‘zgaruvchini e’lon qilish vaqtida emas, balki qiymat tayinlash paytidagi tur bilan bog‘lanadi. Bunda ma’lumotlar turlari avtomatik ravishda aniqlanadi va ularni dastur kodida aniq e'lon qilish shart emas. Dasturning turli qismlarida bir xil o‘zgaruvchi har xil turdagi qiymatlarni olishi mumkin.

Python tilida turlarni e’lon qilish konstruksiyasi mavjud emas, bajarilgan ifodalar sintaksisining o‘zi yaratilgan va foydalaniladigan ob’ektlar turlarini belgilaydi. Ob’ekt yaratilgandan so‘ng, u butun vaqt davomida o‘ziga xos operatsiyalar majmuasi bilan bog‘lanadi - siz faqat uning turiga tegishli bo‘lgan ob’ekt ustida amallarni bajarishingiz mumkin.

Dinamik tuzilmada o‘zgaruvchining turi faqat ish vaqtida aniqlanadi. O‘zgaruvchilar u bilan bog‘liq tur yoki cheklovlar haqida hech qanday ma’lumotga ega emas: tur ob’ektning nomi emas, balki uning xususiyatidir. O‘zgaruvchilar faqat ma'lum ob’ektlarga va ma'lum vaqt momentlariga havolalarni o‘zida saqlaydi: bir xil o‘zgaruvchi dasturning turli qismlarida har xil turdagi ob’ektlar bilan bog‘lanishi mumkin. O‘zgaruvchidan foydalanishdan oldin unga qiymat berilishi kerak.

Pythondagi ob’ekt, uning turiga qo‘shimcha ravishda, murojaatlar soni bilan tavsiflanadi. Agar ob’ektga havolalar bo‘lmasa, u holda ob’ektlar egallagan xotira avtomatik ravishda bo‘shatiladi (garbage collection).

2.3-§. Python tilida sonli toifa

Pythonda boshqa dasturlash tillaridagi kabi standart sonli obyekt turlari (butun va suzuvchi nuqtali) mavjud. Shuningdek Python kompleks sonlar bilan ishlashni qo‘llab-quvvatlaydi. Sonlarni ifodalashning aniqligi Python tarjimonini yaratishda foydalaniladigan C kompilyatorining aniqligi bilan aniqlanadi. Python shuningdek, katta sonlarni ifodalash aniqligi bilan uzun butun sonlarni taqdim etadi (aniqlik mavjud xotira miqdoriga bog‘liq).

int turi (butun sonlar) foydalanilayotgan arxitektura uchun C kompilyatoridagi long turiga mos keladi. 32-bitli tizimda ishlayotganda, maksimal butun son $2^{31}–1$, minimal esa $–2^{31}$. Toifani aniqlash uchun type() operatoridan foydalaniladi.

```
>>> import sys
>>> a = sys.maxsize
>>> print(a)
9223372036854775807
>>> type(a)
```

<class 'int'>

Avval (maxsize buyrug‘i) int tipidagi maksimal butun sonni topamiz, so‘ngra (type buyrug‘i) a va b sonlarning turlarini aniqlaymiz.

Sakkizlik sonning belgisi sonning boshida 0, undan keyin 0-7 raqamlari. O‘n oltilik raqam uchun sonning boshida Ox yoki OX, keyin esa 0-9 va A-F o‘n oltilik raqamlaridan foydalaniladi.

```
>>> print(10,0o12,0xA)
10 10 10
```

Pythonda haqiqiy sonlar (float turi) C da double turi sifatida amalga oshiriladi. Suzuvchi nuqtali sonlar eksponensial ko‘rinishida tasvirlanadi, bunda nuqta belgisi hamda e yoki E belgilaridan iborat bo‘lishi mumkin:

```
>>> 2**100
1267650600228229401496703205376
>>> 2.**100
1.2676506002282294e+30
```

Python tilidagi kompleks sonlar haqiqiy va mavhum qismlarni ifodalovchi ikkita suzuvchi nuqtali raqamlardan iborat. Ifoda oxiridagi j yoki J belgisi mavhum qismni ko‘rsatish uchun ishlatiladi. Misol uchun

```
>>> z=5+8j
>>> z.real
5.0
>>> z.imag
8.0
```

z kompleks sonining haqiqiy va mavhum qismlari alohida ajratiladi.

Pythonda ikkita qiymat "True" (rost) va "False" (yolg‘on) dan iborat bool mantiqiy toifasiga ega. True va False xuddi 1 va 0 butun sonlari kabi ishlaydi, lekin ekranga chop etilganda ular 1 va 0 raqamlari o‘rniga True va False so‘zlari sifatida ko‘rsatiladi:

```
>>> a=8>10
>>> c=a+1
```

```
>>> print(a,c)
False 1
```

Sonli ob’ektlar bilan ishlash uchun standart math (haqiqiy sonlar) va cmath (kompleks sonlar) modullaridan foydalaniladi. Ilmiy hisoblash uchun ayniqsa muhim bo‘lgan bunday Python kengaytmalari haqida alohida qaraladi.

2.4-§. Python tilida satrli toifa

Pythonda matnli ma’lumotlar bilan ishlash uchun belgilarning tartiblangan ketma-ketligi bo‘lgan satrlar (string tipi) ishlatiladi. Python tilida satrlarni qayta ishlash vositalarining kuchli to‘plami mavjud. Yuqorida ta'kidlaganimizdek, satrlar tilning o‘zgarmas ob’ektlari hisoblanadi.

Satrlar apostrof (') yoki qo‘shtirnoq (") yordamida belgilanadi.

```
>>> a="Salom"
>>> b='Python'
>>> print(a,b)
Salom Python
```

Ko‘pqatorli satrlar bilan ishlash uchun uchtalik apostrof yoki qo‘shtirnoq ishlatiladi.

```
>>> a="'Bu
ko‘p qatorli
satr"'
>>> print(a)
Bu
ko‘p qatorli
satr
>>> b="""Huddi shunday bu
ham ko‘p qatorli
satr"""
>>> print(b)
Huddi shunday bu
ham ko‘p qatorli
```

satr

\ belgisi maxsus belgilar kiritish uchun ishlatiladi. Misol uchun

```
>>> s = 'a\nb\tc'
>>> print(s)
a
b	c
```

\n belgisi yangi satrni, \t belgisi esa gorizontal tabulyatsiyani ifodalaydi.

Satr operatsiyalarining asosiy to‘plamiga satrlarni birlashtirish konkatenatsiya operatori + bilan dublikatlash takrorlash operatori * kiradi:

```
>>> si = 'Hello‘
>>> s2 = 'Python'
>>> s3 = '!'
>>> print(si + ' ' + s2 + s3*3)
Hello Python!!!
```

len funksiyasi satr uzunligini qaytaradi:

```
>>> s='Python'
>>> len(s)
6
```

Python tilidagi satr ixtitoriy belgilar ketma-ketligidir. Ushbu ketma-ketlikning alohida elementlari ketma-ketlik pozitsiyasi raqami bo‘yicha indeksatsiya protsedurasi yordamida quyidagi tarzda olinadi:

```
s='Python'
>>> s[1]
'y'
```

Pythonda indekslash 0 dan boshlanadi: birinchi element indeks 0, ikkinchisi indeks 1 va hokazo.Shuningdek, teskari tartibda indekslash ham mumkin:

```
s='Python'
>>> s[-1]
'n'
```

Qatordagi belgilarning bir qismini olish uchun qirqib olish deb nomlanuvchi indekslashning umumiyroq shakli ishlatilishi mumkin. Misol uchun

```
>>> s='Python'
>>> s[2:5]
'tho‘
```

satrdagi 2 (indeks 1) dan 3 gacha bo‘lgan belgilar tanlanadi. Sukunat bo‘yicha, satr qismining chap tomoni nolga teng, o‘ng tomoni esa ketma-ketlikning (satr) uzunligi hisoblanadi:

```
s='Python'
>>> s[1:]
'ython'
```

2.5-§. Python tilida ro‘yxatlar

Pythonda ro‘yxatlar (list turi) boshqa dasturlash tillaridagi massivlarga o‘xshash, ammo ular ko‘proq xususiyatlarga ega. Satrlardan farqli o‘laroq, ro‘yxatlar har qanday turdagi ob’ektlarni o‘z ichiga olishi mumkin: raqamlar, satrlar va hatto boshqa ro‘yxatlar. Shuningdek, ro‘yxatlar o‘zgaruvchan ob’ektlar bo‘lib, ro‘yxatlar to‘g‘ridan-to‘g‘ri o‘sishi va qisqarishi mumkin (ularning uzunligi o‘zgarishi mumkin). Ro‘yxat elementlari kvadrat qavs ichiga olinadi va vergul bilan ajratiladi:

```
l=[123,1.23,'Python']
>>> print(l)
[123, 1.23, 'Python']
```

Ushbu misolda ro‘yxatning har bir elementi alohida turlardan iborat. Pythonda ro‘yxatlar satrlarda qo‘llaniladigan operatsiyalarini qo‘llab-quvvatlaydi

```
>>> l=[123,1.23,'Python']
>>> len(l)
3
>>> l[1:]
[1.23, 'Python']
```

append metodi ro‘yxat hajmini oshiradi va oxiriga yangi element qo‘shadi, pop metodi elementni ro‘yxatdan olib tashlaydi:

```
>>> l=[123,1.23,'Python']
>>> l.append('dastur')
>>> l
[123, 1.23, 'Python', 'dastur']
>>> l.pop(2)
'Python'
>>> l
[123, 1.23, 'dastur']
```

Ro‘yxatning kerakli joyiga yangi elementni kiritish uchun insert metodi qo‘llaniladi, qiymat bilan ko‘rsatilgan elementni olib tashlash uchun remove metodi qo‘llaniladi:

```
>>> l=[5, 2, 3, 4]
>>> l.insert(2,3)
>>> l
[5, 2, 3, 3, 4]
>>> l.remove(4)
>>> l
[5, 2, 3, 3]
```

Ro‘yxat elementlarini saralash uchun sort (o‘sish tartibida tartiblash) va reverse - kamayish tartibida tartiblash metodlari qo‘llaniladi:

```
>>> 1 = [3, 4, 2, 1]
>>> 1. sort ()
>>> 1
[1, 2, 3, 4]
>>> 1.reverse()
>>> 1
[4, 3, 2, 1]
```

Pythonda matritsalar (koʻp oʻlchovli massivlar) bilan ishlashga ichki roʻyxatlar yordamida erishish mumkin. 3x3 matritsaning elementlarini chiqarishga misol:

```
>>> A=[[1,2,3], [4,5,6], [7,8,9]]
>>> A[1]
[4, 5, 6]
>>> A[2][1]
8
```

Birinchi indeks (A[1]) ikkinchi qatorni, ikkinchisi (A[2][1]) 3-qatorning 2-elementini oladi.

2.6-§. Python tilida kortej va lugʻatlar

Kortejlar (tuple toifasi) ham roʻyxatlar kabi ketma-ketliklardir, lekin roʻyxatlardan farqli oʻlaroq, kortejlar oʻzgarmasdir. Shu ma'noda ular satrlarga oʻxshaydi. Kortejni belgilash uchun kvadrat qavslar emas, balki oddiy qavslar ishlatiladi:

```
>>> t=('pt', 10, 1.0)
>>> type(t)
<class 'tuple'>
>>> t[0]
'pt'
```

Obʼektlar toʻplamini dastur komponentlari oʻrtasida oʻtkazishda uzatilayotgan maʼlumotlarning oʻzgarmasligini ta'minlash uchun roʻyxatlar oʻrniga kortejlar qoʻllaniladi.

Lugʻatlar (diet toifasi) - bu kalitlar orqali kirish mumkin boʻlgan obʼektlar toʻplami: kalit-qiymat juftliklari, bu yerda qiymat kalit tomonidan yagona aniqlanadi. Oʻzgarmas maʼlumotlar turi (raqam, satr, kortej) kalit vazifasini bajarishi mumkin. Kalit-qiymat juftlarining tartibi ixtiyoriydir. Lugʻatni belgilashda sistemali qavslardan foydalaniladi, kalit-qiymat juftlarini ajratish uchun vergul qoʻllaniladi va : belgisi kalitni qiymatdan ajratib turadi. Lugʻat elementlariga quyidagi kalit orqali kirish mumkin:

```
>>> dk={'Ism': 'Ali', 'yosh':'26'}
>>> dk
{'Ism': 'Ali', 'yosh': '26'}
>>> dk['Hobby']='futbol'
>>> dk
{'Ism': 'Ali', 'yosh': '26', 'Hobby': 'futbol'}
```

Bu misol, shuningdek, yangi lug‘at yozuvini yaratishni ko‘rsatadi.

Lug‘atlarning o‘ziga xos metodlari mavjud: paydo bo‘lishini tekshirish, kalitlar ro‘yxati, qiymatlar ro‘yxati, nusxa ko‘chirish, standart qiymatni olish, birlashtirish, o‘chirish va hk.

2.7-§. Python tilida fayllar bilan ishlash

Fayl toifasidagi ob’ektlar Python kodi va kompyuterdagi tashqi fayllar o‘rtasida interfeysni ta'minlaydi.

O‘rnatilgan open funksiyasi fayl ob’ektini yaratadi va close tashqi faylga havolani uzadi. open funksiyasining birinchi argumenti fayl nomini o‘z ichiga olgan satr, ikkinchi argument esa fayldan foydalanish rejimi qatoridir. Foydalanish rejimi deganda faylni qanday maqsadda ochish nazarda tutiladi. Bu rejimlar quyidagilar:

❖ **“r”** – **Read** – faylni o‘qish uchun ochish. Agar fayl mavjud bo‘lmasa, xatolik yuz beradi.

❖ **“a”** – **Append** – faylga qo‘shimcha qo‘shish uchun ochish. Agar fayl mavjud bo‘lmasa yangi fayl ochadi.

❖ **“w”** – **Write** – faylga yozish uchun ochish. Agar fayl mavjud bo‘lmasa, yangi fayl ochadi.

❖ **“x”** – **Create** – yangi fayl hosil qilish. Agar bunday fayl mavjud bo‘lsa xatolik yuz beradi.

Bundan tashqari qo‘shimcha 2 ta rejim bor. Ular yuqoridagilar bilan qo‘llaniladi:

❖ **“t”** – **Text** – matn turi ya’ni fayl matndan iborat bo‘ladi.

❖ **“b”-- Binary** – binar rejim (ikkilik). Masalan rasmlarni binary rejimda joylashtirish.

Fayldan yozish va o‘qishga misol:

```
>>> f=open('D:\\data.txt', 'w')
>>> f.write('satr1\nSatr2\n')
>>> f.close()
>>> f=open('D:\\data.txt')
>>> s1=f.readline()
>>> s2=f.readline()
>>> s1
'satr1\n'
>>> s2
'Satr2\n'
```

Faylga yozish uchun write metodi, satr bo‘yicha o‘qish uchun esa readline metodi qo‘llaniladi.

III BOB. PYTHONDA DASTUR YOZISH QOIDALARI

Ushbu bobda Python dasturlash tilida identifikatorlarni e’lon qilish, dasturchilar qabul qilgan qoidalar bo‘yicha dastur kodini yozish, shart hamda takrorlash operatorlari bloklarini ifodalash bayon etilgan. Shuningdek istisno holatlari bilan ishlash ham qarab o‘tilgan.

3.1-§. Pythonda ishlash bo‘yicha ko‘rsatmalar

Python tilida identifikatorlar. O‘zgaruvchilar nomlari lotin harfi (har qanday holatda) yoki pastki chiziqdan keyin harflar, raqamlar, raqamlar yoki pastki chiziq bilan boshlanishi kerak. Pythonda harflar registri siz yaratgan nomlarda ham, ajratilgan so‘zlarda ham muhimdir.

Tilning kalit (zahiralangan) so‘zlarini identifikator sifatida ishlatish mumkin emas va o‘rnatilgan nomlarni qayta belgilash maqsadga muvofiq emas (3.1-jadval).

3.1-jadval. Python kalit so‘zlari

and	As	assert	Break	class	continue	Def	del
elif	else	except	Exec	finally	for	From	global
If	import	in	Is	lambda	not	Or	pass
Print	raise	return	Try	while	with	yield	

Pythonning o‘rnatilgan versiyasi uchun kalit so‘zlar ro‘yxati quyidagicha ko‘rsatilishi mumkin:

```
>>> import keyword
>>> keyword.kwlist
```

Qo‘shimcha nomlash qoidalari ismning boshida va oxirida pastki chiziqdan foydalanishga tegishli. Masalan, boshida va oxirida ikkita pastki chiziqli ismlar tizim nomlaridir.

Dasturning mantiqiy satrlari. Python dasturining asosiy elementi mantiqiy qatordir. Python dasturi, interpretator nuqtai nazaridan, mantiqiy satrlardan iborat. interpretatorning vazifasi mantiqiy satrlarni ketma-ket qayta ishlashdir. Agar qatorni birlashtirish nazarda tutilgan bo‘lsa, mantiqiy satr bir yoki bir nechta jismoniy

satrlardan iborat bo‘ladi. Jismoniy satr foydalanilayotgan platforma uchun qabul qilingan satr oxiri belgisi bilan tugaydi. Izoh # bilan boshlanadi va jismoniy satr oxirida tugaydi.

Odatda, har bir satrda bitta ko‘rsatma bo‘ladi, bir qatorga bir nechta ko‘rsatmalar yozilsa, ular ; belgisi bilan ajratiladi. Teskari chiziq belgisi (\) bir nechta jismoniy satrlarni bitta mantiqiy qatorga birlashtirish uchun ishlatiladi:

```
print a; print b; c = a
sr = a + b + c \
d + e
```

oddiy, kvadrat va sistemali qavslar ichidagi ifodalarni teskari chiziq belgisidan foydalanmasdan bir nechta jismoniy satrlarga bo‘lish mumkin:

```
1st = ['abc',
123,
1.23]
```

Python ko‘rsatmalarini bir nechta mantiqiy qatorlarga bo‘lish mumkin emas, murakkab ko‘rsatmalar bundan mustasno. Murakkab ko‘rsatmalar asosiy ko‘rsatmadan va blok ichiga kiritilgan ko‘rsatmalardan iborat. Murakkab ko‘rsatma quyidagi shablon bo‘yicha yoziladi

Asosiy ko‘rsatmalar:

Ichki ko‘rsatmalar bloki

ajratuvchi sifatida ikki nuqtadan foydalaniladi.

Ichki blok chap chegaradan bir xil chekinishlar yordamida formatlanadi. Python interpretatori blokning qayerdan boshlanishi va qayerda tugashini chekinish miqdori bilan aniqlaydi. Bitta blok ichida barcha ko‘rsatmalar chap chegaradan bir xil chekinishga ega bo‘lishi kerak.

Chekinish Python tili sintaksisining bir qismidir. Ulardan foydalanish sizga tushunish oson bo‘lgan yagona va qulay o‘qiladigan dastur kodini olish imkonini beradi. Bu kabi dizayn cheklovlari har qanday dasturlash tilida muhim ahamiyatga ega va kodingizning qo‘llanilishiga va uni qayta ishlatishga katta ta'sir ko‘rsatadi.

3.2-§. Pythonda dastur yozish stili

Dastur turi bo‘yicha asosiy cheklovlar dasturlash tilining sintaksisi bilan beriladi va agar u buzilgan bo‘lsa, interpretator sintaksis xatolarini chiqaradi. Dasturlash stili deganda, foydalanish uchun qulay, o‘qilishi oson va samarali dasturlarni olish uchun dasturiy ta’minot ishlab chiquvchilarning ma'lum bir guruhi tomonidan qabul qilinadigan va foydalaniladigan kod dizayniga qo‘shimcha cheklovlar tushuniladi.

Dasturlash stili kodni loyihalashning barcha jihatlariga taalluqlidir: nomlarni tanlash va o‘zgaruvchilar nomlari uchun ishlatiladigan holat, sharhlar uslubi, mantiqiy bloklar, modullar, hujjatlar dizayni va boshqalar. Python tili uchun rasmiy kod yozish stili ishlab chiqilgan (Python Style Guide).

Python tilidagi dasturlash stilining ba’zi pozitsiyalariga e’tibor qaratamiz. Kod yozishda quyidagilar tavsiya etiladi:

• 4 ta probeldan iborat bo‘shliqlardan foydalanish;

• 79 belgidan ko‘p bo‘lmagan jismoniy satrlardan foydalanish;

• mantiqiy satrlarni bilvosita uzish (qavslar ichida);

• davomli satrlarning chekinishini qavslar yoki oldingi qatordagi birinchi operand bilan tekislash;

• ochiladigan qavsdan keyin yoki yopilishdan oldin, vergul, nuqtali vergul oldiga darhol bo‘sh joy qo‘ymang;

• topshiriqlardagi teng belgisi atrofida bir nechta bo‘sh joy qo‘ymang (birlamchi qiymatni belgilash uchun foydalanilganda teng belgisi atrofida bo‘sh joy qolmaydi).

Sharhlarni yozish bo‘yicha quyidagi tavsiyalar mavjud:

• kodni o‘zgartirganda sharhlarni yangilash;

• qisqa sharh uchun oxirida nuqta qo‘ymaslik, matn yozishda odatiy qoidalarga muvofiq uzun sharhlarni yozish yaxshiroqdir;

• izohlarni ingliz tilida yozish;

• kod fragmentiga sharhlar (blok sharhi) sharhlangan kod bilan bir xil chekinish bilan cheklanadi, # belgisidan keyin bitta bo‘sh joy qo‘yiladi, paragraflar

bir xil darajadagi # bilan qator bilan ajratilishi mumkin, blok izohi atrofdagi koddan boʻsh chiziqlar bilan ajratiladi;

• inline izohlar ma'lum bir qatorga tegishli va tez-tez ishlatilmasligi kerak, # belgisi sharhlangan operatordan kamida ikkita boʻsh joy bilan ajratilishi kerak;

• moduldan tashqarida foydalanish uchun moʻljallangan barcha modullar, sinflar, funksiyalar va usullarda ulardan qanday foydalanishni, kiritish va chiqish parametrlarini tavsiflovchi hujjat qatorlari boʻlishi kerak;

• satrli hujjatlar uchun uch qoʻshtirnoqdan foydalanish;

• aniq holatlar uchun bitta satrli hujjatdan foydalaning;

• koʻp qatorli hujjatlar yigʻma qatordan keyin boʻsh satr va batafsilroq tavsifdan iborat boʻlsin;

• koʻp qatorli hujjatlardagi oxirgi paragraf va uning yakunlovchi tirnoqlari orasiga boʻsh qator qoʻying, tirnoqlarni alohida qatorga qoʻying.

Konvensiyalarning uchinchi guruhi Pythondan foydalanadigan har qanday dasturchi tushunishi uchun turli xil ob’ektlarni nomlash qoidalariga tegishli:

• modul nomlarini kichik harflar bilan berish yoki soʻzlarning birinchi harflarini bosh harflar bilan yozish yaxshiroq, C tilida yozilgan kengaytmali modullarning nomlari odatda pastki chiziq bilan boshlanadi;

• sinf nomlarida soʻzlarning birinchi harflari bosh harf bilan yoziladi;

• konstantalarning nomlari (ular qayta belgilanmasligi kerak) katta harflar bilan yozilishi yaxshi;

• Istisno nomlari xato (yoki ogohlantirish) soʻzini oʻz ichiga oladi.

Topshiriq boʻyicha koʻrsatmalar. Topshiriq bayonoti har doim ob’ektga havola yaratadi va hech qachon ob’ektlarning nusxalarini yaratmaydi. = belgisi identifikatorni (mavjud yoki yangi) ob’ekt bilan bogʻlash, ob’ekt atributlarini yaratish va oʻzgartirish, oʻzgaruvchan ketma-ketlik elementlarini oʻzgartirish, berilgan yozuvlarni qoʻshish va oʻzgartirish uchun ishlatiladi.

Ommaviy turdagi topshiriq koʻrsatmasidan foydalanish mumkin

```
>>> a = b = c = 0
```

Xuddi shuningdek, kodni minimallashtirish muammosi ixcham shakl bilan hal qilinadi, masalan, x = x + y o‘rniga biz x += y dan foydalanamiz.

chop etish bayonoti.

U ifodalar ro‘yxatidagi har bir ifodani ekranga chop etadi - bu ko‘rsatma ob’ektlarni matn tasviriga aylantiradi va natijani standart chiqarish qurilmasiga yuboradi.

```
>>> c=5.
>>> d=15
>>> print(c/2, d+1)
2.5 16
```

3.3-§. if shart operatori

Python tilidagi shartli if ifodasi qo‘shma ko‘rsatmaga misol bo‘la oladi. Pythonda ushbu ko‘rsatmaning umumiy tuzilishi quyidagicha:

```
if condition1:
body1
elif condition2:
body2
elif condition3:
body3
elif condition(n-l):
body(n-l)
else:
body(n)
```

Agar condition1 shart rost bo‘lsa, bodyl bajariladi; aks holda, agar condition2 shart rost bo‘lsa, u holda body2 bajariladi va hokazo, haqiqiy shart topilgunga qadar yoki haqiqiy shartlar bo‘lmasa, body(n) bajariladi.

Agar boshqa qism bo‘lmasa, shartli operator hech narsani bajarmaydi.

Shartli operatorga ega oddiy namuna:

```
if a < 0:
b = 1
```

```
elif a == 0:
b = 0
else:
b = -1
```

3.4-§. while takrorlash operatori.

while konstruksiyasi Pythondagi eng koʻp qirrali iteratsiya konstruktsiyasi boʻlib, unda shartli ifoda rost boʻlib baholanishi davom etar ekan, konstruksiyalar bloki bajariladi. while sikli formati quyidagicha:

```
while condition:
body
else:
post-code
```

Bu yerda condition sharti - rost yoki yolgʻon deb baholanadigan shart. Shart rost qiymat qabul qilar ekan, body halqasining tanasi doimiy ravishda bajariladi. Agar shart bajarilmasa, post-code dastur bloki bajariladi.

Agar sikl boshlangan boʻlsa va shart notoʻgʻri boʻlsa, siklning tanasi bajarilmaydi, lekin siklning post-code qismi bir marta bajariladi.

```
s = 0.
i = 1
while i < 100:
s = s + l./i**2
i = i + 1
```

Faqat sikllar ichida ishlatilishi mumkin boʻlgan ikkita oddiy break va continue iboralari mavjud. Agar break bajarilsa, while sikli darhol tugatiladi, hatto post-code qismi ham bajarilmaydi. continue bayoni bajarilganda body sikli tanasi tugatiladi, condition sharti yana baholanadi va sikl boshidan boshlanadi. Agar tanada break iborasi boʻlmasa, while siklining else qismi ishlatilmasligi mumkin.

Til sintaksisi bayonotni talab qiladigan hollarda, hech narsa bajarmaydigan pass bayonoti ishlatiladi.

3.5-§. for takrorlash operatori

for sikli ketma-ketlikdagi har bir element uchun sikl tanasini bajaradi. for operatori satrlar, ro‘yxatlar, kortejlar bilan ishlaydi. Qiymatlar ro‘yxati bilan ishlashda siklni bajarishning umumiy shakli:

```
for variable in list:
body
else:
post-code
```

body tanasi ro‘yxatning har bir elementi uchun bir marta bajariladi. else konstruksiyasi odatda ishlatilmaydi. break va continue ifodalari while siklidagi kabi ishlatiladi. Raqamlar yig‘indisini hisoblash:

```
x = [1, 2, 3, 4, 5]
s = 0
for n in x:
s = s + n
```

range() funksiyasi indekslar bilan ishlash uchun ishlatiladi. Bir argumentga ega range() funksiyasi argumentda ko‘rsatilgan qiymatgacha noldan o‘zgarib turadigan butun sonlar ro‘yxatini hosil qiladi, lekin uni o‘z ichiga olmaydi. Ikki argumentli range() funksiyasida birinchisi diapazonning pastki chegarasi sifatida ko‘rib chiqiladi. Ixtiyoriy uchinchi argument butun sonlar ro‘yxatidagi qadamni belgilaydi, shuning uchun

```
>>> range(-3, 10,2)
[-3, -1, 1, 3, 5, 7, 9]
```

Siklda range() funksiyasidan foydalanish:

```
for i in range(1,10):
    for j in range(1,10):
        print i, j, i*j
```

Ushbu misol ichma-ich sikllar bilan ishlashni ko‘rsatadi.

Istisno interpretator dasturni bajarish jarayonida xatolikka duch kelganda chaqiriladi. Dasturda bunday xatolarni qayta ishlashimiz yoki ularni e’tiborsiz qoldirishimiz mumkin. Ikkinchi holda, interpretator dasturning bajarilishini to‘xtatadi va xato xabarini ko‘rsatadi.

```
>>> a = 0.
>>> b = 2 / a
Traceback (most recent call last):
File "<console>", line 1, in <module>
ZeroDivisionError: float division
```

Mavjud xatolarni aniqlash (nolga bo‘lish) va dasturni bajarishni davom ettirish imkonini beruvchi try konstruksiyasini qo‘shishingiz mumkin. Try konstruksiyasining standart shakli quyidagicha:

```
try:
do something
except error:
do something else
```

Bizning misolimizda dasturning natijasi quyidagicha bo‘ladi

```
a = 0.
try:
b = 2 / a
except ZeroDivisionError:
print(’Division by zero!’)
```

natijaviy satr

```
Division by zero!
```

Batafsil umumiy istisno ishlov beruvchi konstruksiyalarini til hujjatlarida topish mumkin.

IV BOB. FUNKSIYA VA MODULLAR

Ushbu bobda Python dasturlash tilida funksiyalarni e’lon qilish, funksiya parametrlari bilan ishlash, lambda funksiyalar hususiyatlari bayon etilgan. Modullar va ularni yaratish hamda dasturda modullardan foydalanish qoidalari muhokama qilinadi.

4.1-§. def instuksiyasi.

Pythonda funksiyalar dastur tuzilishi va kodni qayta ishlatishni ta'minlaydigan asosiy dasturlash tuzilmalaridan biridir.

Pythonda def instruksiyasi sarlavha qatoridan hamda amallar blokidan iborat:

def name(argl, arg2,..., argN):

body

def ko‘rsatmalari funksiya ob’ekti bog‘lanadigan funksiya nomi name ni va bir nechta argl, arg2, ..., argN argumentlar (parametrlar) ro‘yxatini belgilaydi. Parametrlar chaqiruv nuqtasida funksiyaga uzatiladigan ob’ektlar bilan bog‘langan. Body funksiya tanasi har safar funksiya chaqirilganda interpretator tomonidan bajariladigan dastur kodini hosil qiladi.

return instruksiyasi funksiya tanasining istalgan joyiga joylashtirilishi mumkin. Uning dasturda yozilishi ixtiyoriy bo‘lib, agar u mavjud bo‘lmasa, boshqaruv oqimi funksiya tanasining oxiriga yetganda funksiya tugaydi.

Funksiyani yaratish va uni chaqirishga misol:

```
def yig(x, y):
return x + y
print yig(4, 3), yig(’Python’, ’ dasturi’)
```

dastur natijasi

7 Python dasturi

Funksiyani chaqirish funksiya nomi (yig) bilan amalga oshiriladi, funksiya nomidan keyin qavslar ichida funksiya sarlavhasida ko‘rsatilgan argumentlarga qiymatlab beriladi.

Funksiyaga birinchi murojaatda 4 va 3 sonlarni qo‘shish amalga oshiriladi, ikkinchi holatda biz " Python" va " dasturi" satrlari bilan ishlaymiz, natijada bu satrlarni birlashtirish hosil bo‘ladi. Ushbu misol Pythonda polimorfizmni ko‘rsatadi, bu yerda bir xil funksiya turli xil ma’lumotlar turlari uchun bir xil tarzda amalga oshiriladi.

4.2-§. O‘zgaruvchan argumentlar soni

Python tili o‘zgaruvchan sonli argumentlar bilan funksiyani aniqlash uchun turli imkoniyatlarni amalga oshiradi. Eng oddiy va eng muhim usul - bir yoki bir nechta argumentlar uchun standart qiymatni o‘rnatish. Agar funksiya chaqirilganda tegishli argument ko‘rsatilmagan bo‘lsa, standart argument qiymatlari ishlatiladi. Buning uchun funksiya e’lonida mos keladigan argumentga qiymat belgilashingiz kerak. Funksiya chaqiruvi davomida hech qanday argument berilmasa, o‘zgaruvchi o‘zining standart qiymatini qabul qiladi.

```
def funk(x,y=5):
    return x+5-y
print(funk(2),funk(5,3))
```

Natija

2 7

Pythonda funksiyani ixtiyoriy sonli argumentlar bilan chaqirish mumkin bo‘lgan tarzda belgilash mumkin. Bunday holda, argumentlar kortej sifatida uzatiladi. Argumentlarning o‘zgaruvchan sonidan oldin ixtiyoriy sonli muntazam argumentlar mavjud bo‘lishi mumkin.

Quyida ro‘yxat elementlarining yig‘indisini hisoblash funksiyasidan foydalanamiz

```
def fsum(*argum):
    smm=0
    for arg in argum:
        smm+=arg
    return smm
print(fsum(), fsum(1), fsum(1,2,3))
```

Dastur ishlashi natijasida olingan natija

0 1 6

Argumentlar oddiy lug‘at vositalarida qayta ishlanishi mumkin bo‘lgan yangi lug‘atga yig‘ilganda, def name(**args) konstruktsiyasidan foydalanib, argumentlar argumentlarini kalit orqali o‘tkazish ham mumkin.

4.3-§. lambda funksiyalari

Python lambda konstruktsiyasidan foydalangan holda dastur bajarilishi vaqtida anonim funksiyalarni (masalan, nom bilan bog‘lanmagan funksiyalar) yaratishga imkon beradi. Bu holda funksiya ob’ektlari ifodalar ko‘rinishida yaratiladi.

lambda kalit so‘zidan keyin bir yoki bir nechta argumentlar, so‘ng ikki nuqta va ifoda keladi:

lambda argl, arg2, argN: ifoda.

Oddiy funksiya e’loni (f1) hamda lambda funksiyasi e’loni (f2) o‘rtasidagi farq quyidagi misol yordamida tushuntiriladi.

```
def fl (x):
    return x**2
f2 = lambda x: x**2
print f1(4), f2(4)
```

Natija

16 16

Bunday konstruktsiyalar kichik funksiyalarni yaratish uchun qulay va ularni ishlatadigan kodga funksiya e’lonlarini kiritish imkonini beradi.

Pythonda ismlar fazosi. Dasturning har bir nuqtasi bilan bog‘langan uchta ism fazosi mavjud: mahalliy, global va o‘rnatilgan. Ismlar fazosi nomlarni ob’ektlarga solishtirishni belgilaydi.

Ismlar fazosi vaqtning turli momentlarida yaratiladi va turli amal qilish muddatlarga ega. O‘rnatilgan ismlar fazosi interpretator ishga tushganda yaratiladi va interpretatorning ishlashi davomida mavjud bo‘ladi. Modulning global ismlar fazosi o‘qilganda yaratiladi va odatda interpretator ish tugatilgunga qadar mavjud

bo‘ladi. Funksiyaning mahalliy ismlar fazosi funksiya chaqirilganda yaratiladi va funksiya ichida amal qiladi.

Funksiya tanasi ichida e’lon qilingan nomlar faqat funksiya tanasi ichidagina ma’noga ega bo‘ladi. Bu nomlarga funksiyadan tashqarida murojaat qilish mumkin emas. Funksiya tanasi ichidagi global xususiyatni o‘z ichiga olgan modulidagi o‘zgaruvchi global instruksiyasi yordamida taqdim etiladi.

Ismlar ketma-ket to‘rt bosqichda ko‘rib chiqiladi: mahalliy, keyin qo‘shimcha funksiya (agar mavjud bo‘lsa), keyin global va nihoyat o‘rnatilgan.

4.4-§. Modulni yaratish va foydalanish

Dasturlashga modulli yondashuvda katta masala bir nechta kichikroq masalalarga bo‘linadi, ularning har biri alohida modul tomonidan bajariladi. Ular orasidagi aloqalarni minimallashtiradigan modullarning shunday tarkibini tuzish juda muhimdir. Uning elementlari o‘rtasida ko‘plab bog‘lanishlarga ega bo‘lgan funksiyalar to‘plamini bitta modulga joylashtirish mantiqan to‘g‘ri keladi.

Python dasturi qo‘shimcha fayllar biriktirilishi mumkin bo‘lgan bitta asosiy fayl sifatida tashkil etilgan. Har bir bunday fayl ko‘rsatmalarni o‘z ichiga olgan alohida moduldir.

Python dasturlarida foydalaniladigan modullar kelib chiqishi bo‘yicha oddiy modullarga (Pythonda yozilgan) va boshqa dasturlash tilida yozilgan kengaytma modullariga bo‘linadi (masalan, C).

Modulni Python dasturiga kiritish uchun u asosiy fayl yoki boshqa dastur fayllari tomonidan import qilinishi kerak.

Modul yaratish uchun matnli faylga Python kodini kiritish va uni .py kengaytmasi bilan saqlash kerak. Bunday fayl, sukunat bo‘yicha, Python moduli hisoblanadi. Modulning yuqori darajasida tayinlangan barcha nomlar uning atributlariga (modul ob’ekti bilan bog‘langan nomlar) aylanadi va mijozlar foydalanishi mumkin bo‘ladi.

Modul yaratishga misol

```
# module1 (file module1.py)
def times(x, y):
```

```
        return x * y
    def add(x, y):
        return x + y
```

bu modul ikkita funksiyani o‘z ichiga oladi: times va add.

Ushbu moduldan asosiy main.py faylida foydalanamiz. Buning uchun import konstruksiyasidan foydalaniladi:

```
# main (file main.py)
import modulel
print module1.add(5, 2), module1.times(’Modul ’, 5)
```

Dastur natijasi quyidagicha

7 Modul Modul Modul Modul Modul

Asosiy modulda import konstruksiyasi bajarilishi natijasida to‘liq modul ob’ektiga yo‘l ko‘rsatuvchi nom (module1) dan foydalanish mumkin bo‘ladi. Uning atributlariga (times va add funksiyalari) murojaat qilganda, siz modul nomini nuqta bilan ishlatishingiz kerak (module1.add () va modul1.times ()).

Import qilingan modulga havola nomini o‘zgartirish uchun import ... as konstruktsiyasidan foydalaniladi.

```
# main (file main.py)
import module1 as m1
s=m1.add('Import ', 'as ')
print(m1.add(s,' m1'))
```

Dastur natijasi

Import as m1

Ba'zan modulni to‘liq emas, balki uning sohasidagi alohida nomlarni import qilish qulayroqdir. Bunday holda, from ... import iborasi ishlatiladi. Bizning misolimizda

```
# main (file main.py)
from module1 import add
print add('Import', 'add')
```

Import qilinadigan nom (add) from ... import koʻrsatmasini oʻz ichiga olgan sohagaga koʻchiriladi va uni oʻz ichiga olgan modul nomini koʻrsatmasdan foydalanish mumkin.

Import qilinadigan nomlari roʻyxati oʻrniga * belgisidan foydalangan holda from...import koʻrsatmasining maxsus shakli mavjud. Bunday holda, barcha nomlar import qilinadi:

```
# main (file main.py)
from module1 import *
print(add('Import ', 'all'), times('!',4))
Import all !!!!
```

Modulda qanday nomlar aniqlanganligini bilish uchun siz dir() funksiyasidan foydalanishingiz mumkin. U satrlarning tartiblangan roʻyxatini qaytaradi:

```
# main (file main.py)
import module1
print(dir(module1))
['__builtins__', '__cached__', '__doc__', '__file__', '__loader__', '__name__',
 '__package__', '__spec__', 'add', 'times']
```

Import qilish interpretator tomonidan faqat bir marta birinchi import yoki from ... import instruksiyasi bilan amalga oshiriladi. Keyingi import qilish operatsiyalari allaqachon yuklangan modulning ob'ektini oladi. Modullarni qayta yuklash reload funksiyasi tomonidan amalga oshiriladi.

4.5-§. Modulni qidirish

Modulni import qilishda Python interpretatori modulni quyidagi ketma-ketlikda qidiradi:

• joriy katalogda (asosiy fayl joylashgan);

• modul topilmasa, Python PYTHONPATH muhit oʻzgaruvchisidagi har bir katalogni qidiradi;

• agar yuqoridagilardan topilmasa, interpretator Python standart kutubxona modullari oʻrnatilgan kataloglarni tekshiradi.

Qidiruvning bunday tashkil etilishi sababli barcha dastur modullarini bitta katalogga joylashtirish maqsadga muvofiqdir. Dasturni ishga tushirganingizda qidiruv manzilini sys.path ro‘yxati tarkibiga qarab bilib olishingiz mumkin (Python standart kutubxonasining bir qismi bo‘lgan sys modulini import qilgandan keyin):

```
# main (file main.py)
import sys
import module1
print sys.path
['C:\\Users\\MTC\\AppData\\Roaming\\JetBrains\\PyCharmCE2020.3\\scratches',
'C:\\Users\\MTC\\PycharmProjects\\Qo‘shish',
'C:\\Users\\MTC\\AppData\\Local\\Programs\\Python\\Python37\\python37.zip',
'C:\\Users\\MTC\\AppData\\Local\\Programs\\Python\\Python37\\DLLs',
'C:\\Users\\MTC\\AppData\\Local\\Programs\\Python\\Python37\\lib',
'C:\\Users\\MTC\\AppData\\Local\\Programs\\Python\\Python37',
'C:\\Users\\MTC\\PycharmProjects\\Qo‘shish\\venv',
'C:\\Users\\MTC\\PycharmProjects\\Qo‘shish\\venv\\lib\\site-packages']
```

4.6-§. Standart modullar

Qo‘shimcha modullarning keng to‘plamiga ega bo‘lgan standart kutubxonalar Python dasturlash tilining jozibasidir. Ushbu to‘plam o‘nlab yirik modullarni o‘z ichiga oladi va keng tarqalgan vazifalarni qo‘llab-quvvatlaydi.

Operatsion tizim bilan ishlash uchun modullar majmuasi platformalararo ilovalarni yozish imkonini beradi. Shuningdek, ko‘plab tarmoq protokollari va Internet formatlari bilan ishlash, pochta xabarlarini tahlil qilish va yaratish, XML bilan ishlash uchun vositalar mavjud. Shuningdek, regulyar ifodalar, matn kodlari, multimediya formatlari, kriptografik protokollar, arxivlar, ma’lumotlarni seriyalashtirishni qo‘llab-quvvatlash bilan ishlash modullar ham mavjud. Ilmiy hisoblash uchun qiziq bo‘lgan Python standart kutubxonasining ba’zi xususiyatlari keyinroq yoritiladi.

Modullar haqida batafsil ma’lumot olish uchun Python standart kutubxonasi qo‘llanmasidan foydalanish mumkin. Buning uchun Пуск (Start) | Python x.x | Python Manual oynasida The Python Standard Library bo‘limiga kiriladi.

Modul paketlari. Yirik dasturiy mahsulotlarda ierarxik fayl strukturasini tashkil qilish uchun katalogdan foydalanish qulay. Python nafaqat modul nomlarini, balki katalog nomlarini ham import qilish imkonini beradi. Ushbu kontekstdagi katalog paketdir va Pythonda bunday import operatsiyasi paketlarni import qilish deb ataladi.

Paketlarni import qilishning asosiy xususiyati import va from ... import ko‘rsatmalarda mos modulga yo‘lni ko‘rsatishdir. Paket dir1 katalog sifatida aniqlansin. Unda bizga kerak bo‘lgan module1 modulni o‘z ichiga olgan dir2 kichik katalog mavjud (fayl module1.py). Keyin ushbu modulni import qilish quyidagi ko‘rsatma bo‘yicha amalga oshiriladi

```
# main (file main.py)
import dir1.dir2.module1
```

Bunday holda, dir1 katalogi modulni qidirish yo‘lida bo‘lishi kerak.

V BOB. MATEMATIK PYTHON

O‘rnatilgan funksiyalar va Python standart kutubxonasi dasturlashning turli xil vazifalarini hal qilish uchun keng imkoniyatlar beradi. Bundan tashqari, Python foydalanuvchi kutubxonalari bilan kengaytirilishi mumkin. Ushbu bobda sonli tahlil masalalarini hal qilish vositalari muhokama qilinadi. Python standart kutubxonasining matematik modullari hamda NumPy paketi haqida qisqacha tavsif berilgan.

5.1-§. O‘rnatilgan funksiyalar va standart kutubxona

Pythonda bir qator o‘rnatilgan funksiyalar mavjud, ulardan ba’zilarini biz allaqachon tanishdik. O‘rnatilgan funksiyalarning batafsil tavsifi bilan Python qo‘llanmasidan tanishishingiz mumkin (Пуск (Start) | Python x.x | Python Manual). Bu yerda kelgusi hisob-kitoblarni amalga oshirishda kerak bo‘ladigan ba’zilari bilan tanishamiz.

Aralash tipdagi sonlar ustida arifmetik amallar bajarilganda, umumiy yuqori turga avtomatik o‘tkazish amalga oshiriladi. Sonlar ierarxiyasi: butun sonlar, uzun butun sonlar, suzuvchi nuqtali sonlar, kompleks sonlar. Xususan, Python interpretatori, agar ularning qiymatlari oddiy butun son formatiga sig‘maydigan darajada katta bo‘lsa, avtomatik ravishda butun sonlarni uzun butun sonlarga aylantiradi. Majburiy toifa almashtirish uchun int(), long(), float(), kompleks() funksiyalaridan foydalaniladi. Ulardan foydalanishga misollar:

```
>>> a=1
>>> b=-2.3
>>> c='4.5'
>>> print(a+b)
-1.2999999999999998
>>> print(int(b))
-2
>>> print(float(c))
4.5
>>> print(complex(a,b))
```

(1-2.3j)

Core Python faqat bir nechta matematik funksiyalarni qo‘llab-quvvatlaydi. Masalan, abs(x) raqamning mutlaq qiymatini, ya’ni berilgan sonning modulini |*x*|:

```
print(abs(-1.2), abs(3+4j))
```

1.2 5.0

Maksimal va minimal qiymatlarni topish uchun max(x) va min(x) funksiyalaridan foydalaniladi:

```
>>> a=[1,5,8,10]
>>> b=[4,8,3,2]
>>> print(max(a), min(b))
```

10 2

```
>>> print(max(8,5,6,-1))
```

8

pow(x, y) funksiyasi *x* ni *y* darajasiga ko‘taradi.

```
>>> print(pow(2,10))
```

1024

```
>>> print(2**10)
```

1024

Standart kutubxona modullari. Pythonning juda katta miqdordagi standart kutubxona modullari mavjud bo‘lib, biz ulardan bir nechtasini ko‘rib o‘tamiz. sys moduli dasturning ishlash muhiti, Python interpretatori haqidagi ma’lumotlarni o‘z ichiga olgan o‘zgaruvchilarga kirishni ta'minlaydi. Ixtiyoriy argumentga ega exit (arg) funksiyasi dastur bajarilishini tugatadi. Dasturni to‘xtatish va xato xabarini ko‘rsatish uchun exit('error') dan foydalanishingiz mumkin. platform instruksiyasi foydalanilayotgan platforma haqidagi ma’lumotlarni ko‘rsatadi. Interpretatorning standart kiritish, chiqish va xato oqimlariga mos keladigan fayl obyektlari sys modulining stdin, stdout, stderr obyektlari bilan bog‘langan.

Operatsion tizim bilan o‘zaro aloqa os moduli tomonidan qo‘llab-quvvatlanadi. Xususan, getcwd() funksiyasi joriy ishchi katalogni ifodalovchi qatorni qaytaradi. Joriy ishchi katalogni (path ga) o‘zgartirish uchun chdir(path) dan

foydalaning. path nomli faylni (katalogni) o‘chirish uchun siz remove(path) (removedirs(path)) funksiyasidan foydalanishingiz mumkin. rename (src, dst) funksiyasi fayl (katalog) src nomini dst ga o‘zgartirishga yordam beradi.

Dasturdagi to‘siqlarni aniqlash uchun dasturning alohida qismlarini bajarish uchun vaqt oralig‘ini o‘lchash kerak. Shu maqsadda siz vaqt bilan ishlash uchun time modulidan foydalanishingiz mumkin. Platformangizning boshlang‘ich nuqtasi (davr boshlanishi 1970 yil 1 yanvarda soat 00:00) gmtime(0) funksiyasi tomonidan chiqariladi:

```
>>> import time
>>> print(time.gmtime(0))
time.struct_time(tm_year=1970, tm_mon=1, tm_mday=1, tm_hour=0, tm_min=0, tm_sec=0, tm_wday=3, tm_yday=1, tm_isdst=0)
```

Amallarni bajarish vaqtini aniqlash uchun clock() funksiyasidan foydalanish mumkin, u joriy protsessor vaqtini soniyalarda haqiqiy son sifatida qaytaradi. Sleep(x) funksiyasi jarayonni x soniya to‘xtatib turadi. Python tilidagi dasturlarning ishlash jarayonini to‘liqroq tahlil qilish standart kutubxonaga kiritilgan profile va pstats maxsus modullari yordamida amalga oshiriladi.

Python obyektlarini saqlash va nusxalash uchun juda foydali funksiya Python obyektlarini baytlar ketma-ketligiga va aksincha (seriyalashtirish) o‘zgartiruvchi pickle moduli tomonidan amalga oshiriladi. Shu tarzda olingan baytlar ketma-ketligi tashqi xotirada saqlanishi yoki aloqa kanallari orqali uzatilishi mumkin. pickle moduli Pythonda qo‘llaniladi va sekinroq ishlaydi. Tezroq ishlash uchun siz C da amalga oshirilgan cPickle modulidan foydalanishingiz mumkin.

Ma’lumotlarni saqlash uchun mydata.dat fayliga ma’lumotlarni yozish uchun quyidagi dasturdan foydalanamiz:

```
import pickle
data=("abc",1.23, [1,2,3])
file=open("mydata.dat","wb")
pickle.dump(data,file)
file.close()
```

file argumenti write() usuliga ega bo‘lgan fayl obyekti bo‘lishi kerak. dump(data,file) ma’lumotlarni serializatsiyalash uchun ishlatiladi.

Ma’lumotlarni teskari chiqarish jarayoni quyidagi dastur tomonidan amalga oshiriladi:

```
import pickle
file = open("mydata.dat", "rb")
mydata = pickle.load(file)
file.close()
print(mydata)
```

Natija

```
('abc', 1.23, [1, 2, 3])
```

load(file) avval serializatsiyalangan obyektlarni tiklash va qaytarish uchun ishlatiladi.

Pythonning dastlabki versiyalarida satrlar bilan ishlash uchun string modulidan foydalanilgan. Hozirgi vaqtda ushbu modulning funksiyalari satrlarda usullar sifatida amalga oshirilmoqda.

Satrlarni ajratish va birlashtirishga misol:

```
import string
s = "one, two, three"
r = s.split(",")
print(r)
print("".join(r))
```

```
['one', ' two‘, ' three']
one two three
```

Boshqa foydali satrlar bilan ishlash usullari sifatida qidirish va almashtirish uchun find() va replace() ni qarash mumkin.

5.2-§. Matematik modullar

Haqiqiy sonlar bilan ishlashning matematik funksiyalari math moduliga kiritilgan. Kompleks sonlar bilan ishlash cmath modulidan foydalaniladi. Ushbu modullardagi ko‘pgina funksiyalar bir xil nomga ega, ammo natijalar butunlay farq

qilishi mumkin. Xususan, manfiy sonning kvadrat ildizini hisoblashda (math.sqrt(-1) va cmath.sqrt(-1) funksiyalari).

Matematik modulning funksiyalari 5.1-jadvalda keltirilgan. Agar argument z harfi bilan belgilansa, u holda funksiya math modulida ham, smath modulida ham aniqlanadi.

5.1-jadval. math va cmath modullaridagi funksiyalar

Funksiya	Tavsifi
acos(z)	z ning arkkosinusi
acosh(2)	z ning giperbolik arkkosinusi
asin(z)	z ning arksinusi
asinh(z)	z ning giperbolik arksinusi
atan(z)	z ning arktangensi
atan2(y,x)	atan(y/x)
atanh(z)	z ning giperbolik arktangensi
ceil(x)	x dan katta yoki teng eng kichik butun son
copysign(x, y)	y belgisi bilan x
cos(z)	z ning kosinusi
cosh(z)	z ning giperbolik kosinusi
degrees(x)	x burchakni radiandan gradusga aylantirish
exp(z)	eksponenta (e^z)
fabs(x)	x ning absolyut qiymati
factorial(x)	x factorial
floor(x)	x dan kichik yoki teng eng katta butun son
fmod(x,y)	x ni y ga bo‘lgandagi qoldiq
frexp(x)	x ning mantissa va ko‘rsatkichini juftlik (m,i) sifatida qaytaradi, bu yerda m

	haqiqiy va i butun son bo‘lib, x = $m2^i$ bo‘ladi.
hypot(x,y)	$\sqrt{x^2 + y^2}$
ldexp(m,i)	frexp(x) ga teskari funksiya ($m2^i$)
log(z)	z ning natural logarifmi
log10(z)	z ning o‘nlik logarifmi
modf(x)	x ning butun va kasr qismini (p, q) juftlik ko‘rinishida qaytaradi. Ikkala qismda asl sonning ishorasi mavjud
phase(z)	z ning murakkab qiymatli qutb burchagi
polar(z)	qutb koordinatalaridagi z kompleks soni (r, φ)
pow (x,y)	x^y
radians(x)	x burchakni gradusdan radianga aylantiring
rect(r, φ)	qutb koordinatalaridan kartezianga o‘tish
sin(z)	z ning sinusi
sinh(z)	z ning giperbolik sinusi
sqrt(z)	z ning kvadrat ildizi
tan(z)	z ning tangensi
tanh(z)	z ning giperbolik tangensi

math va cmath modullari ikkita haqiqiy doimiyni aniqlaydi: pi-π soni va e-e soni.

Shuni ham ta'kidlash kerakki, random moduli turli xil taqsimotlar uchun tasodifiy sonlarni yaratishga imkon beradi. Tasodifiy haqiqiy son r ($0.0 \le r < 1.0$) ni hosil qilish uchun random() funksiyasidan foydalaning. uniform(a, b) funksiyasi $a \le r < b$ oraliqdagi tasodifiy haqiqiy r sonni qaytaradi. randrange(start, stop, step) funksiyasi range(start, stop, step) tasodifiy butun sonni olish uchun ishlatiladi.

Oddiy taqsimlangan tasodifiy sonlar (Gauss taqsimoti) ketma-ketligidagi sonni yaratish gauss(mu, sigma) funksiyasi orqali amalga oshiriladi, bu yerda mu– matematik kutilma, sigma–esa standart og‘ishdir. Yana bir imkoniyat - bir xil parametrlar bilan normalvariate funksiyadan foydalanish. Tasodifiy sonlarning boshqa ketma-ketliklaridan sonlarni yaratish mumkin (beta taqsimoti, eksponensial taqsimot, gamma taqsimoti, logarifmik normal taqsimot va boshqalar).

5.3-§. NumPy paketi. Python hisoblash yadrosi

Interpretatsiyalanuvchi tillarda amalga oshiriladigan matematik algoritmlar, odatda, C kabi kompilyatsiya qilingan tillardan foydalanishga qaraganda ancha sekinroqdir. Pythonda ko‘p sonli hisoblash algoritmlari uchun past unumdorlik muammosi ko‘p o‘lchovli massivlarni qo‘llab-quvvatlashga qaratilgan maxsus kutubxonalar va ular bilan ishlash uchun ko‘plab funksiyalar va operatorlar yordamida hal qilinadi. Odatdagi holat - hisoblash algoritmi massivlar va matritsalar ustidagi amallar ketma-ketligiga asoslanadi va bu sharoitda Python dasturi C dasturi kabi tez ishlaydi.

Hisoblash paketining eng qiziqarli namunasi bu NumPy. NumPy ko‘plab raqamli ilovalar uchun zarur bo‘lgan ko‘p o‘lchovli massivlar bilan hisoblash uchun modullarni o‘z ichiga oladi. Shu sababli NumPy MATLAB va GNU Octave (MATLAB analogi) kabi tizimlar uchun xos xususiyatlarni taqdim etadi.

NumPy oddiy va ishlatish uchun qulay bo‘lib, ko‘p o‘lchovli massivlar bilan ishlash vositalariga qo‘shimcha ravishda quyidagilar uchun modullarni o‘z ichiga oladi:

- chiziqli algebra masalalarini yechish;
- Furye konvertatsiyasi;
- tasodifiy sonlar massivlarini yaratish.

Hisoblash matematikasi masalalarini yechish uchun boshqa vositalar NumPy paketi asosida qurilgan, ulardan ba’zilarini quyida ko‘rib chiqamiz. Ushbu holat NumPy to‘plamini Python standart kutubxonasiga kiritilmagan bo‘lsada, asosiy matematik paket sifatida tasniflash imkonini beradi.

5.4-§. NumPy da massivlar

NumPy paketi yangi ma’lumotlar turini, N o‘lchovli massivni (ndarray) aniqlaydi.

Massiv - bir yoki bir nechta indekslar bilan identifikatsiya qilingan bir xil turdagi ma’lumotlarni saqlash uchun ma’lumotlarning tartiblangan to‘plami. NumPy paketida massiv odatda o‘zgarmas uzunlikda bo‘ladi va bir xil turdagi dtype ma’lumotlar birliklarini saqlaydi (Data type objects). Amaldagi massiv indekslari soni har xil bo‘lishi mumkin. Bitta indeksli massivlar bir o‘lchovli (vektorlar), ikki indeksli massivlar ikki o‘lchovli (matritsalar) deb ataladi.

Har bir o‘q bo‘ylab massivning o‘lchamlari soni va uzunligi massivning shakli deb ataladi (the shape of the array). NumPy da massivning shakli N ta natural sondan iborat kortej sifatida tasvirlangan, uning uzunligi (N) massivning o‘lchami, kortej elementlari esa massivning mos o‘q bo‘ylab uzunligidir.

2x3 butun sonlardan iborat ikki o‘lchovli massivga misol:

```
import numpy as np
a = np.array([[1, 2, 3], [4, 5, 6]])
print(type(a))
print(a.shape)
print(a.dtype)
```

Natija:

```
<class 'numpy.ndarray'>
(2, 3)
int32
```

Massiv elementlari sifatida barcha sonli tiplardan foydalanish mumkin

5.2-jadval. Massiv elementlarining turlari

Ma’lumotlar toifasi	Tavsifi
bool	mantiqiy (True yoki False)

int	butun son (odatda int32 yoki int64)
int8	bayt (-128..127)
int16	butun son (-32768..32767)
int32	butun son (-2147483648..2147483647)
int64	butun son (-9223372036854775808 .. 9223372036854775807)
uint8	ishorasiz butun son (0..255)
uint16	ishorasiz butun son (0..65535)
uint32	ishorasiz butun son (0..4294967295)
uint64	ishorasiz butun son (0..18446744073709551615)
float	float64 ning qisqa shakli
float32	birinchi tartibli aniqlikdagi haqiqiy son
float64	ikkinchi tartibli aniqlikdagi haqiqiy son
complex64	kompleks son (haqiqiy va mavhum qismlar uchun float32)
complexl28	kompleks son (haqiqiy va mavhum qismlar uchun float64)

Massivni yaratishning eng oddiy usuli array() funksiyasidan foydalanish bo‘lib, u massiv elementlarini turi bo‘yicha aniq yoki bilvosita ichki joylashtirilgan ro‘yxatlar sifatida belgilaydi.

```
import numpy as np
a=np.array([[1, 2, 3], [4, 5, 6]], 'float')
print(a)
print(a.shape)
print(a.dtype)
[[1. 2. 3.]
 [4. 5. 6.]]
```

(2, 3)

float64

Ro‘yxatdan massiv yaratish va teskari protsedura (tolist() usuli yordamida) quyidagi misolda tasvirlangan:

```
import numpy as np
lst = [[1, 2, 3], [4, 5, 6]]
a = np.array(lst)
print(a)
print(a.tolist())
[[1 2 3]
 [4 5 6]]
[[1, 2, 3], [4, 5, 6]]
```

Matritsalar bilan ishlashda birlik, nol matritsalar kabi maxsus matritsalar bilan ishlashga to‘g‘ri keladi. Shuningdek barcha elementlari 1 dan iborat bo‘lgan matritsalar ham mavjud. NumPy da bunday massivlarni ishga tushirish vositalari mavjud.

zeros(sh, dt) funksiyasi sh o‘lchamdagi nol elementlari va ixtiyoriy dt argumenti – element turi bilan massiv yaratish imkonini beradi. Xuddi shunday, ones(sh, dt) birlik elementlardan iborat massiv hosil qiladi. Asosiy diagonalda birlar va boshqa elementlar uchun nol bo‘lgan matritsani hosil qilish uchun eye(n,m) funksiyasidan, kvadrat matritsa uchun eye(n) yoki identity(n) dan foydalaniladi. Massiv yaratishga misollar:

```
import numpy as np
a0 = np.zeros((2, 3), 'float')
print('a0:\n', a0)
a1 = np.ones((3, 2), 'int')
print('a1:\n', a1)
a2 = np.eye(3, 2, dtype=int)
print('a2:\n', a2)
a3 = np.identity(4, 'float')
```

```
print('a3:\n', a3)
```

Natija:

```
a0:
 [[0. 0. 0.]
 [0. 0. 0.]]
a1:
 [[1 1]
 [1 1]
 [1 1]]
a2:
 [[1 0]
 [0 1]
 [0 0]]
a3:
 [[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

Bir o‘lchovli massivlarni (vektorlarni) shakllantirishning maxsus protseduralari mavjud. arange(start, stop, step) funksiyasi step qadam bilan start qiymatidan stop qiymatigacha (stop bu oraliqqa kirmaydi) qiymatlar qatorini hosil qiladi. Ushbu konstruktsiya array(range (start, stop, step)) ga o‘xshaydi, lekin u nafaqat butun sonlar bilan ishlaydi. linspace(start, stop, n) funksiyasi qiymatlari start dan stop gacha (shu jumladan stop ham) teng masofada joylashgan n ta elementdan iborat bir o‘lchovli massivni yaratadi. linspace(start, stop, n) funksiyasi n ta elementdan iborat bir o‘lchovli massivni yaratadi, ularning qiymatlari start dan stop gacha (shu jumladan stop ham) teng taqsimlanadi.

```
import numpy as np
a0 = np.array(range(1,10,3))
print('a0:\n', a0)
```

```
a1 = np.arange(0.,10.,2.)
print('a1:\n', a1)
a2 = np.linspace(0.,1.,11)
print('a2:\n', a2)
```

Natija:

```
a0:
 [1 4 7]
a1:
 [0. 2. 4. 6. 8.]
a2:
 [0.  0.1 0.2 0.3 0.4 0.5 0.6 0.7 0.8 0.9 1. ]
```

5.5-§. Massivlar ustida bajariladigan amallar

Massivlarni nusxalash uchun copy() usulidan, massiv shaklini o‘zgartirish uchun shape() usulidan foydalanishingiz mumkin. Xuddi shunday natijaga reshape() ni chaqirish orqali erishiladi. resize() funksiyasi reshape() funksiyasiga o‘xshaydi, lekin u asl va olingan massivdagi elementlar soni mos kelmasa ham ishlaydi.

```
import numpy as np
a0 = np.linspace(1.,10.,10)
print('a0:\n', a0)
a2 = np.copy(a0)
a1 = np.reshape(a0, (2, 5))
print('a1:\n', a1)
a2.shape = (-1,5)
print('a2:\n', a2)
a3 = np.resize(a2, (3, 4))
print('a3:\n', a3)
```

Natija:

```
a0:
 [ 1.  2.  3.  4.  5.  6.  7.  8.  9. 10.]
a1:
 [[ 1.  2.  3.  4.  5.]
```

```
 [ 6.  7.  8.  9. 10.]]
a2:
 [[ 1.  2.  3.  4.  5.]
 [ 6.  7.  8.  9. 10.]]
a3:
 [[ 1.  2.  3.  4.]
 [ 5.  6.  7.  8.]
 [ 9. 10.  1.  2.]]
```

Bu yerda -1 massiv shaklini tanlashda (bizning misolimizda (2,-1)) massivning bir o‘qi bo‘ylab elementlar sonini bilvosita belgilash uchun ishlatiladi - u elementlarning umumiy soni bilan hisoblanadi.

Massiv elementini o‘zgartirish uchun indeks bo‘yicha kirish, massivning bir qismi uchun esa bo‘lakdan foydalaniladi. Ba’zi asosiy xususiyatlar quyidagi misolda tasvirlangan.

```
import numpy as np
a = np.reshape(np.linspace(1,12,12),(3,4))
print('a:\n', a)
print('element(2,3): ', a[2,3])
print('row1: ', a[1])
print('last row: ', a[-1,:])
print('column 1: ', a[:,1])
print('window 2x2: \n', a[1:3,0:2])
```

Natija:

```
a:
 [[ 1.  2.  3.  4.]
 [ 5.  6.  7.  8.]
 [ 9. 10. 11. 12.]]
element(2,3):  12.0
row1:  [5. 6. 7. 8.]
```

```
last row:  [ 9. 10. 11. 12.]
column 1:  [ 2.  6. 10.]
window 2x2:
 [[ 5.  6.]
 [ 9. 10.]]
```

Massiv elementlarini faylga yozish (binar, NumPy formati, .npy kengaytmasi) save() funksiyasi, fayldan o‘qish esa load() funksiyasi yordamida amalga oshiriladi.

```
import numpy as np
a = np.reshape(np.linspace(1.,8.,8),(2,4))
print('a:\n', a)
np.save('D:/data.npy', a)
a1 = np.load('D:/data.npy')
print('a1:\n', a1)
```

Natija:

```
a:
 [[1. 2. 3. 4.]
 [5. 6. 7. 8.]]
a1:
 [[1. 2. 3. 4.]
 [5. 6. 7. 8.]]
```

Massivlarni matnli faylga yozish va o‘qishda tofile() va fromfile() funksiyalaridan foydalaniladi.

5.6-§. Massivlar ustida bajariladigan matematik funksiyalar

NumPy massiv elementlarida standart arifmetik amallarni amalga oshiradi.

5.3-jadval. Massivlar ustidagi arifmetik amallar

Funksiya	tavsifi
add(x, y)	elementlar bo‘yicha qo‘shish
subtract(x, y)	elementlar bo‘yicha ayirish
multiply(x, y)	elementlar bo‘yicha ko‘paytirish
divide(x, y)	elementlar bo‘yicha bo‘lish
power(x,y)	elementlar bo‘yicha darajaga ko‘tarish
negative(x)	massiv elementlarining ishorasini o‘zgartiradi

Massivlarda elementar arifmetik amallardan foydalanishga misollar:

```
import numpy as np
a0 = np.linspace(1., 10., 10).reshape(2, 5)
print('a0:\n', a0)
a1 = np.ones(10).reshape(2, 5)
a1 = np.add(a1, 1.)
print('a1:\n', a1)
a2 = np.add(a0, a1)
print('a2:\n', a2)
a3 = np.multiply(a0, a1)
print('a3:\n', a3)
a4=np.power(a0, a1)
print('a4:\n',a4)
```

Natija:

```
a0:
 [[ 1.  2.  3.  4.  5.]
 [ 6.  7.  8.  9. 10.]]
a1:
 [[2. 2. 2. 2. 2.]
 [2. 2. 2. 2. 2.]]
```

```
a2:
 [[ 3.  4.  5.  6.  7.]
 [ 8.  9. 10. 11. 12.]]
a3:
 [[ 2.  4.  6.  8. 10.]
 [12. 14. 16. 18. 20.]]
a4:
 [[  1.   4.   9.  16.  25.]
 [ 36.  49.  64.  81. 100.]]
```

Massivlarda element bo‘yicha bajariladigan boshqa funksiyalar:

Trigonometrik: sin(x), cos(x), tan(x), arcsin(x), arccos(x), arctan(x), hypot(x, y), arctan2(x,y), degrees(x), radians(x);

Giperbolik: sinh(x), cosh(x), tanh(x), arcsinh(x), arccosh(x), arctanh(x);

Eksponensial va algoritmik: exp(x), expm1(x) (=exp(x)-1), log(x), logl0(x), log2(x), log1p(x)(=log(1+x)).

Boshqa funksiyalar Python standart kutubxonasining math modulidagi matematik funksiyalar yozuviga mos keladi (3.1-jadvalga qarang).

```
import numpy as np
a0 = np.linspace(0., np.pi, 9)
print('a0:\n', a0)
a1 = np.sin(a0)
print('a1:\n', a1)
a2 = np.cos(a0)
print('a2:\n', a2)
a3 = np.multiply(a1, a1) + np.multiply(a2, a2) - 1.
print('a3:\n', a3)
```

Natija:

```
a0:
 [0.         0.39269908 0.78539816 1.17809725 1.57079633 1.96349541
```

```
 2.35619449  2.74889357  3.14159265]
a1:
 [0.00000000e+00 3.82683432e-01 7.07106781e-01 9.23879533e-01
 1.00000000e+00 9.23879533e-01 7.07106781e-01 3.82683432e-01
 1.22464680e-16]
a2:
 [ 1.00000000e+00  9.23879533e-01  7.07106781e-01  3.82683432e-01
  6.12323400e-17 -3.82683432e-01 -7.07106781e-01 -9.23879533e-01
 -1.00000000e+00]
a3:
 [0.00000000e+00 0.00000000e+00 2.22044605e-16 0.00000000e+00
 0.00000000e+00 0.00000000e+00 0.00000000e+00 0.00000000e+00
 0.00000000e+00]
```

Elementlari kompleks sonlardan iborat bo‘lgan massivlar bilan ishlash uchun quyidagi funksiyalar mavjud: angle(x) (phase(x) funksiyasiga o‘xshash), rcal(x), imag(x) va conj(.x), bu kompleks-bog‘langan elementli massivni qaytaradi.

```
import numpy as np
a0 = np.linspace(1., 10., 10).reshape(2, 5)
print('a0:\n', a0)
a1 = np.add(a0, 1.j)
print('a1:\n', a1)
a2 = np.conj(a1)
print('a2:\n', a2)
a3 = np.real(a1)
print('a3:\n', a3)
a4 = np.imag(a1)
print('a4:\n', a4)
```

Natija:

```
a0:
```

```
[[ 1.  2.  3.  4.  5.]
 [ 6.  7.  8.  9. 10.]]
a1:
[[ 1.+1.j  2.+1.j  3.+1.j  4.+1.j  5.+1.j]
 [ 6.+1.j  7.+1.j  8.+1.j  9.+1.j 10.+1.j]]
a2:
[[ 1.-1.j  2.-1.j  3.-1.j  4.-1.j  5.-1.j]
 [ 6.-1.j  7.-1.j  8.-1.j  9.-1.j 10.-1.j]]
a3:
[[ 1.  2.  3.  4.  5.]
 [ 6.  7.  8.  9. 10.]]
a4:
[[1. 1. 1. 1. 1.]
 [1. 1. 1. 1. 1.]]
```

5.7-§. Matritsali obyektlar

Hisoblash amaliyotida N o‘lchamli massivlar o‘rniga ko‘pincha vektorlar va matritsalar (bir o‘lchovli va ikki o‘lchovli massivlar) bilan ishlash qulayroqdir. Xususan, dasturlashda massivlar bilan elementli amallar o‘rniga matritsali amallardan foydalanish mumkin. Matritsa turlari bilan ishlash MATLAB ilmiy hisoblash tizimining hisoblash asosi hisoblanadi.

NumPy to‘plamida N o‘lchovli massivning umumiy turiga (ndarray) qo‘shimcha ravishda vektor, matritsa va tenzor ob’ektlarining kichik sinflari mavjud. Bizni NumPy matritsa ob’ektlari (matritsalari) chiziqli algebra masalalarini yechishda foydalanish nuqtai nazaridan qiziqtiradi. Bu yerda ndarray bilan solishtirganda matritsadan foydalanishning o‘ziga xos xususiyatlarini qayd etamiz. Ko‘rinib turibdiki, bir tomondan, ma’lumotlar umumiy turi sifatida massivlar bilan ishlash ko‘proq imkoniyatlarni beradi. Boshqa tomondan, matritsali masalalar ko‘pincha ilovalar uchun o‘zini-o‘zi ta'minlaydi va umumiy massivlar bilan ishlash uchun keng imkoniyatlarga ehtiyoj yo‘q.

Matritsa ob’ektlari uchun satrni indekslash mumkin, matritsa elementlari MATLAB sintaksisiga ko‘ra satr va ustun uchun ajratuvchi sifatida bo‘sh joy bilan belgilanadi.

```
import numpy as np
a = np.mat('1 2 3; 4 5 6')
b = np.mat([[1, 2, 3], [4, 5, 6]])
print(type(a), 'a:\n', a)
print(type(b), 'b:\n', b)
```

Natija:

```
<class 'numpy.matrix'> a:
 [[1 2 3]
 [4 5 6]]
<class 'numpy.matrix'> b:
 [[1 2 3]
 [4 5 6]]
```

Chiziqli algebraning asosiy amallari matritsalarni ko‘paytirishdir (maxsus holatda matritsalarni vektor bo‘yicha va vektorni vektor bo‘yicha). NumPy da ndarray va matritsa ob’ektlaridan foydalanilganda, matritsalarni ko‘paytirish sintaksisi boshqacha. ndarray da * belgisi element bo‘yicha ko‘paytirishga to‘g‘ri keladi, matritsani ko‘paytirish uchun dot() funksiyasi qo‘llaniladi. matrix da * belgisi matritsalarni ko‘paytirish uchun, multiply() funksiyasi esa elementi bo‘yicha ko‘paytirish uchun ishlatiladi.

```
import numpy as np
a =   np.array([[1,2],     [3,     4]])
b =   np.array([[5,6],     [7,     8]])
print('ndarray - a*b:\n', a*b)
print('ndarray - dot(a,b):\n', np.dot(a, b))
c =   np.mat(a)
d =   np.mat(b)
```

```
print('matrix - c*d:\n', c*d)
print('matrix - multiply(c,d):\n', np.multiply(c, d))
```

Natija:

```
ndarray - a*b:
 [[ 5 12]
 [21 32]]
ndarray - dot(a,b):
 [[19 22]
 [43 50]]
matrix - c*d:
 [[19 22]
 [43 50]]
matrix - multiply(c,d):
 [[ 5 12]
 [21 32]]
```

Bu misol ndarray obyektlarini matritsaga aylantirish uchun mat() funksiyasidan foydalanadi. U massivlarni aniqlash uchun zeros(), one(), eye(), linspace() kabi funksiyalar yordamida matritsalarni aniqlashda foydalanish mumkin.

Massivlar bilan ishlashda transpozitsiyalangan matritsani olish uchun (satrlarni berilgan matritsa ustunlari bilan almashtirish) .T atributidan foydalanish qulay.

Matritsa ob’ektlari uchun quyidagi atributlar ham qo‘llaniladi:

• .H – kompleks bog‘langan transpozitsiya;

• .I – teskari matritsa;

• .A – matritsani ikki o‘lchovli massivga aylantirish,

```
import numpy as np
a = np.mat(np.linspace(1, 4, 4).reshape(2,2))
print('linspace - reshape -> a:\n', a)
print('a: ', type(a))
```

```
b = a.I
print('a.T:\n', a.T)
print('a.I:\n', b)
print('a*a.I:\n', a*b)
print('a.A: ', type(a.A))
```

Natija:

```
linspace - reshape -> a:
 [[1. 2.]
 [3. 4.]]
a:  <class 'numpy.matrix'>
a.T:
 [[1. 3.]
 [2. 4.]]
a.I:
 [[-2.   1. ]
 [ 1.5 -0.5]]
a*a.I:
 [[1.0000000e+00 0.0000000e+00]
 [8.8817842e-16 1.0000000e+00]]
a.A:  <class 'numpy.ndarray'>
```

5.8-§. NumPy paketi chiziqli algebra elementlari

NumPy paketi shuningdek chiziqli algebraning asosiy operatsiyalarini (linalg moduli) amalga oshiradi. Keling, avvalo xarakteristik matritsani hisoblash uchun funksiyalarni eslatib o‘tamiz.

Vektor normasini (masalan, Evklid) va kvadrat matritsa normasini (masalan, Frobenius) hisoblash uchun norm() funksiyasidan foydalaniladi. Kvadrat matritsaning shartli raqamini hisoblash uchun cond(), determinant det(), matritsaning izi uchun NumPy paketidan trace() funksiyalaridan foydalaniladi.

```
import numpy as np
```

```
a = np.mat( [[1, 2], [3, 4]])
print('a:\n', a)
print('norm a: ', np.linalg.norm(a))
print('cond a: ', np.linalg.cond(a))
print('det a: ', np.linalg.det(a))
print('trace a: ', np.trace(a))
```

Natija:

```
a:
 [[1 2]
 [3 4]]
norm a:  5.477225575051661
cond a:  14.933034373659268
det a:  -2.0000000000000004
trace a:  5
```

Chiziqli algebra moduli matritsani ko‘phadlarga ajratish funksiyalariga ega. Simmetrik musbat aniq A matritsa uchun Choleskiy bo‘linishi A = LL* bo‘lganda sodir bo‘ladi, bu yerda L pastki uchburchak matritsa (cholesky() funksiyasi). Umumiy matritsalar uchun QR-ajralish A = QR bo‘lganda qo‘llaniladi, bu yerda Q ortogonal matritsa va R yuqori uchburchak matritsa (qr() funksiyasi). Matritsaning singulyar qiymatining bo‘linishi svd() funksiyasi tomonidan amalga oshiriladi.

```
import numpy as np
a = np.mat([[2, 1], [3, 4]])
b = (a + a.T) / 2
print('a:\n', a)
print('a:\n', b)
print('lower-triangular (Cholesky) matrix:\n', np.linalg.cholesky(b))
q,r = np.linalg.qr(a)
print('orthonormal matrix q:\n', q)
print('upper-triangular matrix r:\n', r)
```

```
print('q*r:\n', q*r)
Natija:
a:
 [[2 1]
 [3 4]]
a:
 [[2. 2.]
 [2. 4.]]
lower-triangular (Cholesky) matrix:
 [[1.41421356 0.        ]
 [1.41421356 1.41421356]]
orthonormal matrix q:
 [[-0.5547002  -0.83205029]
 [-0.83205029  0.5547002 ]]
upper-triangular matrix r:
 [[-3.60555128 -3.88290137]
 [ 0.          1.38675049]]
q*r:
 [[2. 1.]
 [3. 4.]]
```

Chiziqli algebrada chiziqli algebraik tenglamalar sistemasini yechishga alohida e’tibor beriladi. Teskari matritsani hisoblash uchun inv() funksiyasidan foydalaniladi. Pseudo-teskari matritsalar pinv() yordamida hisoblanadi. solve() funksiyasi tenglamalar sistemasini yechish uchun mo‘ljallangan.

```
import numpy as np
a = np.mat([[4, 1], [2, 3]])
print('a:\n', a)
b = np.linalg.inv(a)
print('b = a^(-1):\n', b)
```

```
print('a*b:\n', a*b)
c = np.mat( [1, 1]).T
print('c:\n', c)
x = np.linalg.solve(a, c)
print('solution x:\n', x)
print('a*x - c:\n', a*x-c)
Natija:
a:
 [[4 1]
 [2 3]]
b = a^(-1):
 [[ 0.3 -0.1]
 [-0.2  0.4]]
a*b:
 [[ 1.00000000e+00  0.00000000e+00]
 [-1.11022302e-16  1.00000000e+00]]
c:
 [[1]
 [1]]
solution x:
 [[0.2]
 [0.2]]
a*x - c:
 [[0.]
 [0.]]
```

nxm ($n \geq m$) o‘lchamdagi A matritsa va o‘ng tomoni b bo‘lgan eng kichik kvadratlar usulida r=b-Ax qoldiq normasini minimallashtiruvchi x vektor (psevdo-yechim) mavjud. NumPy eng kichik kvadratlar usulini amalga oshirish uchun lstsq() funksiyasidan foydalanadi.

```
import numpy as np
a = np.mat([[1, 2], [4, 3], [5, 6]])
print('a:\n', a)
b = np.mat ([7, 8, 9]). T
print('b:\n', b)
x, resids, rank, s = np.linalg.lstsq(a, b)
print('solution x:\n', x)
print('resids r:\n', resids)
y = np.linalg.pinv(a)*b
print('function pinv -> y:\n', y)
```

Natija:

```
a:
 [[1 2]
 [4 3]
 [5 6]]
b:
 [[7]
 [8]
 [9]]
solution x:
 [[0.55737705]
 [1.37704918]]
resids r:
 [[20.49180328]]
function pinv -> y:
 [[0.55737705]
 [1.37704918]]
```

Yuqoridagi misoldan ko‘rinib turibdiki, biz pinv() funksiyasi yordamida bir xil natijaga erishishimiz mumkin.

Endi chiziqli algebraning spektral masalalarini yechish uchun NumPy paketining imkoniyatlarini qayd etamiz. Xos qiymatni topish uchun umumiy matritsalar uchun eigvals() funksiyasidan, Ermit yoki haqiqiy simmetrik matritsalar uchun eigvalsh() funksiyasidan foydalaniladi. Xos qiymat va xos vektorlar mos ravishda eig() va eigh() funksiyasida hisoblanadi.

```
import numpy as np
a = np.mat([[1.5, 2.], [3., 4.]])
print('a:\n', a)
b = (a + a.T) / 2
c = (a - a.T) / 2
print('b:\n', b)
print('c:\n', c)
lb = np.linalg.eigvalsh(b)
print('eigenvalues of b:\n', lb)
lc = np.linalg.eigvals(c)
print('eigenvalues of c:\n', lc)
la, v = np.linalg.eig(a)
print('eigenvalues of a:\n', la)
print('eigenvectors of a:\n', v)
```

Natija:

```
a:
 [[1.5 2. ]
 [3.  4. ]]
b:
 [[1.5 2.5]
 [2.5 4. ]]
c:
 [[ 0.  -0.5]
 [ 0.5  0. ]]
eigenvalues of b:
```

```
 [-0.04508497  5.54508497]
eigenvalues of c:
 [0.+0.5j 0.-0.5j]
eigenvalues of a:
 [0.  5.5]
eigenvectors of a:
 [[-0.8        -0.4472136 ]
 [ 0.6        -0.89442719]]
```

Simmetrik haqiqiy matritsaning xos qiymatlari (bizning holatlarimizda b matritsasi) haqiqiy, kosasimmetrik matritsaning (c matritsasi) esa faqat xayoliydir.

5.9-§. Ko‘phadlar bilan ishlash

NumPy paketi ko‘phadlar ustida asosiy amallarni qo‘llab-quvvatlaydi. Ko‘phadlar koeffitsientlar ro‘yxati sifatida ko‘rsatilgan, masalan, [1,2,3] hadlar $x^2 + 2x + 3$ ko‘phadga to‘g‘ri keladi. Ko‘phadni ko‘rsatish uchun poly1d() funksiyasidan foydalaniladi, p ko‘phadning qiymati x nuqtada p(x) yoki polyval(p, x) funksiyasidan foydalanib hisoblanadi.

```
import numpy as np
p = np.poly1d([1, 2, 3])
print(np.poly1d(p))
print(type(p))
print(p(0.5), np.polyval(p, 0.5))
```

Natija:

```
   2
1 x + 2 x + 3
<class 'numpy.poly1d'>
4.25 4.25
```

Ko‘phadlar ustidagi arifmetik amallar quyidagi funksiyalar bilan bajariladi: polyadd() - qo‘shish, polymul() - ko‘paytirish va polydiv() - bo‘lish.

```
import numpy as np
p = np.poly1d([1, 2, 3, 4, 5])
q = np.poly1d([6, 7, 8])
print('p:\n', p)
print('q:\n', q)
print('p+q:\n', np.polyadd(p, q))
print('p*q:\n', np.polymul(p, q))
d, r = np.polydiv(p, q)
print('bo‘lish:\n', d)
print('qoldiq:', r)
```

Natija:

```
p:
   4     3     2
1 x + 2 x + 3 x + 4 x + 5
q:
   2
6 x + 7 x + 8
p+q:
   4     3     2
1 x + 2 x + 9 x + 11 x + 13
p*q:
   6      5      4      3      2
6 x + 19 x + 40 x + 61 x + 82 x + 67 x + 40
bo‘lish:
        2
0.1667 x + 0.1389 x + 0.1157
qoldiq:
2.079 x + 4.074
```

Koʻphadning ildizlarini hisoblash uchun roots(p) funksiyasi ishlatiladi, p.r qisqaroq shakli. Koʻphadlarni differensiallash (polyder()) va integrallash (polyint()) operatsiyalari ham qoʻllab-quvvatlanadi.

```
import numpy as np
p = np.poly1d([1, 2, 3, 4])
print('p:\n', p)
r = np.roots(p)
print('ildiz:\n', np.roots(p))
print('koʻphad:\n', np.poly(r))
print('ikkinchi hosila:', np.polyder(p, 2))
```

Natija:

```
p:
   3     2
1 x + 2 x + 3 x + 4
ildiz:
 [-1.65062919+0.j         -0.1746854 +1.54686889j -0.1746854 -1.54686889j]
koʻphad:
 [1. 2. 3. 4.]
ikkinchi hosila:
6 x + 4
```

Bu yerda poly() funksiyasi koʻphadni berilgan ildiz boʻyicha hisoblaydi.

5.10-§. NumPy paketining boshqa xususiyatlari

NumPy paketi asosan turli xil sohadagi massivlar bilan ishlashga qaratilgan kuchli vositalarni oʻz ichiga oladi. Xususan, saralash va qidirish usullarini qayd etish mumkin. Ilmiy hisoblashlarga e’tibor qaratilsa, Furyening tez almashtirish usullarini alohida ta'kidlash kerak. Quyida SciPy paketini koʻrib chiqayotganda ular haqida batafsilroq gaplashamiz.

Pythonda random tasodifiy sonlarni yaratish moduli Python standart kutubxonasining bir qismidir. Katta tasodifiy namunalarni tayyorlash uchun NumPy

paketidagi random modulidan foydalaniladi. Ushbu modul turli xil taqsimot va xususiyatlarning tasodifiy sonlari massivlarini yaratish funksiyalarini o‘z ichiga oladi.

rand() funksiyasi (0,1) oraliqda teng taqsimlangan tasodifiy sonlar massivini yaratish uchun ishlatiladi. randint() funksiyasi berilgan oraliqda teng taqsimlangan raqamlarning berilgan shaklidagi massivni qaytaradi.

```
import numpy as np
a = np.random.rand(2, 3)
print('tasodifiy bir xil taqsimot qiymatlari a:\n', a)
b = np.random.randint(0, 100, (2, 5))
print('tasodifiy butun sonlar b:\n', b)
```

Natija:

```
tasodifiy bir xil taqsimot qiymatlari a:
 [[0.59204603 0.52503205 0.27468164]
 [0.50291574 0.01886887 0.44296607]]
tasodifiy butun sonlar b:
 [[91 67 54 74 67]
 [14 61 63 64 53]]
```

Muayyan qonunlarga bo‘ysunadigan tasodifiy sonlar massivlarini hosil qilish funksiyalari ham mavjud. Misol uchun, normal taqsimlangan sonlarni (Gauss taqsimoti) olish uchun normal() funksiyasi qo‘llaniladi, uning birinchi argumenti berilgan o‘rtacha (matematik kutish), ikkinchisi standart og‘ishdir.

```
import numpy as np
a = np.random.normal(0, 1, (2, 4))
print('tasodifiy normal taqsimot qiymatlari a:\n', a)
```

Statistik ma’lumotlarni qayta ishlash uchun NumPy paketining ba’zi xususiyatlarini ta'kidlaymiz. Butun massivning yoki uning alohida qismining

minimal va maksimal qiymatlarini (amax(), amin()), oʻrtacha qiymatlarni (mean(), median()) va standart ogʻish (std()) hisoblash uchun funksiyalar mavjud.

```
import numpy as np
a = np.array([[8, 1, 7], [3, 9, 2]])
print('a massiv:\n', a)
print('a ning eng katta elementi:', np.amax(a))
print('a ning oʻrta arifmetik qiymati:', np.mean(a))
print('standart ogʻish:', np.std(a))
print('1 satr boʻylab oʻrtacha:', np.mean(a, 1))
```

Natija:

```
a massiv:
 [[8 1 7]
 [3 9 2]]
a ning eng katta elementi: 9
a ning oʻrta arifmetik qiymati: 5.0
standart ogʻish: 3.1091263510296048
1 satr boʻylab oʻrtacha: [5.33333333 4.66666667]
```

Tasodifiy oʻzgaruvchilarning berilgan namunalari uchun korrelyatsiya va gistogrammalarni hisoblash ham mumkin.

VI BOB. MATPLOTLIB PAKETI

Ilmiy hisoblashlarda olingan natijalarni jozibali namoyish etish grafiklar orqali amalga oshiriladi. Ushbu bobda Matplotlib paketi va uning yordamida grafiklarni yaratish qaraladi. Funksiya grafigini chizish qoidalari, chiziq rangi va turlari bilan ishlash, tasvir o‘lchamlari bilan ishlash ushbu bobda bayon etilgan. Shuningdek bir o‘lchamli, ikki o‘lchamli hamda uch o‘lchamli grafika bilan ishlash qoidalari muhokama qilinadi.

6.1-§. Ilmiy grafika

Hisoblashlar natijalarini tahlil qilishda turli grafiklarni qurishga asoslangan ilmiy hisoblash natijalarini vizuallashtirishga katta e’tibor beriladi. Ko‘pgina Python kengaytmalari foydalanuvchilarga ko‘p turdagi kuchli va moslashuvchan chizma vositalarini taqdim etadi. Hozirgi kunda keng tarqalgan postprotsessor dasturiy vositalari foydalanuvchiga ma’lumotlar fayllari bilan ishlash, natijalarni grafik ko‘rinishda taqdim etish, chizmani tanlangan standart grafik formatda saqlash va grafikni chop etish imkonini beradi.

Eng muhim vazifalardan bir o‘lchovli yoki ikki o‘lchovli ma’lumotlarni vizualizatsiya qilishdir. Bir o‘lchovli (1D) ma’lumotlar (grafiklar) bilan bir o‘zgaruvchili funksiyani, ikki o‘lchovli (2D) ma’lumotlar bilan - ikki o‘zgaruvchili funksiya bog‘lanadi. Birinchi navbatda yuqori sifatli bir o‘lchovli va ko‘p o‘lchovli grafiklarni yaratishga qaratilgan ba’zi Python kengaytmalariga e’tibor qaratiladi.

Pythonda hisoblash ishlarini vizualizatsiya qilish uchun asosiy dasturiy mahsulot Matplotlib to‘plamidir. U hisoblashlarning asosiy ehtiyojlarini qondiradi, yaxshi ishlab chiqilgan va hujjatlashtirilgan, faol ishlab chiqilgan hamda jamoatchilik tomonidan keng qo‘llab-quvvatlanadi. Ushbu fikrlarni hisobga olgan holda, Matplotlib paketini matematik Pythonning grafik yadrosi sifatida ko‘rib chiqiladi. Paketning asosiy xususiyatlari quyida batafsil ko‘rib chiqiladi.

Ilmiy hisoblashda grafiklarni tayyorlashning an'anaviy vositalaridan Gnuplot paketini ta'kidlash kerak. Gnuplotning o‘ziga xos buyruq tizimi mavjud, u buyruqlar qatori rejimida va fayldan skriptlarni talqin qilish rejimida (buyruq rejimi) interaktiv ishlash mumkin. Gnuplot grafiklarni bevosita ekranga va png, eps, svg,

jpeg kabi asosiy grafik formatlardagi fayllarga o‘tkazadi. LaTeX nashriyot tizimi uchun eksportni qo‘llab-quvvatlanadi. Pythonda Gnuplot bilan ishlash Gnuplot.py paketi tomonidan qo‘llab-quvvatlanadi. Bu xotiradagi ma’lumotlar massivlari asosida grafiklar chizish, fayllardan ma’lumotlarni o‘qish va matematik funksiyalar grafiklari uchun Python dasturlarida Gnuplotdan foydalanish imkonini beradi.

DISLIN ilmiy grafika kutubxonasi notijorat maqsadlarda foydalanish uchun bepul. U egri chiziqlar, gistogrammalar, diagrammalar, konturlar va sirtlar ko‘rinishida hisoblangan ma’lumotlarni tasvirlash uchun mo‘ljallangan. Quyidagi grafik formatlarni qo‘llab-quvvatlaydi: PostScript, pdf, svg, png, bmp, gif, tiff va boshqalar. DISLIN Fortran 77, Fortran 90/95, C, Perl, Python va Java dasturlash tillari hamda ko‘plab operatsion tizimlarda ishlaydi.

MathGL moslashuvchan platformali ilmiy grafik kutubxonasi katta ma’lumotlar to‘plamlarini tezkor qayta ishlash va tasvirlashni ta'minlaydi. Bir, ikki va uch o‘lchovli massivlar uchun grafiklarning asosiy turlarini raster (png, jpeg, tiff yoki bmp) va vektor (eps va svg) formatlarga eksport qilishni qo‘llab-quvvatlaydi. MathGL kutubxonasidan C++, C, Fortran, Python va boshqa tillarda yozilgan dasturlarda foydalanish mumkin.

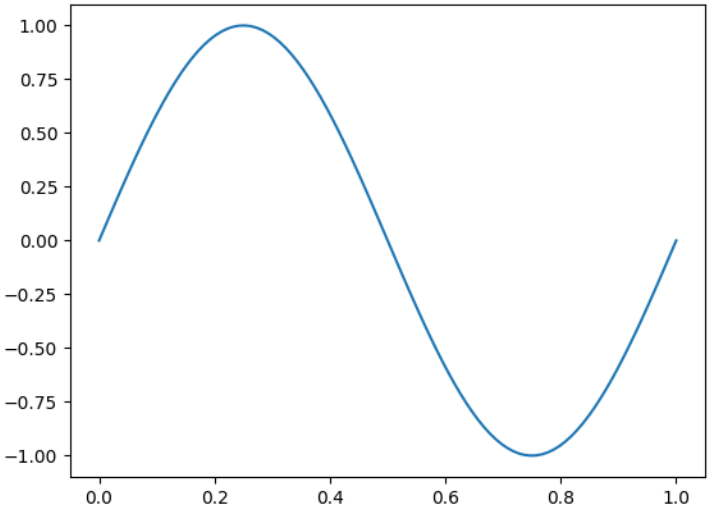
Pythonda ilmiy hisoblash uchun Enthought, Inc mahsulotlari qiziqish uyg‘otadi. Ilmiy vizualizatsiya uchun ikkita paketni taklif etiladi: Chaco va Mayavi. Chaco turli xil grafik formatlarni keng qo‘llab-quvvatlaydigan 2D interaktiv vizualizatsiya uchun ishlatiladi. Mayavi2 to‘plami interaktiv 3D ma’lumotlar vizualizatsiyasini oson yaratadi. Bunga MATLAB va matplotlibda topilganlarga o‘xshash vizualizatsiya funksiyalaridan foydalanib, skalyar, vektor va tenzor 2D va 3D miqdorlarini ko‘rsatish orqali erishiladi.

6.2-§. Matplotlib ning asosiy xususiyatlari

$\sin(2\pi x)$ matematik funksiyaning $0 \le x \le 1$ oraliqda grafigini chizishda Matplotlib to‘plamidan foydalanishga misol keltiraylik. NumPy paketidagi massivlar bilan ishlash funksiyalaridan foydalanib, ikkita bir o‘lchovli 100 ta elementdan iborat *x* va *y* massivlarni yaratamiz. Grafik chizish uchun Matplotlib paketidagi pyplot modulidan foydalanamiz.

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0., 1., 100)
y = np.sin(2*np.pi*x)
plt.plot(x, y)
plt.show()
```

Ushbu dastur ishining natijasi quyidagi grafikdan iborat bo‘ladi (6.1-rasm).



6.1-rasm. Grafik oynasi

Birinchi navbatda show() buyrug‘i bilan chaqiriladigan ushbu dialog oynasida diagramma bilan ishlashning ba’zi interaktiv xususiyatlarini qayd etamiz. U navigatsiya asboblar panelini o‘z ichiga oladi, uning funksionalligi eksperimental tarzda osongina tahlil qilinadi. Sichqoncha yoki klaviatura yorliqlari (yorliq tugmalari) yordamida grafikni siljitish, hajmini o‘zgartirish, grafikning bir qismini tanlash mumkin. Grafikni tanlangan grafik formatda saqlash tugmasi ham mavjud. Siz quyidagi kengaytmali fayllarni tanlashingiz mumkin: png, emf, ps, eps, svg, svgz, raw, rgba va pdf.

Dasturlashda grafiklar bilan interaktiv ishlash imkoniyati natijalarni grafik tasvirlashda muhim ahamiyatga ega. Dastur ishining yakunida natijalarni grafik

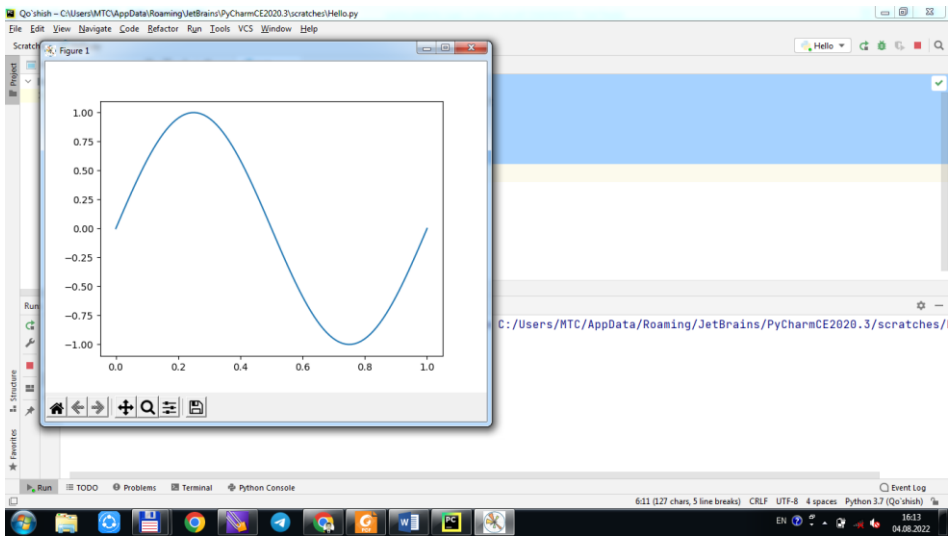
fayllar koʻrinishida saqlashga katta ahamiyat beriladi. Matplotlib toʻplamida bu xususiyat savefig() buyrugʻi bilan amalga oshiriladi.

savefig() funksiyasida birinchi talab qilinadigan argument fayl nomidir. Fayl quyidagi rastr va vektor formatlaridan birida yozilgan boʻlishi mumkin (6.1-jadval). Boshqa ixtiyoriy parametrlar ma’lumotlarni chiqarishni boshqarish imkonini beradi. Xususan, tasvirning fonini oʻzgartirish (facecolor parametri, standart boʻyicha 'w' - oq), grafik formatni (format parametri, standart boʻyicha png) yoki piksellar sonini (dpi parametri) tanlash mumkin. Windows metafayllari bilan ishlash uchun pyemf10 modulini oʻrnatish kerak, ammo emf ga eksport qilishda hali ham ba’zi muammolar mavjud. Chizmalarni tayyorlash uchun vektor formati sifatida eps va bitmap sifatida png foydalanish tavsiya etilishi mumkin.

6.1-jadval. Qoʻllab-quvvatlanadigan rasm formatlari

Kengaytma	Nomi
Emf	Windows Metafile
Eps	Encapsulated PostScript
Pdf	Portable Document Format
Png	Portable Network Graphics
Ps	PostScript
raw, rgba	Raw RGB A bitmap
svg, svgz	Scalable Vector Graphics

PyCharm muhitida yaratilgan grafiklar alohida oynada namoyon boʻladi. Grafiklarni turli formatda saqlash mumkin, muhit sukunat boʻyicha png formatda saqlashni tavsiya etadi.



6.2-rasm. PyCharm muhitida grafiklarni tasvirlash oynasi

6.3-§. Grafiklarni chizish

Grafiklarni chizishning asosiy funksiyasi plot() funksiyasi bo‘lib, ushbu funksiyaning argumentlarini o‘rnatish orqali chiziqlar turini va qalinligini, ularning rangini nazorat qilish, markerlardan foydalanish mumkin. Ushbu funksiyadan foydalanishning barcha imkoniyatlarini umumiy holda yordam buyrug‘i orqali topish mumkin (misol uchun help(plt.plot)).

Chiziqlar turini tanlashning ba’zi variantlari 6.2-jadvalda ko‘rsatilgan.

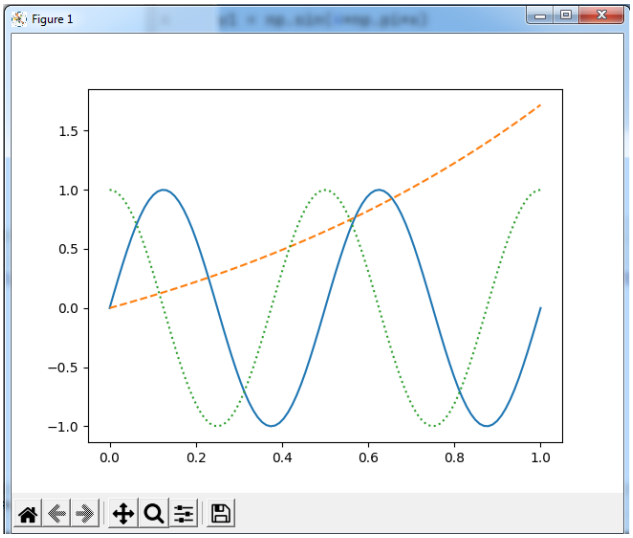
6.2-jadval. Grafik chiziqlar

Belgilar	Chiziq turlari
’-’	Uzluksiz chiziq
’--’	Uzlukli chiziq
’-.’	Uzlukli-nuqtali chiziq
’:’	Nuqtali chiziq

6.3-rasmda quyidagi dasturning natijasini aks ettirilgan. Sukunat bo‘yicha (6.1-rasmga qarang) uzluksiz chiziqdan foydalaniladi.

```
import numpy as np
import matplotlib.pyplot as plt
x=np.linspace(0., 1., 100)
```

```
y1 = np.sin(4*np.pi*x)
y2 = np.exp(x) - 1.
y3 = np.cos(4*np.pi*x)
plt.plot(x, y1,'-',x,y2, '--', x, y3, ':')
plt.show()
```



6.3-rasm. Chiziq turlari

Grafikdagi alohida nuqtalarni ko‘rsatish uchun markerlardan foydalaniladi, ularning ba’zilari 6.3-jadvalda ko‘rsatilgan.

6.3-jadval. Marker turlari

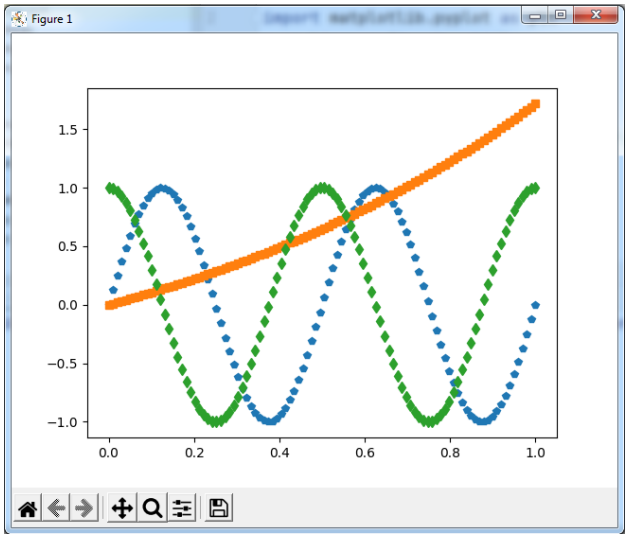
Belgi	Tavsif	Belgi	Tavsif
’.’	Qalin nuqta	’s’	Kvadrat
‘o’	Aylana	’p’	Besh qirrali yulduz
’v’	Pastga qaragan uchburchak	’*’	*
’^’	Yuqoriga qaragan uchburchak	’+’	+
’<’	Chapga qaragan uchburchak	’x’	X
’>’	O‘ngga qaragan uchburchak	’d’	Romb

Yuqoridagi dasturdagi

plt.plot(x, y1,'-',x,y2, '--', x, y3, ':')

kod satrini quyidagi tarzda o‘zgartirish natijasida grafika 6.4-rasmdagi kabi o‘zgaradi

plt.plot(x, y1,'p',x,y2, 's', x, y3, 'd')



6.4-rasm. Markerli grafika

Grafikda chiziqlarni tasvirlashda siz turli xil ranglardan foydalanishingiz mumkin (6.4-jadval). Chiziq turlarini, markerlarni rang bilan birlashtirib, grafikdagi chiziqlarni ajratish mumkin. Masalan, plot(x, y, 'ro:') funksiyasidan foydalanilganda chizma doira belgili qizil chiziq shaklida chiziladi.

6.4-jadval. Chizish ranglari

Belgi	Rang	Belgi	Rang
’b’	Ko‘k	’m’	To‘q qizil
’c’	Zangori	’r’	Qizil
’g‘	Yashil	’w’	Oq
’k’	Qora	’y’	Sariq

Grafikda chiziq qalinligi va marker o‘lchamlarini o‘zgartirish mumkin. Buning uchun plot() funksiyasining linewidth(lw) va markerssize(ms) argumentlari qo‘llaniladi, masalan, plot(x, y, lw=2, ws=4), sukunat bo‘yicha ikkala parametrning qiymati 1 ga teng.

6.4-§. Dizayn elementlari

Grafikka qo‘shimcha tahrirlashlar uning sarlavhasi, o‘qlar nomi, koordinatalar panjarasi va afsona bilan beriladi. Matplotlib paketida bu funksiyalar title(), xlabel(), ylabel(), grid(), legend() funksiyalari tomonidan quvvatlanadi.

Diagrammaga sarlavha title() funksiyasi (sarlavha satri majburiy argument) orqali qo‘yiladi. Siz matematik formulalar uchun LATEX nashriyot tizimi buyruqlaridan foydalanishingiz mumkin. Matplotlib-da rus tilini qo‘llab-quvvatlash alohida harakatlarni talab qiladi, shuning uchun dastlab ingliz tilidan foydalanish tavsiya qilinadi. Shrift o‘lchamini boshqarish uchun fontsize parametridan quyidagi qiymatlar bilan foydalaniladi: ’large’ (katta), ’medium’ (o‘rta), ’small’ (kichik) yoki aniq o‘lcham. Sarlavhaning vertikal holati verticalalignment(va) parametri bilan belgilanadi, u quyidagi qiymatlarni oladi: ’top’ (yuqori), ’baseline’ (asosiy), ’bottom’ (quyi), ’center’ (markaz). Gorizontal joylashuv ’center’ (markaz), ’left’ (chap), ’right’ (o‘ng) qiymatlari bilan horizontalalignment (ha) parametri bilan belgilanadi.

Koordinata o‘qlarini belgilash uchun xlabel(), ylabel() funksiyalaridan foydalaniladi. axis() funksiyasi x va y diapazonlarini boshqaradi. Yangi diapazonlar axis(xmin, xmax, ymin, ymax) buyrug‘i bilan belgilanadi. Ushbu funksiyada o‘qlar bilan qo‘shimcha manipulyatsiyalar qilish imkoniyati mavjud. Masalan, axis('off') buyrug‘i o‘qlarni olib tashlaydi, axis('equal') o‘qlar bo‘ylab o‘zgarishlar diapazonlarini tekislaydi va hokazo.

Bir nechta grafiklarni chizishda, alohida grafiklarni afsona bilan aniqlash foydalidir. Afsona qo‘shish uchun plot() funksiyasida label argumentini har bir chizma uchun satr sifatida belgilash orqali legend() funksiyasidan foydalaniladi. Ikkinchi imkoniyat yorliqlarni legend() funksiyasiga argument sifatida satrlar

ro‘yxati yoki to‘plami sifatida aniq belgilashni o‘z ichiga oladi. Afsonaning joylashtirilishi loc argumentiga muvofiq amalga oshiriladi (6.5-jadval).

6.5-jadval. Grafikda afsona joylashishi

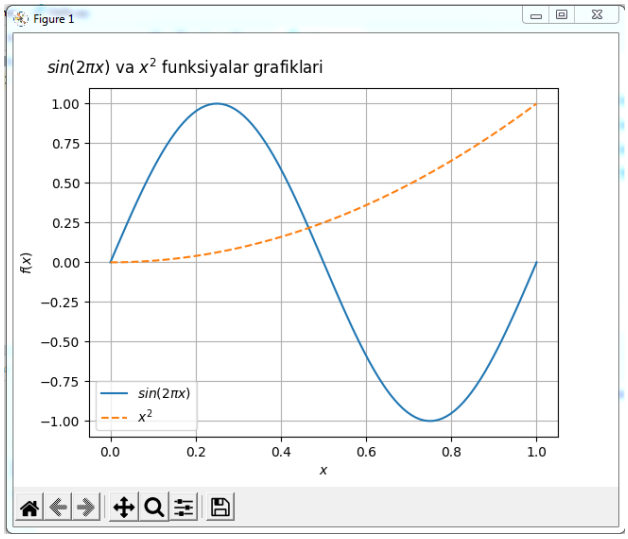
Qator	Kod	Qator	Kod
’best’	0	’center left’	6
’upper right’	1	’center right’	7
’upper left’	2	’lower center’	8
’lower left’	3	’upper center’	9
’lower right’	4	’center’	10
’right’	5		

Koordinatalar to‘rini ko‘rsatish uchun chiziqlar xossalarini belgilash imkoniyati bilan grid() funksiyasidan foydalaniladi. To‘r oralig‘i ustidan qo‘shimcha nazorat xticks() va yticks() funksiyalari tomonidan ta'minlanadi.

Quyidagi dastur grafikni loyihalashning ba’zi imkoniyatlarini ko‘rsatadi (6.5-rasm).

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0., 1., 100)
y1 = np.sin(2*np.pi*x)
y2 = x*x
plt.plot(x, y1, '-', label='$sin(2\pi x)$')
plt.plot(x, y2,'--', label='$x^2$')
plt.title('$sin(2\pi x)$ va $x^2$ funksiyalar grafiklari' , \
   fontsize='large', va='bottom', ha='right')
plt.xlabel('$x$')
plt.ylabel('$f(x)$')
plt.legend(loc='lower left')
plt.grid(True)
```

```
plt.show()
```

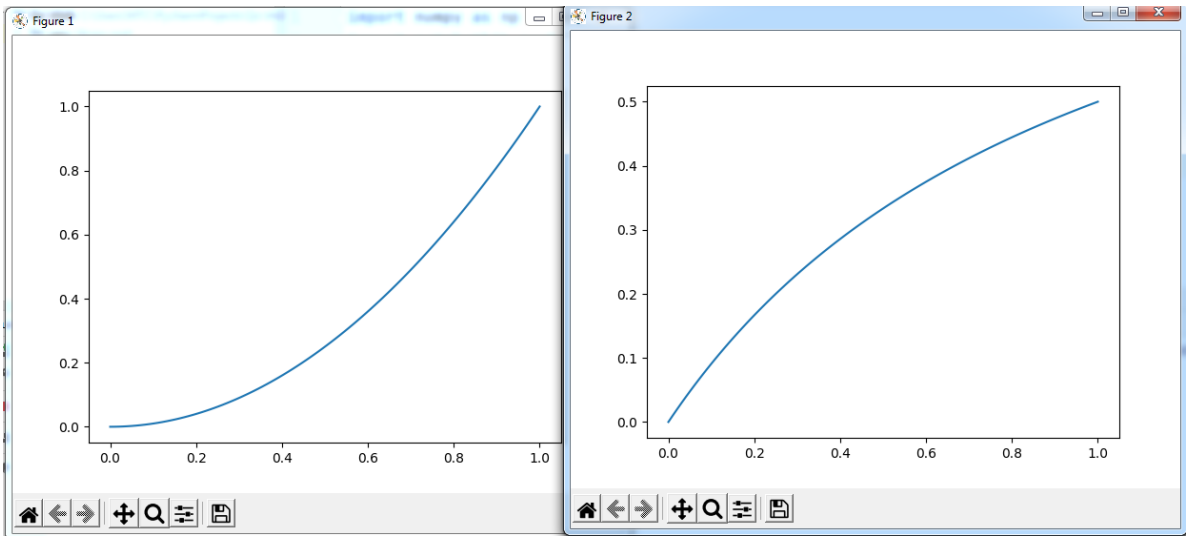


6.5-rasm. Grafik dizayni bilan ishlash

Yuqorida biz bitta grafikni chizish imkoniyatlarini koʻrib chiqdik. Amaliyotda bir oynada bir nechta grafikni chizishga toʻgʻri keladi. Hisoblash amaliyotida bir nechta shaklni tayyorlash kerak boʻlishi mumkin va ularning har biri bir nechta grafiklarni oʻz ichiga olishi mumkin.

Ikkita shakl bilan ishlashga misol, 6.6-rasmda koʻrsatilgan, quyidagicha amalga oshiriladi.

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0., 1., 100)
plt.figure(1)
plt.plot(x, x*x)
plt.figure(2)
plt.plot(x, x/(1+x))
plt.show()
```

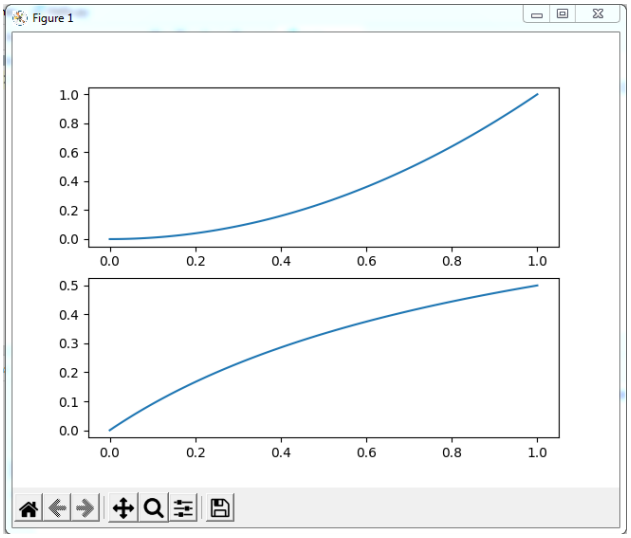


6.6-rasm. Grafiklarni alohida oynalarda tasvirlash

Bitta rasmga bir nechta grafiklarni joylashtirishning bir nechta imkoniyatlari mavjud. Birinchi imkoniyat subplot() funksiyasidan foydalanish bilan bog‘liq bo‘lib, u rasmni matritsa elementlari kabi joylashtirilgan bir xil o‘lchamdagi bir nechta to‘rtburchak maydonlarga ajratadi. Ushbu funksiyaning birinchi argumenti qatorlar sonini (numrows), ikkinchisi - faol rasmdagi ustunlar sonini (numcols) va uchinchisi - grafik raqamini (fignum) aniqlaydi. Agar numrows*numcols < 10 bo‘lsa, vergul qo‘yish ixtiyoriy, shuning uchun subplot(1,2,1) o‘rniga subplot(121) dan foydalanish mumkin.

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0., 1., 100)
plt.figure(1)
plt.subplot(211)
plt.plot(x, x*x)
plt.subplot(212)
plt.plot(x, x/(1+x))
plt.show()
```

Dastur natijasi 6.7-rasmda aks etadi:

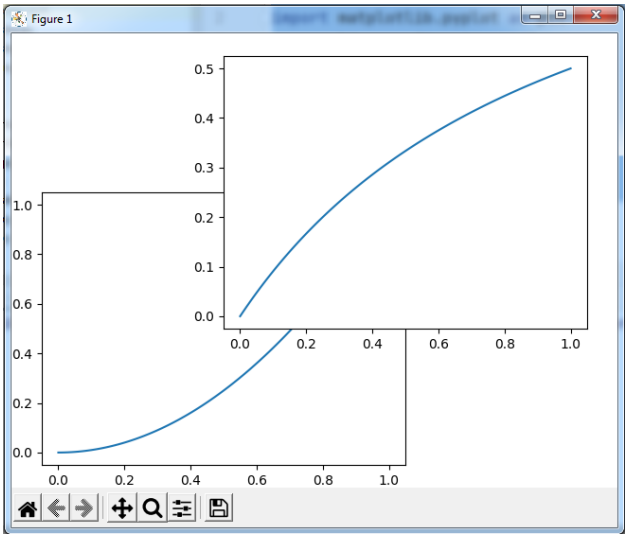


6.7-rasm. Bir oynada ikkita grafika

Grafiklarni rasmga qo‘lda joylashtirish mumkin. Buning uchun axes() funksiyasidan foydalaniladi. Ushbu funksiyaning argumenti ro‘yxat [left, bottom, width, height] ([chap, pastki, kenglik, balandlik]) bo‘lib, u nisbiy koordinatalarda rasmdagi grafik o‘rnini belgilaydi (barcha qiymatlar 0 dan 1 gacha). Ushbu imkoniyatlardan foydalanish quyidagi dasturda ifodalangan

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0., 1., 100)
plt.figure(1)
plt.axes([0.05, 0.05, 0.6, 0.6])
plt.plot(x, x*x)
plt.axes([0.35, 0.35, 0.6, 0.6])
plt.plot(x, x/(1+x))
plt.show()
```

natija quyidagi rasmda namoyish etilgan.



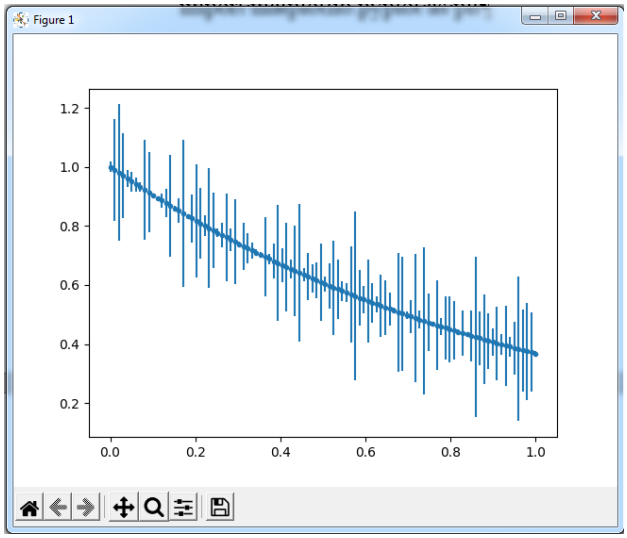
6.8-rasm. Grafiklarni qo‘lda joylashtirish

6.5-§. Bir o‘lchamli grafika

Har qanday ilmiy vizualizatsiya dasturiy vositalarining asosiy imkoniyatlari bir o‘lchovli va ikki o‘lchovli hisoblangan ma’lumotlarning grafik tasvirini o‘z ichiga oladi. Yuqorida keltirilgan plot() funksiyasidan foydalanish bo‘yicha misollar $y=f(x)$ (bir o‘lchovli funksiya) funksiya grafiklarini qurish imkoniyatini ko‘rsatdi. Quyida Matplotlib paketidagi 1D grafikaning boshqa imkoniyatlari bayon etiladi.

Xato bilan ko‘rsatilgan qiymatlar grafiklarini ko‘rsatish uchun errorbar() funksiyasidan foydalaniladi. Abtsissalar massivi va ordinatalar massividan keyin ordinata xatosi uchinchi argument sifatida uzatiladi. Kengaytirilgan xususiyatlar absissani o‘rnatishdagi xatolarni o‘z ichiga oladi. errorbar() funksiyasidan foydalanishga misol quyida ko‘rsatilgan, natija 6.9-rasmda.

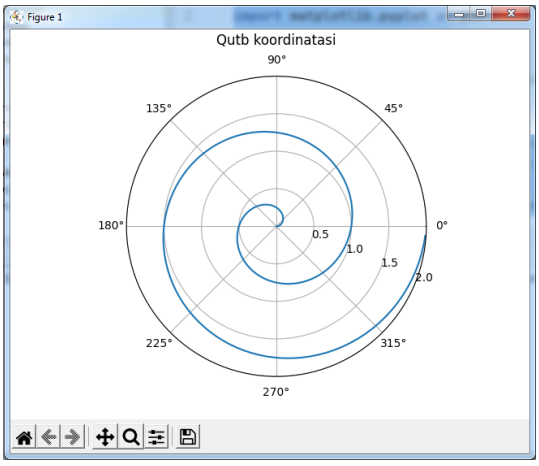
```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0., 1., 100)
y = np.exp(-x)
e = 0.1*np.random.normal(0, 1, 100)
plt.errorbar(x, y, e, fmt='.')
plt.show()
```



6.9-rasm. Grafikada xatoliklarni ifodalash

Qutb (ρ,φ) koordinatalarini chizishning eng oddiy usuli plot() o‘rniga polar() funksiyasidan foydalanishdir. Bu funksiyaning birinchi argumenti φ burchak, ikkinchisi ρ. Qutb koordinatasi bo‘yicha chizish dasturi:

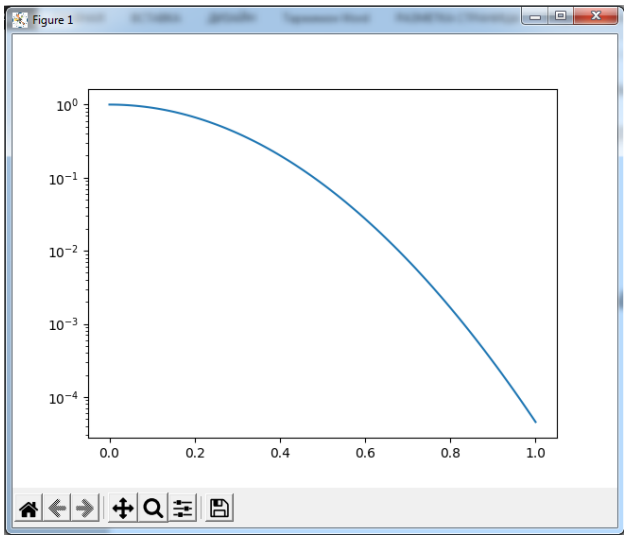
```
import numpy as np
import matplotlib.pyplot as plt
phi = np.arange(0, 2, 0.01); ro = 2 * np.pi * phi
fig, ax = plt.subplots(subplot_kw={'projection': 'polar'})
ax.plot(ro, phi); ax.set_rmax(2)
ax.set_rticks([0.5, 1, 1.5, 2]); ax.set_rlabel_position(-22.5);
ax.grid(True); ax.set_title("Qutb koordinatasi", va='bottom')
plt.show()
```



6.10-rasm. Qutb koordinatalari

Bir yoki ikkala koordinata o‘qi bo‘ylab ko‘proq ko‘rinadigan grafik yaratish uchun siz logarifmik masshtabni tanlashingiz mumkin. Bunday grafiklarni qurish uchun plot() o‘rniga loglog() (har ikki o‘qda logarifmik masshtab), semilogx() (x o‘qi bo‘yicha logarifmik masshtab), semilogy (y o‘qi bo‘yicha logarifmik masshtab) funksiyalaridan foydalaniladi. Bunga quyidagi dastur yordamida olingan 6.11-ramsdagi grafik misol bo‘ladi.

```
import numpy as np
import matplotlib.pyplot as plt
x = np.linspace(0., 1., 100)
y = np.exp(-10*x*x)
plt.semilogy(x,y)
plt.show()
```

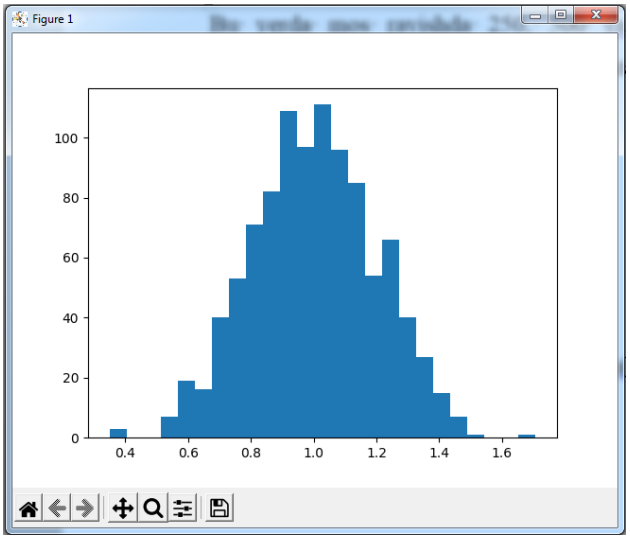


6.11-rasm. Logarifmik masshtab

Berilgan elementlarning o‘zgarish chastotasining tegishli guruhlash oralig‘iga bog‘liqligini grafik tasvirlash uchun gistogrammalar qo‘llaniladi. Gistogrammalarni yaratish hist() funksiyasi tomonidan amalga oshiriladi, uning birinchi argumenti ma’lumotlar massivi, ikkinchisi esa intervallar sonini belgilaydi. Bu yerda mos ravishda 250, 500 va 1000 tasodifiy normal taqsimlangan

o‘zgaruvchilarning namunasi uchun uchta gistogramma misoli keltirilgan (6.12-rasm).

```
import numpy as np
import matplotlib.pyplot as plt
mu = 1
sigma =0.2
x = mu + sigma*np.random.randn(1000)
plt.hist(x, 25)
plt.show()
```

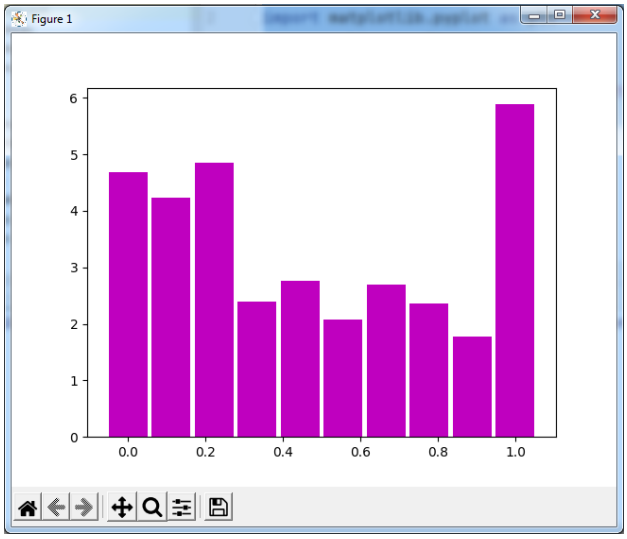


6.12-rasm. Gistogramma

Ba'zan natijalarni ko‘rish uchun turli xil diagrammalardan - ustunli va doiraviy diagrammalardan foydalanishga to‘g‘ri keladi. Ustunli diagrammalar bar() funksiyasi (gorizontal diagrammalar uchun barh()) yordamida chiziladi. Ustunli diagrammadan foydalanish quyidagi dasturda misol sifatida keltirilgan (6.13-rasm):

```
import numpy as np
import matplotlib.pyplot as plt
pos = np.linspace(0., 1., 10)
val = 3 + np.random.randn(10)
```

```
plt.bar(pos, val, align='center', width=0.1, color='m')
plt.show()
```



6.13-rasm. Ustunli diagramma

Bunday vositalar taqdim etilgan kichik hajmdagi ma’lumotlarga ega bo‘lgan biznes grafiklari uchun odatiy bo‘lib, ilmiy hisoblash uchun unchalik ahamiyatga ega emas.

6.6-§. Ikki o‘lchamli grafika

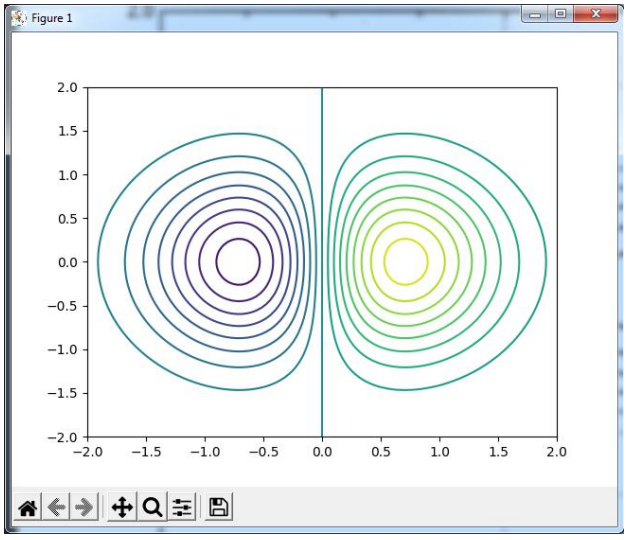
Raqamli hisob-kitoblarni amalga oshirishda ikki o‘lchovli ma’lumotlarning vizualizatsiyasiga katta e’tibor beriladi. Shu asosda, alohida yuzalar va bo‘limlarda vizualizatsiya yordamida ko‘proq umumiy uch o‘lchovli ma’lumotlarni ko‘rsatish uchun ma'lum vositalar amalga oshiriladi.

Ikki o‘lchovli $z=f(x,y)$ funksiya grafigini qurish masalasi ko‘rib chiqiladi. Qiymatlar to‘rtburchakli to‘rga, ya’ni $m \times n$ o‘lchamli ikki o‘lchovli massivga (matritsaga) o‘rnatilgan deb hisoblanadi. x va y massivlari mos ravishda n va m o‘lchamli vektorlar yoki z bilan bir xil o‘lchamdagi matritsalar (bir xil satrlar yoki ustunlar bilan) bo‘lishi mumkin.

Darajali chiziqlar (izoliniyalar) $z=f(x,y)$ sirtning $z=const$ tekisliklar bilan kesishishi natijasida hosil bo‘lgan egri chiziqlar. Ularni qurish uchun kontur() funksiyasidan foydalaniladi. Ushbu funksiyaga standart murojaat to‘rni belgilaydigan x, y massivlarini, so‘ngra z funksiya qiymatlari qatorini va darajalar sonini belgilashni o‘z ichiga oladi.

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(-2., 2., 101)
y = np.linspace(-2., 2., 101)
X, Y = np.meshgrid(x, y)
z = X * np.exp(-X**2 - Y**2)
plt.contour(x, y, z, 20)
plt.show()
```

Ushbu dasturning natijasi 6.14-rasmda ko‘rsatilgan.



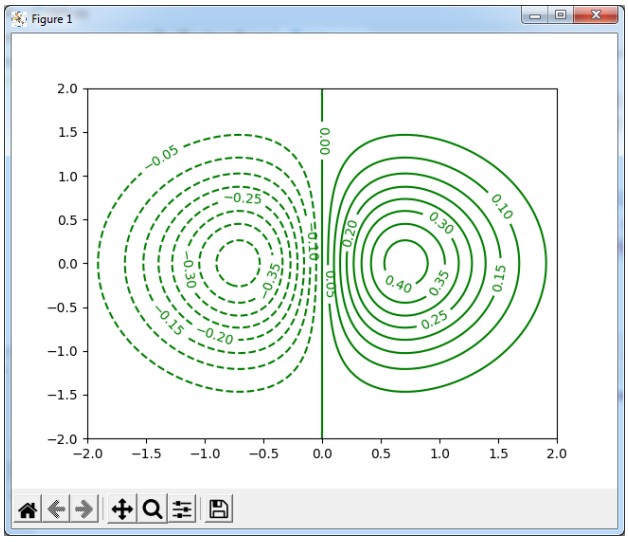
6.14-rasm. Darajali chiziqlar

Bu yerda meshgrid() funksiyasi *x*, *y* tekisligida bir o‘lchovli *x* va *y* massivlar bilan aniqlangan ikki o‘lchovli *X*, *Y* massivlar ko‘rinishidagi to‘rni belgilaydi. Massivlar ustidagi amallar yordamida ikkita o‘zgaruvchining funksiyalari ushbu funksiya yordamida osonlik bilan hisoblanadi.

Siz darajali chiziqlarning qiymatlarini ko‘rsatishingiz mumkin. Bu clabel() funksiyasi yordamida amalga oshiriladi, uning birinchi argumenti kontur orqali

qaytariladigan ContourSet obyektining natijasidir. Quyidagi dasturda ushbu operatorning ishlashi ko‘rsatiladi, natija 6.15-rasmda ko‘rsatilgan:

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(-2., 2., 101)
y = np.linspace(-2., 2., 101)
X, Y = np.meshgrid(x, y)
z = X * np.exp(-X**2 - Y**2)
v = np.linspace(-0.5, 0.5, 21)
cs = plt.contour(x, y, z, v, colors='g‘)
plt.clabel(cs, inline=True, fontsize=10)
plt.show()
```

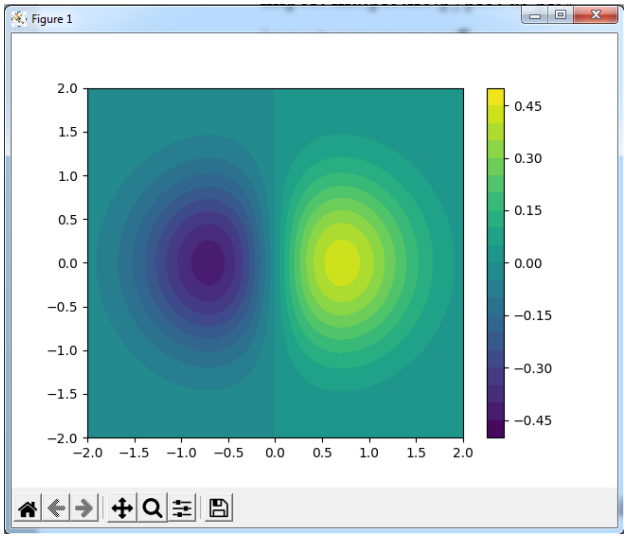


6.15-rasm. Turli darajadagi chiziqlarni boshqarish

Izoliniyalar rangini tanlash uchun ranglar parametridan foydalaniladi (6.4-jadval). Shtrixli chiziqlar salbiy qiymatlarga ega bo‘lgan izoliniyalarni ko‘rsatish uchun ishlatiladi. Ushbu misolda kontur chiziqlari aniq ko‘rsatilgan (v massivi).

Darajali chiziqlar orasidagi to‘ldirilgan maydonlar bilan grafiklarni chizish uchun contourf() funksiyasidan foydalaning. Bu funksiyadan foydalanish quyidagi misolda keltirilgan.

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(-2., 2., 101)
y = np.linspace(-2., 2., 101)
X, Y = np.meshgrid(x, y)
z = X * np.exp(-X**2 - Y**2)
v = np.linspace(-0.5, 0.5, 21)
plt.contourf(x, y, z, v)
plt.colorbar()
plt.show()
```



6.16-rasm. To‘ldirilgan darajali chiziqlar

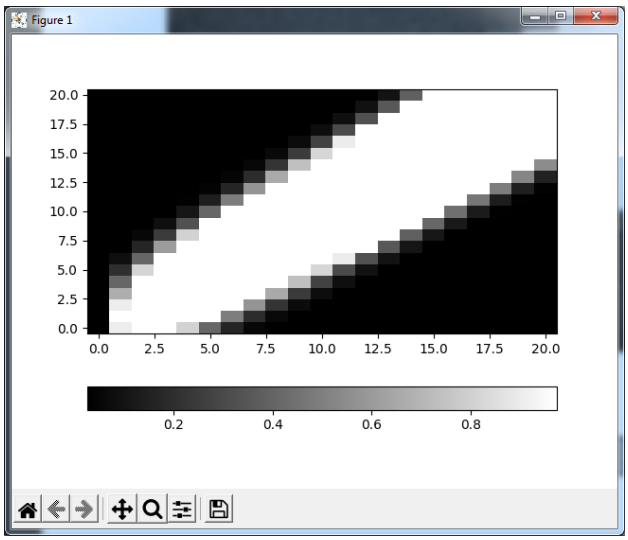
Rang paneli colorbar() yordamida ko‘rsatilgan satrlardagi qiymatlar bilan ko‘rsatiladi. Matplotlib paketi turli palitralardan foydalanadi (ularning ba’zilari 6.6-jadvalda ko‘rsatilgan, sukunat bo‘yicha hsv qiymatidan foydalaniladi).

6.6-jadval Palitralar

Nomlanishi	Tavsifi
’autumn’	qizil va sariq ranglar
’cool’	ko‘k va binafsha ranglarning soyalari
’flag‘	qizil, oq, ko‘k va qora ranglar arashmasi
’gray’	kulrangdagi chiziqli palitra
’hot’	qora, qizil, sariq va oq ranglar aralashmasi
’hsv’	kamalak ranglari
’pink’	pushti rang
’spectral’	spektral palitrasi
’spring‘	sariq va binafsha ranglarning soyalari
’summer’	yashil va sariq ranglarning soyalari
’winter’	ko‘k va yashil soyalar

Ba'zan hisoblashda siz matritsani ko‘rsatishingiz kerak. Buning uchun turli vositalardan (imshow(), matshow()) foydalanish mumkin. Quyidagi misolda palitrani tanlash uchun pcolor() funksiyasidan foydalanish qaraladi (6.17-rasm).

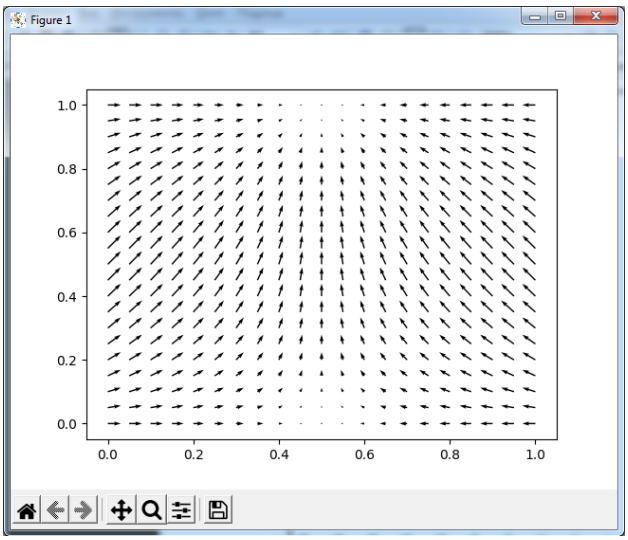
```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0, 20, 21)
y = np.linspace(0, 20, 21)
X, Y = np.meshgrid(x, y)
z = X * np.exp(-0.1*(X-Y)**2)
plt.pcolor(x, y, z, vmin=0.025, vmax=0.975, cmap='gray')
plt.colorbar(orientation='horizontal')
plt.show()
```



6.17-rasm. Matritsaning grafik koʻrinishi

Shuningdek, vektor maydonlarini tasvirlash uchun Matplotlib paketining minimal imkoniyatlari ham mavjud. Har bir tugundagi vektor komponentlari uchun strelka chiziladi, uning yoʻnalishi vektor yoʻnalishiga toʻgʻri keladi, uzunligi esa vektor uzunligiga proporsionaldir. Bu tasvir quiver() funksiyasi tomonidan amalga oshiriladi. Bu funksiyadan foydalanishga misol quyidagi dasturda keltirilgan, natija esa 6.18-rasmda tasvirlangan.

```
import matplotlib.pyplot as plt
import numpy as np
x = np.linspace(0., 1., 21)
y = np.linspace(0., 1., 21)
X,Y = np.meshgrid(x, y)
U = np.cos(np.pi*X)
V = np.sin(np.pi*Y)
plt.quiver(x, y, U, V)
plt.show()
```

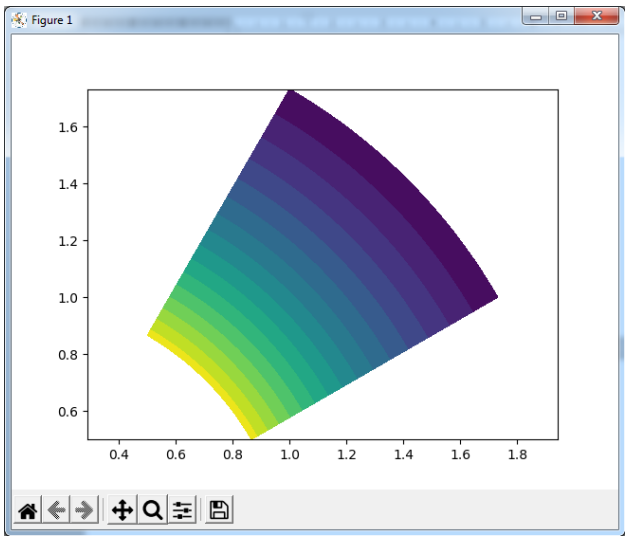


6.18-rasm. Vektor maydoni

Yuqorida to‘rtburchakli panjarada ma’lumotlarni tasvirlash bo‘yicha misollar keltirildi. Quyida ko‘proq umumiy sharoitlarda ikki o‘lchovli grafiklarni qurish uchun ba’zi imkoniyatlar qayd etiladi.

Izoliniyalar ixtiyoriy tuzilgan to‘rlarda qurilishi mumkin, agar to‘r to‘rtburchakka topologik ekvivalent bo‘lsa. Bunday holda, nafaqat funksiyaning o‘zi, balki to‘r tugunlarining har bir koordinatasi ham ikki o‘lchovli massiv sifatida ifodalanadi. Masalan, bir xil silindrsimon panjara ustiga qurilgan yechimning (6.19-rasm) tasvirini qaraylik.

```
import matplotlib.pyplot as plt
import numpy as np
ro = np.linspace(1., 2., 21)
phi = np.linspace(np.pi/6, np.pi/3, 21)
Ro, Phi = np.meshgrid(ro, phi)
X = Ro*np.cos(Phi); Y = Ro*np.sin(Phi)
Z = np.exp(-Ro)
plt.contourf(X, Y, Z, 20)
plt.axis('equal')
plt.show()
```



6.19-rasm. Polyar to‘r

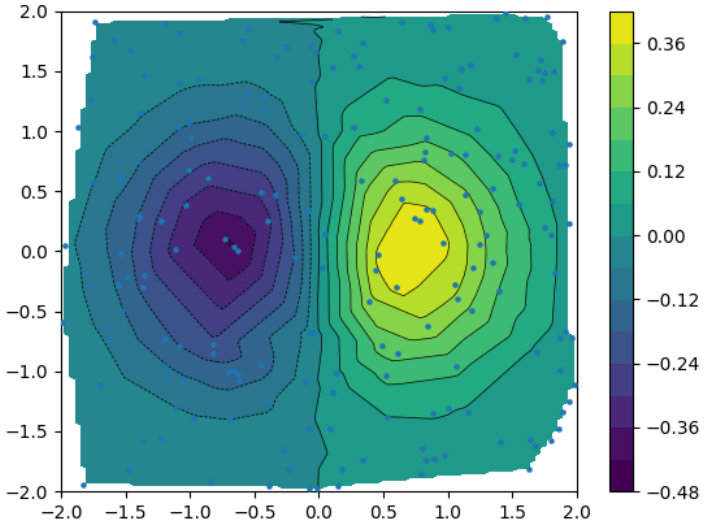
Ixtiyoriy tugunlar to‘plamidagi ma’lumotlar uchun Matplotlib paketidagi vizualizatsiya muammosi quyidagicha hal qilinadi. Birinchidan, berilgan tugunlar to‘plami uchun Delaunay triangulyatsiyasi tuziladi va ma’lumotlar tuzilgan panjaraga interpolyatsiya qilinadi, shundan so‘ng grafik chiziladi. Interpolatsiya muammosi mlab modulidan griddata() funksiyasi yordamida hal qilinadi. Yana bir imkoniyat Natgrid kengaytmasidan foydalanishdir (mpl_toolkits.natgrid Matplotlib paketidan alohida o‘rnatiladi).

x, y, z koordinatalarining bir o‘lchovli massivlari x, y va interpolyatsiya qilingan z funksiyasining berilgan qiymatlari bo‘lsin. To‘rtburchak panjara odatdagidek bir o‘lchovli yoki ikki o‘lchovli X va Y massivlari bilan belgilanadi. Ushbu yangi to‘rdagi qiymatlarning ikki o‘lchovli massivi griddata(x, y, z, X, Y) funksiyasi bilan aniqlanadi. Quyidaig dastur tomonidan tartibsiz tugunlar to‘plamida ma’lumotlarni vizualizatsiya qilish misoli keltirilgan:

```
from matplotlib.mlab import griddata
import matplotlib.pyplot as plt
import numpy as np
x = np.random.uniform(-2, 2, 250)
y = np.random.uniform(-2, 2, 250)
```

```
z = x*np.exp(-x**2 - y**2)
X = np.linspace(-2., 2., 100)
Y = np.linspace(-2., 2., 100)
Z = griddata(x, y, z, X, Y)
plt.contourf(X, Y, Z, 20, cmap = plt.cm.spectral)
plt.colorbar()
plt.scatter(x, y, marker='o', c='k', s=5)
plt.xlim(-2,2)
plt.ylim(-2,2)
plt.show()
```

Dasturning natijasi 6.20-rasmda ko‘rsatilgan.



6.20-rasm. Tartibsiz tugunlar to‘plami

Bu yerda scatter funksiyasi tanlangan o‘lcham va rangdagi markerlarni (s va c parametrlari) berilgan nuqtalarda diagrammada joylashtirish uchun ishlatiladi.

6.7-§. Uch o‘lchamli vizualizatsiya

Ikki o‘lchovli ma’lumotlarni vizualizatsiya qilishda biz *x*, *y* tekislikning alohida nuqtalaridagi funksiya qiymatini rang bilan bog‘ladik, ma’lumotlarni tekislikda aks ettirdik. Ikkinchi imkoniyat ikki o‘lchovli funksiya qiymatini uchinchi

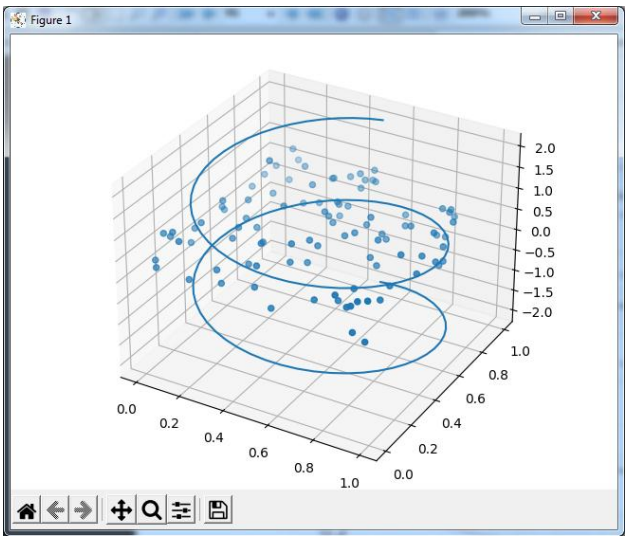
Dekart koordinatasi z ga bog‘lash bilan bog‘liq. Bunday holda, vizualizatsiya ob’ekti asosan uch o‘lchovli ob’ekt bo‘lgan sirtdir, ya’ni ikki o‘lchovli funksiyaning uch o‘lchovli tasviri.

3D vizualizatsiya ob’ektlari bilan ishlash imkoniyatlari Matplotlib paketining mplot3D modulida qo‘llab-quvvatlanadi. Ikki o‘lchovli funksiyalarning vizualizatsiyasini muhokama qilishdan oldin, oddiyroq ob’ektlarni ko‘rib chiqaylik.

Alohida nuqtalarni (scatter() funksiyasi) va fazoda parametrik egri chiziqni (plot() funksiyasi) chizish misoli 6.21-rasmda keltirilgan.

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = Axes3D(fig)
x = np.random.sample(100)
y = np.random.sample(100)
z = np.random.sample(100)
ax.scatter(x, y, z)
theta = np.linspace(0., 4 * np.pi, 100)
xl=0.5* (1+ np.sin(theta))
yl = 0.5 * (1 + np.cos(theta))
zl = np.linspace(-2, 2, 100)
ax.plot(xl, yl, zl)
plt.show()
```

Bu dastur natijasi quyidagi rasmda ifodalangan

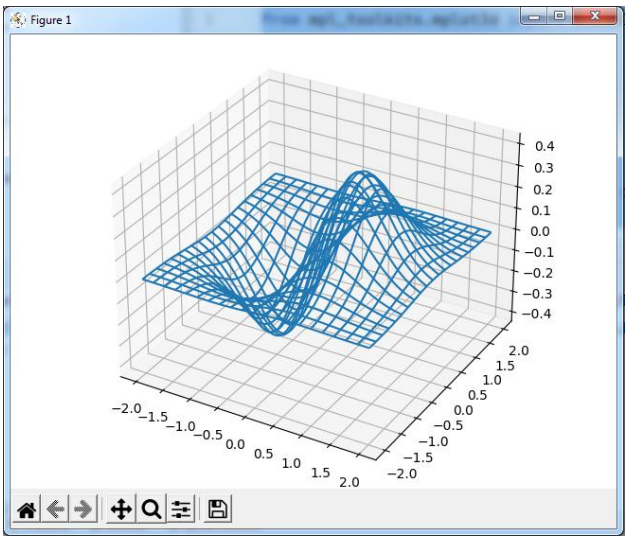


6.21-rasm. Fazodagi nuqtalar va egri chiziq

plot() va scatter() funksiyalarining parametrlari pyplot modulining shu kabi funksiyalarida topilgan parametrlarga yaqin. Ochilgan interaktiv oynada sichqonchani aylantirish va kattalashtirish orqali ko‘rsatilgan ob’ektning masshtabini o‘zgartirish uchun qo‘shimcha imkoniyatlar mavjud.

Soha sirtini panjarasimon tasvirlash uchun plot_wireframe() funksiyasidan foydalaniladi, uning argumentlari ko‘rsatilgan sirt koordinatalarining ikki o‘lchovli massivlaridir. Soha sirtini tasvirlash quyidagi dasturda keltirilgan, natijasi 6.22-rasmda tasvirlangan. (har bir yo‘nalishda massivning har beshinchi elementini chizadi).

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure(); ax = Axes3D(fig)
x = np.linspace(-2., 2., 101)
y = np.linspace(-2., 2., 101)
X, Y = np.meshgrid(x, y)
Z = X * np.exp(-X**2 - Y**2)
ax.plot_wireframe(X, Y, Z, rstride=5, cstride=5)
plt.show()
```

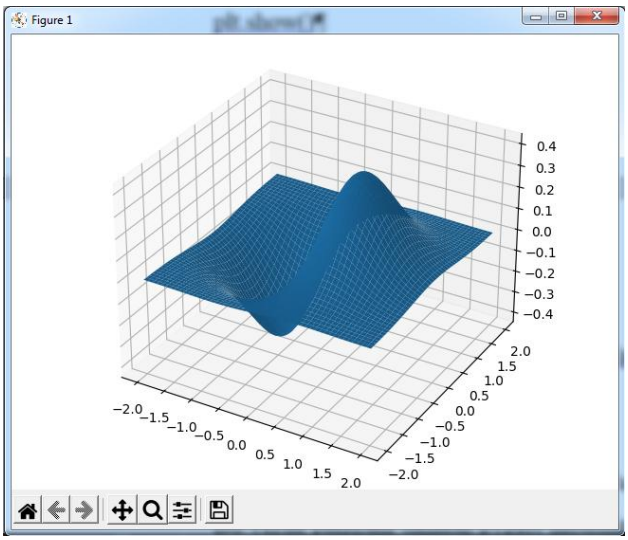


6.22-rasm. Panjarasimon tasvir

Ikki o‘lchovli hisoblangan ma’lumotlarni yanada ifodali ko‘rish uchun rang va balandlik bog‘lashlari ko‘pincha funksiya qiymatlari bilan birlashtiriladi. Bunday holda, grafik plot_surface() funksiyasi yordamida panjara to‘rtburchaklarini to‘ldirish bilan chiziladi.

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = Axes3D(fig)
x = np.linspace(-2., 2., 101)
y = np.linspace(-2., 2., 101)
X, Y = np.meshgrid(x, y)
Z = X * np.exp(-X**2 - Y**2)
ax.plot_surface(X, Y, Z, rstride=2, cstride=2)
plt.show()
```

Dasturning natijasi 6.23-rasmda ko‘rsatilgan.

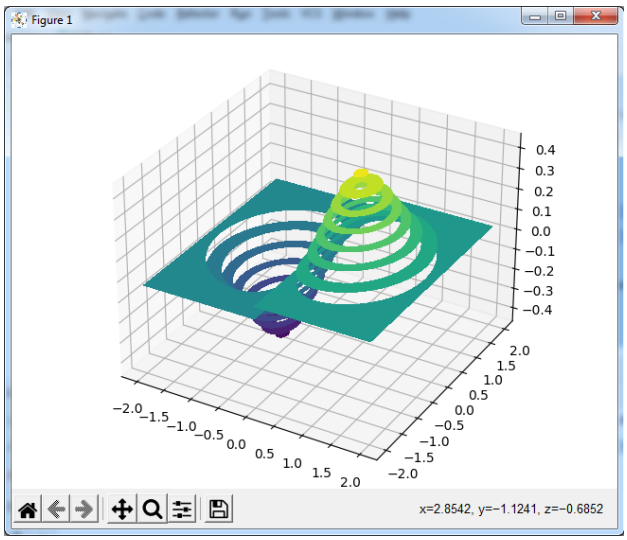


6.23-rasm. Soha yuzasi

Bu yerda palitrani tanlash pyplot modulining mos keladigan funksiyalariga o‘xshash tarzda belgilanadi.

Mplot3D moduli tomonidan taqdim etilgan 3D izoliniyalarni chizish qobiliyatini alohida ta'kidlash kerak. pyplot modulining mos funksiyalariga juda o‘xshash contour() va contourf() funksiyalaridan foydalaniladi. Bu funksiyalardan foydalanish quyidaig dasturda keltirilgan

```
import numpy as np
import matplotlib.pyplot as plt
from mpl_toolkits.mplot3d import Axes3D
fig = plt.figure()
ax = Axes3D(fig)
x = np.linspace(-2., 2., 101)
y = np.linspace(-2., 2., 101)
X, Y = np.meshgrid(x, y)
Z = X * np.exp(-X**2 - Y**2)
ax.contourf(X, Y, Z, 15)
plt.show()
```



6.24-rasm. Uch o‘lchamli konturlar

Umuman olganda, Matplotlib to‘plami ilmiy vizualizatsiyaning asosiy ehtiyojlarini qondiradi va uning asosida hisoblash ma’lumotlarini qayta ishlash uchun dasturiy ta’minot tuzilishi mumkin.

VII BOB. SCIPY PAKETI

Ushbu bob ilmiy hisoblashlarni amalga oshirishning kuchli dasturiy ta’minotlaridan biri bo‘lgan SciPy paketi imkoniyatlariga bag‘ishlanadi. Ushbu paketning qism paketlari, chiziqli algebra, optimallashtirish, interpolatsiya, integral hamda differensial masalalarni yechishga mo‘ljallangan funksiyalar haqida qisqacha bayon qilinadi.

7.1-§. Pythonda ilmiy hisoblash

Amaliy matematik modellashtirish matematik modellarni sonli tadqiq etish asosida amalga oshiriladi. Matematik modellarga differensial va algebraik tenglamalar sistemalari bilan bog‘langan hususiy hosilali differensial tenglamalarning chiziqli bo‘lmagan sistemalari kiradi. Ularning taqribiy yechimi sonli tahlil algoritmlariga asoslangan bo‘lib, ular orasida chiziqli va chiziqli bo‘lmagan tenglamalar sistemalarini va approksimatsiyalash algoritmlarini yechish usullarini qayd etishimiz mumkin.

Python kengaytmalari orasida hisoblash matematikasi masalalarini hal qilishga qaratilgan ko‘plab vositalar mavjud. Ilmiy hisoblash uchun mo‘ljallangan Python paketlaridan SciPy ni qayd etish mumkin.

Ilmiy hisoblash uchun Python modullarining yana bir to‘plami Scientific-Python deb nomlanadi. U vektorlar, tensorlar va kvaternionlar bilan ishlash vositalarini o‘z ichiga oladi, vektor va tenzor maydonlarini o‘zgartirishni qo‘llab-quvvatlaydi, avtomatik differensiatsiya, interpolyatsiya, polinomlar bilan ishlash, elementar statistik masalalar va minimallashtirish masalalari, shuningdek MPI (Message Passing Interface- Xabarni uzatish interfeysi) interfeyslari mavjud, ikki o‘lchovli va uch o‘lchovli vizualizatsiya bilan ishlaydi. Ushbu xususiyatlarning aksariyati SciPy ga kiritilgan.

Ehtimol, Python(x,y) ning eng katta va juda xilma-xil to‘plami Python tilida ilmiy va muhandislik dasturlash uchun barcha asosiy ishlanmalarni o‘z ichiga olgan. U nafaqat barcha asosiy matematik va grafik paketlarni, balki, bir nechta integratsiyalashgan ishlab chiqish vositalarini, grafik foydalanuvchi interfeysini ishlab chiqishni qo‘llab-quvvatlash vositalarini, C/C++ va Fortran tillari bilan

ishlash va parallel hisoblash tizimlarida hisob-kitoblarni qo‘llab-quvvatlashni o‘z ichiga oladi.

Xuddi shuningdek SciPy paketini muvaffaqiyatli to‘ldiradigan ilmiy hisoblashning maxsus masalalarini hal qilishga qaratilgan dasturiy ta’minotni ham ta'kidlaymiz. Masalan, Numexpr paketi massivlar bilan NumPy ga qaraganda tezroq ishlaydi. SciPy asosiy ilmiy hisoblash paketiga muvaffaqiyatli qo‘shimchaning ikkinchi misoli sifatida PyTrilinos ni ta'kidlash mumkin. PyTrilinos, xususan, vektorlarni, siyrak va zich matritsalarni, chiziqli tenglamalar sistemalarini hisoblash tizimlari parallel arxitekturasini hisobga olgan holda, aniq va iterativ usullar bilan sonli yechishni qurish va ulardan foydalanish masalalarini hal qiladigan Trilinos kutubxonalari to‘plamiga interfeysni taqdim etadi.

Oddiy differensial tenglamalar uchun chegaraviy masalalarni sonli yechish uchun bvp dan foydalanish mumkin. Chegaraviy masalalarni SfePy yordamida chekli elementlar usuli bilan yechish mumkin. Ushbu dastur tutash muhitlar mexanikasining ikki o‘lchovli va uch o‘lchovli masalalarini (issiqlik o‘tkazuvchanlik, elastiklik, Navier-Stoks tenglamalari) yechishga imkon beradi.

7.2-§. SciPy paketining umumiy xususiyatlari

SciPy paketi matematika, tabiiy fanlar va muhandislik uchun ochiq kodli dasturiy ta’minotdir. SciPy N o‘lchamli NumPy massivlari bilan qulay va tez ishlashni ta'minlaydi. Bundan tashqari, SciPy paketi raqamli integratsiya va optimallashtirish dasturlari kabi ko‘plab samarali hisoblangan sonli tahlil protseduralarini taqdim etadi.

Yuqorida ta'kidlaganimizdek, NumPy paketi hisoblash uchun eng muhim ob’ekt – N o‘lchovli massiv bilan ishlash uchun keng imkoniyatlarni taqdim etadi. Ushbu ob’ektlar bilan ishlash uchun alohida elementlar va guruhlarni yaratish, qayta tartibga solish, tanlash kabi kuchli vositalar mavjud. Bundan tashqari, NumPy paketi chiziqli algebra masalalari bilan bog‘liq bo‘lgan sonli tahlil modullarini o‘z ichiga oladi, Furye qatorlariga yoyishni amalga oshiradi va tasodifiy sonlar bilan ishlaydi.

SciPy paketi ilmiy hisoblashning turli sohalari bo‘yicha hisoblashni ta'minlaydigan paketlar to‘plami yoki subpackages (kichik paketlar) sifatida tashkil

etilgan. Python da dasturiy ta’minot uchun qabul qilingan terminologiyani hisobga olgan holda, biz ushbu paketlar haqida SciPy paketining modullari sifatida gaplashamiz. Aynan shu nuqtai nazardan, xususan, biz NumPy paketini SciPy paketining moduli sifatida ko‘rib chiqamiz.

SciPy paketi quyidagi (alifbo tartibida) modullarni o‘z ichiga oladi.

Clustering package (scipy.cluster). Klasterli tahlili funksiyalari taqdim etiladi, bu ob’ektlarning (jarayonlarning) berilgan namunasini bir-biriga o‘xshash bo‘lmagan kichik to‘plamlarga bo‘lish bilan bog‘liq bo‘lib, har bir klaster o‘xshash ob’ektlardan iborat bo‘lishi va turli klasterlarning ob’ektlari sezilarli darajada farqlanishi sharti bilan klasterlar deb ataladi. Klasterlash vazifasi statistik ishlov berishga taalluqlidir va ob’ektlar orasidagi masofa bo‘yicha klasterlarni ierarxik yaratish, kuzatish vektoridan masofalar matritsasi, statistik ma’lumotlarni guruhlar bo‘yicha hisoblash, dendrogrammalar yordamida klasterlarni vizualizatsiya qilishni o‘z ichiga oladi.

Constants (scipy.constants). Fizik konstantalar va qiymatlarni bir o‘lchov birligidan boshqasiga o‘tkazish uchun funksiyalar to‘plamini taqdim etadi. Xalqaro miqyosda qabul qilingan fundamental fizik konstantalar to‘plami CODATA (Fan va texnologiya ma’lumotlari qo‘mitasi) tomonidan taqdim etilgan. Qiymatlar, birliklar va nisbiy aniqlik mavjud.

Fourier transforms (scipy.fftpack). Modul signallar spektrini olish va ularning spektrlaridan signallar shaklini tiklash uchun to‘g‘ridan-to‘g‘ri va teskari Furyening tez almashtirishlarini (FFT) amalga oshirish uchun mo‘ljallangan. Diskret Furye almashtirishi kompleks (yoki haqiqiy) sonlar ketma-ketligini kompleks sonlar ketma-ketligiga aylantiradi. Bularni amalga oshirish taniqli va mashhur FFTPACK paketiga asoslangan.

Integration and ODEs (scipy.integrate). Integralning funksional va jadvalli spesifikatsiyasi bilan integrallarni hisoblash funksiyalari keltirilgan. Integratsiya modulining ikkinchi qismi oddiy differensial tenglamalar sistemasi uchun Koshi masalasini sonli yechish uchun mo‘ljallangan. Dasturiy ta’minotni amalga oshirish FORTRAN ODEPACK paketiga asoslangan.

Interpolation (scipy.interpolate). Interpolatsiya moduli bir o‘lchovli va ikki o‘lchovli funksiyalarni interpolyatsiya qilish va silliqlash algoritmlarini amalga oshiradi. Kubik splaynlar yordamida approksimatsiya qilish va silliqlashga alohida e’tibor beriladi. FORTRAN FITPACK dasturiy paketining ko‘pgina xususiyatlaridan foydalaniladi.

Input and output (scipy.io). SciPy paketida ma’lumotlarni kiritish/chiqarish NumPy ning asosiy o‘qish va yozish imkoniyatlarini to‘ldiradi. NumPy hisob-kitoblarning asosiy ob’ektlari bo‘lgan massivlarni o‘qish va yozishni qo‘llab-quvvatlaydi. SciPy paketi boshqa turli formatlar (ma’lumotlar, audio, video, tasvirlar, MATLAB va boshqalar) uchun ba’zi kiritish-chiqarish imkoniyatlarini amalga oshiradi.

Linear algebra (scipy.linalg). NumPy paketi asosiy chiziqli algebra masalalarini hal qiladi. ATLAS LAPACK va BLAS chiziqli algebra kutubxonalari ustiga qurilgan SciPy paketi yangi imkoniyatlarni taqdim etadi. Xususan, matritsa ob’ektlari bilan ishlashda matritsa funksiyalari bilan hisob-kitoblar qo‘llab-quvvatlanadi.

Maximum entropy models (scipy.maxentropy). Axborot nazariyasida maksimal entropiya modellarini tanlash protseduralarini o‘z ichiga oladi.

Miscellaneous routines (scipy.misc). SciPy paketining ixtisoslashtirilgan modullari sifatida tasniflanmagan oz sonli turli xil foydali funksiyalar taqdim etilgan.

Multi-dimensional image processing (scipy.ndimage). Modul qiymatlar massivlari bo‘yicha operatsiyalar asosida tasvirlarni tahlil qilish va qayta ishlash uchun mo‘ljallangan. Chiziqli va chiziqli bo‘lmagan filtrlash funksiyalari taqdim etilgan, interpolyatsiya va binar morfologiya masalalarini hal qilish uchun uskunalar mavjud.

Orthogonal distance regression (scipy.odr). Bog‘liq o‘zgaruvchilarda xatolar mavjud bo‘lganda modelning parametrlarini topish uchun eng kichik kvadratlar usuli qo‘llaniladi. odr moduli mustaqil o‘zgaruvchilarga ham xatolar kiritilganda umumiyroq vaziyatda parametrlarni baholash uchun samarali

algoritmlarni taqdim etadi. Jarayonni amalga oshirish ODRPACK paketiga asoslangan.

Optimization and root finding (scipy.optimize). Optimallashtirish moduli standart optimallashtirish algoritmlarining koʻpchiligini, ma'lum bir sohada haqiqiy funksiyaning ekstremumini (minimal yoki maksimal) topish masalalari yechimlarini ta'minlaydi. Shartli va shartsiz optimallashtirish masalalarini yechishning asosiy algoritmlari amalga oshirilgan. Modul shuningdek, chiziqli boʻlmagan tenglamalar sistemalarining yechimlarini topish funksiyalarini oʻz ichiga oladi.

Signal processing (scipy.signal). Signallarni qayta ishlash funksiyalari taqdim etiladi, ular haqiqiy yoki kompleks sonlar massivlari sifatida qaraladi. Modul signalni filtrlash masalalarini hal qilish uchun baʼzi funksiyalarni, shuningdek filtrlarni ishlab chiqish vositalarini oʻz ichiga oladi. 1D va 2D maʼlumotlar uchun interpolatsiya B-splaynlar yordamida taqdim etiladi.

Sparse matrices (scipy.sparse). Asosan nol elementlarga ega matritsalar siyrak matritsa boʻladi. Bunday turdagi matritsalar differensial tenglamalar uchun chegaraviy masalalarni sonli yechishda ilmiy hisoblar uchun xosdir. Sparse moduli siyrak matritsalarni yaratish, saqlash va oʻzgartirish funksiyalarini amalga oshiradi. Turli xil maʼlumotlar tuzilmalari qoʻllab-quvvatlanadi, xususan, siyrak matritsalar uchun standart CSR (Compressed Sparse Row) va CSC (Compressed Sparse Column) formatlari.

Sparse linear algebra (scipy.sparse.linalg). SciPy paketining sparse.linalg moduli asosiy chiziqli algebra masalalarini yechish uchun ishlatilishi mumkin boʻlgan funksiyalarni taqdim etadi. Nosimmetrik va nosimmetrik siyrak matritsalar bilan tenglamalar sistemasini yechishning toʻgʻridan-toʻgʻri va asosiy iterativ usullari, siyrak matritsalar uchun xos qiymatlar va xos funksiyalarni hisoblash algoritmlari amalga oshiriladi.

Spatial algorithms and data structures (scipy.spatial). Fazoviy maʼlumotlar tuzilmalari va algoritmlari bilan ishlash uchun moʻljallangan. spatial modulida KD daraxtlari bilan ishlash, xususan, eng yaqin qoʻshni, taxminiy qoʻshni topish va barcha eng yaqin juftlarni topish funksiyalari mavjud.

Special functions (scipy.special). special moduli SciPy ni matematik fizikaning ko‘plab maxsus funksiyalari bilan ta'minlaydi (Eyri funksiyalari, Bessel funksiyalari, gipergeometrik funksiyalar va boshqalar).

Statistical functions (scipy.stats). SciPy paketining stats modulida statistik ma’lumotlarni qayta ishlash uchun ko‘plab asosiy protsedural mavjud. Diskret va uzluksiz taqsimotlar uchun tasodifiy o‘zgaruvchilar generatorlari to‘plamini amalga oshiradi.

Image Array Manipulation and Convolution (scipy.stsci). O‘rtacha va ekstremal qiymatlarni hisoblash, tasvirlar massivlari uchun konvolyutsiyalar qo‘llab-quvvatlanadi.

C/C++ integration (scipy.weave). weave moduli yordamida Python kodiga C/C++ kodini kiritish ta'minlanadi, bu hisob-kitoblarni sezilarli darajada tezlashtirishi mumkin.

SciPy paketi modullarining ko‘pchiligi bevosita ilmiy hisoblashlarga qaratilgan, birinchi navbatda hisoblash matematikasi masalalarini hal qilish bilan bog‘liq.

7.3-§. SciPy paketida chiziqli algebra masalalari

SciPy paketi chiziqli algebra masalalarini yechish uchun keng xususiyatlarni taqdim etadi. Yuqorida ta'kidlaganimizdek, bu vazifalar NumPy paketida taqdim etilgan, ammo bu yerda biz linalg moduli tomonidan taqdim etilgan vositalarni qisqacha tasvirlab beramiz. Ushbu modul matritsalar va vektorlar normalarini hisoblash, matritsalarni teskarisini topish va ko‘paytuvchilarga yoyish, chiziqli tenglamalar sistemasini yechish funksiyalarini o‘z ichiga oladi, sizga xos qiymatlar va xos vektorlarni topish va matritsa funksiyalarini hisoblash imkonini beradi. Ilmiy hisoblashda ayniqsa muhim bo‘lgan siyrak matritsalar bilan chiziqli algebra masalalarini yechish uchun sparse.linalg moduli mo‘ljallangan.

SciPy da, NumPy paketi kabi, biz ham massiv, ham matritsalar bilan ishlay olamiz. Matritsalarni belgilashda MATLAB sintaksisi (qatorlar bo‘yicha kiritish) qo‘llab-quvvatlanadi.

Matritsaning determinantini hisoblash uchun det() funksiyasi mavjud. norm() funksiyasi matritsa yoki vektor normasini topish uchun mo‘ljallangan. Ushbu funksiyaning ikkinchi argumenti 1, 2 va inf (sukut bo‘yicha 2) qiymatlarini oladi va mos ravishda $p = 1,2,\infty$ da l_p vektor normasini va subordinatsiyali matritsa normasini tanlashga mos keladi. Bu yerda inf cheksizlikni bildiradi (NumPy paketida belgilangan).

Misol tariqasida, natijada nolga erishadigan funksiyalar uchun $0 \le x \le 1$ oraliqda teskari belgisi bo‘lgan ikkinchi tartibli ayirma hosilasiga mos keladigan matritsani ko‘rib chiqamiz. h ichki tugun $x_i, i = 0,1,\ldots,m-1$ va $h = 1/(m+1)$ bosqichli bir xil to‘r ishlatiladi. $m \times m$ o‘lchovli A matritsaning determinantini va turli normalarini hisoblaymiz, Xoleskiy yoyilmasini bajaramiz, teskari matritsasini topamiz va $b = 1$ uchun $Ay = b$ tenglamaning yechimini olamiz.

```
import numpy as np
from scipy import linalg
import matplotlib.pyplot as plt
m = 25
h = 1. / (m+1)
A = np.mat(np.zeros((m, m),'float'))
for i in range(1, m):
    A[i-1,i] = - 1. / h**2
    A[i,i] =2. / h**2
    A[i,i-1] = - 1. / h**2
A[0,0] = 2. / h**2
A[m-1, m-1] = 2. / h**2
plt.matshow(A)
print('det:', linalg.det(A))
print('norm-1:', linalg.norm(A, 1))
print('norm-2:', linalg.norm(A, 2))
print('norm-inf:',linalg.norm(A, np.inf))
```

```
L = linalg.cholesky(A)
plt.matshow(L)
B = linalg.inv(A)
plt.matshow(B, cmap='gray')
b = np.mat(np.ones((m), 'float'))
y = B*b.T
plt.figure(4)
x = np.linspace(h, m*h, m)
plt.plot(x, y, '--',x, x*(1-x)/2,':')
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.show()
```

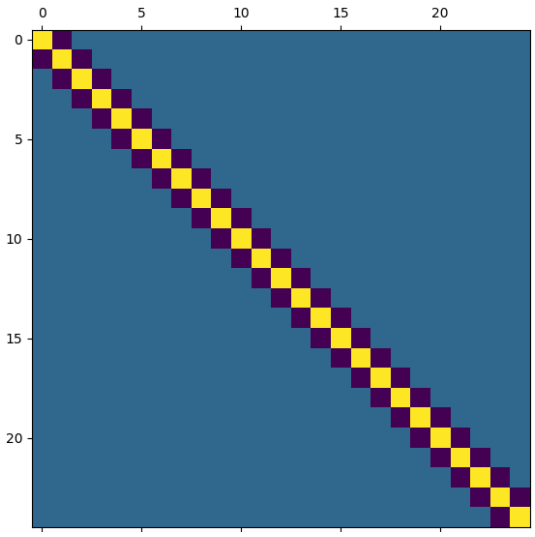
Natija:

det: 1.4576080109926803e+72

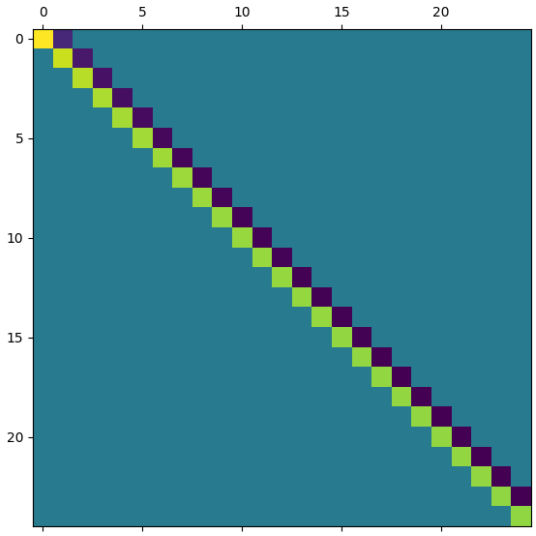
norm-1: 2703.9999999999995

norm-2: 2694.1423977805694

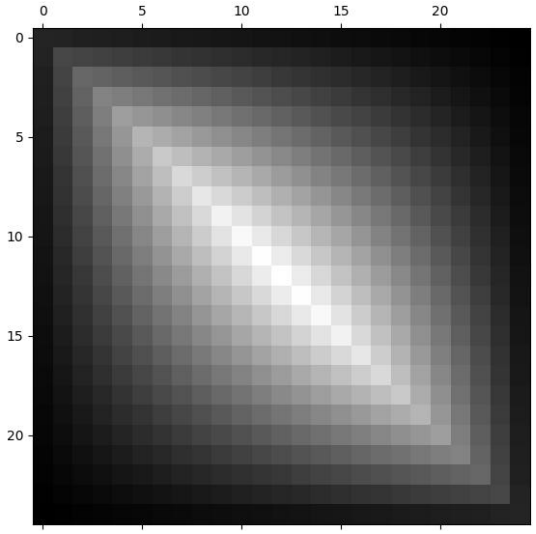
norm-inf: 2703.9999999999995



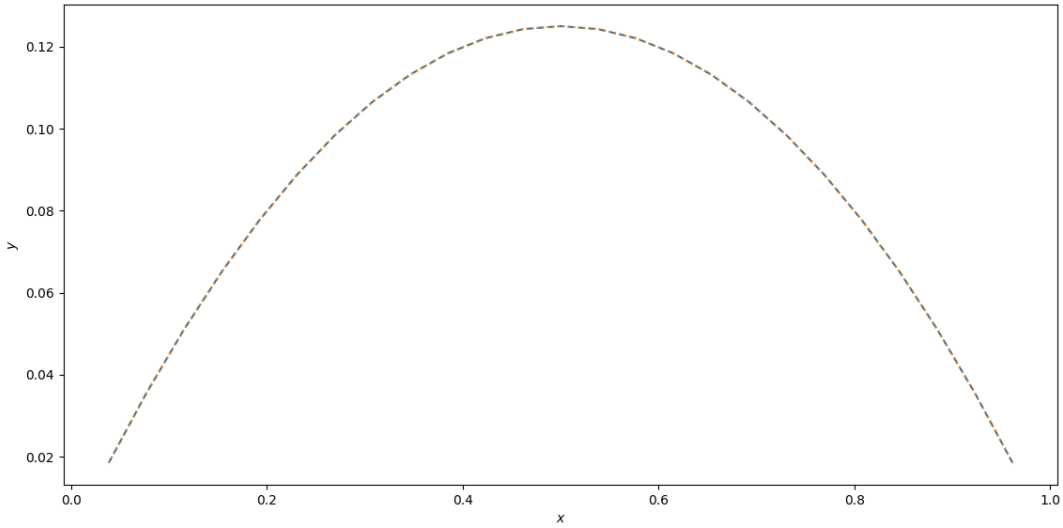
7.1-rasm. *A* matritsaning chizmasi



7.2-rasm. *U* matritsa (*A=UU**)



7.3-rasm. $B=A^{-1}$ matritsa chizmasi



7.4-rasm. Tenglama yechimi

7.1-rasm *A* matritsaning chizmasini koʻrsatadi (Matplotlib paketidagi matshow () funksiyasi ishlatilgan). A matritsa simmetrik va tridiagonaldir.

Bizning misolimizda tenglamalar sistemasini yechish teskari matritsadan foydalanish bilan amalga oshirilgan. Ikkinchi imkoniyat solve() funksiyasidan foydalanish bilan bogʻliq ($y = B * b.T$ oʻrniga y=linalg.solve(A,b.T) ishlatiladi). sol_banded() funksiyasi lentasimon matritsali tenglamalar sistemasini yechish uchun moʻljallangan.

A matritsaning simmetrik va musbat aniqlanganligini hisobga olib, uni A=LL*=U*U yoyilma shaklda taqdim etildi (matritsaning chiziqli omillari 7.2-rasmda kuzatilgan). Teskari matritsa 7.3-rasmda keltirilgan. Aniq yechim bilan taqqoslash 7.4-rasmda koʻrsatilgan - ikkinchi hosilaning ayirma operatori ikkinchi darajali koʻphadlar uchun aniq berilgan.

Koʻproq umumiy matritsalar uchun linalg moduli LU yoyilma funksiyalarini ta'minlaydi. Bundan tashqari, har qanday xosmas matritsalar bilan ishlash uchun *A=PLU* umumiy koʻpaytuvchilarga ajratish - lu() funksiyasi yordamida amalga oshiriladi. Bu yerda L - diagonal hadlari birga teng boʻlgan quyi uchburchakli matritsa, U - yuqori uchburchakli matritsa, P - almashtirish matritsasi. Boshqa LU yoyilma xususiyatlari lu_factor(), lu_solve() funksiyalarida amalga oshiriladi.

Boshqa turdagi matritsa yoyilmalari uchun ham funksiyalar mavjud:

• Xoleskiy yoyilma – cholesky(), cholesky_banded(), cho_factor(), cho_solve() funksiyalari;

• QR yoyilma – qr() funksiyasi;

• singulyar (SVD) yoyilma –svd(), svdvals(), diagsvd() funksiyalari;

• Shura yoyilmasi – schur(), rsf2csf() funksiyalari.

Eng kichik kvadratlar usulini silliqlashtiruvchi koʻphadning koeffitsientlarini tanlash masalasida qoʻllanilishini koʻramiz. $y_i, i = 0,1,\ldots,m$ funksiyaning qiymatlari $x_i = ih, i = 0,1,\ldots,m$ nuqtada ma'lum boʻlsin. Biz ushbu ma'lumotlarni $n < m$ boʻlganda $z(a;x) = a_0 + a_1 x + \ldots + a_n x^n$ polinom yordamida yaqinlashamiz. $a_k, k = 0,1,\ldots,n$ koeffitsientlar minimal shart

$$J(a) = \sum_{i=1}^{m} (z(a; x_i) - y_i)^2$$

dan topiladi

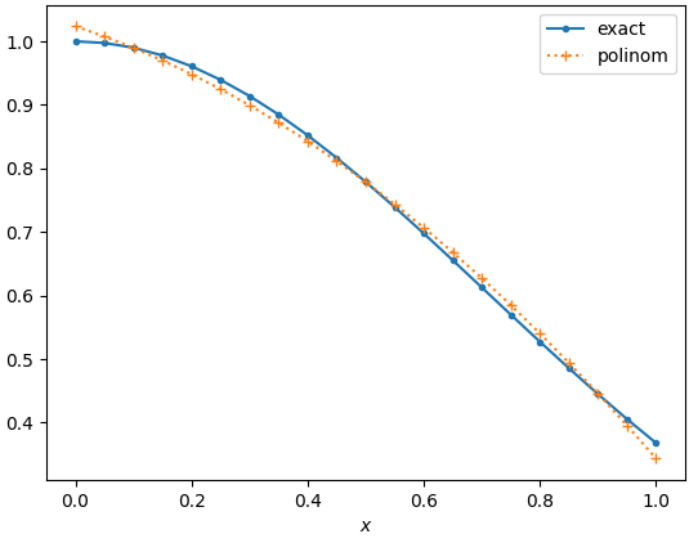
Matritsali ko‘rinishda quyidagicha yoziladi $J(a) = \|Aa - b\|^2$.

Eng kichik kvadratlar usulini lstsq() funksiyasi yordamida hisoblash quyidagi dastur orqali berilgan.

```
import numpy as np
from scipy import linalg
import matplotlib.pyplot as plt
m = 20
n = 3
h = 1./m
A = np.mat(np.zeros((m+1, n+1), 'float'))
x = np.zeros((m+1), 'float')
y = np.zeros((m+1), 'float')
y1 = np.zeros((m+1), 'float')
for i in range(0, m+1):
    xx = i*h
    x[i] = xx
    y[i] = np.exp(-xx**2)
    for k in range(0, n):
        A[i,k] = xx**k
a,resid, rank, sigma = linalg.lstsq(A,y)
for i in range(0, m+1):
    xx = i*h
    sum = 0.
    for k in range(0, n):
        sum+=a[k]*xx**k
    y1[i] = sum
plt.plot(x, y,'.-',    label='exact')
```

```
plt.plot(x, y1, '+:', label='polinom')
plt.xlabel('$x$')
plt.legend()
plt.show()
```

Natija:



7.5-rasm. Polinomni approksimatsiya qilish

7.5-rasm e^{-x^2} funksiyani uchinchi darajali ko‘phad bilan $0 \le x \le 1$ kesmada approksimatsiyalash aniqligini ko‘rsatadi. lstsq() funksiyasi o‘rniga siz chiziqli tenglamalar sistemasining psevdo-yechimini topish imkonini beruvchi pinv() yoki pinv2() dan foydalanishingiz mumkin.

Quyida matritsalarning xos qiymatlarini topishga misol qaraladi. Davriy to‘r funksiyalari uchun markaziy hosila operatorining xos qiymatlari qidiriladi:

$$Ay = \frac{y_{i+1} - y_{i-1}}{2h}, \ i = 0,1,\ldots,m-1,$$

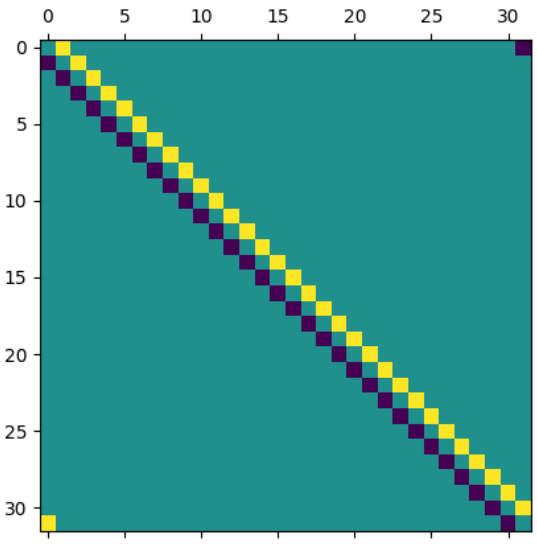
bunda $y_{i+m} = y_i, h = 1/m$. Egri-simmetrik A matritsaning xos qiymatlari faqat sof mavhumdir:

$$\lambda_k = \frac{\sqrt{-1}}{h}\sin(2\pi kh), \ k = 0,1,\ldots,m-1.$$

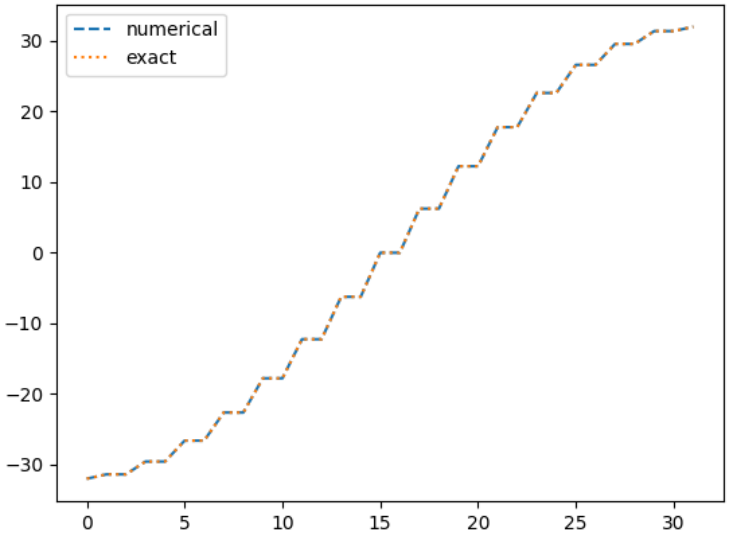
Quyida linalg modulining eigvals() funksiyasidan foydalangan holda ushbu matritsaning xos qiymatlarini hisoblash dasturi keltirilgan (7.6-rasmda matritsaning

chizmasi ko‘rsatilgan). 7.7-rasmda aniq va taqribiy hisoblangan xos qiymatlarni taqqoslash berilgan.

```
import numpy as np
from scipy import linalg
import matplotlib.pyplot as plt
m = 32; h = 1./m;
A = np.mat(np.zeros((m, m), 'float'))
for i in range(1, m):
    A[i-1,i] = 1. / (2*h)
    A[i,i-1] = -1. / (2*h)
A[0,m-1] = -1. / (2*h)
A[m-1,0] = 1. / (2*h)
plt.matshow(A); plt.figure(2)
la = linalg.eigvals(A)
laI = np.sort(la.imag)
ii = np.linspace(0, m-1, m)
plt.plot(ii, laI,'--', label='numerical')
laE= np.sort(1./h * np.sin(2*np.pi*ii*h))
plt.plot(ii, laE,':', label='exact')
plt.legend(loc=0)
plt.show()
```



7.6-rasm. *A* matritsaning chizmasi



7.7-rasm. Aniq va taqribiy hisoblangan xos qiymatlarni taqqoslash

eig() funksiyasi xos qiymatlarni ham, xos funksiyalarni ham topish uchun ishlatiladi. Pythonda matritsalarning maxsus sinflari uchun spektral masalalarni yechish algoritmlari ham amalga oshirilgan: kompleks Ermit va haqiqiy simmetrik matritsalar uchun eigvalsh() va eigh() funksiyalari, lentasimon matritsalar uchun eigvals_banded() va eig_banded() funksiyalari.

linalg moduli matritsa funksiyalari bilan ishlashni qo‘llab-quvvatlaydi. Matritsali eksponentani

$$e^A = \sum_{k=0}^{\infty} \frac{1}{k!} A^k$$

hisoblash uchun uchta funksiyadan foydalaniladi: expm() - Pade approksimatsiyalash, expm2() - spektral yoyilma, expm3() - Teylor qatori. Logarifmni hisoblash uchun logm() funksiyasi mavjud. Matritsali sinus, kosinus va tangens (sinm(), cosm() va tanm() funksiyalari), giperbolik sinus, kosinus va tangens (sinhm(), coshm() va tanhm() funksiyalari) ham amalga oshiriladi. Matritsa sgn (sign) funksiyasini hisoblash uchun signm() dan foydalaniladi, kvadrat ildiz sqrtm(). Har qanday Python funksiyasi bilan bog‘langan matritsa funksiyalarini (funm() yordamida) hisoblash imkoniyatiga ham ta’kidlash kerak.

Quyida Koshi masalasini hal qilishda matritsali eksponentadan foydalanishga misol keltirilgan:

$$\frac{dy}{dt} + Ay = 0,\ 0 < t \le T,$$

$$y(0) = u.$$

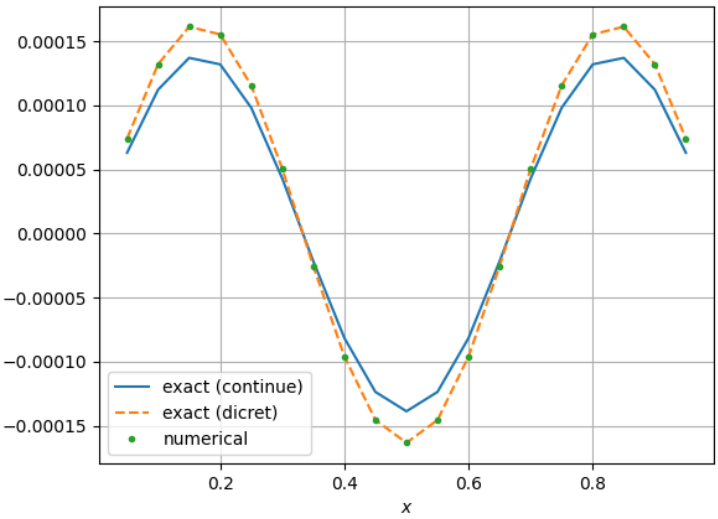
Doimiy (t dan bog‘liq bo‘lmagan) matritsa *A* uchun yechim quyidagicha ifodalanadi

$$y(t) = e^{-At}u.$$

Dasturda A matritsa [0,1] interval chegaralarida nolga teng bo‘ladigan to‘r funksiyalarining ikkinchi ayirmali hosilasiga mos keladi. Diskret va uzluksiz masalaning aniq yechimi $\varphi_k = \sqrt{2}\sin(\pi kx), k = 1,2,\ldots$ funksiya bo‘yicha yoyilma shaklida ifodalanadi.

```
import numpy as np
from scipy import linalg
import matplotlib.pyplot as plt
n = 3
m = 19
t = 0.1
h = 1. / (m+1)
A = np.mat(np.zeros((m, m), 'float'))
for i in range(1, m):
    A[i-1,i] = - 1. / h**2
    A[i,i] = 2. / h**2
    A[i,i-1] = - 1. / h**2
A[0,0] = 2. / h**2
A[m-1,m-1] = 2. / h**2
x = np.zeros((m), 'float')
y = np.zeros((m), 'float')
for i in range(0, m):
```

```
    xx = (i+1)*h
    x[i] = xx
    y[i] = np.sin(np.pi*n*xx)
y0 = np.exp(-(np.pi*n)**2 * t) * y
y1 = np.exp(-4/h**2*(np.sin(np.pi*n*h/2))**2 * t) * y
B = np.mat(linalg.expm(-A*t))
y2 = B*np.mat(y).T
plt.plot(x, y0,'-', label='exact (continue)')
plt.plot(x, y1, '--', label='exact (dicret)')
plt.plot(x, y2, '.', label='numerical')
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.legend(loc=0)
plt.grid(True)
plt.show()
```



7.8-rasm. $t = 0.1$ da Koshi masalasi yechimi

Bu yerda

$$\lambda_k = \pi^2 k^2, \quad \tilde{\lambda}_k = \frac{4}{h^2}\sin^2\left(\frac{\pi k h}{2}\right)$$

mos ravishda uzluksiz va to‘r masalalarining xos qiymatlari.

expm() funksiyasi o‘rniga expm2() funksiyasidan foydalanish bir xil natijalar beradi, lekin expm3() bir xil emas (chunki matritsali eksponentani hisoblashda Teylor seriyasining yomon yaqinlashuvi).

Chiziqli algebra hisoblash algoritmlarini to‘liq qo‘llab-quvvatlash uchun chiziqli tenglamalar sistemasini yechishning iterativ usullarini qo‘shish mumkin. Tegishli funksiyalarni siyrak matritsani yechish moduli sparse.linalgda topish mumkin (ular avval linalg modulida edi). Oldindan ko‘rib chiqilgan masala uchun qo‘shma gradient usulidan foydalanishga misol (7.4-rasmga qarang) quyidagi dastur tomonidan berilgan.

```
import numpy as np
from scipy.sparse import linalg
import matplotlib.pyplot as plt
m = 25
h = 1. / (m+1)
A = np.mat(np.zeros((m, m), 'float'))
for i in range(1, m):
    A[i-1,i] = - 1. / h**2
    A[i,i] = 2. / h**2
    A[i,i-1] = - 1. / h**2
A[0,0] = 2. / h**2
A[m-1,m-1] = 2. / h**2
b =np.mat(np.ones((m), 'float')).T
help(linalg.cg)
y, info = linalg.cg(A, b, tol=1.e-06)
x = np.linspace(h, m*h, m)
plt.plot(x, y, '--', x, x*(1-x)/2, ':')
plt.xlabel('$x$')
plt.ylabel('$y$')
plt.show()
```

sparse.linalg moduli qo‘shma gradient usulidan (cg() funksiyasi) tashqari, chiziqli tenglamalar sistemasini yechish uchun boshqa iterativ usullarni o‘z ichiga oladi:

• cgs() – kvadratik qo‘shma gradient usuli;

• qmr() – kvaziminimal xatoliklar usuli;

• gmres() – minimal farqlarning umumlashtirilgan usuli;

• bicg() – biqo‘shma gradient usuli;

• bicgstab() – stabilizatsiya bilan biqo‘shma gradient usuli.

Bu iterativ usullar ham to‘liq, ham siyrak matritsalar bilan bog‘liq masalalarni yechishda qo‘llaniladi.

7.4-§. SciPy paketining interpolyatsiya moduli

SciPy paketining interpolyatsiya moduli bir o‘zgaruvchili to‘r funksiyalarining polinomlari va splaynlari orqali interpolyatsiya va tekislashning standart masalalarini hal qilish uchun vositalarni taqdim etadi. Shuningdek, oddiy to‘r yoki tekislikning ixtiyoriy nuqtalarida berilgan ikki o‘lchamli ma’lumotlarini interpolyatsiya qilish funksiyalari ham mavjud.

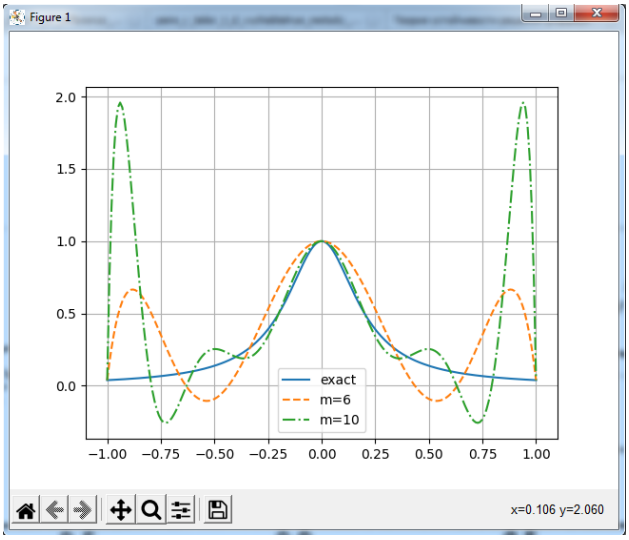
$x_0, x_1,\ldots,x_m$ tugunlarda $y = y(x)$ funksiya $y_0, y_1,\ldots,y_m$ berilgan qiymatlarni qabul qilsin. Polinom interpolyatsiyasi bilan polinom $p(x) = a_0 x^m + a_1 x^{m-1} + \ldots + a_m$ tuziladi, bunda $p(x_k) = y_k$, $k = 0,1,\ldots,m$. Interpolyatsiya qiluvchi polinomni yaratish uchun lagrange() funksiyasidan foydalanish mumkin.

barycentric_interpolate() va krogh_interpolate() funksiyalarida interpolyatsiya qiluvchi polinom qiymatlarini hisoblash uchun yanada barqaror algoritmlar qo‘llaniladi.

Runge $y = (1 + 25x^2)^{-1}$ funksiyasi uchun interpolyatsiya qiluvchi ko‘phadni qurishga misol keltiramiz (7.9-rasm).

```
import numpy as np
from scipy import interpolate
import matplotlib.pyplot as plt
```

```
x = np.linspace(-1, 1., 200)
plt.plot(x, 1./(1 + 25*x*x), label='exact')
x1 = np.linspace(-1, 1., 7); y1 = 1./(1 + 25*x1*x1)
p1 = interpolate.lagrange(x1, y1)
plt.plot(x, p1(x), '--',label='m=6')
x2 = np.linspace(-1, 1., 11); y2 = 1./(1 + 25*x2*x2)
yb = interpolate.barycentric_interpolate(x2, y2, x)
plt.plot(x, yb, '-.',label='m=10')
plt.grid(True)
plt.legend(loc=0)
plt.show()
```



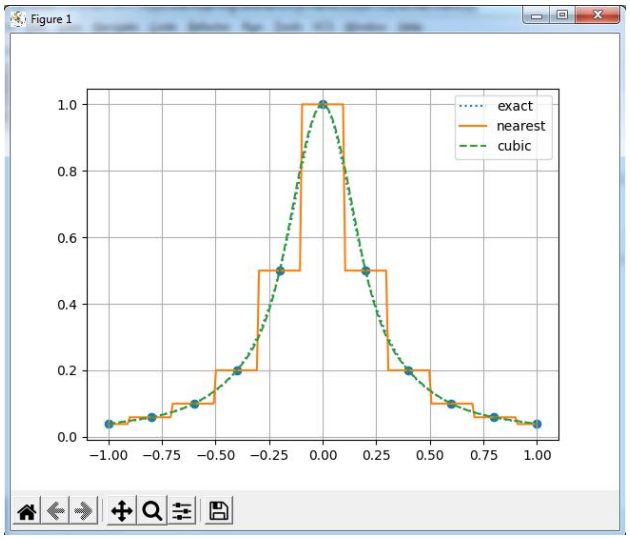
7.9-rasm. Ko‘phad bo‘yicha yaqinlashish

Ilmiy hisoblash amaliyotida qismlarga bo‘lingan ko‘phadli yaqinlashuvlarga (splaynlar) ko‘proq e’tibor beriladi. Bir o‘lchovli interpolyatsiyaning asosiy funksiyasi interp1d() bo‘lib, uning birinchi argumenti tugunlar massivi, ikkinchisi qiymatlar massividir. kind parametri bo‘lakli-polinom interpolyatsiyasi turini belgilaydi. kind = ‘linear’ (standart qiymat) bilan bo‘lakli-chiziqli interpolyatsiya qo‘llaniladi (chiziqli splaynlar), kind = ‘cubic’ - kubik splaynlar, kind = ‘nearest’, ‘zero‘ qiymatlari bo‘lakli-doimiy yaqinlashishlar bilan bog‘langan. Bu parametrning butun qiymat turi tugunlar orasidagi interpolyatsiya qiluvchi polinom darajasiga mos

keladi. Interpolyatsiya tugunlarida hosilalarni belgilashda piecewise_polynomial_interpolate() funksiyasi yordamida bo‘lakli- polinom yaqinlashuvi tuziladi.

Runge funksiyasini interpolyatsiya qilishda interp1d() funksiyasidan foydalanishga misol keltiramiz (7.10-rasmga qarang).

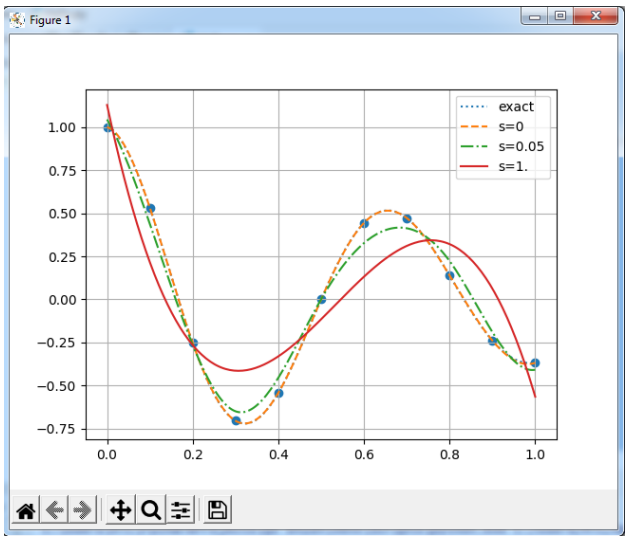
```
import numpy as np
from scipy import interpolate
import matplotlib.pyplot as plt
x = np.linspace(-1, 1., 200)
plt.plot(x, 1./(1 + 25*x*x),':', label='exact')
x1 = np.linspace(-1, 1., 11); y1 = 1./(1 + 25*x1*x1)
plt.scatter(x1,y1)
f1 = interpolate.interp1d(x1,y1,kind='nearest')
plt.plot(x, f1(x), '-',label='nearest')
f2 = interpolate.interp1d(x1,y1,kind=3)
plt.plot(x, f2(x), '--',label='cubic')
plt.grid(True)
plt.legend(loc=0)
plt.show()
```



7.10 Bo‘lakli-polinomli yaqinlashish

Interpolyatsiya moduli splaynlar bilan ishlash uchun koʻplab vositalarni oʻz ichiga oladi. Jumladan, faqat berilgan nuqtadagi shplaynning qiymatini hisoblabgina qolmay, balki splaynning koeffitsientlarini ham topish, splaynning hosilasini hisoblash, berilgan oraliqdagi integralni, shplaynning ildizlarini topish mumkin. Misol sifatida, tekislovchi B-splayn qurilishini qaraymiz (7.11-rasm). B-splayn splrep() funksiyasi yordamida quriladi va splayn qiymatlarini hisoblash uchun splev() ishlatiladi. s parametri silliqlash parametridir.

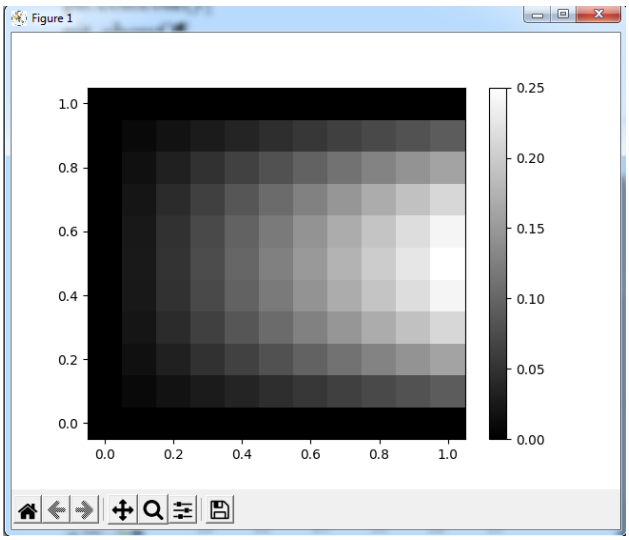
```
import numpy as np
from scipy import interpolate
import matplotlib.pyplot as plt
x = np.linspace(0., 1., 200)
plt.plot(x, np.cos(3*np.pi*x)*np.exp(-x),':', label='exact')
x1 = np.linspace(0., 1., 11)
y1 = np.cos(3*np.pi*x1)*np.exp(-x1)
plt.scatter(x1, y1)
tck1 = interpolate.splrep(x1, y1, s=0)
s1 = interpolate.splev(x, tck1, der=0)
plt.plot(x, s1, '--', label='s=0')
tck2 = interpolate.splrep(x1, y1, s=0.05)
s2 = interpolate.splev(x, tck2, der=0)
plt.plot(x, s2,'-.',label='s=0.05')
tck3 = interpolate.splrep(x1, y1, s=1.)
s3 = interpolate.splev(x, tck3, der=0)
plt.plot(x, s3, '-', label='s=1.')
plt.grid(True)
plt.legend(loc=0)
plt.show()
```



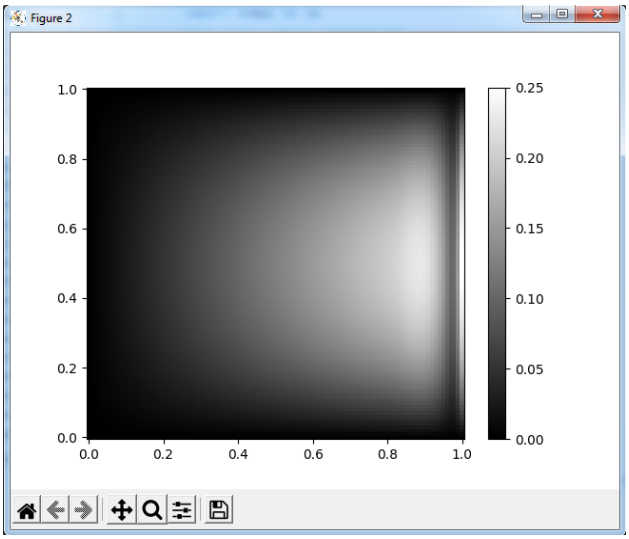
7.11 Splaynlar bilan tekislash

Ikki o‘lchovli to‘rda ikki o‘lchovli ma’lumotlarni interpolyatsiya qilish uchun siz bisplrep() (B-splaynni topish) va bisplev() (ikki o‘lchovli B-splayn qiymatlarini hisoblash) funksiyalaridan foydalanishingiz mumkin. 7.12-rasmda 11x11 o‘lchamdagi to‘rning haqiqiy funksiyasini ko‘rsatadi, interpolyatsiya natijasi (101 x 101 panjara) shaklda ko‘rsatilgan.

```
import numpy as np
from scipy import interpolate
import matplotlib.pyplot as plt
x, y = np.mgrid[0:1:11j, 0:1:11j]
z = y*(1-y)*x; plt.figure(1)
plt.pcolor(x, y, z, cmap='gray')
plt.colorbar()
x1, y1 = np.mgrid[0:1:101j, 0:1:101j]
tck = interpolate.bisplrep(x, y, z, s=0)
z1 = interpolate.bisplev(x1[:,0], y1 [0,:], tck)
plt.figure(2)
plt.pcolor(x1, y1, z1, cmap='gray')
plt.colorbar()
plt.show()
```



7.12-rasm. To‘r funksiya

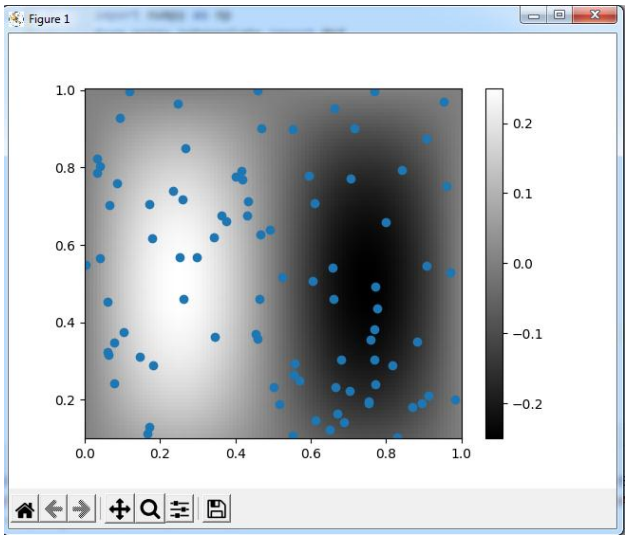


7.13-rasm. Splayn interpolyatsiya

Tartibsiz to‘rdagi ma’lumotlar bilan ishlash uchun siz radial asosli funksiyalarga asoslangan interpolyatsiya yoki tekislashni qo‘llashingiz mumkin. Quyida Rbf() funksiyasidan foydalanishga misol keltirilgan (7.14-rasmga qarang).

```
import numpy as np
from scipy.interpolate import Rbf
import matplotlib.pyplot as plt
x = np.random.rand(100); y = np.random.rand(100)
z = np.sin(2*np.pi*x)*y*(1-y)
rbf=Rbf(x,y,z,epsilon=2)
x1=np.linspace(0.,1.,101); y1=np.linspace(0.,1.,101)
```

```
X1, Y1 = np.meshgrid(x1, y1)
Z1 = rbf(X1, Y1)
plt.pcolor(X1, Y1, Z1, cmap='gray')
plt.colorbar(); plt.scatter(x,y)
plt.xlim(0,1); plt.ylim(0.1)
plt.show()
```



7.14-rasm. Tartibsiz to‘rda interpolyatsiya

Bir o‘zgaruvchili funksiya qiymatlarini uning qiymatlaridan ba’zi nuqtalarda taqriban tiklash masalalari ko‘rib chiqiladi. Bir o‘lchovli interpolyatsiya uchun an’anaviy yondashuv interpolyatsiya nuqtalarida berilgan qiymatlarni qabul qiladigan algebraik polinomlarni qurish bilan bog‘liq. Bo‘lakli-silliq polinomlardan foydalanish yanada istiqbolli hisoblanadi.

7.5-§. SciPy paketining optimallashtirish moduli

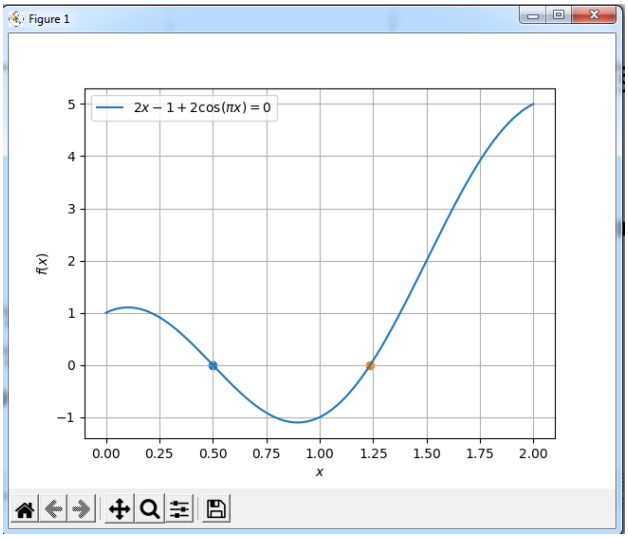
Hisoblash matematikasining asosiy masalalari orasida ko‘p o‘zgaruvchili funksiyalarni minimallashtirish masalalarini (optimallashtirish masalalarini) qayd etish mumkin. Minimalni qidirish ko‘pincha qo‘shimcha cheklovlar ostida amalga oshiriladi - shartli optimallashtirish. Bunday masalalarni sonli yechish uchun iterativ usullar qo‘llaniladi. Cheklovlar bilan bog‘liq masalalarda jarima funksiyalari usullari qo‘llaniladi. Ko‘rib chiqilayotgan sinfning eng oddiy masalasi bir o‘lchamli funksiyaning minimalini topishdir.

SciPy to‘plamining optimallashtirish moduli chiziqli bo‘lmagan tenglamalar va ularning sistemalarini yechish, chegaralangan va chegaralanmagan bir o‘lchamli funksiyalarni va ko‘plab o‘zgaruvchili funksiyalarni minimallashtirish vositalarini taqdim etadi. $f(x) = 0$ nochiziqli tenglamaning yechimini topish masalasini taqribiy yechish uchun turli usullardan foydalanish mumkin. $f(x)$ funksiya ishorasini o‘zgartiradigan ($f(a) f(b) < 0$) [a, b] oraliqdagi ildizini topish uchun Brent usulini (brentq() funksiyasi) qo‘llash mumkin. Quyida [0,1] va [1,2] oraliqlarda $2x - 1 + 2\cos(\pi x) = 0$ tenglamaning yechimi izlanadi.

```
import numpy as np
import matplotlib.pyplot as pit
import scipy.optimize as optimize
def f(x):
    return 2*x - 1 + 2*np.cos(np.pi*x)
x = np.linspace(0., 2., 201)
y = f(x)
pit.plot(x, y, label='$2x-1+2\\cos(\\pi x) = 0$')
x0 = optimize.brentq(f, 0., 1.)
x1 = optimize.brentq(f, 1., 2.)
print('x:', x0, ', ', x1)
pit.scatter(x0, 0)
pit.scatter(x1, 0)
pit.legend(loc=0)
pit.xlabel('$x$')
pit.ylabel('$f(x)$')
pit.grid(True)
pit.show()
```

Natija:

x: 0.5 , 1.2364844482415167



7.15-rasm. Chiziqsiz tenglama yechimi

Chiziqli bo‘lmagan tenglamalar sistemasini yechish uchun fsolve() funksiyasidan foydalaniladi, bunda sistema tenglamalarini va yechimga dastlabki yaqinlashishni ko‘rsatishish kerak. Yuqoridagi $2x-1+2\cos(\pi x)=0$ tenglama

$$1-2x_0-x_1=0,$$
$$x_1-2\cos(\pi x_0)=0$$

sistema sifatida qabul qilinadi.

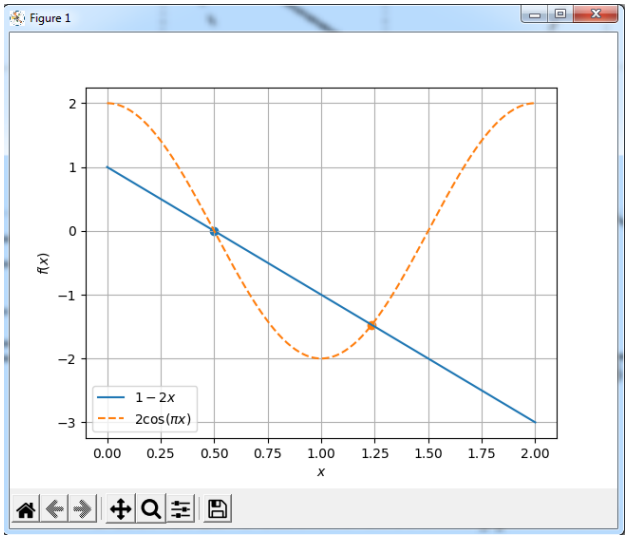
Keyingi dasturda tenglamalar sistemasi yechimi fsolve() funksiyasi yordamida olinadi.

```
import numpy as np
import matplotlib.pyplot as pit
import scipy.optimize as optimize
def F(x):
    out = [1 - 2*x[0] - x[1]]
    out.append(x[1] - 2*np.cos(np.pi*x[0]))
    return out
x = np.linspace(0., 2., 201)
y1 = 1 - 2*x; y2 = 2*np.cos(np.pi*x)
pit.plot(x, y1,      '-',label='$1 - 2x$')
pit.plot(x, y2, '--', label='$2\\cos(\\pi x)$')
```

```
x0 = optimize.fsolve(F, [0.5, 0.])
x1 = optimize.fsolve(F, [1.5, 0.])
print('x:', x0, ',', x1)
pit.scatter(x0[0], x0[1]); pit.scatter(x1[0], x1[1])
pit.legend(loc=0); pit.xlabel('$x$')
pit.ylabel('$f(x)$'); pit.grid(True)
pit.show()
```

Natija:

x: [0.5 0.] , [1.23648445 -1.4729689]



7.16-rasm. Tenglamalar sistemasini yechish

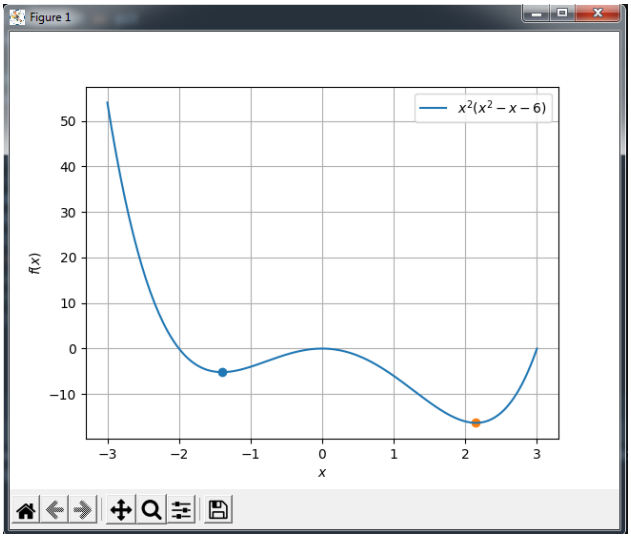
Bir o‘lchamli funksiyalarni minimallashtirish uchun turli xil algoritmlar qo‘llaniladi: masalan, golden() funksiyasida oltin kesim usuli, brent() da Brent (teskari parabolik interpolyatsiya) usuli. Quyidagi dastur $f(x) = x^2(x^2 - x - 6)$ funksiyani [-3,0] va [0,3] oraliqlarda minimal qiymatlarini topish uchun fminbound() funksiyasidan foydalanadi.

```
import numpy as np
import matplotlib.pyplot as pit
import scipy.optimize as optimize
```

```
def f(x):
    return x*x*(x*x - x - 6)
x = np.linspace(-3., 3., 201); y = f(x)
pit.plot(x, y, label='$x^2(x^2 - x -6)$')
x0 = optimize.fminbound(f, -3., 0.)
x1 = optimize.fminbound(f, 0., 3.)
print('x:' , x0, ',', x1); pit.scatter(x0, f(x0))
pit.scatter(x1, f(x1)); pit.legend(loc=0)
pit.xlabel('$x$'); pit.ylabel('$f(x)$'); pit.grid(True);
pit.show()
```

Natija:

x: -1.3971801357198503 , 2.1471818673842034



7.17-rasm. Bir o‘lchamli funksiyani minimallashtirish

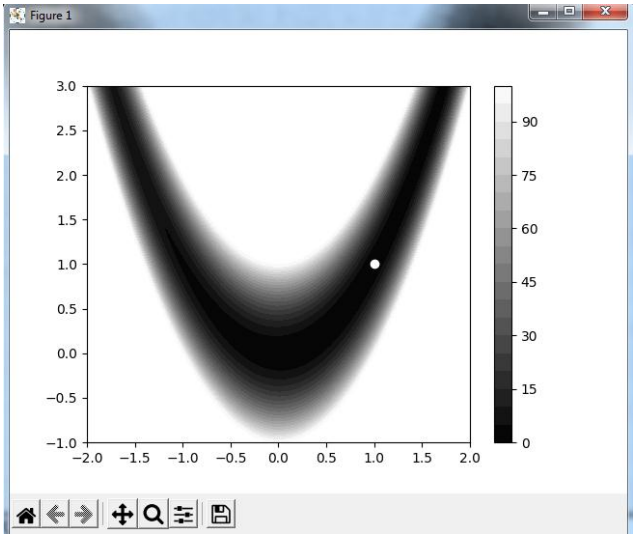
Ko‘p o‘zgaruvchili funksiyalarni minimallashtirishda odatda kvazi-Nyuton usullari qo‘llaniladi. Bunday holda, biz minimallashtiruvchi funksiyaning hosilalarini aniq ko‘rsatishimiz mumkin yoki ular hisob-kitoblar jarayonida taxminan hisoblab chiqiladi. Quyida Simpleks usulini amalga oshiradigan fmin() funksiyasidan foydalanish qaraladi. $F(x) = 100(x_1 - x_0^2)^2 + (1 - x_0)^2$ (ikki o‘lchamli Rosenbrok funksiyasi) funksiyaning minimalini qidiramiz.

```
import numpy as np
```

```
import matplotlib.pyplot as pit
import scipy.optimize as optimize
def f(x):
    return 100*(x[1] -x[0]**2)**2 + (1 - x[0])**2
x = np.linspace(-2., 2., 101); y = np.linspace(-1., 3., 101)
X, Y = np.meshgrid(x, y); z = f([X, Y])
v = np.linspace(0, 100, 21)
pit.contourf(x, y, z, v, cmap=pit.cm.gray)
pit.colorbar(); x0 = [0, 0]
xmin = optimize.fmin(f, x0)
pit.scatter(xmin[0], xmin[1], c='w')
print('xmin:', xmin)
pit.show()
```

Natija:

```
Optimization terminated successfully.
        Current function value: 0.000000
        Iterations: 79
        Function evaluations: 146
xmin: [1.00000439 1.00001064]
```



7.18-rasm. Rosenbrok funksiyasini minimallashtirish

Hisoblash amaliyotida kvadratlar yig‘indisi bo‘lgan funksiyani minimallashtirishga alohida e’tibor beriladi. $y = f(p;x)$ funksiyaning $p_k = k = 0,1,\ldots,n$ parametrlarga chiziqli bo‘lmagan funksional bog‘liqligi $x_i, \widetilde{y}_i, i = 0,1,\ldots,m$ $(m \geq n)$ taqribiy ma’lumotlar to‘plamida tiklanganidagi odatiy holat. Buning uchun

$$J(p) = \sum \left(f(p;x_i) - \widetilde{y}_i \right)^2$$

funksional minimallashtiriladi. Buning uchun leastsq() funksiyasidan foydalanish mumkin.

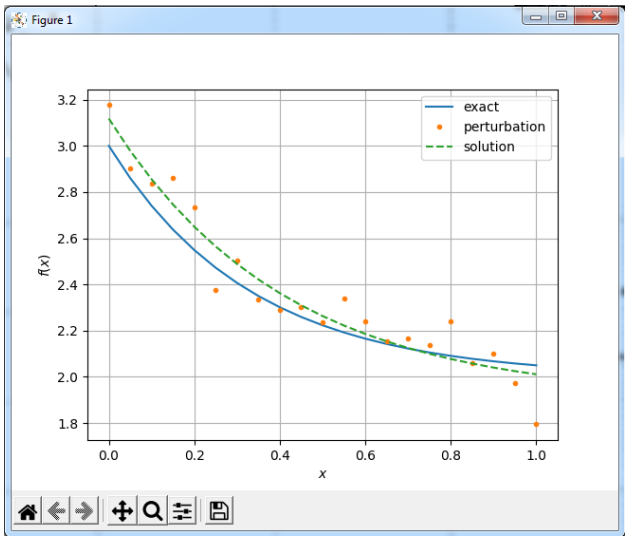
Dastur shovqinli ma’lumotlardan foydalangan holda $y = a + be^{-cx}$ ko‘rinishidagi *a*, *b*, *c* parametrlarini qidiradi. Hisoblash natijalari quyidagi rasmda ko‘rsatilgan.

```
import numpy as np
import matplotlib.pyplot as pit
import scipy.optimize as optimize
def res(p, y, x):
    a, b, c= p
    err=y-a-b*np.exp(-c*x)
    return err
x = np.linspace(0., 1., 21)
a,b, c=2, 1, 3
y =a +b*np.exp(-c*x)
np.random.seed(0)
y0 = y +0.1 * np.random.randn(len(x))
pit.plot(x, y,'-',     label='exact')
pit.plot(x, y0, '.',label='perturbation')
p0 = [0, 0, 0]
plsq = optimize.leastsq(res, p0,args=(y0, x))
```

```
p1 = plsq[0]
print('’a, b, c:' , p1)
y1 = p1[0] + p1[1] *np.exp(-p1[2] *x)
pit.plot(x, y1,       '--',label='solution')
pit.legend(loc=0)
pit.xlabel('$x$')
pit.ylabel('$f(x)$')
pit.grid(True)
pit.show()
```

Natija:

’a, b, c: [1.9051214 1.21298056 2.43970228]



7.19-rasm. Parametrli identifikatsiya

Umuman olganda, biz chegaralangan minimallashtirish masalalarini ko‘rib chiqamiz, ular odatda tengsizliklar sifatida shakllantiriladi. Eng oddiy holatda, mustaqil o‘zgaruvchilar diapazonlarida chegaralar mavjud. Shartli minimallashtirish uchun fmin_cobyla() funksiyasidan foydalanish mumkin.

7.6-§. SciPy paketining integrate moduli

SciPy to‘plamining integrate moduli hisoblash matematikasining ikkita sinfiga kiruvchi masalalarni hal qilish uchun dasturiy ta’minotni taqdim etadi.

Quyida uzluksiz va to‘r funksiyalarning integrallarini hisoblash masalalari ko‘rib chiqiladi. Ikkinchi sinf masalalari oddiy differensial tenglamalar sistemasi uchun Koshi masalasini sonli yechish bilan bog‘liq.

SciPy paketining integrate modulidagi sonli kvadratur formulalalarini ikki turga bo‘lish mumkin: integralni Python funksiyasi sifatida qabul qiluvchi formulalar va berilgan nuqtalarda integral ifodalari bilan massivlarni oladigan formulalar. Birinchi turdagi funksiyalar Gauss kvadraturasidan (quad(), quadrature(), fixed_quad()) foydalansa, ikkinchi turdagi funksiyalarda Nyuton-Kotes usullari (trapz(), simps() va romb()) qo‘llaniladi.

Quadrature() funksiyasi Pythonda amalga oshirilgan adaptiv Gauss kvadraturasi qism dasturidir. Quadrature() o‘rnatilgan tartibli Gauss kvadraturasi uchun, talab qilinadigan aniqlikka erishilgunga qadar tartibni oshirgan holda, fixed_quad() funksiyasini qayta-qayta chaqiradi. quad() funksiyasi FORTRAN kutubxonasi QUADPACK dasturlari uchun qobiq bo‘lib, u tezlik bo‘yicha yuqori unumdorlikka va ko‘proq xususiyatlarga ega (masalan, cheksiz integral chegaralarini qo‘llab-quvvatlash). Shuning uchun odatda quad funksiyasidan foydalanish afzalroq va quyida biz ushbu kvadratur funksiyasidan foydalanamiz. Biroq, bu funksiyalarning barchasi yaqin qiymatlarniberadi va ko‘pincha bir-biri bilan almashtirilishi mumkin. Ular birinchi argument sifatida integral hisoblanadigan funksiyani oladilar, ikkinchi va uchinchi argumentlar esa quyi va yuqori integrallash chegaralari hisoblanadi. Aniq misol sifatida integral $\int_{-1}^{1} e^{-x^2} dx$ ni sonli hisoblashni ko‘rib chiqamiz. Ushbu integralni SciPy ning quad() funksiyasidan foydalangan holda hisoblash uchun biz avval integral uchun funksiyani aniqlaymiz va keyin quad() funksiyani chaqiramiz:

```
import numpy as np
import matplotlib.pyplot as pit
import scipy.integrate as integrate
def f(x):
    return np.exp(-x**2)
```

```
val, err = integrate.quad(f, -1, 1)
print(val,err)
val
1.493648265624854
Err
1.6582826951881447e −14
```

quad() funksiya integralning sonli natijasi o‘z ichiga olgan val-kortejni va err-absolyut xatolikni qaytaradi. Absolyut va nisbiy xatoliklar uchun mos ravishda ixtiyoriy epsabs va epsrel kalit so‘z argumentlari yordamida o‘rnatilishi mumkin. Agar *f* funksiyasi bir nechta o‘zgaruvchini qabul qilsa, quad() tartib funksiyani birinchi argumenti ustida birlashtiradi. Biz ixtiyoriy ravishda qo‘shimcha argumentlar qiymatlarini argument args kalit so‘zi orqali quad() funksiyaga integral funksiyasiga o‘tkazish orqali belgilashimiz mumkin.

Misol uchun, agar biz a=1, b=2 va c=3 parametrlarining o‘ziga xos qiymatlari uchun $\int_{-1}^{1} ae^{-(x-b)^2/c^2}dx$ ni hisoblamoqchi bo‘lsak, biz barcha qo‘shimcha argumentlarni oladigan integral uchun funksiyani belgilashimiz va keyin a, b va c qiymatlarini args=(1, 2, 3) orqali belgilab, quad() funksiyasiga o‘tkazishimiz mumkin:

```
def f(x, a, b, c):
        return a * np.exp(-((x - b)/c)**2)
val, err = integrate.quad(f, -1, 1, args=(1, 2, 3))
val
1.2763068351022229
err
1.4169852348169507e −14
```

Biz integrallamoqchi bo‘lgan o‘zgaruvchi birinchi argument bo‘lmagan funksiyalar bilan ishlaganda, lambda funksiyadan foydalanib, argumentlarni

o‘zgartirishimiz mumkin. Misol uchun, agar biz $J_0(x)$ integrali birinchi turdagi nolinchi tartibli Bessel funksiyasi bo‘lgan $\int_0^5 J_0(x)dx$ integralini hisoblamoqchi bo‘lsak, integralni hisoblash uchun scipy.special modulidan jv() funksiyasidan foydalanish qulay bo‘ladi. jv() funksiyasi v va x argumentlarini oladi va haqiqiy qiymatli v tartibi uchun birinchi turdagi Bessel funksiyasi va x da hisoblanadi.

jv() funksiyasidan quad() uchun integral sifatida foydalanish uchun jv() argumentlarini o‘zgartirishimiz kerak. Lambda funksiyasi bilan biz buni quyidagi tarzda qilishimiz mumkin:

```
from scipy.special import jv
f = lambda x: jv(0, x)
val, err = integrate.quad(f, 0, 5)
val
0.7153119177847678
err
2.47260738289741e −14
```

Ushbu texnika yordamida biz har qanday funksiyaning argumentlarini ixtiyoriy o‘zgartirishimiz va har doim integral o‘zgaruvchisi birinchi argument bo‘lgan funksiyani olishimiz mumkin, shuning uchun funksiya quad() funksiyasi uchun integrand sifatida ishlatilishi mumkin.

quad() tartibi cheksiz integral chegaralarini qo‘llab-quvvatlaydi. Cheksiz integral chegaralarini ko‘rsatish uchun biz NumPy'da np.inf sifatida qulay bo‘lgan float('inf') cheksizning suzuvchi nuqtali tasviridan foydalanamiz. Masalan, $\int_{-\infty}^{\infty} e^{-x^2} dx$ integral qaralsin. Buni quad() yordamida hisoblash quyidagicha bo‘ladi

```
f = lambda x: np.exp(-x**2)
val, err = integrate.quad(f, -np.inf, np.inf)
val
1.7724538509055159
```

```
err
1.4202636780944923e −08
```

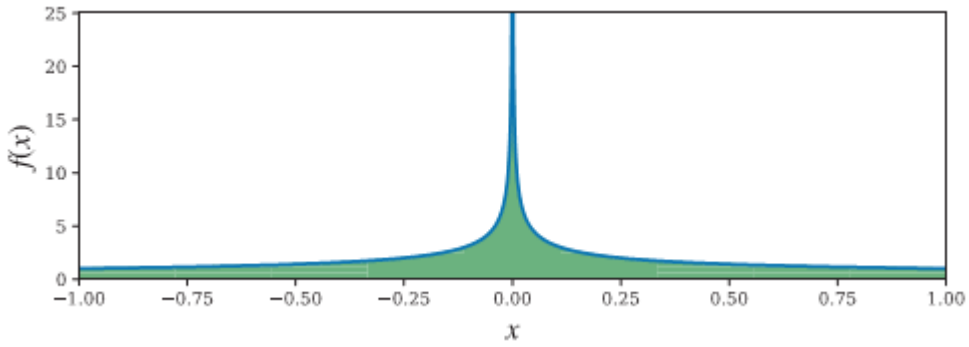
Biroq, quadrature() va fixed_quad() funksiyalari faqat cheklangan integral chegaralarini qo‘llab-quvvatlaydi.

Bir oz qo‘shimcha yo‘l-yo‘riq bilan, quad() funksiyasi integrallanadigan singulyativ ko‘plab integrallarni ham hisoblay oladi. Misol uchun $\int_{-1}^{1} \frac{1}{\sqrt{x}} dx$ integralni hisoblash. Integral x=0 da uzilishga ega, lekin integralning qiymati farqlanmaydi va uning qiymati 4 ga teng. Quad() yordamida bu integralni oddiygina hisoblashga urinish uzilishga ega integral tufayli muvaffaqiyatsiz bo‘lishi mumkin:

```
f = lambda x: 1/np.sqrt(abs(x))
a, b = -1, 1
print(integrate.quad(f, a, b))
(inf, inf)
```

Bunday holatlarda, 7.20-rasmda ko‘rsatilganidek, integralning o‘zini qanday tutishi haqida tushunchaga ega bo‘lish uchun uning grafigini tuzish foydali bo‘lishi mumkin.

```
fig, ax = plt.subplots(figsize=(8, 3))
    x = np.linspace(a, b, 10000)
    ax.plot(x, f(x), lw=2)
    ax.fill_between(x, f(x), color='green', alpha=0.5)
    ax.set_xlabel("$x$", fontsize=18)
    ax.set_ylabel("$f(x)$", fontsize=18)
    ax.set_ylim(0, 25)
    ax.set_xlim(-1, 1)
```



7.20-rasm. quad() funksiyasi yordamida hisoblanishi mumkin bo‘lgan chekli integralli (yashil/soyali maydon) uzilishga ega integralga misol.

Bunday holda, integralni hisoblash muvaffaqiyatsiz tugaydi, chunki integral Gauss kvadraturasi qoidasidagi namunaviy nuqtalardan birida (o‘rta nuqtada) aniq uzilishga ega. Point kalit so‘z argumentlari yordamida oldini olish kerak bo‘lgan nuqtalar ro‘yxatini belgilash orqali quad() tartibini boshqarishimiz mumkin va joriy misolda point=[0] dan foydalanish quad() funksiyasiga integralni to‘g‘ri hisoblash imkonini beradi:

integrate.quad(f, a, b, points=[0])

(4.0,5.684341886080802e −14)

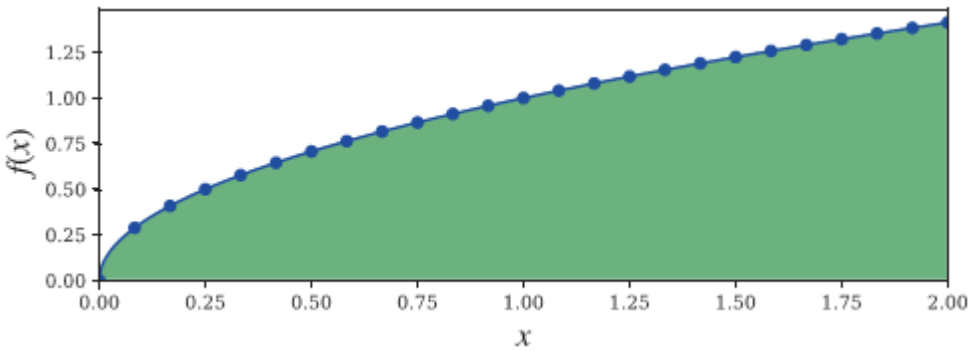
Jadvalli integral. quad() funksiyasi integralni Python funksiyasi yordamida aniqlanganda integrallarni hisoblash uchun mos keladi, uni ixtiyoriy nuqtalarda hisoblashi mumkin (bu maxsus kvadratur qoidasi bilan aniqlanadi). Biroq, ko‘p hollarda biz faqat oldindan belgilangan nuqtalarda ko‘rsatilgan integralga ega bo‘lishimiz mumkin, masalan, [a, b] integrallash oralig‘idagi teng masofada joylashgan nuqtalarda. Bunday turdagi holat, masalan, integral muayyan integrallash tartibi bilan boshqarilmaydigan tajribalar yoki kuzatishlar natijasida olinganda yuzaga kelishi mumkin. Bunday holda Nyuton-Kotes kvadraturasidan foydalanishimiz mumkin.

SciPy integratsiya modulida trapz() va simps() funksiyalarida trapetsiya qoidasi va Simpson qoidasi amalga oshiriladi. Bu funksiyalar birinchi argument sifatida integrallash oralig‘idagi nuqtalar to‘plamidagi integrand qiymatlari bo‘lgan y massivini oladi va ixtiyoriy ravishda ikkinchi argument sifatida berilgan

nuqtalarning x qiymatlarini yoki muqobil ravishda nuqtalar orasidagi dx oralig‘ini belgilaydigan turli oraliqli x massivini oladi. E’tibor bering, berilgan nuqtalar bir xil bo‘lishi shart emas, lekin ular oldindan ma'lum bo‘lishi kerak.

Tanlangan qiymatlar bilan berilgan funksiyaning integralini qanday hisoblashni ko‘rish uchun 7.21-rasmda ko‘rsatilganidek, [0, 2] integrallash oralig‘ida integralning 25 ta nuqtasini olib, $\int_0^2 \sqrt{x}dx$ integralni hisoblaymiz:

```
f = lambda x: np.sqrt(x)
a, b = 0, 2
x = np.linspace(a, b, 25)
y = f(x)
fig, ax = plt.subplots(figsize=(8, 3))
ax.plot(x, y, 'bo‘)
xx = np.linspace(a, b, 500)
ax.plot(xx, f(xx), 'b-')
ax.fill_between(xx, f(xx), color='green', alpha=0.5)
ax.set_xlabel(r"$x$", fontsize=18)
ax.set_ylabel(r"$f(x)$", fontsize=18)
```



7.21-rasm. Integrallash nuqtalari bilan belgilangan jadvalli qiymatlar sifatida berilgan. Integral soyali maydonga mos keladi.

Integralni hisoblash uchun biz x va y massivlarni trapz() yoki simps() usullariga o‘tkazishimiz mumkin. Esda tutingki, y massivi birinchi argument sifatida uzatilishi kerak:

```
val_trapz = integrate.trapz(y, x)
val_trapz
1.88082171605
val_simps = integrate.simps(y, x)
val_simps
1.88366510245
```

trapz() va simps() funksiyalari hech qanday xato natijalarini taqdim etmaydi, ammo bu misol uchun biz integralni analitik tarzda hisoblashimiz va ikkita usul bilan hisoblangan raqamli natijalar bilan solishtirishimiz mumkin:

```
val_exact = 2.0/3.0 * (b-a)**(3.0/2.0)
val_exact
1.8856180831641267
val_exact - val_trapz
0.00479636711328
val_exact - val_simps
0.00195298071541
```

Bizda mavjud bo‘lgan barcha ma’lumotlar berilgan nuqtalar massivi bo‘lganligi sababli, biz trapz() yoki simps() dan ham aniqroq yechimlarni hisoblashni so‘ray olmaymiz. Aniqlikni oshirishning yagona variantlari berilgan nuqtalar sonini ko‘paytirish (agar asosiy funksiya noma'lum bo‘lsa, bu qiyin bo‘lishi mumkin) yoki ehtimol yuqori tartibli usuldan foydalanishdir.

Integrate moduli, shuningdek, romb() funksiyasi bilan Romberg usulini amalga oshirishni ta'minlaydi. Romberg usuli Nyuton-Kotes usulidir, lekin trapetsiya usulining yaqinlashuvini tezlashtirish uchun Richardson ekstrapolyatsiyasidan foydalanadi; ammo bu usul namuna nuqtalarining bir tekis

joylashishini, shuningdek, 2^{n+1} namuna nuqtalarining mavjudligini talab qiladi, bunda *n* butun sondir. Trapz() va simps() usullari singari, romb() birinchi argument sifatida integral namunalari bo‘lgan massivni oladi, lekin ikkinchi argument (agar berilgan bo‘lsa) dx namunaviy nuqta oralig‘i bo‘lishi kerak:

```
x = np.linspace(a, b, 1 + 2**6)
len(x)
65
y = f(x)
dx = x[1] - x[0]
val_exact - integrate.romb(y, dx=dx)
0.000378798422913
```

Biz bu yerda muhokama qilgan SciPy integrallash funksiyalari orasida simps(), ehtimol, eng foydalisi hisoblanadi, chunki u foydalanish qulayligi (namuna nuqtalarida hech qanday cheklovlar yo‘q) va nisbatan yaxshi aniqlik o‘rtasida yaxshi muvozanatni ta'minlaydi.

Uzluksiz argumentli funksiyaning aniq integrallarini hisoblash uchun asosan quad() funksiyasi foydalaniladi. Misol sifatida (7.22-rasmga qarang) berilgan $f(x) = e^{-4x}\sin(4\pi x)$ funksiya uchun $j(x) = \int_0^x f(t)dt$ integralni [0,1] kesmada teng oraliqli to‘r bo‘yicha hisoblanadi.

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integrate
def f(x):
    return np.exp(-4*x)*np.sin(4*np.pi*x)
m = 101
x = np.zeros((m), 'float')
y = np.zeros((m), 'float')
y1 = np.zeros((m), 'float')
```

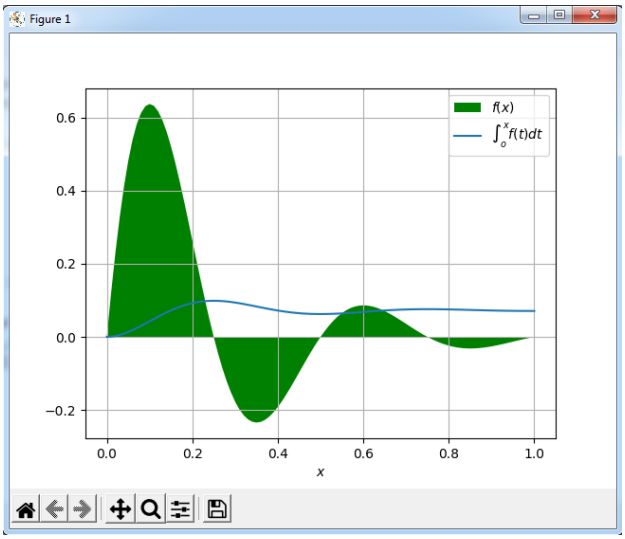
```
for i in range(0, m):
    x[i] = i/(m-1.)
    y[i] = integrate.quad(f, 0, x[i]) [0]
    y1[i] = f(x[i])
plt .fill (x, f(x), 'g‘, label='$f(x)$')
plt.plot(x, y, label='$\\int_o^x f(t) dt$')
I0 = integrate.quad(f, 0, 1)[0]
I1 = integrate.trapz(y1, x)
I2 = integrate.simps(y1, x)
print('int(quad): ', I0); print('int(trapz): ', I1)
print('int(simps)', I2)
plt.legend(loc=0); plt.xlabel('$x$')
plt.grid(True)
plt.show()
```

Natija:

int(quad): 0.07093294896505287

int(trapz): 0.0708301283865725

int(simps) 0.0709330243013272



7.22-rasm. $f(x) = e^{-4x} \sin(4\pi x)$ funksiyani integrallash

To‘r funksiyalari uchun trapetsiya va Simpsonning kvadratur formulalari qo‘llaniladi (mos ravishda trapz() va simps() funksiyalari). Bitta o‘zgaruvchi uchun integrallash protseduralariga asoslanib, ikki va uch o‘zgaruvchiga (dblquad() va tplquad()) integrallash funksiyalari ham amalga oshiriladi.

SciPy ning integrate moduli ikkita ODE hisoblovchi interfeysini taqdim etadi: integrate.odeint va integrate.ode. odeint funksiyasi ODEPACK-dan LSODA hisoblovchi interfeysi bo‘lib, u qat’iy bo‘lmagan masalalar uchun Adams usuli va qat’iy masalalar uchun BDF usuli o‘rtasida avtomatik ravishda almashinadi. Bundan farqli o‘laroq, integrate.ode sinfi bir qator turli xil hisoblovchilarga ob’ektga yo‘naltirilgan interfeysni taqdim etadi: VODE va ZVODE hisoblovchilar (ZVODE murakkab qiymatli funksiyalar uchun VODE variantidir), LSODA hal qiluvchi va dopri5, dop853 to‘rtinchi va sakkizinchi tartibli Dormand-Prens usullari (ya’ni, Runge-Kutta usullari turlari) moslashuvchan bosqichli o‘lchamlarga ega. integrate.ode tomonidan taqdim etilgan ob’ektga yo‘naltirilgan interfeys yanada moslashuvchan bo‘lsa-da, odeint funksiyasi ko‘p hollarda sodda va foydalanish uchun qulayroqdir.

Oddiy differensial tenglamalar sistemalari uchun Koshi masalasini yechish imkoniyati alohida e’tiborga loyiqdir. Integrate moduli bunday vazifalar uchun ode() va odeint() funksiyalaridan foydalanadi. Ular qat’iy bo‘lmagan va qat’iy ODE tizimlari uchun masalalarni hal qilish uchun mo‘ljallangan.

Koshi masalasini qaraymiz (Lotka-Volterra modeli)

$$\frac{dy_0}{dt} = y_0 - y_0 y_1, \quad \frac{dy_1}{dt} = -y_1 + y_0 y_1, \quad 0 < t \le T, \quad y_0(0) = y_0^0, \quad y_1(0) = y_1^0.$$

Tenglamalar sistemasi vektor shaklida yozilsa

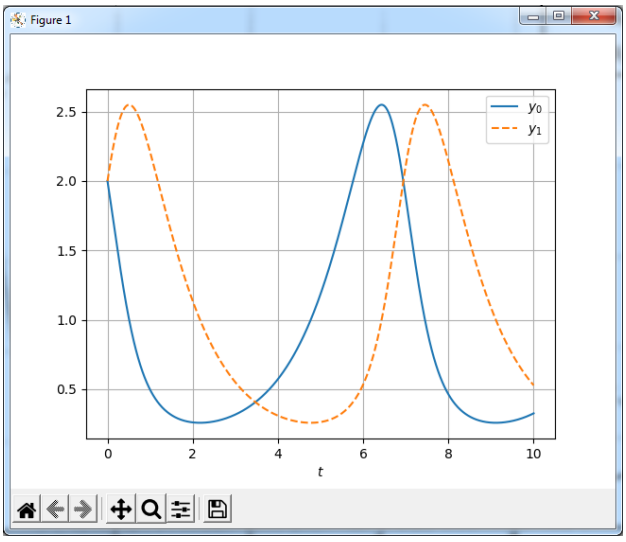
$$\frac{dY}{dt} = F(Y,t), \quad 0 < t \le T$$

bunda

$$Y = \begin{bmatrix} y_0 \\ y_1 \end{bmatrix}, \quad F(Y,t) = \begin{bmatrix} y_0 - y_0 y_1 \\ -y_1 + y_0 y_1 \end{bmatrix}.$$

Dasturda yechim [0,10] oraliqda odeint() funksiyasi yordamida olinadi (7.23-rasmga qarang).

```
import numpy as np
import matplotlib.pyplot as plt
import scipy.integrate as integrate
def f(y, t):
    return [y [0] - y[0]*y[1],
            -y[1] + y[0] *y[1]]
t = np.linspace(0, 10,1000); y0 = [2, 2]
Y = integrate.odeint(f, y0, t); r0, r1 = Y.T
plt.plot(t, r0,'-',label='$y_0$')
plt.plot(t, r1,'--',label='$y_1$')
plt.legend(loc=0); plt.xlabel('$t$')
plt.grid(True)
plt.show()
```



7.23-rasm. ODT sistemasi uchun Koshi masalasi

ODE sistemalari uchun Koshi masalasini yechish uchun odeint() funksiyasidan foydalanganda Yakobiy matritsasi (Yakobiy, tizimning o‘ng tomonlarining hosilalari)ni analitik ko‘rsatish orqali hisoblash samaradorligini oshirish mumkin.

FOYDALANILGAN ADABIYOTLAR RO`YXATI

1. Вабищевич П.Н. Численные методы: Вычислительный практикум. — М.: Книжный дом «ЛИБРОКОМ», 2010. — 320 с.
2. Robert Johansson. Numerical Python: Scientifi Computing and Data Science Applications with Numpy, SciPy and Matplotlib. Apress.
3. Бизли Д.М. Язык программирования Python. Справочник. — Киев.: ДиаСофт, 2000.
4. Каханер Д., Моулер К., Нэш С. Численные методы и программное обеспечение. — М.: Мир, 2001.
5. Лейнингем И. Освой самостоятельно Python за 24 часа. — Киев: Вильямс, 2001.
6. Лутц Марк. Изучаем Python. — СПб: Символ-Плюс, 2009.
7. Лутц Марк. Программирование на Python. — СПб: Символ-Плюс, 2002.
8. Ляшко И. И., Макаров В. Л., Скоробогатько А. А. Методы вычислений. — Киев: Высшая школа, 1977.
9. Марчук Г. И. Методы вычислительной математики. — М.: Лань, 2009.
10. Devert, A. (2014). matplotlib Plotting Cookbook. Mumbai: Packt.
11. J. Steele, N. I. (2010). Beautiful Visualization. Sebastopol: O’Reilly.
12. McKinney, W. (2013). Python for Data Analysis. Sebastopol: O’Reilly.
13. Milovanovi, I. (2013). Python Data Visualization Cookbook. Mumbai: Packt.
14. Tosi, S. (2009). Matplotlib for Python Developers. Mumbai: Packt.
15. Yau, N. (2011). Visualize this. Indianapolis: Wiley.
16. www.python.org
17. www.matplotlib.org

MUNDARIJA

QAYDLAR UCHUN

A. X. Toyirov

Pythonda matematik hisoblashlarni dasturlash
(uslubiy qo‘llanma)

Muharrir: Bekpo‘lat Umurqulov
Mas’ul muharrir: Sevara Shoimova
Texnik muharrir: Feruzjon Shaimov
Musahhih: Ruzixol Shaimova
Sahifalovchi: Muhriddin Yuldashov

Nashriyot litsenziyasi № 013751
Bosishga 06.12.2021-yilda ruxsat etildi. Bichimi 60x84 $1/16$. Ofset qog‘ozi. Times New Roman garniturasi. Shartli bosma tobog‘i 7. Nashr bosma tabog‘i 6.51. 112-sonli shartnoma. 201-sonli buyurtma. Adadi 100 nusxada. Erkin narxda.

NASHRIYOT VA BOSMAXONA MANZILI:
“Bekshox print servis” nashriyoti. Termiz shahri,
“Navbog‘” ko‘chasi, 20-uy.
Tel: (+998 91) 972-77-03