Ташкентский институт Текстильной и Легкой Промышленности

Факультет: Автоматизация и Управление

Курсовая работа

по информационной технологии

Тема: Python

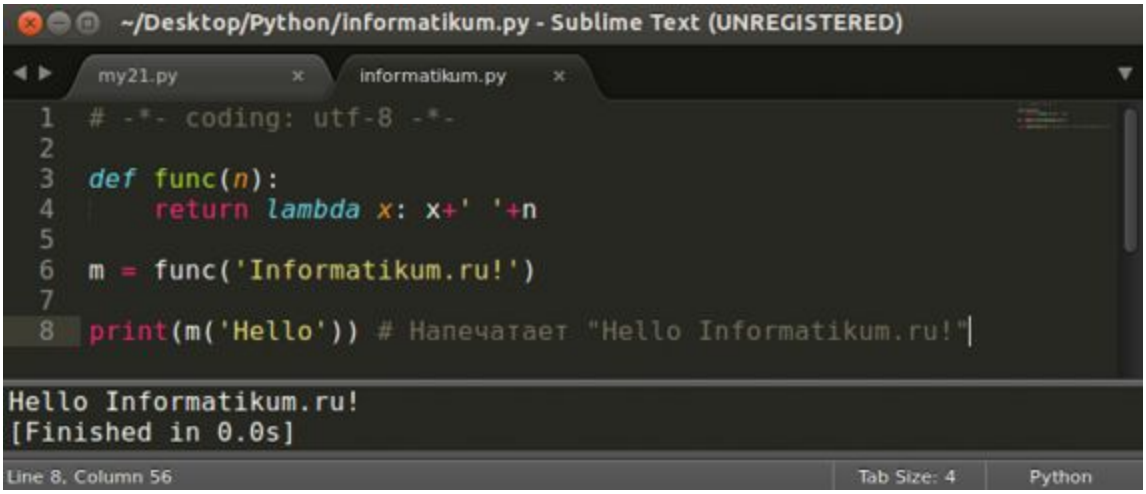
Выполнил: Олимов Азизбек 19р-15

Принял: Абдурахмонов А.А.

Тошкент 2016

Основные свойства

Python не требует явного объявления переменных, является регистро-зависим (переменная var не эквивалентна переменной Var или VAR — это три разные переменные) объектно-ориентированным языком.



Синтаксис

Во первых стоит отметить интересную особенность Python. Он не содержит операторных скобок (begin..end в pascal или {..}в Си), вместо этого блоки выделяются отступами: пробелами или табуляцией, а вход в блок из операторов осуществляется двоеточием. Однострочные комментарии начинаются со знака фунта «#», многострочные — начинаются и заканчиваются тремя двойными кавычками «"""».

Чтобы присвоить значение пременной используется знак «=», а для сравнения —

«==». Для увеличения значения переменной, или добавления к строке используется оператор «+=», а для уменьшения — «-=». Все эти операции могут взаимодействовать с большинством типов, в том числе со строками. Например

<*blockquote*>>>> *myvar* = <*font color="#ff4500"*>*3*</*font*>

>>> *myvar* += <*font color="#ff4500"*>*2*</*font*>

>>> *myvar* -= <*font color="#ff4500"*>*1*</*font*>

<*font color="#808080"*>"""Это многострочный комментарий

Строки заключенные в три двойные кавычки игнорируются"""

>>> *mystring* = <*font color="#483d8b"*>*"Hello"*</*font*>

>>> *mystring += " world."*

>>> *print mystring*

Hello world.

 # Следующая строка меняет

значения переменных местами. (Всего одна строка!)

>>> *myvar, mystring = mystring, myvar</blockquote>*

Структуры данных

Python содержит такие структуры данных как списки (lists), кортежи (tuples) и словари (dictionaries). Списки — похожи на одномерные массивы (но вы можете использовать Список включающий списки — многомерный массив), кортежи — неизменяемые списки, словари — тоже списки, но индексы могут быть любого типа, а не только числовыми. "Массивы" в Python могут содержать данные любого типа, то есть в одном массиве может могут находиться числовые, строковые и другие типы данных. Массивы начинаются с индекса 0, а последний элемент можно получить по индексу -1 Вы можете присваивать переменным функции и использовать их соответственно.

*<blockquote>>>> sample = [1, ["another", "list"], ("a", "tuple")] #*Список состоит из целого числа, другого списка и кортежа

>>> *mylist = ["List item 1", 2, 3.14] #*Этот список содержит строку, целое и дробное число

>>> *mylist[0] = "List item 1 again" #*Изменяем первый (нулевой) элемент листа mylist

>>> *mylist[-1] = 3.14 #*Изменяем последний элемент листа

>>> *mydict = {"Key 1": "Value 1", 2: 3, "pi": 3.<font*

*color="#ff4500">14} #*Создаем словарь, с числовыми и целочисленным индексами

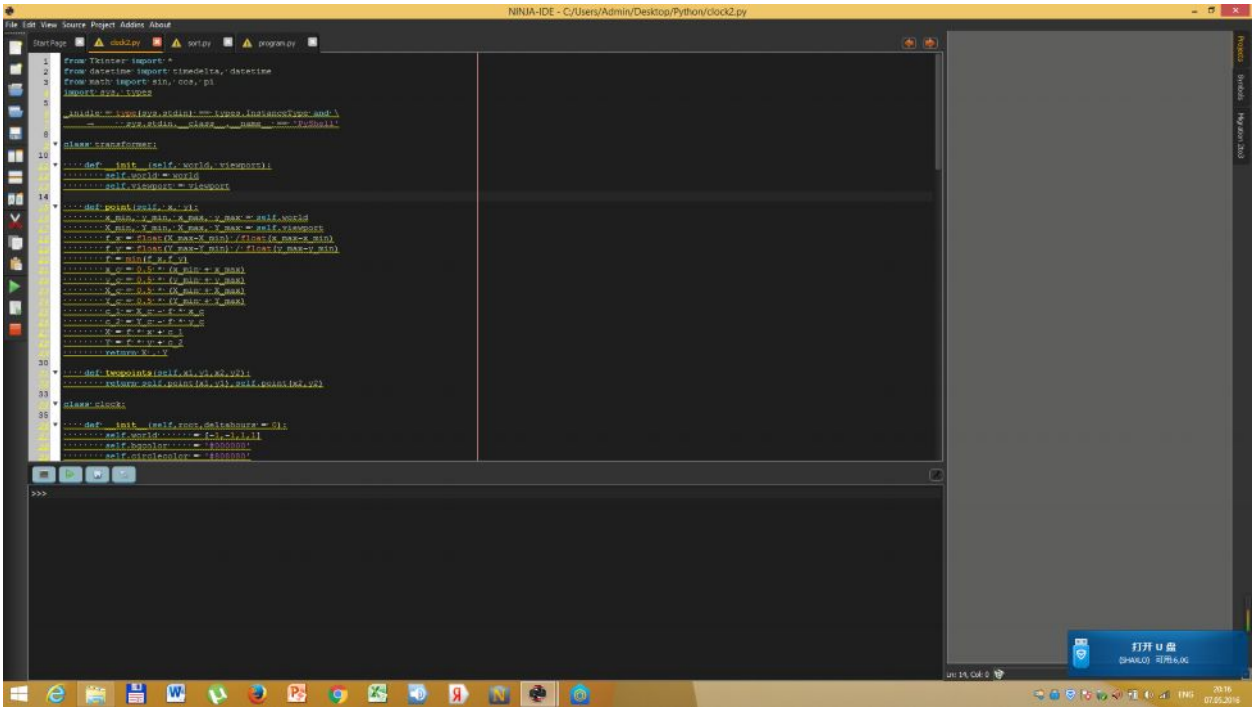
*>>> mydict["pi"] = 3.15 #*Изменяем элемент словаря под индексом "pi".

*>>> mytuple = (1, 2, 3) #*Задаем кортеж

*>>> myfunction = len #*Python позволяет таким образом объявлять синонимы функции

>>> print myfunction(list)

3</blockquote>



Вы можете использовать часть массива, задавая первый и последний индекс через двоеточие «:». В таком случае вы получите часть массива, от первого индекса до второго не включительно. Если не указан первый элемент, то отсчет начинается с начала массива, а если не указан последний — то масив считывается до последнего элемента. Отрицательные значения определяют положение элемента с конца. Например:

<blockquote>>>> mylist = ["List item 1", 2, 3.14]

>>> print mylist[:] #Считываются все элементы массива

['List item 1', 2, 3.1400000000000001]

>>> print mylist[0:2] #Считываются нулевой и первый элемент массива.

['List item 1', 2]

>>> print mylist[-3:-1] #Считываются элементы от нулевого (-3) до второго (-1) (не включительно)

['List item 1', 2]

>>> print mylist[1:] #Считываются элементы от первого, до последнего

[2, 3.14]</blockquote>

Строки

Строки в Python обособляются кавычками двойными «"» или одинарными «'». Внутри двойных ковычек могут присутствовать одинарные или наоборот. К примеру строка «Он сказал 'привет'!» будет выведена на экран как «Он сказал 'привет'!». Если нужно использовать строку из несколько строчек, то эту строку надо начинать и заканчивать тремя двойными кавычками «"""». Вы можете подставить в шаблон строки элементы из кортежа или словаря. Знак процента «%» между строкой и кортежем, заменяет в строке символы «%s» на элемент кортежа. Словари позволяют вставлять в строку элемент под заданным индексом. Для этого надо использовать в строке конструкцию «%(индекс)s». В этом случае вместо «%(индекс)s» будет подставлено значение словаря под заданным индексом.

<blockquote>>>>print "Name: %s\nNumber: %s\nString: %s" % (myclass.name, 3, 3 * "-")

Name: Poromenos

Number: 3

String: ---

strString = """Этот текст расположен

на нескольких строках"""

>>> print "This %(verb)s a %(noun)s." % {"noun": "test", "verb": "is"}

This is a test.</blockquote>

Операторы

Операторы while, if, for составляют операторы перемещения. Здесь нет аналога оператора select, так что придется обходиться if. В операторе for происходит сравнение переменной и списка. Чтобы получить список цифр до числа <number> — используйте функцию range(<number>). Вот пример использования операторов

<blockquote>rangelist = range(10) #Получаем список из десяти цифр (от 0 до 9)

>>> print rangelist

[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

for number in rangelist: #Пока переменная number (которая каждый раз увеличивается на единицу) входит в список...

Проверяем входит ли переменная

numbers в кортеж чисел (3, 4, 7, 9)

if number in (3, 4, 7, 9): #Если переменная number входит в кортеж (3, 4, 7, 9)...

Операция «break» обеспечивает

выход из цикла в любой момент

break

else:

«continue» осуществляет "прокрутку"

цикла. Здесь это не требуется, так как после этой операции

в любом случае программа переходит опять к обработке цикла

continue

else:

«else» указывать необязательно. Условие выполняется

если цикл не был прерван при помощи «break».

pass # Ничего не делать

if rangelist[1] == 2:

print "The second item (lists are 0-based) is 2"

elif rangelist[1] == 3:

print "The second item (lists are 0-based) is 3"

else:

print "Dunno"

while rangelist[1] == 1:

pass</blockquote>

Функции

Для объявления функции служит ключевое слово «def». Аргументы функции задаются в скобках после названия функции. Можно задавать необязательные аргументы, присваивая им значение по умолчанию. Функции могут возвращать кортежи, в таком случае надо писать возвращаемые значения через запятую. Ключевое слово «lambda» служит для объявления элементарных функций .

<blockquote># arg2 и arg3 - необязательные аргументы, принимают значение объявленное по умолчни,

 # если не задать им другое значение при вызове функци.

def myfunction(arg1, arg2 = 100, arg3 = "test"):

return arg3, arg2, arg1

#Функция вызывается со значением первого аргумента - "Argument 1", второго - по умолчанию, и третьего - "Named argument".

>>>ret1, ret2, ret3 = myfunction("Argument 1", arg3 = "Named argument")

 # ret1, ret2 и ret3 принимают значения "Named argument", 100, "Argument 1" соответственно

>>> print ret1, ret2, ret3

Named argument 100 Argument 1

 # Следующая запись эквивалентна def f(x): return x + 1

functionvar = lambda x: x + 1

>>> print functionvar(1)

2</blockquote>

Классы

Язык Python ограничен в множественном наследовании в классах. Внутренние переменные и внутренние методы классов начинаются с двух знаков нижнего подчеркивания «__» (например «__myprivatevar»). Мы можем также присвоить значение переменной класса извне. Пример:

<blockquote>class Myclass:

common = 10

def __init__(self):

self.myvariable = 3

def myfunction(self, arg1, arg2):

return self.myvariable

 # Здесь мы объявили класс Myclass. Функция __init__ вызывается автоматически при инициализации классов.

>>> classinstance = Myclass() # Мы инициализировали класс и переменная myvariable приобрела значение 3 как заявлено в методе инициализации

>>> classinstance.myfunction(1, 2) #Метод myfunction класса Myclass возвращает значение переменной myvariable

3

Переменная common объявлена во всех классах

>>> classinstance2 = Myclass()

>>> classinstance.common

10

>>> classinstance2.common

10

 # Поэтому, если мы изменим ее значение в классе Myclass изменятся

*# и ее значения в объектах, инициализированных классом Myclass *

>>> Myclass.common = 30

>>> classinstance.common

30

>>> classinstance2.common

30

 # А здесь мы не изменяем переменную класса. Вместо этого

*# мы объявляем оную в объекте и присваиваем ей новое значение *

>>> classinstance.common = 10

>>> classinstance.common

10

>>> classinstance2.common

30

>>> Myclass.common = 50

*# Теперь изменение переменной класса не коснется *

 # переменных объектов этого класса

>>> classinstance.common

10

>>> classinstance2.common

50

Следующий класс является наследником класса Myclass

наследуя его свойства и методы, ктому же класс может

наследоваться из нескольких классов, в этом случае запись

такая: class Otherclass(Myclass1, Myclass2, MyclassN)

class Otherclass(Myclass):

def __init__(self, arg1):

self.myvariable = 3

print arg1

>>> classinstance = Otherclass("hello")

hello

>>> classinstance.myfunction(1, 2)

3

Этот класс не имеет совйтсва test, но мы можем

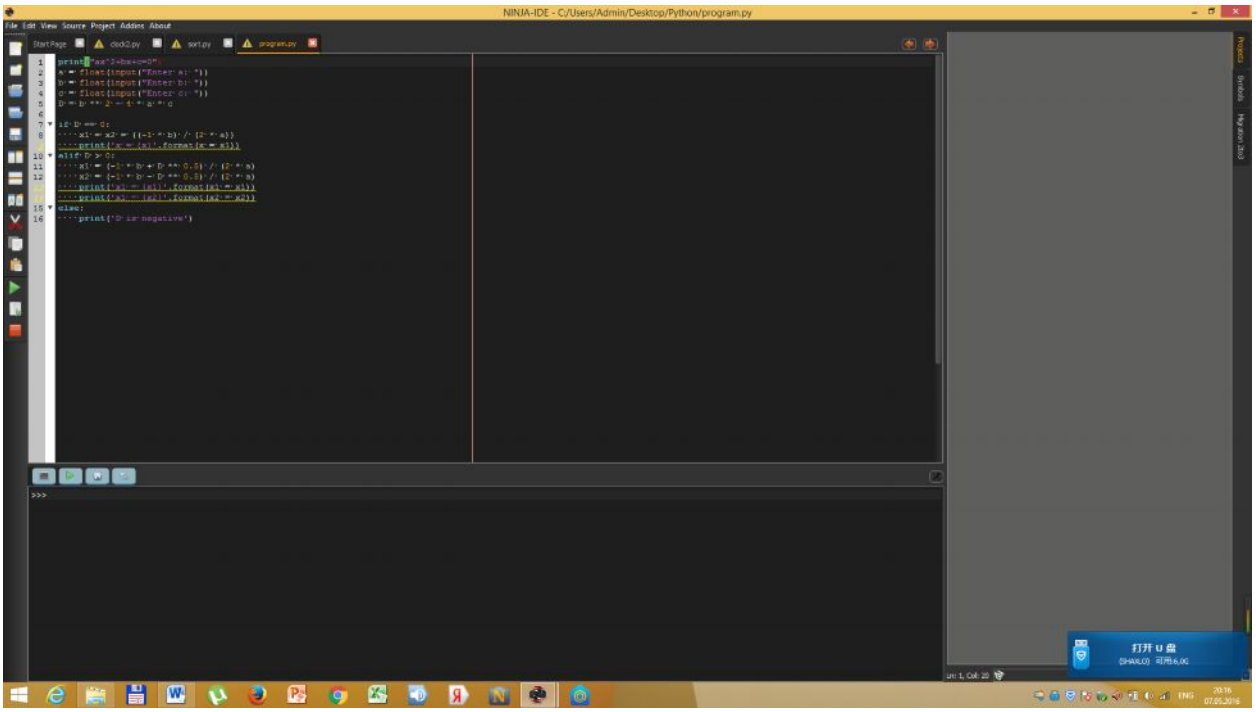
объявить такую переменную для объекта. Причем

тэта переменная будет членом только classinstance.

>>> classinstance.test = 10

>>> classinstance.test

10</blockquote>



Исключения

Исключения в Python имеют структуру try-except [exceptionname]:

<blockquote>def somefunction():

try:

 # Деление на ноль вызывает ошибку

10 / 0

except ZeroDivisionError:

 # Но программа не "Выполняет недопустимую операцию"

 # А обрабатывает блок исключения соответствующий ошибке «ZeroDivisionError»

print "Oops, invalid."

>>> fnexcept()

Oops, invalid.</blockquote>

Импорт

Внешние библиотеки можно подключить процедурой «import [libname]», где [libname] — название подключаемой библиотеки. Вы так же можете использовать команду «from [libname] import [funcname]», чтобы вы могли использовать функцию [funcname] из библиотеки [libname]

<blockquote>import random #Импортируем библиотеку «random»

from time import clock #И заодно функцию «clock» из библиотеки «time»

randomint = random.randint(1, 100)

>>> print randomint

64</blockquote>

Работа с файловой системой

Python имеет много встроенных библиотек. В этом примере мы попробуем сохранить в бинарном файле структуру списка, прочитать ее и сохраним строку в текстовом файле. Для преобразования структуры данных мы будем использовать стандартную библиотеку «pickle»

<blockquote>import pickle

mylist = ["This", "is", 4, 13327]

*# Откроем файл C:\binary.dat для записи. Символ «r» *

 # предотвращает замену специальных сиволов (таких как \n, \t, \b и др.).

myfile = file(r"C:\binary.dat", "w")

pickle.dump(mylist, myfile)

myfile.close()

myfile = file(r"C:\text.txt", "w")

myfile.write("This is a sample string")

myfile.close()

myfile = file(r"C:\text.txt")

>>> print myfile.read()

'This is a sample string'

myfile.close()

 # Открываем файл для чтения

myfile = file(r"C:\binary.dat")

loadedlist = pickle.load(myfile)

myfile.close()

>>> print loadedlist

['This', 'is', 4, 13327]</blockquote>

Особенности

Условия могут комбинироваться. 1 < a < 3 выполняется тогда, когда a больше 1, но меньше 3.

Используйте операцию «del» чтобы очищать переменные или элементы массива.

Python предлагает большие возможности для работы со списками. Вы можете использовать операторы объявлении структуры списка. Оператор for позволяет задавать элементы списка в определенной последовательности, а if — позволяет выбирать элементы по условию.

<blockquote>>>> lst1 = [1, 2, 3]

>>> lst2 = [3, 4, 5]

>>> print [x * y for x in lst1 for y in lst2]

[3, 4, 5, 6, 8, 10, 9, 12, 15]

>>> print [x for x in lst1 if 4 > x > 1]

[2, 3]

Оператор «any» возвращает true, если хотя

бы одно из условий, входящих в него, выполняется.

>>> any(i % 3 for i in [3, 3, 4, 4, 3])

True

Следующая процедура подсчитывает количество

подходящих элементов в списке

>>> sum(1 for i in [3, 2, 4, 5, 8, 10, 9, 12, 15])

color="#ff4500">3, 4, 4, 3] if i == 3)

3

>>> del lst1[0]

>>> print lst1

[2, 3]

>>> del lst1</blockquote>

Глобальные переменные объявляются вне функций и могут быть прочитанны без каких либо объявлений. Но если вам необходимо изменить значение глобальной переменной из функции, то вам необходимо объявить ее в начале функции ключевым словом «global», если вы этого не сделаете, то Python объявит переменную, доступную только для этой функции.

<blockquote>number = 5

def myfunc():

 # Выводит 5

print number

def anotherfunc():

Это вызывает исключение, поскольку глобальная апеременная

не была вызванна из функции. Python в этом случае создает

одноименную переменную внутри этой функции и доступную

только для операторов этой функции.

print number

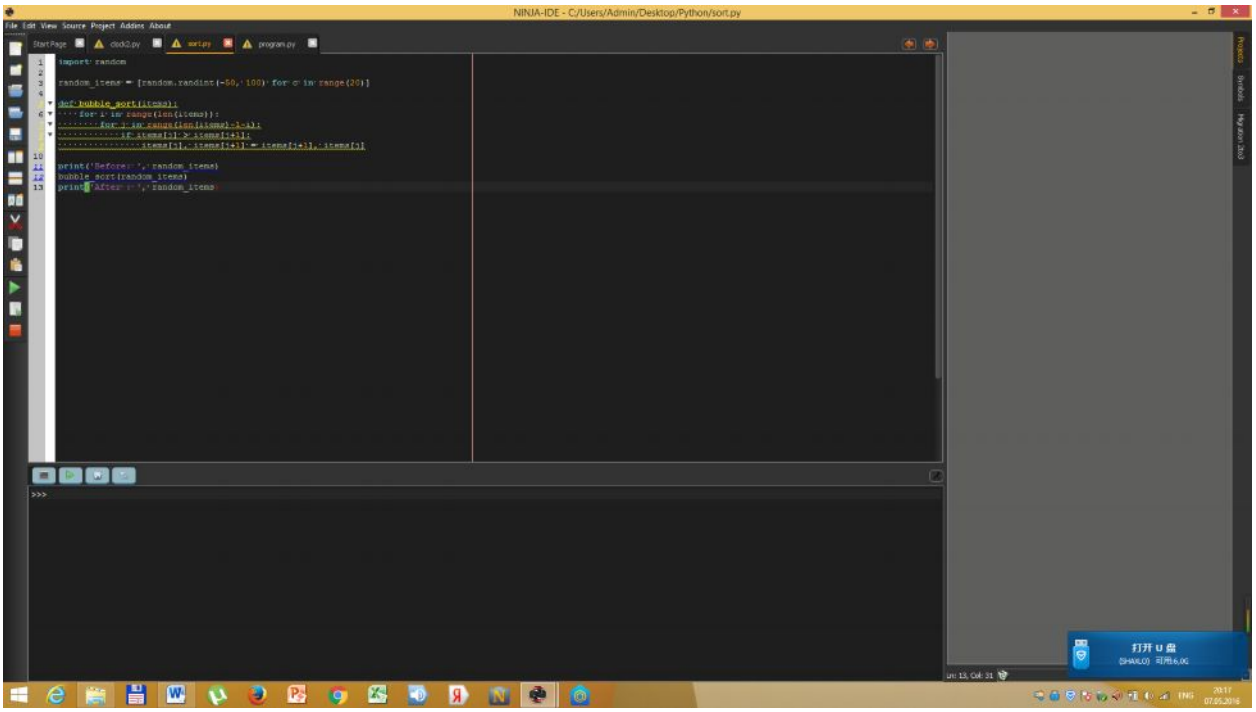
number = 3

def yetanotherfunc():

global number

И только из этой функции значение переменной изменяется.

number = 3</blockquote>



Эпилог

Разумеется в этой статье не описываются все возможности Python. Я надеюсь что эта статья поможет вам, если вы захотите и в дальнейшем изучать этот язык программирования.

Преимущества Python

Скорость выполнения программ написанных на Python очень высока. Это связанно с тем, что основные библиотеки Python

написаны на C++ и выполнение задач занимает меньше времени, чем на других языках высокого уровня.

В связи с этим вы можете писать свои собственные модули для Python на C или C++

В стандартных библиотеках Python вы можете найти средства для работы с электронной почтой, протоколами

Интернета, FTP, HTTP, базами данных, и пр.

Скрипты, написанные при помощи Python выполняются на большинстве современных ОС. Такая переносимость обеспечивает Python применение в самых различных областях.

Python подходит для любых решений в области программирования, будь то офисные программы, вэб-приложения, GUI-приложения и т.д.

Над разработкой Python трудились тысячи энтузиастов со всего мира. Поддержкой современных технологий в стандартных библиотеках мы можем быть обязаны именно тому, что Python был открыт для всех желающих.