Лабораторна робота №2

Тема: Функції у мові Python

Мета: навчитися створювати власні функції у мові Python,

Теоретична частина

Функції

Функція в python – об'єкт, який приймає аргументи і повертає значення. Зазвичай функція визначається за допомогою інструкції def.

Визначимо найпростішу функцію:

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

Інструкція **return** каже, що потрібно повернути значення. У нашому випадку функція повертає суму x і y.

Тепер ми її можемо викликати:

```
>>>
>>> add(1, 10)
11
>>> add('abc', 'def')
'abcdef'
```

Функція може бути будь-якої складності і повертати будь-які об'єкти (списки, кортежі, і навіть функції).

Функція може і не закінчуватися інструкцією return, при цьому функція поверне значення None:

```
>>>
>>> def func():
...     pass
...
>>> print(func())
None
```

А ргументи функції

Функція може приймати будь-яку кількість аргументів чи не приймати їх зовсім. Також поширені функції з довільним числом аргументів, функції з позиційними і іменованими аргументами, обов'язковими і необов'язковими.

```
>>>
>>> def func(a, b, c=2): # c - необов'язковий аргумент
...     return a + b + c
...
>>> func(1, 2)  # a = 1, b = 2, c = 2 (за замовчуванням)
5
>>> func(1, 2, 3)  # a = 1, b = 2, c = 3
6
>>> func(a=1, b=3)  # a = 1, b = 3, c = 2
6
>>> func(a=3, c=6)  # a = 3, c = 6, b не визначений - виникне помилка
```

Функція також може приймати змінну кількість позиційних аргументів, тоді перед ім'ям ставиться *:

```
>>>
>>> def func(*args):
...     return args
...
>>> func(1, 2, 3, 'abc')
(1, 2, 3, 'abc')
>>> func()
()
>>> func(1)
(1,)
```

Як видно з прикладу, args – це кортеж з усіх переданих аргументів функції, і зі змінною можна працювати так само, як і з кортежем.

Функція може приймати і довільне число іменованих аргументів, тоді перед ім'ям ставиться **:

```
>>>
>>> def func(**kwargs):
...     return kwargs
...
>>> func(a=1, b=2, c=3)
{'a': 1, 'c': 3, 'b': 2}
>>> func()
{}
>>> func(a='python')
{'a': 'python'}
```

В змінній kwargs у нас зберігається словник, з яким ми, знову-таки, можемо робити все, що нам заманеться.

Анонімні функції, інструкція lambda

Анонімні функції можуть містити лише один вислів, але і виконуються вони швидше. Анонімні функції створюються за допомогою інструкції lambda. Крім цього, їх не обов'язково привласнювати змінній, як ми робили з інструкцією def func ():

```
>>>
>>> func = lambda x, y: x + y
>>> func(1, 2)
3
>>> func('a', 'b')
'ab'
>>> (lambda x, y: x + y)(1, 2)
3
>>> (lambda x, y: x + y)('a', 'b')
'ab'
```

lambda функції, на відміну від звичайної, не потрібна інструкція return, а в іншому, вона поводиться точно так же:

```
>>>
>>> func = lambda *args: args
>>> func(1, 2, 3, 4)
(1, 2, 3, 4)

f = lambda x: x**2 + 4
Те ж саме, що:
def f(x):
   return x**2 + 4

У загальному випадку будь-яка конструкція виду:
def g(arg1, arg2, arg3, ...):
   return expression
Може бути записана як:
g = lambda arg1, arg2, arg3, ...:expression
```

Lambda-функції дуже зручні для того, щоб визначати невеликі функції "на льоту" і тому дуже популярні серед багатьох програмістів.

Lambda-функції часто використовуються для швидкого визначення функції як аргумент іншої функції.

Doc strings

В Python є домовленість про те, що рядки документації (doc strings) вставляються відразу після заголовка функції. Doc strings містять короткий опис мети функції та пояснюють сенс аргументів і значень. Doc strings розміщаються в потрійних лапках """, які дозволяють розбивати текст між ними в кілька рядків. Ось два приклади використання рядків документації у функціях, короткий і довгий:

```
def C2F(C):
   """Convert Celsius degrees (C) to Fahrenheit."""
   return (9.0/5)*C + 32

def line(x0, y0, x1, y1):
   """
   Compute the  coefficients a and b in the mathematical
   expression for a straight line y = a*x + b that goes
   through two points (x0, y0) and (x1, y1).

   x0, y0: a point on the line (floats).
   x1, y1: another point on the line (floats).
   return: coefficients a, b (floats) for the  line  (y=a*x+b).
   """
   a = (y1 - y0)/float(x1 - x0)
   b = y0 - a*x0
   return a, b
```

Запам'ятайте, що рядки документації повинні розташовуватися до всіх інструкцій функції. Doc strings це не просто коментарі, вони мають більші можливості. Наприклад, для вище описаної функції лінії виконується команда:

print(C2F.__doc__)

в результаті якої (як ви можете перевірити самі) виводиться весь укладений вміст Doc strings.

Модулі

Підключення модуля зі стандартної бібліотеки

Підключити модуль можна за допомогою інструкції імпорту. Наприклад, підключимо модуль OS для отримання поточної директорії:

```
>>>
>>> import os
>>> os.getcwd()
'C:\\Python33'
```

Після ключового слова import вказується назва модуля. Однією інструкцією можна підключити декілька модулів, хоча цього не рекомендується робити, так як це знижує читаність коду. Імпортуємо модулі time і random.

```
>>>
>>> import time, random
>>> time.time()
1376047104.056417
>>> random.random()
0.9874550833306869
```

Після імпортування модуля його назва стає змінною, через яку можна отримати доступ до атрибутів модуля. Наприклад, можна звернутися до константи e, розташованої в модулі math:

```
>>>
>>> import math
>>> math.e
2.718281828459045
```

Варто відзначити, що якщо зазначений атрибут модуля не буде знайдений, збудиться виняток AttributeError. А якщо не вдасться знайти модуль для імпортування, то ImportError.

Використання псевдонімів

Якщо назва модуля занадто довга, або вона вам не подобається з якихось інших причин, то для неї можна створити псевдонім, за допомогою ключового слова as.

```
>>>
>>> import math as m
>>> m.e
2.718281828459045
```

Тепер доступ до всіх атрибутів модуля math здійснюється тільки за допомогою змінної m, а змінної math в цій програмі вже не буде (якщо,

звичайно, ви після цього не напишете import math, тоді модуль буде доступний як під ім'ям m, так і під ім'ям math).

Інструкція from

Підключити певні атрибути модуля можна за допомогою інструкції from. Вона має кілька форматів:

from <Назва модуля> **import** <Атрибут 1> [**as** <Псевдонім 1>], [<Атрибут 2> [**as** <Псевдонім 2>] ...]

from <Назвамодуля> **import** *

Перший формат дозволяє підключити з модуля тільки зазначені вами атрибути. Для довгих імен також можна призначити псевдонім, вказавши його після ключового слова as.

```
>>>
>>> from math import e, ceil as c
>>> e
2.718281828459045
>>> c(4.6)
5
```

Імпортовані атрибути можна розмістити на декількох рядках, якщо їх багато, для кращої читання коду:

```
>>>
>>> from math import (sin, cos,
...          tan, atan)
```

Другий формат інструкції from дозволяє підключити всі (точніше, майже все) змінні з модуля. Для прикладу імпортуємо всі атрибути з модуля sys:

```
>>>
>>> from sys import *
>>> version
'3.3.2 (v3.3.2:d047928ae3f6, May 16 2013, 00:03:43) [MSC v.1600 32 bit (Intel)]'
>>> version_info
sys.version_info (major = 3, minor = 3, micro = 2, releaselevel = 'final', serial = 0)
```

Слід зауважити, що не всі атрибути будуть імпортовані. Якщо в модулі визначена змінна __all__ (список атрибутів, які можуть бути підключені), то будуть підключені тільки атрибути з цього списку. Якщо змінна __all__ не визначена, то будуть підключені всі атрибути, які не починаються з нижнього підкреслення. Крім того, необхідно враховувати, що імпортування всіх атрибутів з модуля може порушити простір імен головної програми, так як змінні, що мають однакові імена, будуть перезаписані.

Створення свого модуля на Python

Створимо файл mymodule.py, в якому визначимо якісь функції:

```
def hello():
    print('Hello, world!')
```

```
def fib(n):
    a = b = 1
    for i in range(n - 2):
        a, b = b, a + b
    return b
```

Тепер в цій же папці створимо інший файл, наприклад, main.py:

```
import mymodule

mymodule.hello()
print(mymodule.fib(10))
```

На екран буде виведено:

```
Hello, world!
55
```

Як назвати модуль?

Пам'ятайте, що ви (або інші люди) будуть його імпортувати і використовувати в якості змінної. Модуль неможна назвати так, як і ключове слово. Також імена модулів неможна починати з цифри. І не варто називати модуль так, як будь-яку з вбудованих функцій. Тобто, звичайно, можна, але це створить великі незручності при його подальшому використанні.

Куди помістити модуль?

Туди, де його потім можна буде знайти. Шляхи пошуку модулів вказані в змінній sys.path. У нього включені поточна директорія (тобто модуль можна залишити в папці з основною програмою), а також директорії, в яких встановлено python. Крім того, змінну sys.path можна змінювати вручну, що дозволяє покласти модуль в будь-яке зручний для вас місце (головне, не забути в головній програмі модифікувати sys.path).

Чи можна використовувати модуль як самостійну програму?

Можна. Однак треба пам'ятати, що при імпортуванні модуля його код виконується повністю, тобто, якщо програма щось друкує, то при її імпортуванні це буде надруковано. Цього можна уникнути, якщо перевіряти, чи запущений скрипт як програма, або імпортований. Це можна зробити за допомогою змінної __name__, яка визначена в будь-якій програмі, і дорівнює "__main__", якщо скрипт запущений в якості головної програми, і ім'я, якщо він імпортований. Наприклад, mymodule.py може виглядати ось так:

```
def hello():
    print('Hello, world!')

def fib(n):
    a = b = 1
    for i in range(n - 2):
        a, b = b, a + b
    return b
```

```
if __name__ == "__main__":
    hello()
    for i in range(10):
        print(fib(i))
```

Завдання:

Розробіть функції для здійснення наступних операцій зі списками:

1. Швидке сортування;
2. Пошук елементу за значенням;
3. Пошук послідовності елементів;
4. Пошук перших п’яти мінімальних елементів;
5. Пошук перших п’яти максимальних елементів;
6. Пошук середнього арифметичного;
7. Повернення списку, що сформований з початкового списку, але не містить повторів (залишається лише перший з однакових елементів).

Помістіть функції в окремий модуль. Реалізуйте програму, яка використовує всі функції зі створеного модуля. Зробити описи Doc strings для кожної реалізованої функції.

Контрольні питання:

1. Як створити функцію у мові Python?
2. Що таке модулі та пакети?
3. Які бувають способи підключення модулів та пакетів?
4. Що таке анонімні функції та інструкція lambda?
5. Чи можна використати модуль як самостійну програму?