**Технически Университет – София,**

**Факултет „Компютърни системи и технологии“**

**Упражнение 8 – Създаване приложение с модули**

**Понятие за модул**

Модулът е файл, съдържащ както дефиниции на функции, класове, променливи, така и операции. В Python името на файла е името на модула с файлово разширение ***.py.***

**Пример 1:** Създаване на модул за извеждане на съобщението *„Hello“* и съхраняванет му под името ***mymodule.py***:

**def greeting(name):  
 print("Hello, " + name)**

Извикването на модула се реализира чрез изпълняването на инструкцията ***import*** в друг файл, създаден на Python, спазвайки следния синтаксис:

**import mymodule  
mymodule.greeting("Jonathan")**

**Резултат:**

Hello, Jonathan

Когато интерпретаторът срещне оператор за импортиране, той го импортира ако модулът е наличен в търсената директория. Директорията за търсене е списъкът от файлове, които се претърсват от страна на интерпретатора.

Например, за импортирането на модула ***support.py***, в горната част на скрипта трябва да се постави следната команда:

**Пример 2:**

**#!/usr/bin/python**

**# Import module support**

**import support**

**# Now you can call defined function that module as follows**

**support.print\_func("Zara")**

**Резултат:**

Hello : Zara

Съдържащите се в модула изпълними инструкции и дефиниции на функции се изпълняват само първия път, когато името на модула бъде срещнато в оператор за импортиране. Допустимо е едни модули да импортират други модули. Обикновено всички ***import*** оператори се поставят в началото на модул. Имената на импортираните модули се поставят в таблицата с глобални символи на модула за импортиране.

За импортирането на имена от модул директно в таблицата със символи на модула за импортиране се използва оператора ***import*** при следния синтаксис:

**from modname import name1[, name2[, ... nameN]]**

**Пример 3:**

**>>> from fibo import fib, fib2**

**>>> fib(500)**

**Резултат:**

**0 1 1 2 3 5 8 13 21 34 55 89 144 233 377**

За импортиране на всички имена, които модулът дефинира се прилага следния синтаксис:

**from modname import \***

**Пример 4:**

**>>> from fibo import \***

**>>> fib(500)**

**Резултат:**

0 1 1 2 3 5 8 13 21 34 55 89 144 233 377

**Забележка:**

1. Всеки модул се импортира само веднъж на сесия на интерпретатора. Следователно, при промяна в модулите е необходимо презареждане на интерпретаторите.

2. Друг вариант е презареждането на предварително импортиран модул, което се изпълнява чрез функцията ***reload ()***. Тази реализация се препоръчва за случаите, когато е извършена редакция в кода на модула с използването на външен редактор и е необходимо да се тества редактираната версия без да се напуска интерпретатора на Python. Върнатата стойност е обектът на модула. Синтаксисът е както следва:

**reload(module\_name)**

където, ***module\_name*** е името на модула, който трябва да бъде презареден.

**Променливи в модул**

При създаване на модули могат да се използват промеливи от тип масив, обекти, речници, и др.

**Пример 5:** Нека следващия код бъде съхранен под името ***mymodule.py***:

**person1 = {  
 "name": "John",  
 "age": 36,  
 "country": "Norway"  
}**

**Пример 6:** Импортирайте модула с име ***mymodule*** и осъществете достъп до речника ***person1***:

**import mymodule  
a = mymodule.person1["age"]  
print(a)**

**Локализиране на модули**

При импортиране на модул интерпретаторът на Python извършва търсене за модула в следната последователност:

* Текуща директория;
* Ако модулът не бъде открит в текущата директория, Python проверява всяка директория в променливата ***PYTHONPATH***;
* Ако модулът не бъде открит и в променливата ***PYTHONPATH***, то Python проверява директорията по подразбиране.

**Променливата PYTHONPATH**

Променливата ***PYTHONPATH*** се състои се от списък с директории и се задава със следния синтаксис за ОС Windows:

**set PYTHONPATH = c:\python20\lib;**

**Глобални и локални променливи**

В Python променливите могат да бъдат с глобален или локален обхват. Променливите, създадени извън дадена функция, се наричат глобални променливи. Достъпът до глобална променлива може да се осъществява както вътре, така и извън функцията.

**Пример 7 (за дефиниране на глобална променлива):**

**#variable defined in global scope**

**var='Hello world'**

**#function accessing the global variable**

**def my\_function():**

**print(var)**

**#calling the function**

**my\_function()**

**Коментар:** В посочения пример се създава променливата ***var***, на която е присвоява стойността ***Hello world***. Променливата се дефинира извън функцията ***my\_function***. При извикването на функцията се отпечатва стойността на променливата. Поради това, че във функцията не е дефинирана променлива със същото име, то ***var*** се отпечатва.

Променливата, дефинирана в рамките на функцията, се нарича **локална променлива**. Достъпът до нея се осъществява само в рамките на функцията.

**Пример 8 (за дефиниране на локална променлива):**

**def my\_function():**

**#variable defined in local scope**

**var='Hello world'**

**print(var)**

**#calling the function**

**my\_function()**

**Обхват на локални и глобални променливи**

Ако се дефинира променлива с едно и също име както във функция, така и извън нея, то приоритет се дава на локалната променлива пред глобалната.

**Пример 9 (за обхват на локална и глобална променлива):**

**var='global scope'**

**def my\_function():**

**var='local scope'**

**print(var)**

**my\_function()**

**print(var)**

**Резултат:**

local scope

global scope

**Забележка:** Промяната на стойността на променливата в локалния обхват не оказва влияние върху стойността на глобалната променлива.

### **Функцията dir()**

Вградената функция ***dir ()*** връща сортиран списък от низове, съдържащ имената на всички модули, променливи и функции, дефинирани в модул.

**Пример 10:**

**#!/usr/bin/python**

**# Import built-in module math**

**import math**

**content = dir(math)**

**print content**

**Резултат:**

['\_\_doc\_\_', '\_\_file\_\_', '\_\_name\_\_', 'acos', 'asin', 'atan',

'atan2', 'ceil', 'cos', 'cosh', 'degrees', 'e', 'exp',

'fabs', 'floor', 'fmod', 'frexp', 'hypot', 'ldexp', 'log',

'log10', 'modf', 'pi', 'pow', 'radians', 'sin', 'sinh',

'sqrt', 'tan', 'tanh']

Където: променливата *\_\_name\_\_* е името на модула, а *\_\_file\_\_* е името на файла, от който модула е зареден.

**Пакети в Python**

Пакетът е йерархична структура на файловата директория, която дефинира приложение на Python, състоящо се от модули и подпакети.

**Пример 11:**

Нека са налице три файла, разработени на Python, всеки от които се намира в директорията ***Phone*** и изпълнява определена функция, както следва:

1. файлът ***Pots.py*** има следния програмен код:

**#!/usr/bin/python**

**def Pots():**

**print "I'm Pots Phone"**

1. файлът *Isdn.py, е аналогичен на Pots.py и изпълнява функцията* Isdn() за извеждане на съобщението „“;
2. файлът G3.py, характеризиращ се с подобен програмен код както предните два файла, но изпълняващ функцията G3(), извеждайки съобщението „“.

Нека създадем четвърти файл *\_\_init\_\_.py* в директорията Phone, чрез който да изпълним всички горепосочени функции. За целта в този файл трябва да се извърши импортиране чрез добавянето на следния синтаксис в *\_\_init\_\_.py*:

**from Pots import Pots**

**from Isdn import Isdn**

**from G3 import G3**

**Резултат:** всички посочени класове са налични, когато се импортира пакета Phone:

**#!/usr/bin/python**

**# Now import your Phone Package.**

**import Phone**

**Phone.Pots()**

**Phone.Isdn()**

**Phone.G3()**

Изпълнението на посочения програмен код осигурява следния резултат:

I'm Pots Phone

I'm 3G Phone

I'm ISDN Phone

**Задачи:**

**1.** Да се състави програма, която реализира приложение за изчисляване и извеждане лицата на геометрични фигури: триъгълник, квадрат, правоъгълник, ромб и трапец, като страните и височините на фигурите се въвеждат от клавиатурата като входни данни.

**2.** Да се създаде приложение за реализиране на калкулатор, изчисляващо аритметичните операции събиране, изваждане, умножение и деление на въведени от клавиатурата цели числа. Операциите да се реализират чрез модули, които да бъдат извиквани и изпълнявани чрез пакет.