Tema 9.
Module și pachete în Python

Ce ne așteaptă?

1. Importul modulelor
2. Pachete în Python
3. Module standard în Python

1. Ce reprezintă un modul?
2. Cum se importă un modul?
3. Care se importă un modul cu schimbarea numelui?
4. Care se importă doar unele componente ale modulului?
5. Cum se execută un modul importat?
6. Care este rolul variabilei speciale __name__?
7. Care este rolul funcției dir()?
8. Ce reprezintă un pachet?
9. Cum se clasifică modulele/pachetele?
10. Care este rolul modulului math?
11. Care este rolul modulului random?
12. Care este rolul modulului platform?
13. Cum se descarcă și se instalează modulele ne-încorporate

1. Importul modulelor

Noțiune de modul și importul acestuia

- **Modul – fișier salvat cu extensia .py ce poate conține clase, metode, funcții sau variabile**
- **Cuvântul cheie *import* – permite importul componentelor unui modul în alt modul**
- **Sintaxa generală de import:**

  import *nume_modul*

- **Sintaxa de utilizare a componetelor importate:**

  nume_modul.nume_componeta

Exemple de import și utilizare a componentelor importante

- **Exemplu de modul salvat cu numele suma_produs.py**

```
variabila = 10
def suma_doua_numere(a, b):
    print(f"Suma dintre {a} si {b} este {a+b} ")
def produsul_doua_numere(a, b):
    print(f"Produsul dintre {a} si {b} este {a*b} ")
```

- **Exemplu de modul salvat modulul_meu.py care importă componentele lui suma_produs.py**

```
import suma_produs
print(suma_produs.variabila)
suma_produs.suma_doua_numere(5, 2)
suma_produs.produsul_doua_numere(2, 3)
```

Aliasing-ul modului și cuvântul cheie *from*

- Aliasing-ul redenumirea componentelor importate
- Sintaxa aliasing-ului unui modul:

  import *nume_modul* as *nume_nou*
- Exemplu de utilizare

```
import suma_produs as calc
print(calc.variabila)
calc.suma_doua_numere(5, 2)
calc.produsul_doua_numere(2, 3)
```

- Cuvântul-cheie *from* – permite importul doar a unor componente ale modulului ce apoi pot fi apelate fară a specifica denumirea modulului
- Sintaxa de import utilizand cuvantul cheie from:

  from *nume_modul* import *nume_componenta*
- Exemplu de utilizare – nu mai este necesară scrierea numelui modului la apelare

```
from suma_produs import variabila, suma_doua_numere
print(variabila)
suma_doua_numere(5, 2)
```

import * și aliasing-ul componentelor modulului

- import * - permite importul tuturor componentelor modulului
- Exemplu de utilizare

```
from suma_produs import *
print(variabila)
suma_doua_numere(5, 2)
produsul_doua_numere(5,2)
```

- Aliasing-ul componentelor permite redenumirea componentelor importate
- Exemplu de utilizare

```
from suma_produs import variabila as x, suma_doua_numere as suma
print(x)
suma(5, 2)
```

Execuția modulului importat

- Modul suma_produs.py cu apelarea funcțiilor

```
variabila = 10

def suma_doua_numere(a, b):
    print(f"Suma dintre {a} si {b} este {a+b} ")

def produsul_doua_numere(a, b):
    print(f"Produsul dintre {a} si {b} este {a*b} ")

suma_doua_numere(30,50)
produsul_doua_numere(3,5)
```

- Modul modulul_meu.py care importă modul suma_produs.py

```
import suma_produs
```

Variabila specială __name__

- __name__ - variabila ce păstrează numele modulului.
- __name__ returneaza valoare __main__ daca este apelata în modulul curent sau numele modului daca modulul curent este importat.
- Exemplu de afisare a valorii variabilei __name__ în modulul curent suma_produs.py

```
variabila = 10

def suma_doua_numere(a, b):
    print(f"Suma dintre {a} si {b} este {a+b} ")

def produsul_doua_numere(a, b):
    print(f"Produsul dintre {a} si {b} este {a*b} ")

print(f'Variabila __name__ are valoarea: {__name__}')
```

- Exemplu de afisare a valorii variabilei __name__ în modulul_meu.py

```
import suma_produs
```

Utilizarea variabilei __name__

- Modul suma_produs.py cu utilizarea variabila __name__

```
variabila = 10

def suma_doua_numere(a, b):
    print(f"Suma dintre {a} si {b} este {a+b} ")

def produsul_doua_numere(a, b):
    print(f"Produsul dintre {a} si {b} este {a*b} ")

if __name__=="__main__":
    suma(30,50)
    produsul(3,5)
```

- Modul modulul_meu.py care importă modul suma_produs.py

```
import suma_produs
```

Funcția dir()

- **dir() – returnează lista tuturor componentelor unui modul:**
  - __doc__ - informația despre modul
  - __name__ - numele modului
  - __file__ - calea către locația modului
  - __spec__ - specificațiile modului
  - __package__ - numele pachetului în care se află modului
  - funcții, clase, metode, variabile definite de utilizator
- **Exemplu de utilizare a funcției dir()**

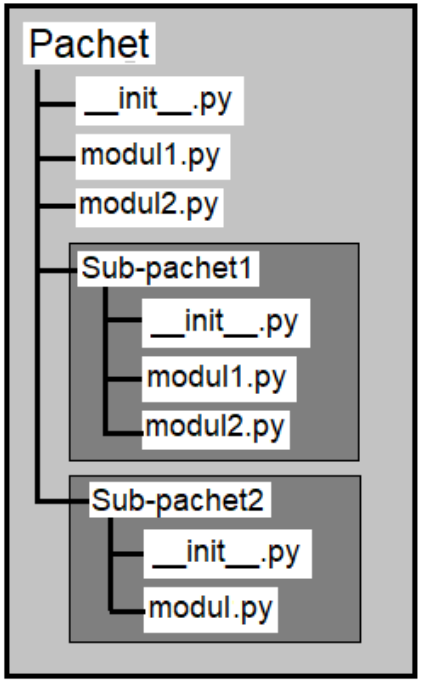
```
import suma_produs

print(dir(suma_produs))
print(suma_produs.__name__)
print(suma_produs.__doc__)
print(suma_produs.__file__)
print(suma_produs.__package__)
```

2. Pachete în Python

Structura unui pachet

- **pachet – un folder ce conține mai multe module**
- **un pachet va conține un fișier __init__.py**
- **__init__.py poate fi un fișier gol**
- **un pachet poate conține mai multe sub-pachete**

- **Totalizare:**
  - Modul = grup de variabile, funcții, metode și clase
  - Pachet = grup de module
  - Bibliotecă = grup de pachete



3. Module standard în Python

Clasificarea module

- **Module proprii create de utilizator**
- **Module existente:**
  - Module create de propriul grup de lucru
  - Module încorporate în Python

    https://docs.python.org/3/py-modindex.html
  - Module ale comunității Python

    https://pypi.org/

Modulul math

- Conține funcții și constante matematice:
  https://docs.python.org/3/library/math.html#module-math

- Exemplu de utilizare a modulului math

```
import math

print(dir(math))

print(math.sin(20))
print(math.ceil(3.2))
print(math.floor(3.2))
print(math.log(40))
print(math.log(40,6))
print(math.radians(90))
print(math.atan(180))
print(math.pow(2, 3))
print(math.sqrt(36))
Print(math.pi)
```

Modulul random

- Conține mecanisme de operare cu numere pseudo-aliatoare
- https://docs.python.org/3/library/random.html#module-random

- Exemplu de utilizare a modulului random

```
import random

print(dir(random))

random.seed(42)  # creaza numere pseudo-aliatoare
print(random.random())   #numar aliator in banda 0-1
print(random.randint(2,7))  # numar intreg aliator in banda specificata
print(random.randrange(3, 30, 2)) # numar intreg aliator generat conform range()

lista = [45, 67, 30, 20 ,54, 70, 34, 86]
print(random.choice(lista))  # numar aliator dintr-o lista
print(random.sample(lista, 3)) # mai multe numere aliatoare dintr-o lista
random.shuffle(lista)  # amesteca ordinea elementelor in lista
print(lista)
```

Modulul platform

- Conține funcții de generarea a informație despre dispozitiv si componentele soft
https://docs.python.org/3/library/platform.html#module-platform

- Exemplu de utilizare a modulului platform

```
import platform

print(dir(platform))

print(platform.system())   # sistemul de operare
print(platform.win32_ver())  # versiunea sistemului windows
print(platform.win32_edition()) # editia sistemului windows
print(platform.node())  # numele dispozitivului in retea
print(platform.processor()) # tipul procesorului
print(platform.python_compiler()) # tipul interpretorului Python
print(platform.python_version())  # versiunea Python
```

- Alte module importante în Python: os și sys

Importul modulelor comunității Python

- **Importul simplu al unui modul ne-încorporat Python = ModuleNotFoundError**

```
import pandas
```

- **Descărcarea și instalarea modulelor ne-încorporat Python**

https://utm-my.sharepoint.com/:f:/g/personal/pavel_nicolaev_tlc_utm_md/EgDbVwX6JPZCjQzHeuXbmE4BocPnM_3fyVj6KGhcMc0seA?e=TXQcIE

  - Descărcarea de pe net și instalarea fișierelor de tip .msi sau .exe - nerecomandat
  - Descărcarea și instalarea cu comanda pip = python install program
    - Comanda "**pip install nume_modul**" in linia de comanda - se va instala universal
    - Comanda "**pip install nume_modul**" in terminalul proiectului - se va instala local

Module specializate

- **Data Science și Machine Learning**
  - Pandas
  - Numpy
  - Matplotlib
  - Seaborn
  - Scikit-Learn
  - Keras
  - TensorFlow
  - PyTorch
- **Programarea WEB**
  - Django
  - Flask
  - Web2Py
  - Bootle
  - CherryPy
  - Pyramid
  - Dash
  - CubicWeb
- **Programarea GUI**
  - Tkinter
  - PyQT5
  - PySide2
  - Kivy
- **Game Development**
  - Pygame
  - PyGlet
  - Pandas3D
  - Arcade
- **Automatizare**
  - PyAutoGUI
  - Pywinauto
  - BeautifulSoup
  - Selenium
- **Rețilistica**
  - Netmiko
  - NAPALM
  - Requests
  - NCCClient
- **Procesare imagini**
  - OpenCV
  - Scikit-Image
  - Mahotas
  - Pymagick