LAB 5. Module și pachete.

Să cunoască:

- noţiuni generale despre bibliotecile Python

Să fie capabil:

- să acceseze bibliotecile și funcțiile predefinite de modulul respectiv Python

---

Ce este un modul?

Considerați un modul ca fiind același cu o bibliotecă de coduri.

Un fișier care conține un set de funcții pe care doriți să le includeți în aplicația dvs.

Creați un modul

Pentru a crea un modul, trebuie doar să salvați codul dorit într-un fișier cu extensia de fișier `.py`:

Salvați acest cod într-un fișier numit`mymodule.py`

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

Utilizați un modul

Acum putem folosi modulul pe care tocmai l-am creat, folosind `import`instrucțiunea:

Importați modulul numit mymodule și apelați funcția de salut:

```
import mymodule

mymodule.greeting("Jonathan")
```

Notă: Când utilizați o funcție dintr-un modul, utilizați sintaxa: *module_name.function_name* .

Variabile în modul

Modulul poate conține funcții, așa cum este deja descris, dar și variabile de toate tipurile (matrice, dicționare, obiecte etc):

Exemplu

Salvați acest cod în fișier`mymodule.py`

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

Exemplu

Importați modulul numit mymodule și accesați dicționarul person1:

```
import mymodule

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

Denumirea unui modul

Puteți denumi fișierul modul cum doriți, dar trebuie să aibă extensia fișierului `.py`

Redenumirea unui modul

Puteți crea un alias atunci când importați un modul, utilizând `as`cuvântul cheie:

Exemplu

Creați un alias pentru `mymodule`apelat `mx`:

```
import mymodule as mx

a = mx.person1["age"]
print(a)
```

Module încorporate

Există mai multe module încorporate în Python, pe care le puteți importa oricând doriți.

Exemplu

Importați și utilizați `platform`modulul:

```
import platform

x = platform.system()
print(x)
```

Folosind funcția dir().

Există o funcție încorporată pentru a lista toate numele funcțiilor (sau numele variabilelor) dintr-un modul. Funcția `dir()`:

Exemplu

Listați toate denumirile definite aparținând modulului platformei:

```
import platform

x = dir(platform)
print(x)
```

Notă: Funcția dir() poate fi folosită pe *toate* modulele, de asemenea pe cele pe care le creați singur.

Import din modul

Puteți alege să importați doar părți dintr-un modul, folosind `from`cuvântul cheie.

Exemplu

Modulul numit `mymodule`are o funcție și un dicționar:

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

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

Exemplu

Importați numai dicționarul person1 din modul:

```
from mymodule import person1

print (person1["age"])
```

Notă: Când importați folosind `from` cuvântul cheie, nu utilizați numele modulului când vă referiți la elementele din modul. Exemplu: `person1["age"]`, nu ~~`mymodule.person1["age"]`~~

Python Datetime

Curmalele Python

O dată în Python nu este un tip de date propriu, dar putem importa un modul numit `datetime`să lucreze cu datele ca obiecte date.

Exemplu

Importați modulul datetime și afișați data curentă:

```
import datetime

x = datetime.datetime.now()
print(x)
```

Data de ieșire

Când executăm codul din exemplul de mai sus, rezultatul va fi:

```
2025-04-07 07:34:53.992768
```

Data conține an, lună, zi, oră, minut, secundă și microsecundă.

Modulul `datetime`are multe metode de a returna informații despre obiectul dată.

Iată câteva exemple, veți afla mai multe despre ele mai târziu în acest capitol:

Exemplu

Returnează anul și numele zilei săptămânii:

```
import datetime

x = datetime.datetime.now()

print(x.year)
print(x.strftime("%A"))
```

Crearea obiectelor date

Pentru a crea o dată, putem folosi `datetime()` clasa (constructorul) modulului `datetime`.

Clasa `datetime()` necesită trei parametri pentru a crea o dată: an, lună, zi.

Exemplu

Creați un obiect data:

```
import datetime

x = datetime.datetime(2020, 5, 17)

print(x)
```

Clasa `datetime()` preia, de asemenea, parametri pentru oră și fus orar (oră, minut, secundă, microsecundă, tzone), dar aceștia sunt opționali și au o valoare implicită de `0`, (`None`pentru fusul orar).

Metoda strftime().

Obiectul `datetime`are o metodă de formatare a obiectelor date în șiruri de caractere care pot fi citite.

Metoda este numită `strftime()` și ia un parametru, `format`, pentru a specifica formatul șirului returnat:

Exemplu

Afișează numele lunii:

```
import datetime

x = datetime.datetime(2018, 6, 1)

print(x.strftime("%B"))
```

O referință a tuturor codurilor de format legal:

Directive	Description	Example	Code
%a	Weekday (short)	Wed	print(x.strftime("%a"))
%A	Weekday (full)	Wednesday	print(x.strftime("%A"))
%w	Weekday (0=Sun, 6=Sat)	3	print(x.strftime("%w"))
%d	Day of month (01–31)	07	print(x.strftime("%d"))
%b	Month (short)	Apr	print(x.strftime("%b"))
%B	Month (full)	April	print(x.strftime("%B"))
%m	Month (01–12)	04	print(x.strftime("%m"))
%y	Year (2-digit)	25	print(x.strftime("%y"))
%Y	Year (4-digit)	2025	print(x.strftime("%Y"))
%H	Hour (00–23)	14	print(x.strftime("%H"))
%I	Hour (01–12)	02	print(x.strftime("%I"))
%p	AM/PM	PM	print(x.strftime("%p"))
%M	Minute (00–59)	30	print(x.strftime("%M"))
%S	Second (00–59)	45	print(x.strftime("%S"))

`%f`	**Microsecond (000000–999999)**	`123456`	`print(x.strftime("%f"))`
`%z`	**UTC offset**	`+0200`	`print(x.strftime("%z"))`
`%Z`	**Timezone name**	`EET`	`print(x.strftime("%Z"))`
`%j`	**Day of year (001–366)**	`097`	`print(x.strftime("%j"))`
`%U`	**Week number (Sun-start)**	`14`	`print(x.strftime("%U"))`
`%W`	**Week number (Mon-start)**	`14`	`print(x.strftime("%W"))`
`%c`	**Locale date and time**	`Mon Apr 7 14:30:00 2025`	`print(x.strftime("%c"))`
`%x`	**Locale date**	`04/07/25`	`print(x.strftime("%x"))`
`%X`	**Locale time**	`14:30:00`	`print(x.strftime("%X"))`
`%%`	**Literal %**	`%`	`print(x.strftime("%%"))`

Python Math

Python are un set de funcții matematice încorporate, inclusiv un modul extins de matematică, care vă permite să efectuați sarcini matematice pe numere.

Funcții matematice încorporate

Funcțiile `min()` și `max()` pot fi utilizate pentru a găsi cea mai mică sau cea mai mare valoare într-un iterabil:

Exemplu

```
x = min(5, 10, 25)
y = max(5, 10, 25)

print(x)
print(y)
```

Funcția `abs()` returnează valoarea absolută (pozitivă) a numărului specificat:

Exemplu

```
x = abs(-7.25)

print(x)
```

Funcția returnează valoarea lui x la puterea lui y (x^y). `pow(x, y)`

Exemplu

Returnează valoarea lui 4 la puterea lui 3 (la fel ca 4 * 4 * 4):

```
x = pow(4, 3)

print(x)
```

Modulul de matematică

Python are, de asemenea, un modul încorporat numit `math`, care extinde lista de funcții matematice.

Pentru a-l folosi, trebuie să importați `math`modulul:

```
import math
```

După ce ați importat `math`modulul, puteți începe să utilizați metodele și constantele modulului.

Metoda, `math.sqrt()` de exemplu, returnează rădăcina pătrată a unui număr:

Exemplu

```
import math

x = math.sqrt(64)

print(x)
```

Metoda `math.ceil()` rotunjește un număr în sus la cel mai apropiat număr întreg, iar `math.floor()` metoda rotunjește un număr în jos la cel mai apropiat număr întreg și returnează rezultatul:

Exemplu

```
import math

x = math.ceil(1.4)
y = math.floor(1.4)

print(x) # returns 2
print(y) # returns 1
```

Constanta `math.pi` returnează valoarea lui PI (3.14...):

Exemplu

```
import math

x = math.pi

print(x)
```

Învață NumPy [+:

NumPy este o bibliotecă Python.

NumPy este folosit pentru a lucra cu matrice.

NumPy este prescurtarea pentru „Numerical Python”.

Ce este NumPy?

NumPy este o bibliotecă Python folosită pentru lucrul cu matrice.

De asemenea, are funcții pentru lucrul în domeniul algebrei liniare, transformării Fourier și matricelor.

NumPy a fost creat în 2005 de Travis Oliphant. Este un proiect open source și îl puteți folosi liber.

NumPy înseamnă Numerical Python**.**

De ce să folosiți NumPy?

În Python avem liste care servesc scopului matricelor, dar procesează lent.

NumPy își propune să ofere un obiect matrice care este de până la 50 de ori mai rapid decât listele tradiționale Python.

Obiectul matrice din NumPy se numește `ndarray`, oferă o mulțime de funcții de sprijin care fac lucrul cu `ndarray`foarte ușor.

Array-urile sunt foarte frecvent utilizate în știința datelor, unde viteza și resursele sunt foarte importante.

Data Science: este o ramură a informaticii în care studiem cum să stocăm, să utilizăm și să analizăm datele pentru a obține informații din acestea.

Instalarea NumPy

Dacă aveți deja instalate Python și PIP pe un sistem, atunci instalarea NumPy este foarte ușoară.

Instalați-l folosind această comandă:

```
C:\Users\Your Name>pip install numpy
```

Dacă această comandă eșuează, atunci utilizați o distribuție python care are deja instalat NumPy, cum ar fi Anaconda, Spyder etc.

Import NumPy

Odată ce NumPy este instalat, importați-l în aplicațiile dvs. adăugând `import`cuvântul cheie:

```
import numpy

arr = numpy.array([1, 2, 3, 4, 5])

print(arr)
```

Ce oferă NumPy?

Ce face	Exemplu scurt
Creează array-uri	`np.array([1, 2, 3])`
Operații vectoriale rapide	`a + b`, `a * b`, `a ** 2`
Funcții matematice	`np.sin(x)`, `np.exp(x)`, `np.mean(x)`
Indexare, slicing, broadcasting	`a[1:3]`, `a[a > 0]`
Algebra liniară	`np.dot(A, B)`, `np.linalg.inv(A)`
Numere aleatoare	`np.random.rand(3,3)`

```python
import numpy as np

a = np.array([1, 2, 3])
b = np.array([4, 5, 6])

print("Suma:", a + b)       # [5 7 9]
print("Produs:", a * b)     # [ 4 10 18]
print("Media:", np.mean(a)) # 2.0
```

De ce e atât de folosit?

- Viteză: scris în C, deci operațiile sunt rapide.
- Ușurință: simplifică codul comparat cu folosirea listelor clasice.
- Integrare: e baza pentru alte biblioteci importante: Pandas, SciPy, Scikit-learn, TensorFlow, PyTorch etc.
- Eficiență: gestionează eficient memoria pentru array-uri mari.

Exercitii:

1. Generează și afișează un număr aleatoriu între 1 și 100 folosind modulul `random.`

2. Calculează și afișează rădăcina pătrată a unui număr dat folosind modulul `math.`

3. Afișează data și ora curentă într-un format simplu folosind modulul `datetime.`

4. Creează un fișier text nou și scrie un mesaj simplu în el folosind modulul `os.`

5. Generează o listă de 5 culori aleatorii folosind modulul `random` și afișează-le.

6. Alege și afișează un nume de animăluț dintr-o listă dată folosind modulul `random.`

7. Sortează o listă de numere întregi în ordine crescătoare folosind modulul `random.`

8. Creează un joc de ghicit numere între 1 și 10 folosind modulul `random.`

9. Criptează și decriptează un cuvânt simplu folosind modulul `cryptography.`

10. Afișează primele 5 litere din alfabet în ordine folosind modulul `string.`

11. Afișează conținutul unui fișier text dat folosind modulul `os.`

12. Generează o listă de 10 numere pare folosind modulul `random.`

13. Alege și afișează un fruct dintr-o listă dată folosind modulul `random.`

14. Găsește și afișează suma a două numere întregi date folosind modulul `math.`

15. Afișează elementele unui dicționar dat folosind modulul `random.`

16. Calculează și afișează rezultatul adunării a două numere întregi folosind modulul `math.`

17. Alege și afișează o lună dintr-un an dat folosind modulul `random.`

18. Generează și afișează un număr prim între 1 și 50 folosind modulul `random.`

19. Afișează și sortează o listă de litere în ordine alfabetică folosind modulul `random.`

20. Alege și afișează o zi din săptămână folosind modulul `random.`