# Tema 15.
# Fișiere binare și foldere în Python

## Ce ne așteaptă?

1. Fișiere binare în Python
2. Arhivarea și dezarhivarea fișierelor
3. Foldere în Python

1. Ce reprezintă fișierele binare?
2. Care sunt regimurile de lucru cu fișierele binare?
3. Cum se realizează citirea datelor din fișierele binare?
4. Cum se realizează înscrierea datelor în fișierele binare?
5. Cum se realizează arhivarea fișierelor în Python?
6. Cum se realizează dearhivarea fișierelor în Python?
7. Cum se determină calea către folderul de program în Python?
8. Cum se modifică calea folderului curent în Python?
9. Cum se citește conținutul unui folder în Python?
10. Cum se creează foldere în Python?
11. Cum se șterg foldere în Python?
12. Cum se verifică prezența unui fișier într-un folder în Python?

# 1. Fișiere binare în Python

## Lucru cu fișierele binare

- **Fișiere binare – fișiere în care datele se stochează sub formă de baiți**
- **Fișierele binare dispun de aceleași regimuri de lucru cu sufix b:**
  - Regimul rb – permite citirea datelor dintr-un fișier binar
  - Regimul wb – permite ștergerea și înscrierea de noi date într-un fișier binar
  - Regimul ab – permite completarea datelor într-un fișier binar
  - Regimul r+b – permite citirea datelor și înscrierea altor date într-un fișier binar
  - Regimul w+b – permite ștergerea, înscrierea de noi date și citirea acestora dintr-un fișier binar
  - Regimul a+b – permite completarea datelor și citirea acestora într-un fișier binar

# Citirea datelor din fișiere binare

- **Pentru lucru cu fișierele binare este necesară deschiderea și închiderea acestora prin metodologii identice fișierelor textuale:**

- **Pentru citirea conținutului fișierului binar se utilizează metoda read() cu sintaxa:**

***nume_variabila_baiti = nume_variabila*.read()**

- **Exemplu de utilizare** (necesită existența unui fișier munte.jpg în folderul de program)**:**

```
with open("munte.jpg", "rb") as f:
    baiti=f.read()
    print(baiti)
```

# Înscrierea datelor în fișiere binare

- **Pentru înscrierea conținutului într-un fișier binar se utilizează metoda write() cu sintaxa:**

*nume_variabila*.write(*nume_variabila_baiti*)

- **Exemplu de utilizare:**

```
with open("munte.jpg", "rb") as f:
    baiti=f.read()
with open("munte2.jpg", 'wb') as f1:
    f1.write(baiti)
    print("O noua imagine este disponibila cu numele: munte2.jpg")
```

# 2. Arhivarea și dezarhivarea fișierelor

## Arhivarea zip a fișierelor în Python

- **Clasa ZipFile – permite arhivarea și dezarhivarea zip a fișierelor**
- **Sintaxa creării unui obiect ZipFile pentru arhivarea fișierelor:**

  ***nume_variabila_arhiva*** **=ZipFile(*“nume_arhiva.zip”*, “w”, “ZIP_DEFLATED”)**

- **Adăugarea fișierelor în arhivă se realizează cu metoda *write()* cu sintaxa:**

  ***nume_variabila_arhiva*.write(*nume_fișier*)**

- **Exemplu de utilizare:**

```
from zipfile import *
with ZipFile("arhiva.zip", 'w', ZIP_DEFLATED) as f:
    f.write("Welcome.txt")
    f.write("munte.jpg")
    f.write("munte2.jpg")
    print("Fișierul arhiva.zip a fost creat cu succes")
```

# Dezarhivarea zip a fișierelor în Python

- **Sintaxa creării unui obiect ZipFile pentru dezarhivarea fișierelor:**

  ***nume_variabila_arhiva* =ZipFile(*"nume_arhiva.zip"*, "r", ZIP_STORED)**

- **Metoda *namelist()* permite vizualizarea denumirilor fișierelor arhivate:**

  ***nume_lista_denumiri* = *nume_variabila_arhiva*.namelist()**

- **Metoda *extract()* permite extragerea unui fișier din arhiva conform numelui acestuia:**

  ***nume_variabila_arhiva*.extract(*"nume_fișier"*,)**

- **Metoda *extractall()* permite extragerea tuturor fișierelor arhivate:**

  ***nume_variabila_arhiva*.extractall()**

# Exemplu de dezarhivare zip în Python

- **Exemplu de utilizare:**

```
from zipfile import *

with ZipFile("arhiva.zip", 'r', ZIP_STORED) as f:

    lista_denumiri=f.namelist()
    print(lista_denumiri)

    for nume in lista_denumiri:
        print(nume)

    f.extract('munte.jpg')

    f.extractall()
```

# 3. Foldere în Python

## Folderul de program

- **Modulul os –** permite diferite funcționalități ale sistemului de operare inclusiv lucru cu sistemul de foldere

- **Folderul de program –** folderul în care se află fișierul Python executat

- **Folderul curent –** folderul din care programul citește datele, implicit este forderul de program

- **Metoda *getcwd()* – permite vizualizarea căii folderului curent**

```
import os
cale_program=os.getcwd()
print(f"Folderul de program are calea: {cale_program}")
```

- **Metoda *listdir()* – permite vizualizarea conținutului folderului curent**

```
import os
files = os.listdir()
for f in files:
    print(f)
```

# Folderul curent

- Metoda *chdir("cale")* – permite modificare folderului curent

```
import os
print(f'Calea folderului curent initial este: {os.getcwd()} si contine:')
for f in os.listdir():
    print(f)
os.chdir("E:/PCLP") #în loc de E:/PCLP poate fi cale unui alt folder existent
print(f'Calea noului folder curent este: {os.getcwd()} si contine:')
for f in os.listdir():
    print(f)
```

- Utilizarea fișierelor din diferite foldere

```
import os
with open("Welcome.txt", 'r') as f:
    data=f.read()
    print(data)
os.chdir("E:/PCLP")
with open("Welcome.txt", 'r') as f:
    data=f.read()
    print(data)
```

# Conținutul unui folder

- Metoda *walk("cale_folder")* – permite vizualizarea conținutului tuturor subfolderelor

```
import os
calea_curenta = os.getcwd()
print(calea_curenta)
for root, dirs, files in os.walk(calea_curenta):
    print(root)
    print(dirs)
    for fisiere in files:
        print(fisiere)
```

# Crearea folderelor

- Metoda *mkdir()* – permite crearea unui subfolder în folderul curent

```
import os
os.mkdir("SubFolder")
print("Subfolderul a fost creat in folderul de curent")
```

- Metoda *makedirs()* – permite crearea unei ierarhii de foldere în folderul curent

```
import os
os.makedirs("SubFolder1/SubFolder2/SubFolder3")
print("In folderul curent a fost creat SubFolder1 care contine
SubFolder2, care la randul lui contine SubFolder3")
```

# Ștergerea în folderelor

- Metoda remove*(nume_fisier)* – permite ștergerea unui fișier din folderul curent

```
import os
os.remove('munte2.jpg')
print("Fișierul 'munte2.jpg' a fost sters cu succes")
```

- Metoda *rmdir()* – permite ștergerea unui subfolder gol din folderul curent

```
import os
os.rmdir("SubFolder")
print("Folderul SubFolder a fost sters")
```

- Metoda *removedirs()* – permite ștergerea unei ierarhii de subfoldere goale din folderul curent

```
import os
os.removedirs("SubFolder1/SubFolder2/SubFolder3")
print("Ierarhia de foldere goale a fost stearsa")
```

# Verificarea prezenței unui fișier

- **Submodulul path –** componentă a modului os ce permite diferite funcționalități de gestionare a căilor către fișiere/foldere

- **Metoda isfile() –** o metodă a submodului path ce verifică prezența fișierului în folderul curent

- **Exemplu de utilizare:**

```
import os
nume_fisier=input("Introduceti numele fisierului: ")
if os.path.isfile(nume_fisier):
    with open(nume_fisier, "r") as f:
        data=f.read()
        print(data)
else:
    print(f"Fisierul '{nume_fisier}' nu exista în folderul curent")
```

# Metode ale submodulului path

- **Metoda dirname() –** reîntoarce calea folderului parinte a folderului curent
- **Metoda exists() –** verifică daca calea introdusă este una valabilă în sistemul de foldere
- **Metoda isdir() –** verifică daca calea introdusă este o cale către un alt folder sau un fișier
- **Exemplu de utilizare:**

```
import os
cale = os.getcwd()
print(cale)
print(os.path.dirname(cale))
print(os.path.exists(cale))
print(os.path.isdir('Welcome.txt'))
```