

## Lucrarea 4. Librăria Pandas

### A. Obiectivele lucrării

1. Familiarizarea cu funcționalitatea și sintaxa librăriei **Pandas**.
2. Analiza unui set de date real, utilizând funcționalitățile librăriei **Pandas**

### B. Introducere

**Pandas** (prescurtare de la **Python Data Analysis**) este o librărie **Python** utilizată pentru manipularea și analiza datelor. În principiu, ea oferă structuri de date și funcții pentru manipularea tabelelor și a seriilor de date, oferind posibilitatea de a analiza, explora, curăța și manipula cantități mari de date. Utilizând **Pandas** se pot trage concluzii pe baza analizei statistice a cantităților mari de date [4].

**Pandas** are la bază **NumPy**, de la care adoptă mare parte a sintaxei, doar că, spre deosebire de **NumPy** care este construit astfel încât să lucreze cu matrice cu valori numerice de același tip (deci matrice omogenă), **Pandas** poate lucra cu date eterogene, adică în aceeași structură de date pot fi tipuri de date diferite.

După instalarea librăriei **Pandas** (aşa cum s-a văzut în prima lucrare), pentru a fi folosită într-un program aceasta trebuie importată scriind la începutul programului instrucțunea `import` urmată de numele librăriei (`pandas`). Totuși, pentru a economisi timp, putem să utilizăm un alias mai scurt pentru numele librăriei, în mod uzual, pentru librăria **Pandas** folosindu-se alias-ul `pd`:

```
main.py
import pandas as pd
```

Pe lângă structurile de date din limbajul de bază **Python** (liste, tupluri, seturi și dicționare), **Pandas** introduce două noi structuri [4]:

*Tabelul 1 – Structuri de date definite de Pandas*

Structură de date	Descriere
Series	tablou unidimensional, etichetat, ce poate conține date de orice tip
DataFrame	tablou bidimensional (tabel), cu ambele axe (rânduri și coloane) etichetate.

### C. Serii de date

Seriile de date (*Series*) sunt tablouri unidimensionale etichetate. Acestea pot conține orice tip de date, cu condiția ca toate valorile din serie să fie de același tip. Inițializarea seriilor se poate face utilizând funcția *Series* care poate primi unul sau mai mulți parametrii:

1. *data* – obiect iterabil (lista, dicționar, matrice *NumPy*, etc) ce conține valorile proprii zise ale seriei de date; dacă este de tip dicționar, cheile vor reprezenta etichetele valorilor.
2. *index* – vector de aceeași dimensiune cu *data* conținând etichetele; dacă lipsește, atunci seria de date va avea ca etichete valori de la 0 la numărul de elemente.
3. *dtype* – tipul de date al seriei specificat în același mod ca la matricele NumPy; dacă lipsește, tipul de date va fi moștenit de la *data*.

Exemplu de inițializare a seriilor de date utilizând o listă, fără a specifica etichetele:

**main.py**

```
import pandas as pd

A = pd.Series([7, 5, 3])
print(A)
```

**rezultat**

0	7
1	5
2	3
dtype: int64	

Exemplu de inițializare a seriilor de date utilizând o listă, cu etichetele specificate:

**main.py**

```
import pandas as pd

A = pd.Series([7, 5, 3], ["a", "b", "c"])
print(A)
```

**rezultat**

a	7
b	5
c	3
dtype: int64	

Exemplu de inițializare a seriilor de date utilizând un dicționar, fără a specifica etichetele:

**main.py**

```
import pandas as pd

d = {"a": 7, "b": 8, "c": 9}

A = pd.Series(d)
print(A)
```

**rezultat**

```
a    7
b    8
c    9
dtype: int64
```

Exemplu de inițializare a seriilor de date utilizând un dicționar, cu etichetele specificate:

**main.py**

```
import pandas as pd

d = {"a": 7, "b": 8, "c": 9}

A = pd.Series(d, ["c", "d", "e", "a", "b"])
print(A)
```

**rezultat**

```
c    9.0
d    NaN
e    NaN
a    7.0
b    8.0
dtype: float64
```

Indexarea elementelor seriilor de date se face în mod asemănător indexării dicționarelor (adică specificând etichetele între paranteze pătrate). Exemplu:

**main.py**

```
import pandas as pd

note = pd.Series({"Ion": 7, "Vasile": 8, "George": 9})
note["Ion"] = 2

print(note["Vasile"])
```

**rezultat**

```
8
```

Seriile de date acționează foarte asemănător cu matricele unidimensionale *NumPy*, un obiect de tip **Series** fiind un argument valid pentru marea majoritate a funcțiilor definite de

**NumPy**. De asemenea, seriile de date pot fi convertite în matrice unidimensionale **NumPy** utilizând metoda `to_numpy()`:

**main.py**

```
import pandas as pd
import numpy as np

date1 = pd.Series([1, 2, 3, 4, 5, 6])
date2 = np.sqrt(date1)

print(date2)
print(date2.to_numpy())
```

**rezultat**

```
0    1.000000
1    1.414214
2    1.732051
3    2.000000
4    2.236068
5    2.449490
dtype: float64
[1.        1.41421356 1.73205081 2.          2.23606798 2.44948974]
```

Între două serii de date se pot aplica oricare din operațiile aritmetice de bază (adunare, scădere, înmulțire, împărțire, ridicare la putere), caz în care operațiile se vor realiza element cu element, cu mențiunea că cele două serii vor fi aliniate după etichete:

**main.py**

```
import pandas as pd

note_examen = pd.Series({"Ion":7, "Vasile":8, "George":9, "Alin":2})
note_laborator = pd.Series({"Ion": 2, "George":10, "Vasile":6})

media = (note_examen + note_laborator) / 2
print(media)
```

**rezultat**

```
Alin      NaN
George    9.5
Ion       4.5
Vasile    7.0
```

**Pandas** definește, pentru obiectele de tip **Series**, o serie de metode pentru manipularea și analiza datelor [4]. Cele mai uzuale pot fi observate în tabelul următor:

Tabelul 2 - Metode pentru manipularea și analiza seriilor de date

<b>Metode pentru manipularea seriilor de date</b>	
X.count()	returnează numărul total de elemente diferite de <b>NaN/null</b> din serie
X.drop(l)	returnează o copie a seriei cu elementele ale căror etichete sunt specificate de lista l sterse
X.drop_duplicates()	returnează o copie a seriei cu elementele cu valori duplicate eliminate
X.pop(i)	sterge elementul cu eticheta i din serie
X.astype(d)	convertește seria la tipul de date specificat de d
<b>Metode pentru analiza statistică a datelor</b>	
X.min()	returnează valoarea minimă dintr-o serie
X.max()	returnează valoarea maximă dintr-o serie
X.mean()	returnează valoarea media dintr-o serie
X.median()	returnează mediana unei serii
X.mode()	returnează modul unei serii
X.std()	returnează deviația standard a unei serii
X.describe()	generează o statistică descriptivă a datelor dintr-o serie

## D. DataFrame

DataFrame-ul este o structură de date bidimensională, cu ambele axe (rânduri și coloane) etichetate. Practic, obiectele de tip **DataFrame** pot fi văzute ca serii de serii. Acestea pot conține orice tip de date, cu condiția ca toate valorile de pe o coloană să fie de același tip.

Inițializarea **DataFrame**-urilor se poate face utilizând funcția `DataFrame` care poate primi unul sau mai mulți parametrii [4]:

1. `data` – dicționar de iterabile (liste, serii de date etc); cheile dicționarului vor reprezenta etichetele coloanelor, în timp ce valorile vor reprezenta valorile obiectului pe coloana respectivă.
2. `index` – vector conținând etichetele rândurilor. Dacă lipsește, atunci liniile vor avea ca etichete valori de la 0 la numărul de elemente.

Exemplu:

main.py				
<pre>import pandas as pd</pre>				
df = pd.DataFrame({	"Nume": ["Ion", "Vasile", "Gheorghe"],			
"Nota Examen": [10, 4, 7],				
"Nota Laborator": [9, 8, 6]				
})				
<pre>print(df)</pre>				
rezultat				
	Nume	Nota Examen	Nota Laborator	
0	Ion	10	9	
1	Vasile	4	8	
2	Gheorghe	7	6	

Totuși, pentru că **Pandas** este dedicat manipulării și analizei datelor de mari dimensiuni, cea mai comună metodă de inițializare a unui **DataFrame** este prin importarea dintr-o sursă externă (de obicei fișier CSV). Spre exemplu, dându-se un fișier CSV care conține informații despre emisiunile și filmele Netflix (<https://www.kaggle.com/shivamb/netflix-shows>), importarea acestuia se poate face astfel:

main.py				
<pre>import pandas as pd</pre>				
df = pd.read_csv('netflix_titles.csv')				
<pre>print(df)</pre>				
rezultat				
	show_id	...		description
0	s1	...		As her father nears the end of his life, filmm...
1	s2	...		After crossing paths at a party, a Cape Town t...
2	s3	...		To protect his family from a powerful drug lor...
3	s4	...		Feuds, flirtations and toilet talk go down amo...
4	s5	...		In a city of coaching centers known to train I...
...	...	...		...
8802	s8803	...		A political cartoonist, a crime reporter and a...
8803	s8804	...		While living alone in a spooky town, a young g...
8804	s8805	...		Looking to survive in a world taken over by zo...
8805	s8806	...		Dragged from civilian life, a former superhero...
8806	s8807	...		A scrappy but poor boy worms his way into a ty...
<pre>[8807 rows x 12 columns]</pre>				

Se poate observa că tabelul are 8807 rânduri și 12 coloane, dar afișarea se face trunchiat. De asemenea, implicit, prima linie din fișierul CSV acționează ca și etichete pentru coloane, în timp ce rândurile au fost etichetate cu o secvență de numere de la 0 la 8806.

Pentru că afișarea se face trunchiat, se poate utiliza metoda `info()` [4], pentru a vedea ce coloane sunt disponibile și ce tip de date conțin ele.

### main.py

```
import pandas as pd

df = pd.read_csv('netflix_titles.csv')

df.info()
```

### rezultat

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 8807 entries, 0 to 8806
Data columns (total 12 columns):
 #   Column           Non-Null Count  Dtype  
---  --  
 0   show_id          8807 non-null   object 
 1   type              8807 non-null   object 
 2   title             8807 non-null   object 
 3   director          6173 non-null   object 
 4   cast               7982 non-null   object 
 5   country            7976 non-null   object 
 6   date_added        8797 non-null   object 
 7   release_year      8807 non-null   int64  
 8   rating             8803 non-null   object 
 9   duration            8804 non-null   object 
 10  listed_in          8807 non-null   object 
 11  description         8807 non-null   object 
dtypes: int64(1), object(11)
memory usage: 825.8+ KB
```

Implicit, la importarea unui fișier CSV, etichetele rândurilor sunt generate ca o secvență de numere de la 0 la numărul total de rânduri. Totuși putem seta una din coloanele tabelului ca reprezentând etichetele rândurilor:

### main.py

```
import pandas as pd
df = pd.read_csv('netflix_titles.csv')
df = df.set_index("show_id")
print(df)
```

### rezultat

	type	...	description
show_id	...		
s1	Movie	...	As her father nears the end of his life, fi...
s2	TV Show	...	After crossing paths at a party, a Cape Tow...

De asemenea, dacă în urma importării unui fișier CSV, tipul de date aferent fiecărei coloane nu a fost stabilit în mod corect, utilizând metoda `astype` căreia îi dăm ca parametru un dicționar ale cărui perechi cheie – valoare reprezintă perechi coloană – tip data putem seta tipurile de date corecte. Pe lângă tipurile clasice de date, se pot seta și alte tipuri precum `datetime64[ns]` (care să țină minte o dată) sau `category` (pentru coloanele care specifică categorii).

<b>main.py</b>	
<pre>import pandas as pd  df = pd.read_csv('netflix_titles.csv') df.set_index("show_id", inplace=True)  df = df.astype({     "type": "category",     "release_year": "int8",     "date_added": "datetime64[ns]") df.info()</pre>	
<b>rezultat</b>	
	<pre>&lt;class 'pandas.core.frame.DataFrame'&gt; Index: 8807 entries, s1 to s8807 Data columns (total 11 columns):  #   Column      Non-Null Count  Dtype   ---   0   type        8807 non-null   category  1   title       8807 non-null   object    2   director    6173 non-null   object    3   cast        7982 non-null   object    4   country     7976 non-null   object    5   date_added  8797 non-null   datetime64[ns]  6   release_year 8807 non-null   int32    7   rating      8803 non-null   object    8   duration    8804 non-null   object    9   listed_in   8807 non-null   object    10  description  8807 non-null   object   dtypes: category(1), datetime64[ns](1), int32(1), object(8) memory usage: 731.2+ KB</pre>

### a) Indexarea DataFrame-urilor

Selectarea unei singure coloane se poate face scriind, după numele variabilei de tip `DataFrame`, între paranteze patrate eticheta coloanei respective, rezultând o serie ce conține doar datele de pe acea coloană. Exemplu:

**main.py**

```
import pandas as pd

df = pd.read_csv('netflix_titles.csv')
titluri = df["title"]

print(titluri)

rezultat

0      Dick Johnson Is Dead
1          Blood & Water
2            Ganglands
3    Jailbirds New Orleans
4        Kota Factory
...
8802           Zodiac
8803        Zombie Dumb
8804       Zombieland
8805           Zoom
8806         Zubaan
Name: title, Length: 8807, dtype: object
```

Dacă între parantezele pătrate se dă o listă de etichete de coloane, va rezulta tot un **DataFrame** conținând doar coloanele specificate. Există mai multe modalități de a extrage elemente, rânduri, coloane sau subseturi, dar cele mai eficiente dintre ele includ utilizarea proprietăților `loc` (indexare pe baza etichetelor rândurilor și coloanelor) și `iloc` (indexare pe baza poziției rândurilor și coloanelor) [4]. Exemplu:

**main.py**

```
import pandas as pd

df = pd.read_csv('netflix_titles.csv')
df = df.set_index("show_id")

print(df.iloc[5, 1])
print(df.iloc[5:7, 1:3])
print(df.loc["s6", 'director'])
```

**rezultat**

show_id	title	director
s6	Midnight Mass	Mike Flanagan
s7	My Little Pony: A New Generation	Robert Cullen, José Luis Ucha
		Mike Flanagan

Dacă la utilizarea acestor proprietăți se folosește doar un index / o etichetă, atunci el va reprezenta rânduri, deci astfel putem extrage un rând dintr-un **DataFrame**:

main.py	
<code>import pandas as pd</code>	
<code>df = pd.read_csv('netflix_titles.csv')</code>	
<code>print(df.iloc[0])</code>	
<b>rezultat</b>	
show_id	s1
type	Movie
title	Dick Johnson Is Dead
director	Kirsten Johnson
cast	NaN
country	United States
date_added	September 25, 2021
release_year	2020
rating	PG-13
duration	90 min
listed_in	Documentaries
description	As her father nears the end of his life, film...
Name: 0, dtype: object	

### b) Filtrarea datelor

O sarcină tipică în procesarea datelor este filtrarea acestora, adică extragerea anumitor înregistrări. **Pandas** oferă mai multe metode de a face acest lucru, una din ele fiind utilizarea metodei `query` care primește ca parametru un sir de caractere ce specifică expresia ce descrie filtrarea dorită [4]. Spre exemplu, dacă se dorește selectarea doar a filmelor/emisiunilor lansate după anul 2020:

main.py	
<code>import pandas as pd</code>	
<code>df = pd.read_csv('netflix_titles.csv')</code>	
<code>filme_noi = df.query("release_year &gt; 2020")</code>	
<code>print(filme_noi["title"])</code>	
<b>rezultat</b>	
1	Blood & Water
2	Ganglands
3	Jailbirds New Orleans
4	Kota Factory
	...
1696	Polly Pocket
2920	Love Is Blind
8437	The Netflix Afterparty
Name: title, Length: 592, dtype: object	

Bineînțeles, se pot scrie expresii de filtrare mult mai complexe. Tabelul următor prezintă o serie de astfel de expresii:

*Tabelul 3 – Exemple de expresii pentru filtrarea datelor*

Expresie	Descriere
release_year >= 2020	filmele/emisiunile lansate începând cu anul 2020
release_year == 2020 and type == 'Movie'	filmele lansate în anul 2020
release_year in (1999, 2007)	filmele/emisiunile lansate fie în anul 1999 fie în anul 2007
listed_in.str.contains('Documentaries')	filmele/emisiunile care sunt de tip documentar

### c) Analiza datelor

Bineînțeles, dacă se dorește analiza statistică a datele dintr-o coloană, se vor putea utiliza metodele prezentate anterior, specifice seriilor de date, după ce extragem coloana respectivă. Aceste metode se pot aplica și obiectelor de tip DataFrame, caz în care parametrul statistic dorit se va calcula pentru fiecare coloană ce conține un tip de date asupra căruia se poate aplica metoda (de exemplu maximul se poate aplica pentru date numerice dar și pentru siruri de caractere, în timp ce media sau abaterea standard poate fi calculată doar pentru date numerice).

O metodă utilă la analiza statistică a datelor este metoda `groupby` care primește ca parametru o etichetă de coloană și care permite gruparea datelor și aplicarea metodelor de analiză statistică fiecărui grup. Spre exemplu, dacă se dorește să se afle câte emisiuni/filme au fost lansate în fiecare an, se vor extrage coloanele `release_year` și `title`, se va grupa după coloana `release_year` apoi se va aplica metoda `count`:

<b>main.py</b>	
<code>import pandas as pd</code>	
<code>df = pd.read_csv('netflix_titles.csv')</code>	
<code>print(df[["release_year", "title"]].groupby("release_year").count())</code>	
<b>rezultat</b>	
	<code>title</code>
<code>release_year</code>	
1925	1
1942	2
1943	3
...	...
2020	953
2021	592
	[74 rows x 1 columns]

O altă metodă utilă pentru analiza datelor o reprezintă `corr` care va calcula corelația fiecărei perechi de coloane [4]. Practic, această metodă este utilă pentru a cunoaște gradul de similaritate/dependență între oricare două coloane. Exemplu:

**main.py**

```
import pandas as pd
import numpy as np

df = pd.DataFrame({
    "x": np.arange(5, 10, 1),
    "y": np.sqrt(np.arange(5, 10, 1)),
    "z": [3, 2, 8, 1, 9]
})

print(df.corr())
```

**rezultat**

	x	y	z
x	1.000000	0.999067	0.491935
y	0.999067	1.000000	0.458418
z	0.491935	0.458418	1.000000

#### d) Alte metode pentru manipularea **DataFrame-urilor**

*Pandas* pune la dispoziție o serie de metode pentru manipularea **DataFrame-urilor**. Acestea se aplică obiectelor de tip **DataFrame**, dar nu le modifică, ci returnează un alt **DataFrame** cu modificările dorite. Cele mai uzuale dintre aceste metode sunt descrise în tabelul următor [4].

*Tabelul 4 – Exemple de metode de manipulare a **DataFrame-urilor***

Metoda	Descriere
X.drop(labels, axis)	șterge rândurile/coloanele etichetate cu etichetele specificate de parametrul <code>labels</code> ; parametrul <code>axis</code> specifică dacă e vorba de rânduri (0) sau coloane (1)
X.dropna(axis)	șterge rândurile sau coloanele din care lipsesc date; parametrul <code>axis</code> specifică dacă e vorba de rânduri (0) sau coloane (1)
X.drop_duplicates()	șterge rândurile duplicate
X.merge(Y, on)	unește coloanele <b>DataFrame-ului X</b> cu coloanele <b>DataFrame-ului Y</b> aliniindu-le după coloana specificată de parametrul <code>on</code> ; Dacă parametrul <code>on</code> lipsește, alinierea se va face după etichetele liniilor
X.rename(map, axis)	redenumește etichetele rândurilor/coloanelor conform dicționarului <code>map</code> . parametrul <code>axis</code> specifică dacă e vorba de rânduri (0) sau coloane (1)

Exemplu:

**main.py**

```
df1 = pd.DataFrame({  
    "sensor_id": [1, 2, 4, 3, 2, 1, 1],  
    "value": [22, 0.8, 28, 7, 0.6, 21, 20]  
})  
  
df2 = pd.DataFrame({  
    "id_sensor": [1, 2, 3, 4],  
    "sensor_name": ["Temp1", "Pres", "Temp2", "pH"]  
})  
  
df2 = df2.rename({"id_sensor": "sensor_id"}, axis = 1)  
  
print(df1.merge(df2, on="sensor_id"))
```

**rezultat**

	sensor_id	value	sensor_name
0	1	22.0	Temp1
1	1	21.0	Temp1
2	1	20.0	Temp1
3	2	0.8	Pres
4	2	0.6	Pres
5	4	28.0	pH
6	3	7.0	Temp2

## E. Cerințe

Fie fișierul `date_climatice_2016.csv` ce conține valorile minime, maxime și medii ale temperaturilor înregistrate în fiecare zi a anului 2016 de către cele 23 de stații esențiale [5]. Rezolvați următoarele cerințe:

1. Încărcați datele din fișierul CSV dat într-o variabilă de tip **DataFrame** (`df1`) și afișați atât datele, cât și informațiile despre coloane.
2. Ștergeți din `df1` coloanele LAT, LON și R24.
3. Corectați tipurile de date pentru coloanele DATCLIM (**datetime64[ns]**) și CODST (**category**) și reafișați informațiile despre coloane.
4. Încărcați în variabila `df2` de tip **DataFrame** fișierul `stati_i_meteo.csv` ce conține ID-urile și numele stațiilor meteo și afișați atât datele, cât și informațiile despre coloane.

5. Observăm că coloanele aferente ID-urilor stațiilor din cele două **DataFrame**-uri au nume diferite. Redenumiți în df2 coloana ID în CODST și coloana Nume în NUME.
6. Modificați df1 prin a uni cu acest **DataFrame** coloanele din df2 aliniindu-le pe coloana CODST.
7. Calculați și afișați din df1:
  - a. temperatura medie înregistrată (media coloanei TMED)
  - b. temperatura maximă înregistrată (maximul coloanei TMAX) și numele stației la care a fost ea înregistrată (**indiciu**: pentru numele stației va trebui să utilizați metoda query)
  - c. temperatura minimă înregistrată (minimul coloanei TMIN) și numele stației la care a fost ea înregistrată
8. Creați un nou obiect de tip **DataFrame** numit date\_statistice urmând următorii pași:
  - a. selectați din df1 doar coloanele NUME, ALT și TMIN, grupați după coloana NUME, calculați minimul și puneti rezultatele în date\_statistice.
  - b. selectați din df1 doar coloanele NUME și TMED, grupați după coloana NUME, calculați media și uniți date\_statistice cu rezultatul.
  - c. selectați din df1 doar coloanele NUME și TMAX, grupați după coloana NUME, calculați maximul și uniți date\_statistice cu rezultatul.
  - d. afișați datele din date\_statistice.
9. Studiați corelația temperaturilor minime, medii și maxime cu altitudinea (**indiciu**: aplicați metoda corr asupra obiectului date\_statistice).