

# ACADEMIA DE STUDII ECONOMICE DIN BUCUREȘTI FACULTATEA DE CIBERNETICĂ, STATISTICĂ ȘI INFORMATICĂ ECONOMICĂ



Prof. univ. dr. Adina Ileana UȚĂ  
Studenta: Giol Adriana  
Grupa 1078, Seria C

*București, 2020*

# Conținutul proiectului (partea a II-a – Python)

1. Liste .....	3
2. Dicționare.....	6
3. Seturi .....	8
4. Tupluri .....	10
5. Definirea și apelarea funcțiilor, structuri condiționate și repetitive,grafic.....	11
6. Import CSV .....	14
7. Accesarea datelor cu loc și iloc.....	15
8. Tratarea valorilor lipsă, ștergerea coloanelor și a înregistrărilor.....	16
9. Prelucrarea seturilor de date cu Merge .....	18
10. Gruparea datelor și prelucrarea statistică .....	20
11. Conversia unui tip de dată.....	22
12. Regresia liniară multiplă (pachet statsmodels) .....	23
13. Regresia logistică (pachet scikit-learn ).....	25
14. Clusterizare (pachet scikit-learn ).....	28
Bibliografie .....	33

# PARTEA a II-a – PYTHON

## 1. LISTE

### • Descrierea problemei

Managerul televiziunii ArirangTV dorește să realizeze sezonul 2 pentru serialul "Love Rain". Acesta dorește o listă cu actorii principali din acest serial, vrând să facă următoarele modificări asupra ei:

- Dorește crearea listei cu actori principali.
- Vrea să știe câți actori principali sunt.
- Se gândește ca pentru creșterea audienței să o aducă pe actrița Hwang BoRa în lista de actori principali, dar ca rezervă, deci la urma listei.
- Dorește, de asemenea, să îl aducă pe actorul Seo In Guk printre actorii principali.
- Vrea să o elimine pe actrița Im Yoon Ah din distribuție, iar pe actorul din rol principal Jang Geun Suk vrea să îl înlocuiască cu actorul Yoo Seung Ho.
- În urma concursului desfășurat pe site-ul televiziunii ArirangTV lista celor mai apreciați actori a fost inversată, totuși actorul de pe ultima poziție a primit premiul de popularitate.
- La final, managerul dorește ca lista să fie ștearsă din motive de confidențialitate.

### • Informații necesare pentru rezolvare

Pentru realizarea acestei liste avem nevoie de următoarele informații :

- ✓ Actorii care joacă în serialul "Love Rain".

### • Produs software / funcție / metodă de calcul folosită

- ✓ Produs software folosit: Anaconda Navigator, JupyterLab
- ✓ Limbaj: Python
- ✓ Funcția: len(), append(), insert(), remove(), reverse(), pop(), clear()
- ✓ Metoda de calcul folosită:

Am creat o listă de actori numită `actoriLoveRain` și am aflat lungimea listei cu funcția `len()`. Mai apoi, am adăugat-o pe actrița Hwang BoRa la sfârșitul listei folosind funcția `append()`. De asemenea, l-am adăugat pe actorul Seo In Guk pe poziția 3 (numerotarea începe de la 0), și am eliminat-o pe actrița Im Yoon Ah din listă, folosind funcția `remove()`.

În lista actuală am înlocuit actorul de pe prima poziție cu actorul Yoo Seung Ho. Am inversat mai apoi elementele listei de actori cu ajutorul funcției `reverse()`.

Apoi am extras actorul de pe poziția 6 (locul 7 – se începe numerotarea de la 0) cu funcția `pop()`, iar în final, cu ajutorul funcției `clear()`, am golit lista de actori.



- **Codul utilizat**

```
#LUCRUL CU LISTE -> [" "]
```

```
#1. Creare Lista Actori
```

```
actoriLoveRain = ["Jang Geun Suk", "Im Yoon Ah", "Kim Shi Hoo", "Lee Mi Sook", "Know In Ha", "Yoo Hye Ri"]
```

```
print("Lista de actori ai serialului Love Rain este:")
```

```
print(actoriLoveRain)
```

```
#2. Lungime lista actori
```

```
print("Lungime lista:")
```

```
print(len(actoriLoveRain))
```

```
#3. Adaugare actrita Hwang BoRa la sfarsitul listei:
```

```
actoriLoveRain.append("Hwang BoRa")
```

```
print("Dupa adaugarea actritei Hwang BoRa:")
```

```
print(actoriLoveRain)
```

```
#4. Adaugarea actorului Seo In Guk pe pozitia 3 (numerotarea incepe de la 0)
```

```
actoriLoveRain.insert(3, "Seo In Guk")
```

```
print("Dupa adaugarea actorului Seo In Guk pe pozitia 3:")
```

```
print(actoriLoveRain)
```

```
#5. Eliminarea actritei Im Yoon Ah din lista
```

```
actoriLoveRain.remove("Im Yoon Ah")
```

```
print("Dupa eliminarea actritei Im Yoon Ah:")
```

```
print(actoriLoveRain)
```

```
#6. Inlocuirea primului actor cu actorul Yoo Seung Ho
```

```
actoriLoveRain[0] = "Yoo Seung Ho"
```

```
print("Dupa inlocuirea primului actor:")
```

```
print(actoriLoveRain)
```

```
#7. Inversarea elementelor listei de actori
```

```
actoriLoveRain.reverse()
```

```
print("Dupa inversarea listei de actori:")
```

```
print(actoriLoveRain)
```

```
#8. Extragerea actorului de pe pozitia 5 (locul 6 - numerotarea se incepe de la 0)
```

```
print("Actorul de pe pozitia 5 este:")
```

```
actoriLoveRain.pop(5)
```

```
#9. Golire lista
```

```
actoriLoveRain.clear()
```

```
print("Lista de actori se va sterge:")
```

```
print(actoriLoveRain)
```

## • Rezultatele obținute

```

In [1]: #LUCRUL CU LISTE -> [" "]
#1. Creare Lista Actori
actoriLoveRain = ["Jang Geun Suk","Im Yoon Ah","Kim Shi Hoo", "Lee Mi Sook", "Know In Ha","Yoo Hye Ri"]
print("Lista de actori ai serialului Love Rain este:")
print(actoriLoveRain)

Lista de actori ai serialului Love Rain este:
['Jang Geun Suk', 'Im Yoon Ah', 'Kim Shi Hoo', 'Lee Mi Sook', 'Know In Ha', 'Yoo Hye Ri']

In [2]: #2. Lungime lista actori
print("Lungime lista:")
print(len(actoriLoveRain))

Lungime lista:
6

In [3]: #3.Adaugare actrita Hwang BoRa La sfarsitul Listei:
actoriLoveRain.append("Hwang BoRa")
print("Dupa adaugarea actritei Hwang BoRa:")
print(actoriLoveRain)

Dupa adaugarea actritei Hwang BoRa:
['Jang Geun Suk', 'Im Yoon Ah', 'Kim Shi Hoo', 'Lee Mi Sook', 'Know In Ha', 'Yoo Hye Ri', 'Hwang BoRa']

In [4]: #4. Adaugarea actorului Seo In Guk pe pozitia 3 (numerotarea incepe de la 0)
actoriLoveRain.insert(3,"Seo In Guk")
print("Dupa adaugarea actorului Seo In Guk pe pozitia 3:")
print(actoriLoveRain)

Dupa adaugarea actorului Seo In Guk pe pozitia 3:
['Jang Geun Suk', 'Im Yoon Ah', 'Kim Shi Hoo', 'Seo In Guk', 'Lee Mi Sook', 'Know In Ha', 'Yoo Hye Ri', 'Hwang BoRa']

In [5]: #5. Eliminarea actritei Im Yoon Ah din Lista
actoriLoveRain.remove("Im Yoon Ah")
print("Dupa eliminarea actritei Im Yoon Ah:")
print(actoriLoveRain)

Dupa eliminarea actritei Im Yoon Ah:
['Jang Geun Suk', 'Kim Shi Hoo', 'Seo In Guk', 'Lee Mi Sook', 'Know In Ha', 'Yoo Hye Ri', 'Hwang BoRa']

In [6]: #6. Inlocuirea primului actor cu actorul Yoo Seung Ho
actoriLoveRain[0]="Yoo Seung Ho"
print("Dupa inlocuirea primului actor:")
print(actoriLoveRain)

Dupa inlocuirea primului actor:
['Yoo Seung Ho', 'Kim Shi Hoo', 'Seo In Guk', 'Lee Mi Sook', 'Know In Ha', 'Yoo Hye Ri', 'Hwang BoRa']

In [7]: #7. Inversarea elementelor Listei de actori
actoriLoveRain.reverse()
print("Dupa inversarea listei de actori:")
print(actoriLoveRain)

Dupa inversarea listei de actori:
['Hwang BoRa', 'Yoo Hye Ri', 'Know In Ha', 'Lee Mi Sook', 'Seo In Guk', 'Kim Shi Hoo', 'Yoo Seung Ho']

In [8]: #8. Extragerea actorului de pe pozitia 5 (locul 6 - numerotarea se incepe de la 0)
print("Actorul de pe pozitia 5 este:")
actoriLoveRain.pop(5)

Actorul de pe pozitia 5 este:

Out[8]: 'Kim Shi Hoo'

In [9]: #9. Golire lista
actoriLoveRain.clear()
print("Lista de actori se va sterge:")
print(actoriLoveRain)

Lista de actori se va sterge:
[]

```

## • Interpretare:

În urma aplicării funcțiilor menționate mai sus asupra listei de actori, se poate observa că toate cerințele managerului televiziunii ArirangTV au fost îndeplinite cu exactitate, iar lista actorilor a fost ștearsă la final.

## 2. DICTIONARE

### • Descrierea problemei

Managerul televiziunii ArirangTV dorește o corespondență a rolurilor serialului "Love Rain" și actorii care le joacă. Având la dispoziție acest dicționar, managerul televiziunii ArirangTV dorește să realizeze următoarele acțiuni asupra acestui dicționar:

- Afișarea dicționarului.
- Afișarea actriței care interpretează rolul Hanei.
- Corespondența dintre actori și personaje.
- Personajele jucate în serialul Love Rain.
- Eliminarea personajului Yoon Hee.
- Actorii finali ai serialului.

### • Informații necesare pentru rezolvare

Pentru realizarea acestui dicționar avem nevoie de următoarele informații :

- ✓ Actorii care joacă în serialul "Love Rain" reprezentând valorile cheilor dicționarului.
- ✓ Rolurile din serialul "Love Rain" reprezentând cheile dicționarului.

### • Produs software / funcție / metodă de calcul folosită

- ✓ Produs software folosit: Anaconda Navigator, JupyterLab
- ✓ Limbaj: Python
- ✓ Funcția: `get()`, `items()`, `keys()`, `pop()`, `values()`
- ✓ Metoda de calcul folosită:

Am creat un dicționar numit `roluriLoveRain` care cuprinde corespondența dintre actorii din serial și rolurile jucate de aceștia. Cu ajutorul funcției `get()` am extras actrița care joacă rolul Hanei în serial.

De asemenea, am afișat corespondența dintre rolurile jucate în serial și actorii care le joacă, cu ajutorul funcției `items()`.

Afișăm, folosind funcția `keys()`, personajele ce sunt jucate în serial, ca mai apoi să eliminăm personajul Yoon Hee, iar la final afișăm actorii finali ai serialului "Love Rain".

### • Codul utilizat

```
#LUCRUL CU DICTIONARE -> {" ":" "}
#1. Creare dictionar si afisarea lui
roluriLoveRain = {"Jun":"Jang Geun Suk","Hana":"Im Yoon Ah","Sun Ho":"Kim Shi Hoo","Yoon Hee":"Lee Mi Sook","Dong Wook":"Know In Ha","Hye Jun":"Yoo Hye Ri"}
print("Dicționar:")
print(roluriLoveRain)

#2. Afisarea actritei care joaca rolul Hanei
print("Rolul Hanei este jucat de actrita:")
print(roluriLoveRain.get("Hana"))

#3. Corespondenta dintre actori si roluri
print("Corespondenta dintre actori si roluri:")
print(roluriLoveRain.items())

#4. Personajele serialului Love Rain
print("Personajele serialului Love Rain sunt:")
print(roluriLoveRain.keys())


#5. Eliminarea personajului Yoon Hee
print("Dupa eliminarea personajului Yoon Hee:")
roluriLoveRain.pop('Yoon Hee')
```

```
print(roluriLoveRain)
```

```
#6. Actorii Serialului Love Rain
```

```
print("Actorii serialului Love Rain sunt:")
print(roluriLoveRain.values())
```

## • Rezultatele obținute

jupyter 2.Dictionare Last Checkpoint: Last Tuesday at 1:39 AM (unsaved changes)  Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [2]: 

```
#LUCRUL CU DICTIONARE -> {" ":" "}
#1. Creare dictionar si afisarea lui
roluriLoveRain = {"Jun":"Jang Geun Suk","Hana":"Im Yoon Ah","Sun Ho":"Kim Shi Hoo",
                  "Yoon Hee":"Lee Mi Sook","Dong Wook":"Know In Ha","Hye Jun":"Yoo Hye Ri"}
print("Dictionar:")
print(roluriLoveRain)
```

Dictionar:  
{'Jun': 'Jang Geun Suk', 'Hana': 'Im Yoon Ah', 'Sun Ho': 'Kim Shi Hoo', 'Yoon Hee': 'Lee Mi Sook', 'Dong Wook': 'Know In H a', 'Hye Jun': 'Yoo Hye Ri'}

In [3]: 

```
#2. Afisarea actritei care joaca rolul Hanei
print("Rolul Hanei este jucat de actrita:")
print(roluriLoveRain.get("Hana"))
```

Rolul Hanei este jucat de actrita:  
Im Yoon Ah

In [4]: 

```
#3. Corespondenta dintre actori si roluri
print("Corespondenta dintre actori si roluri:")
print(roluriLoveRain.items())
```

Corespondenta dintre actori si roluri:  
dict\_items([('Jun', 'Jang Geun Suk'), ('Hana', 'Im Yoon Ah'), ('Sun Ho', 'Kim Shi Hoo'), ('Yoon Hee', 'Lee Mi Sook'), ('Dong Wook', 'Know In Ha'), ('Hye Jun', 'Yoo Hye Ri')])

In [5]: 

```
#4. Personajele serialului Love Rain
print("Personajele serialului Love Rain sunt:")
print(roluriLoveRain.keys())
```

Personajele serialului Love Rain sunt:  
dict\_keys(['Jun', 'Hana', 'Sun Ho', 'Yoon Hee', 'Dong Wook', 'Hye Jun'])

In [6]: 

```
#5. Eliminarea personajului Yoon Hee
print("Dupa eliminarea personajului Yoon Hee:")
roluriLoveRain.pop('Yoon Hee')
print(roluriLoveRain)
```

Dupa eliminarea personajului Yoon Hee:  
{'Jun': 'Jang Geun Suk', 'Hana': 'Im Yoon Ah', 'Sun Ho': 'Kim Shi Hoo', 'Dong Wook': 'Know In Ha', 'Hye Jun': 'Yoo Hye Ri'}

In [7]: 

```
#6. Actorii Serialului Love Rain
print("Actorii serialului Love Rain sunt:")
print(roluriLoveRain.values())
```

Actorii serialului Love Rain sunt:  
dict\_values(['Jang Geun Suk', 'Im Yoon Ah', 'Kim Shi Hoo', 'Know In Ha', 'Yoo Hye Ri'])

## • Interpretare:

*În urma aplicării funcțiilor menționate mai sus asupra dicționarului actorilor, se poate observa că toate cerințele managerului televiziunii ArirangTV au fost îndeplinite cu exactitate.*



### 3. SETURI

#### • Descrierea problemei

Managerul televiziunii ArirangTV dorește să știe ce directori de seriale au colaborat cu televiziunea Arirang în anul 2019, respectiv în anul 2020. Având la dispoziție cele două seturi de date, managerul dorește să realizeze următoarele acțiuni asupra acestora:

- Afișarea celor două seturi.
- Adăugarea unui director nou, Park Shin Hye, în colecția anului 2020.
- Afișarea directorilor care se regăsesc printre colaborările anului 2020, dar nu și ale anului 2019.
- Lista directorilor ce au avut colaborări cu Arirang TV atât în anul 2019, cât și în 2020.
- Elementele necomune celor doi ani.
- Afișarea tuturor directorilor, atât din anul 2019 cât și 2020.

#### • Informații necesare pentru rezolvare

Pentru realizarea acestor seturi avem nevoie de următoarele informații :

- ✓ Directorii de seriale colaboratori cu Arirang TV în anul 2019, cât și în anul 2020.

#### • Produs software / funcție / metodă de calcul folosită

- ✓ Produs software folosit: Anaconda Navigator, JupyterLab
- ✓ Limbaj: Python
- ✓ Funcția: `add()`, `difference()`, `intersection()`, `symmetric_difference()`, `union()`.
- ✓ Metoda de calcul folosită:

Am creat două seturi de directori, corespondente celor 2 ani, numite `setDirectori2019` și `setDirectori2020`. Mai apoi, l-am adăugat pe directorul Park Shin Hye în setul de directori colaboratori ai anului 2020, folosind funcția `add()`.

De asemenea, am descoperit actorii ce au fost adăugați exclusiv în anul 2020 cu ajutorul funcției `difference()`, scăzând din directorii colaboratori ai anului 2020 pe directorii colaboratori ai anului 2019.

Mai apoi, cu ajutorul funcției `intersection()`, am aflat ce directorii au colaborat atât în anul 2020, cât și în anul 2019, după am aflat ce directorii au contribuit la realizarea serialelor doar într-un singur an, fie el 2019 sau 2020, cu ajutorul funcției `symmetric_difference()`.

La final, am afișat toți directorii care au colaborat cu televiziunea în cei doi ani, cu ajutorul funcției `union()`.

#### • Codul utilizat

```
#LUCRUL CU SETURI -> {" "}
#1. Crearea unui set care contine directorii care au colaborat cu televiziunea Arirang in 2019
setDirectori2019={"Jo Young Kwang","Park Yong Soon","Lee Chang Min","Choi Jung Gyu","Jung Dong Yoon","Kim Young Jo"}
print("Directorii colaboratori in 2019:")
print(setDirectori2019)

#2. Crearea unui set care contine directorii care colaboreaza cu televiziunea Arirang in 2020
setDirectori2020={"Jung Jung Hwa", "Choi Jung Gyu", "Yoon Sung Shik", "Kim Young Jo","Park Eun Young", "Jo Soo Won"}
print("Directorii colaboratori in 2020:")
print(setDirectori2020)

#3. Adaugarea unui director nou in lista anului 2020
setDirectori2020.add("Park Shin Hye")
print("Update Directori 2020:")
print(setDirectori2020)

#4. Directorii care au fost adaugati in 2020
print("Directorii adaugati in 2020:")
print(setDirectori2020.difference(setDirectori2019))

#5. Directorii care au avut colaborari atat in 2019 cat si in 2020 cu televiziunea Arirang TV
```



```
print("Directorii care au avut colaborari cu Arirang TV si in 2019, cat si in 2020")
print(setDirectorii2020.intersection(setDirectorii2019))
#6.Elementele necomune celor 2 seturi
print("Elementele necomune celor 2 seturi:")
print(setDirectorii2020.symmetric_difference(setDirectorii2019))

#7. Afisarea tuturor directorilor care au colaborat in 2020 cu ArirangTV
totiDirectorii=setDirectorii2020.union(setDirectorii2019)
print("Toti directorii:")
print(totiDirectorii)
```

## • Rezultatele obținute

Jupyter 3.Seturi Last Checkpoint: Last Tuesday at 1:42 AM (autosaved) Python 3

File Edit View Insert Cell Kernel Widgets Help Trusted

In [1]: #1. Crearea unui set care contine directorii care au colaborat cu televiziunea Arirang in 2019  
 setDirectorii2019={"Jo Young Kwang","Park Yong Soon","Lee Chang Min","Choi Jung Gyu","Jung Dong Yoon","Kim Young Jo"}  
 print("Directorii colaboratori in 2019:")  
 print(setDirectorii2019)

Directorii colaboratori in 2019:  
 {'Choi Jung Gyu', 'Jung Dong Yoon', 'Park Yong Soon', 'Lee Chang Min', 'Jo Young Kwang', 'Kim Young Jo'}

In [2]: #2. Crearea unui set care contine directorii care colaboreaza cu televiziunea Arirang in 2020  
 setDirectorii2020={"Jung Jung Hwa", "Choi Jung Gyu", "Yoon Sung Shik", "Kim Young Jo","Park Eun Young", "Jo Soo Won"}  
 print("Directorii colaboratori in 2020:")  
 print(setDirectorii2020)

Directorii colaboratori in 2020:  
 {'Jo Soo Won', 'Park Eun Young', 'Choi Jung Gyu', 'Yoon Sung Shik', 'Kim Young Jo', 'Jung Jung Hwa'}

In [3]: #3. Adaugarea unui director nou in lista anului 2020  
 setDirectorii2020.add("Park Shin Hye")  
 print("Update Directorii 2020:")  
 print(setDirectorii2020)

Update Directorii 2020:  
 {'Jo Soo Won', 'Park Eun Young', 'Choi Jung Gyu', 'Park Shin Hye', 'Yoon Sung Shik', 'Kim Young Jo', 'Jung Jung Hwa'}

In [4]: #4. Directorii care au fost adaugati in 2020  
 print("Directorii adaugati in 2020:")  
 print(setDirectorii2020.difference(setDirectorii2019))

Directorii adaugati in 2020:  
 {'Jo Soo Won', 'Park Eun Young', 'Park Shin Hye', 'Yoon Sung Shik', 'Jung Jung Hwa'}

In [5]: #5. Directorii care au avut colaborari atat in 2019 cat si in 2020 cu televiziunea Arirang TV  
 print("Directorii care au avut colaborari cu Arirang TV si in 2019, cat si in 2020")  
 print(setDirectorii2020.intersection(setDirectorii2019))

In [6]: #6.Elementele necomune celor 2 seturi  
 print("Elementele necomune celor 2 seturi:")  
 print(setDirectorii2020.symmetric\_difference(setDirectorii2019))

Elementele necomune celor 2 seturi:  
 {'Jo Soo Won', 'Park Eun Young', 'Jung Dong Yoon', 'Lee Chang Min', 'Park Yong Soon', 'Jo Young Kwang', 'Park Shin Hye', 'Yoon Sung Shik', 'Jung Jung Hwa'}

In [7]: #7. Afisarea tuturor directorilor care au colaborat in 2020 cu ArirangTV  
 totiDirectorii=setDirectorii2020.union(setDirectorii2019)  
 print("Toti directorii:")  
 print(totiDirectorii)

Toti directorii:  
 {'Jo Soo Won', 'Park Eun Young', 'Choi Jung Gyu', 'Jung Dong Yoon', 'Park Yong Soon', 'Lee Chang Min', 'Jo Young Kwang', 'Park Shin Hye', 'Yoon Sung Shik', 'Kim Young Jo', 'Jung Jung Hwa'}

## • Interpretare:

În urma aplicării funcțiilor menționate mai sus asupra seturilor de directori, se poate observa că toate cerințele managerului televiziunii ArirangTV au fost îndeplinite cu exactitate.

## 4. TUPLURI

### • Descrierea problemei

Managerul televiziunii ArirangTV dorește să știe top-ul primelor 5 seriale ce vor apărea în 2020 și care sunt cele mai așteptate seriale de către public de aceea, dorește crearea unui tuplu pentru această cerință. Mai dorește să știe pe ce loc se află serialul "Forest" în acest top.

### • Informații necesare pentru rezolvare

Pentru realizarea acestui dicționar avem nevoie de următoarele informații :

✓ Serialele ce vor apărea în 2020.

### • Produs software / funcție / metodă de calcul folosită

✓ Produs software folosit: Anaconda Navigator, JupyterLab

✓ Limbaj: Python

✓ Funcția: index()

✓ Metoda de calcul folosită:

Am creat un tuplu care cuprinde topul serialelor ce vor apărea în anul 2020. Cum tuplurile sunt structuri de date nemodificabile, la nivel de manipulare, am realizat afișarea tuplului și numărul de ordine al serialului "Forest", cu ajutorul funcției index().

### • Codul utilizat

```
#LUCRUL CU TUPLURI -> (" ")
#1. Crearea tuplului si afisarea lui
seriale2020=("Memorist", "The Cursed", "When the Weather is Fine","Forest", "Dr.Romantic S2")
print("Seriale aparute in 2020:")
print(seriale2020)

#2. Numarul de ordine al serialului "Forest"
print("Numarul de ordine al serialului Forest este:")
print(seriale2020.index("Forest")+1)
```

### • Rezultatele obținute

The screenshot shows the JupyterLab interface with the following content:

**File Edit View Insert Cell Kernel Widgets Help** Trusted Python 3

**In [1]:**

```
#LUCRUL CU TUPLURI -> (" ")
#1. Crearea tuplului si afisarea lui
seriale2020=("Memorist", "The Cursed", "When the Weather is Fine","Forest", "Dr.Romantic S2")
print("Seriale aparute in 2020:")
print(seriale2020)
```

Seriale aparute in 2020:  
( 'Memorist', 'The Cursed', 'When the Weather is Fine', 'Forest', 'Dr.Romantic S2' )

**In [2]:**

```
#2. Numarul de ordine al serialului "Forest"
print("Numarul de ordine al serialului Forest este:")
print(seriale2020.index("Forest")+1)
```

Numarul de ordine al serialului Forest este:  
4

### • Interpretare:

Se poate observa, în urma aplicării funcției index(), că serialul "Forest" se află pe locul 4 în topul celor mai dorite seriale din 2020 de către public.

## 5.DEFINIREA ȘI APELAREA FUNCȚIILOR, STRUCTURI CONDITIONATE ȘI REPETITIVE, GRAFIC

### • Descrierea problemei

Managerul televiziunii ArirangTV dorește să le majoreze actorilor salariul în funcție de vârstă, așa că are nevoie de informații despre actorii care joacă în producțiile televiziunii Arirang TV. De asemenea, managerul dorește la final un grafic reprezentativ structurii salariale ale actorilor principali din producție.

### Informații necesare pentru rezolvare

Pentru realizarea acestor funcții avem nevoie de următoarele informații :

- ✓ **Actori:** cuprinde numele actorilor înregistrați.
- ✓ **Vârste:** vârstele actorilor.
- ✓ **Salarii:** salariile obținute de aceștia în ultimul an.
- ✓ **NrScene:** numărul scenelor la care au participat în decursul ultimului an.

### • Produs software / funcție / metodă de calcul folosită

- ✓ **Produs software folosit:** Anaconda Navigator, JupyterLab
- ✓ **Limbaj:** Python
- ✓ **Funcția:** for(), if-else()  
Funcții utilizate: afisare-actori(), majorare\_salarii(), apel\_funcții(), generare\_piechart()
- ✓ **Metoda de calcul folosită:**

1. Am creat 4 liste ce cuprind informațiile despre actori și anume numele, vâsta salariul și numărul de scene jucate în decursul unui an. Cu ajutorul funcției afisare\_actori() vom afișa detaliile anterior menționate despre actori, în mod unitar. Parcurgerea listelor se va realiza cu o instrucțiune for().
2. A doua funcție implementată, majorare\_salarii() va fi utilizată pentru majorarea salariilor actorilor în felul următor:
  - dacă vârsta este mai mică de 30 de ani, bonusul va fi de 100 de dolari pentru fiecare scenă jucată.
  - dacă vârsta este mai mare de 30 de ani, dar mai mică decât 40, bonusul va fi de 200 de dolari pentru fiecare scenă jucată.
  - dacă vârsta este de minim 40 de ani, bonusul va fi de 300 de dolari pentru fiecare scenă jucată.
3. Cea de a treia funcție, apel\_funcții() are ca scop afișarea datelor inițiale, majorarea salariilor și afișarea datelor actualizate, toate acestea prin apelul funcțiilor create anterior.
4. Generarea graficului a fost realizată cu ajutorul pachetului Matplotlib. Prin intermediul acestuia, am realizat un grafic de tip PieChart, care cuprinde situația salariilor acordate actorilor înregistrați în lista Actori.

### • Codul utilizat

```
#LUCRUL CU FUNCTII, STRUCTURI CONDITIONATE SI REPETITIVE
actori=["Yoo Seung Ho", "Park Min Young", "Park Seo Joon","Lee Jun Ki", "Park Jung Min"]
salarii=[150000,50000,170000,165000,20000]
varste=[26,34,31,37,32]
nrScene=[415,210,356,100,40]

#1. Functie care parcurge listele si afiseaza datele legate de actori in mod unitar
def afisare_actori(actori,salarii,varste,nrScene):
    for i in range(len(actori)):
        actor='Nume:' +actori[i]+' ,Salariu:'+str(salarii[i])+' ,Varsta:'+str(varste[i])+' , numar scene
jucate:' +str(nrScene[i])
        print(actor)
```



```
afisare_actori(actori,salarii,varste,nrScene)
print("\n")
```

#2. Functie care majoreaza salariile actorilor in felul urmator:

#daca varsta este mai mica de 30 de ani, bonusul va fi de 100 de dolari pentru fiecare scena jucata  
 #daca varsta este mai mare de 30 de ani, bonusul va fi de 200 de dolari pentru fiecare scena jucata  
 #daca varsta este mai mare de 40 de ani, bonusul va fi de 300 de dolari pentru fiecare scena jucata

```
def majorare_salarii(salarii, varste, nrScene):
    for i in range(len(actori)):
        if varste[i]<30:
            salarii[i]+=100*nrScene[i]
        elif varste[i]>=30 and varste[i]<40:
            salarii[i]+=200*nrScene[i]
        else:
            salarii[i]+=300*nrScene[i]
```

#3. Functia care afiseaza datele initiale, va majora salariile, iar apoi va afisa datele actualizate

```
def apel_functii(actori,salarii,varste,nrScene):
    print("Date initiale:")
    afisare_actori(actori,salarii,varste,nrScene)
    majorare_salarii(salarii,varste,nrScene)
    print("Date actualizate:")
    afisare_actori(actori,salarii,varste,nrScene)
apel_functii(actori,salarii,varste,nrScene)
```

```
import pandas as pd
import matplotlib.pyplot as plt
```

#4. Functia care genereaza un grafic de tip PieChart referitor la salariile actorilor

```
def generare_piechart(actori,salarii):
    cols=['c','m','r','k','b']
    plt.pie(salarii, labels=actori, colors=cols)
    plt.title('Grafic salarii')
    plt.show()
generare_piechart(actori,salarii)
```

## • Rezultatele obținute

Jupyter 05.Functii si grafic Last Checkpoint: 19 hours ago (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
In [2]: #LUCRUL CU FUNCTII, STRUCTURI CONDITIONATE SI REPETITIVE
actori=["Yoo Seung Ho", "Park Min Young", "Park Seo Joon", "Lee Jun Ki", "Park Jung Min"]
salarii=[150000,50000,170000,165000,20000]
varste=[26,34,31,37,32]
nrScene=[415,210,356,100,40]

#1. Functie care parcurge listele si afiseaza datele legate de actori in mod unitar
def afisare_actori(actori,salarii,varste,nrScene):
    for i in range(len(actori)):
        actor='Nume:' +actori[i]+' ,Salariu:' +str(salarii[i])+' ,Varsta:' +str(varste[i])+' ,numar scene jucate:' +str(nrScene[i])
        print(actor)
    afisare_actori(actori,salarii,varste,nrScene)
    print("\n")

Nume:Yoo Seung Ho,Salariu:150000,Varsta:26,numar scene jucate:415
Nume:Park Min Young,Salariu:50000,Varsta:34,numar scene jucate:210
Nume:Park Seo Joon,Salariu:170000,Varsta:31,numar scene jucate:356
Nume:Lee Jun Ki,Salariu:165000,Varsta:37,numar scene jucate:100
Nume:Park Jung Min,Salariu:20000,Varsta:32,numar scene jucate:40
```

```
In [3]: #2. Functie care majoreaza salariile actorilor in felul urmator:
        #daca varsta este mai mica de 30 de ani, bonusul va fi de 100 de dolari pentru fiecare scena jucata
        #daca varsta este mai mare de 30 de ani, bonusul va fi de 200 de dolari pentru fiecare scena jucata
        #daca varsta este mai mare de 40 de ani, bonusul va fi de 300 de dolari pentru fiecare scena jucata

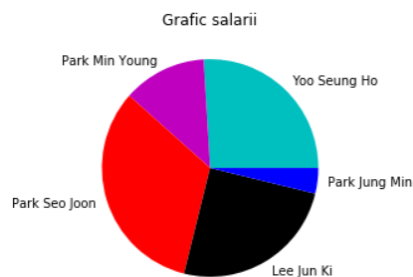
def majoreare_salarii(salarii, varste, nrScene):
    for i in range(len(actors)):
        if varste[i]<30:
            salarii[i]+=100*nrScene[i]
        elif varste[i]>=30 and varste[i]<40:
            salarii[i]+=200*nrScene[i]
        else:
            salarii[i]+=300*nrScene[i]

#3. Functia care afiseaza datele initiale, va majora salariile, iar apoi va afisa datele actualizate
def apel_functii(actors,salarii,varste,nrScene):
    print("Date initiale:")
    afisare_actors(actors,salarii,varste,nrScene)
    majoreare_salarii(salarii,varste,nrScene)
    print("Date actualizate:")
    afisare_actors(actors,salarii,varste,nrScene)
    apel_functii(actors,salarii,varste,nrScene)
```

```
Date initiale:
Nume:Yoo Seung Ho,Salariu:150000,Varsta:26,numar scene jucate:415
Nume:Park Min Young,Salariu:50000,Varsta:34,numar scene jucate:210
Nume:Park Seo Joon,Salariu:170000,Varsta:31,numar scene jucate:356
Nume:Lee Jun Ki,Salariu:165000,Varsta:37,numar scene jucate:100
Nume:Park Jung Min,Salariu:20000,Varsta:32,numar scene jucate:40
Date actualizate:
Nume:Yoo Seung Ho,Salariu:191500,Varsta:26,numar scene jucate:415
Nume:Park Min Young,Salariu:92000,Varsta:34,numar scene jucate:210
Nume:Park Seo Joon,Salariu:241200,Varsta:31,numar scene jucate:356
Nume:Lee Jun Ki,Salariu:185000,Varsta:37,numar scene jucate:100
Nume:Park Jung Min,Salariu:28000,Varsta:32,numar scene jucate:40
```

```
In [6]: import pandas as pd
import matplotlib.pyplot as plt

#4. Functia care genereaza un grafic de tip PieChart referitor la salariile actorilor
def generare_piechart(actors,salarii):
    cols=['c','m','r','k','b']
    plt.pie(salarii, labels=actors, colors=cols)
    plt.title('Grafic salarii')
    plt.show()
    generare_piechart(actors,salarii)
```



### ● Interpretare:

*Se poate observa din rezultate că nu avem actori de peste 40 de ani deci, instrucțiunea if() nu v-a intra pe a treia ramură. Așadar, niciunul din cei 5 actori nu poate primi majorarea salariului cu 300 de dolari, majoritatea actorilor încadrându-se în cea de-a doua ramură a if-ului.*

*De asemenea, din grafic se poate observa că actorul Park Seo Joon este cel mai bine plătit actor din producția televiziunii ArirangTV.*

## 6. IMPORTUL UNUI FIȘIER CSV

### • Descrierea problemei

Managerul televiziunii ArirangTV dorește să importe un fișier CSV pentru a executa anumite operații asupra datelor din acel fișier. Fișierul are în componență date despre primii 5 actori cei mai iubiți din Coreea de Sud, date precum venitul anual, vârsta și numărul de filme și seriale în care au jucat.

Datele înregistrate se pot observa în tabelul următor:

	A	B	C	D	E
1	Top	Actori	Venit	Varsta	Nr filme
2	1	Yoo Seung Ho	191500	26	52
3	2	Park Min Young	92000	34	50
4	3	Park Seo Joon	241200	31	35
5	4	Lee Jun Ki		37	25
6	5	Park Jung Min	28000	32	20

### • Informații necesare pentru rezolvare

Pentru realizarea acestor operații avem nevoie de următoarele informații :

✓ Fișierul CSV.

### • Produs software / funcție / metodă de calcul folosită

✓ Produs software folosit: Anaconda Navigator, JupyterLab, MS Excel

✓ Limbaj: Python

✓ Funcția: `pd.read_csv()`, `print()`

✓ Metoda de calcul folosită:

Am importat fișierul CSV cu ajutorul pachetului Pandas și al funcției `pd.read_csv()` și am afișat datele conținute de acesta cu ajutorul funcției `print()`.

### • Codul utilizat

```
#LUCRUL CU FISIER CSV
#1. Importarea fisierului Actori.csv folosind pachetul pandas
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv('Actori.csv', sep=',',header=0)
print(df)
```

### • Rezultatele obtinute

Jupyter 6. Import csv Last Checkpoint: 04/07/2020 (autosaved)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
In [32]: #LUCRUL CU FISIER CSV
#1. Importarea fisierului Actori.csv folosind pachetul pandas
import pandas as pd
import matplotlib.pyplot as plt
df=pd.read_csv('Actori.csv', sep=',',header=0)
print(df)
```

	Top	Actori	Venit	Varsta	Nr filme
0	1	Yoo Seung Ho	191500.0	26	52
1	2	Park Min Young	92000.0	34	50
2	3	Park Seo Joon	241200.0	31	35
3	4	Lee Jun Ki	NaN	37	25
4	5	Park Jung Min	28000.0	32	20

### • Interpretare:

Se poate observa că valorile au fost introduse asemenea celor din fișierul CSV, iar valoarea lipsă a fost înlocuită cu NaN.



## 7. ACCESAREA DATELOR CU LOC ȘI ILOC

### • Descrierea problemei

Managerul televiziunii ArirangTV știe că pe poziția 2 din topul anterior menționat se află singura actriță din top. Dorește astfel, să știe numele acestei actrițe. De asemenea, managerul știe că această actriță a împlinit anii și dorește să i se actualizeze vârsta.

### • Informații necesare pentru rezolvare

Pentru realizarea acestor operații avem nevoie de următoarele informații :

✓ Datele din fișierul csv.

### • Produs software / funcție / metodă de calcul folosită

- ✓ Produs software folosit: Anaconda Navigator, JupyterLab, MS Excel
- ✓ Limbaj: Python
- ✓ Funcția: `iloc()`, `loc()`
- ✓ Metoda de calcul folosită:

Cu ajutorul funcției `iloc()` am identificat numele actriței din top, de pe poziție 2, și i-am actualizat vârsta cu ajutorul funcției `loc()`.

### • Codul utilizat

```
#2. Folosirea funcției iloc() pentru identificarea numelui actriței
print(df.iloc[1, 1]) #valoarea din rândul 2(pozitia 2), coloana 2(coloana cu numele actorilor)
```

```
#3. Modificarea varstei pentru actrita Park Min Young cu ajutorul funcției loc()
print(df.loc[1, 'Varsta'])
df.loc[1, 'Varsta'] = 35
print(df.loc[1, 'Varsta'])
```

### • Rezultatele obținute

```
In [20]: #2. Folosirea funcției iloc() pentru identificarea numelui actriței
print(df.iloc[1, 1]) #valoarea din rândul 2(pozitia 2), coloana 2(coloana cu numele actorilor)

Park Min Young

In [21]: #3. Modificarea varstei pentru actrita Park Min Young cu ajutorul funcției loc()
print(df.loc[1, 'Varsta'])
df.loc[1, 'Varsta'] = 35
print(df.loc[1, 'Varsta'])

34
35
```

### • Interpretare:

Se poate observa că numele actriței identificate este Park Min Young și aceasta a împlinit vârsta de 35 de ani.

## 8. TRATAREA VALORILOR LIPSĂ, ȘTERGEREA COLONELOR ȘI A ÎNREGISTRĂRIILOR

### • Descrierea problemei

Managerul televiziunii ArirangTV dorește să șteargă coloana cu numărul filmelor în care cei 5 actori au jucat și să păstreze doar top-ul, numele, vârsta și salariul acestora. De asemenea, vrea să verifice dacă există vreun actor care nu și-a declarat venitul și să înlocuiască posibila lipsă a valorii cu mesajul "Declarație nerealizată" și să șteargă actorul care nu și-a declarat venitul din top 5.

### • Informații necesare pentru rezolvare

Pentru realizarea acestor operații avem nevoie de următoarele informații :

✓ Datele din fișierul CSV.

### • Produs software / funcție / metodă de calcul folosită

✓ Produs software folosit: Anaconda Navigator, JupyterLab, MS Excel

✓ Limbaj: Python

✓ Funcția: drop(), isnull(), fillna()

✓ Metoda de calcul folosită:

Pentru a șterge coloana "Nr. filme" am folosit funcția drop() și am specificat numele coloane și axis=1 ce reprezintă coloana. Cu ajutorul funcției isnull() am verificat dacă există vreo valoare lipsă în coloană "Venit", iar cu funcția fillna() am înlocuit valoarea lipsă cu textul "Declarație nerealizată".

La final, am șters, cu ajutorul funcției drop(), actorul care nu și-a declarat venitul.

### • Codul utilizat

#4. Ștergerea coloanei Nr filme

```
df = df.drop("Nr filme", axis=1)
print(df.head())
```

#5. Înlocuirea valorii lipsă a venitului cu mesajul Declarație nerealizată

```
df=pd.read_csv('Actori.csv', sep=',', header=0, usecols=['Venit'])
```

# verifică dacă există valori lipsă

```
print(df['Venit'])
```

```
print(df.loc[df['Venit'].isnull()])
```

```
print('-'*40)
```

```
print(df['Venit'].fillna('Declarație nerealizată'))
```

#6. Ștergerea actorului din Lista a carui declarație despre venit lipsește - rand 3

#Axis 0 reprezintă randurile, iar axis 1 reprezintă coloanele

```
df = df.drop([3], axis=0)
```

```
print(df.head(10))
```

## • Rezultatele obținute

```
In [26]: #4. Ștergere coloanei Nr filme
df = df.drop("Nr filme", axis=1)
print(df.head())
```

	Top	Actori	Venit	Varsta
0	1	Yoo Seung Ho	191500.0	26
1	2	Park Min Young	92000.0	35
2	3	Park Seo Joon	241200.0	31
3	4	Lee Jun Ki	NaN	37
4	5	Park Jung Min	28000.0	32

```
In [31]: #5. Înlocuirea valorii lipsă a venitului cu mesajul Declarație nerealizată
df=pd.read_csv('Actori.csv', sep=',',header=0, usecols=['Venit'])

# verifică dacă există valori lipsă
print(df['Venit'])
print(df.loc[df['Venit'].isnull()])
print('-'*40)
print(df['Venit'].fillna('Declarație nerealizată'))
```

	Venit
0	191500.0
1	92000.0
2	241200.0
3	NaN
4	28000.0

Name: Venit, dtype: float64

```
Venit
3    NaN
-----
```

	Venit
0	191500
1	92000
2	241200
3	Declarație nerealizată
4	28000

Name: Venit, dtype: object

```
In [28]: #6. Ștergerea actorului din lista a carui declaratie despre venit lipseste - rand 3
#Axis 0 reprezinta randurile, iar axis 1 reprezinta coloanele
df = df.drop([3], axis=0)
print(df.head(10))
```

	Top	Actori	Venit	Varsta
0	1	Yoo Seung Ho	191500.0	26
1	2	Park Min Young	92000.0	35
2	3	Park Seo Joon	241200.0	31
4	5	Park Jung Min	28000.0	32

## • Interpretare:

*Se poate observa că actorul care nu și-a declarat venitul a fost șters acesta aflându-se pe poziția 4 în top deci, fiind actorul Lee Jun Ki.*



## 9. PRELUCRAREA SETURILOR DE DATE CU MERGE

### • Descrierea problemei

Managerul televiziunii ArirangTV are două liste (DataFrame-uri): una cu numele filmelor și premiile la care au fost nominalizate și a doua cu numele filmelor și dacă acestea au câștigat acel premiu sau doar au fost nominalizate. Managerul dorește să aibă într-o singură listă toate informațiile acestea.

### • Informații necesare pentru rezolvare

Pentru realizarea acestor operații avem nevoie de următoarele informații :

- ✓ Filmele nominalizate la premii.
- ✓ Premiile la care au fost nominalizate filmele..
- ✓ Nominalizarea sau castigarea premiului.

### • Produs software / funcție / metodă de calcul folosită

- ✓ Produs software folosit: Anaconda Navigator, JupyterLab
- ✓ Limbaj: Python
- ✓ Funcția: merge()
- ✓ Metoda de calcul folosită:

Am creat două DataFrame-uri unul legat de filme și premiu la care au fost nominalizate și al doilea despre filme și câștigarea acelui premiu. Mai apoi, cu ajutorul funcției merge(), am combinat cele două DataFrame-uri într-unul singur, având acces la informații mult mai ușor.

### • Codul utilizat

```
#LUCRUL CU INNER MERGE
```

```
import pandas as pd
```

```
#1. Prima lista cu date contine numele filmelor nominalizate la un premiu si premiul la care sunt nominalizate.
```

```
df1=pd.DataFrame({
    "Film":["Parasite","Hwarang","Remember:The War of the Son","Watcher",
           "Choco Bank","Fashion King","Defendant","Arthdal Chronicles"],
    "Premiu":["Globul de aur","Premiul Award","Globul de aur","Premiul Leopardul de aur",
             "Hong Kong Film Award","Premiul BAFTA","Premiul Oscar","Korean Film Award"],
})
df1
```


```
#2. A doua lista de date contine din nou numele filmelor si daca acestea au castigat premiul sau doar au fost nominalizate.
```

```
df2=pd.DataFrame({
    "Film":["Parasite","Hwarang","Remember:The War of the Son","Watcher",
           "Choco Bank","Fashion King","Defendant","Arthdal Chronicles"],
    "Nominalizare":["Castigat","Nominalizat","Castigat","Castigat","Castigat",
                   "Nominalizat","Nominalizat","Castigat"],
})
df2
```

```
#3. Combinatia dintre cele doua liste de date.
```

```
rezultat=pd.merge(df1,df2,on="Film")
rezultat
```

## • Rezultatele obținute

jupyter 9. Inner Merge Last Checkpoint: 18 hours ago (autosaved)  Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

Run Code

```
In [4]: #LUCRUL CU INNER MERGE
import pandas as pd

#1. Prima lista cu date contine numele filmelor nominalizate la un premiu si premiul la care sunt nominalizate.
df1=pd.DataFrame({
    "Film":["Parasite","Hwarang","Remember:The War of the Son","Watcher","Choco Bank","Fashion King","Defendant",
           "Arthdal Chronicles"],
    "Premiu":["Globul de aur","Premiul Award","Globul de aur","Premiul Leopardul de aur","Hong Kong Film Award",
            "Premiul BAFTA","Premiul Oscar","Korean Film Award"],
})
df1
```

Out[4]:

	Film	Premiu
0	Parasite	Globul de aur
1	Hwarang	Premiul Award
2	Remember:The War of the Son	Globul de aur
3	Watcher	Premiul Leopardul de aur
4	Choco Bank	Hong Kong Film Award
5	Fashion King	Premiul BAFTA
6	Defendant	Premiul Oscar
7	Arthdal Chronicles	Korean Film Award

```
In [5]: #2. A doua lista de date contine din nou numele filmelor si daca acestea au castigat premiul sau doar au fost nominalizate.
df2=pd.DataFrame({
    "Film":["Parasite","Hwarang","Remember:The War of the Son","Watcher","Choco Bank","Fashion King","Defendant",
           "Arthdal Chronicles"],
    "Nominalizare":["Catigat","Nominalizat","Castigat","Castigat","Castigat",
                  "Nominalizat","Nominalizat","Castigat"],
})
df2
```

Out[5]:

	Film	Nominalizare
0	Parasite	Catigat
1	Hwarang	Nominalizat
2	Remember:The War of the Son	Castigat
3	Watcher	Castigat
4	Choco Bank	Castigat
5	Fashion King	Nominalizat
6	Defendant	Nominalizat
7	Arthdal Chronicles	Castigat

```
In [6]: #Combinatia dintre cele doua lista de date.
rezultat=pd.merge(df1,df2,on="Film")
rezultat
```

Out[6]:

	Film	Premiu	Nominalizare
0	Parasite	Globul de aur	Catigat
1	Hwarang	Premiul Award	Nominalizat
2	Remember:The War of the Son	Globul de aur	Castigat
3	Watcher	Premiul Leopardul de aur	Castigat
4	Choco Bank	Hong Kong Film Award	Castigat
5	Fashion King	Premiul BAFTA	Nominalizat
6	Defendant	Premiul Oscar	Nominalizat
7	Arthdal Chronicles	Korean Film Award	Castigat

## • Interpretare:

Se poate observa că majoritatea filmelor nominalizate la premii au și câștigat acel premiu.

## 10. GRUPAREA DATELOR ȘI PRELUCRAREA STATISTICĂ

### • Descrierea problemei

Managerul televiziunii ArirangTV dorește să știe ce distribuție a avut mai mult succes la Oscar pentru a prelungi contractele cu acei actori. Astfel, la premiul Oscar au fost câștigatoare 2 filme. Managerul dorește să știe ce film a avut nota din partea juriului mai mare.

### • Informații necesare pentru rezolvare

Pentru realizarea acestor operații avem nevoie de următoarele informații :

- ✓ Numele celor două filme nominalizate la Oscar.
- ✓ Notele obținute de aceste filme din partea juriului.

### • Produs software / funcție / metodă de calcul folosită

- ✓ Produs software folosit: Anaconda Navigator, JupyterLab
- ✓ Limbaj: Python
- ✓ Funcția: `groupby()`, `mean()`
- ✓ Metoda de calcul folosită:

Am creat un `DataFrame` ce conține numele celor două filme și notele obținute de acestea la Oscar. Cu ajutorul funcției `groupby()` am grupat notele filmelor în funcție de numele filmului iar, mai apoi, am făcut o medie a notelor obținute de aceste filme cu ajutorul funcției statistice `mean()`.

### • Codul utilizat

```
#GRUPAREA DATELOR ȘI PRELUCRAREA STATISTICA
# 1. Notele acordate de juriul de la Oscar cele doua filme nominalizate la "cea mai buna distributie"
import pandas as pd

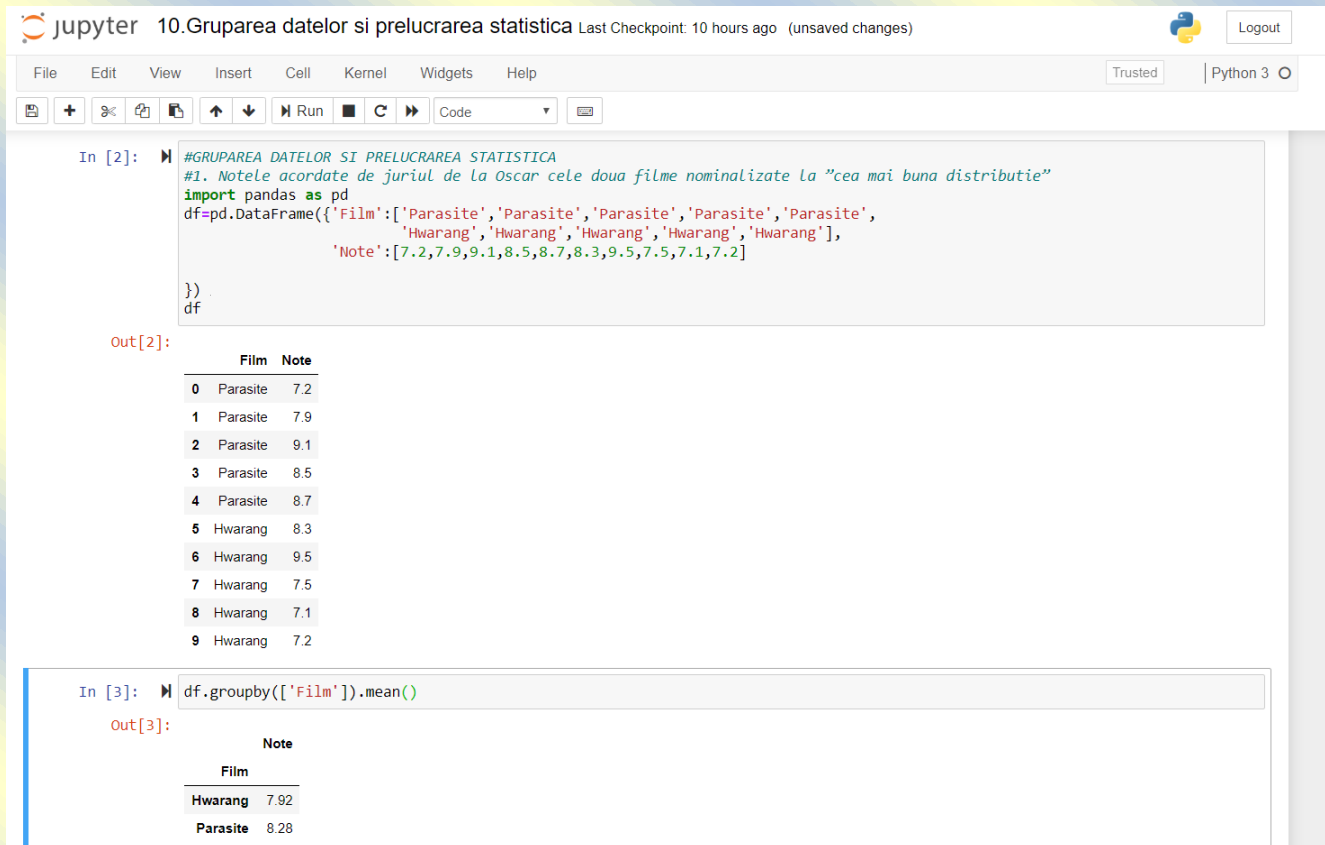
df = pd.DataFrame({'Film': ['Parasite', 'Parasite', 'Parasite', 'Parasite', 'Parasite',
                           'Hwarang', 'Hwarang', 'Hwarang', 'Hwarang', 'Hwarang'],
                  'Note': [7.2, 7.9, 9.1, 8.5, 8.7, 8.3, 9.5, 7.5, 7.1, 7.2]

                  })

df.groupby(['Film']).mean()
```



- Rezultatele obținute**



The screenshot shows a Jupyter Notebook titled "10.Gruparea datelor si prelucrarea statistica". The code in the first cell creates a DataFrame with two columns: 'Film' and 'Note'. The 'Film' column contains the names 'Parasite' and 'Hwarang', and the 'Note' column contains their respective scores. The second cell displays the DataFrame as a table. The third cell calculates the mean score for each film using the 'groupby' method.

```
In [2]: #GRUPAREA DATELOR SI PRELUCRAREA STATISTICA
#1. Notele acordate de juriul de la Oscar cele doua filme nominalizate la "cea mai buna distributie"
import pandas as pd
df=pd.DataFrame({'Film':['Parasite','Parasite','Parasite','Parasite','Parasite',
                        'Hwarang','Hwarang','Hwarang','Hwarang','Hwarang'],
                 'Note':[7.2,7.9,9.1,8.5,8.7,8.3,9.5,7.5,7.1,7.2]
                })
df
```

Out[2]:

	Film	Note
0	Parasite	7.2
1	Parasite	7.9
2	Parasite	9.1
3	Parasite	8.5
4	Parasite	8.7
5	Hwarang	8.3
6	Hwarang	9.5
7	Hwarang	7.5
8	Hwarang	7.1
9	Hwarang	7.2

```
In [3]: df.groupby(['Film']).mean()
```

Out[3]:

	Note
Film	
Hwarang	7.92
Parasite	8.28

- Interpretare:**

*Se poate observa că filmul "Parasite" a avut scorul mai mare dat de jurații de la Oscar.*

# 11. CONVERSIA UNUI TIP DE DATĂ

## • Descrierea problemei

Managerul televiziunii ArirangTV știe că doi dintre actorii din producția sa au luat premiul cel mare la Korean Drama Award. Cunoaște numele și poziția pe care s-au aflat inițial actorii când au fost nominalizați, dar poziția este sub formă de șir de caractere. El nu poate căuta actorii în baza lui de date decât după poziția inițială. Deoarece căutare trebuie să fie doar după date de tip integer se dorește conversia tipului de dată a pozițiilor celor doi actori.

## • Informații necesare pentru rezolvare

Pentru realizarea acestor operații avem nevoie de următoarele informații :

- ✓ Numele actorilor.
- ✓ Poziția inițială.

## • Produs software / funcție / metodă de calcul folosită

- ✓ Produs software folosit: Anaconda Navigator, JupyterLab
- ✓ Limbaj: Python
- ✓ Funcția: `astype(int)`
- ✓ Metoda de calcul folosită:

Am creat inițial un `DataFrame` care să conțină numele actorilor și poziția acestora. Mai apoi, cu ajutorul funcției `astype(int)`, am convertit pozițiile din `string` în `integer`.

## • Codul utilizat

```
#CONVERSIA UNUI STRING IN INTEGER
```

```
import pandas as pd
```

```
Data = {'Actor': ['Park Min Young', 'Lee Jun Ki'],
        'Pozitie': ['21', '25']}
```

```
#Convesia din string in int
```

```
df = pd.DataFrame(Data)
```

```
df['Pozitie'] = df['Pozitie'].astype(int)
```

```
print (df)
```

## • Rezultatele obținute

The screenshot shows a JupyterLab window titled "11. Conversia datelor". The code cell contains the same Python code as shown in the previous block. Below the code, the output of the `print(df)` statement is displayed as a DataFrame:

	Actor	Pozitie
0	Park Min Young	21
1	Lee Jun Ki	25

## • Interpretare:

Se poate observa că actrița Park Min Young se afla inițial pe poziția 21, iar actorul Lee Jun Ki pe poziția 25.

## 12. REGRESIA LINIARĂ MULTIPLĂ (pachet statsmodels)

### • Descrierea problemei

*Pentru a determina în ce măsură variabilele independente contribuie la modificarea variabilei dependente managerul televiziunii ArirangTV dorește un model de regresie liniară multifactorială. Pentru a determina dacă acesta poate fi considerat valid, adică dacă există, sau nu, o legătură liniară între rating-ul înregistrat de primele 5 seriale aflate în curs difuzare și sumele aferente publicității realizată pentru aceste seriale. Publicitatea s-a realizat prin mijloace media și anume prin spot-uri publicitare la televizor, prin reclame pe Internet și amplasarea de panouri publicitare prin orașul Seul.*

### • Informații necesare pentru rezolvare

*Pentru realizarea acestor cerințe avem nevoie de următoarele informații :*

- ✓ *Sumele aferente realizării de publicitate prin mijloacele menționate mai sus.*
- ✓ *Rating-ul înregistrat de cele 5 seriale, exprimat în procente.*

### • Produs software / funcție / metodă de calcul folosită

- ✓ *Produs software folosit: Anaconda Navigator, JupyterLab*
- ✓ *Limbaj: Python*
- ✓ *Funcția: ols.fit(), round()*
- ✓ *Metoda de calcul folosită:*

*Am stabilit că variabilele independente sunt coloanele reprezentate de cele 3 mijloace de publicitate: prin TV, Internet și Panou, iar variabila dependentă este reprezentată de coloana Rating. Prin funcțiile ols() și fit() ale pachetului statsmodels se potrivește un model OSL cu termenul liber pe cele 3 coloane: TV + Internet + Panou.*

*La urmă se afișează parametrii rezultați și se rotunjește rezultatele predicției celor 5 câmpuri, cu ajutorul funcției round().*

### • Codul utilizat

```
#REGRESIE LINIARA MULTIPLA
import pandas as pd
import statsmodels.formula.api as smf

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

X = pd.DataFrame(df, columns=['TV', 'Internet', 'Panou'])
y = df['Rating']
results = smf.ols('y ~ TV + Internet + Panou', data=df).fit()
print("In urma regresiei liniare multiple avem urmatorii parametri:\n")
print(results.params)
print(round(results.predict(df[:5])))
```



## • Rezultatele obținute

Jupyter 12. Regresie Liniara Multipla Last Checkpoint: Last Thursday at 4:24 PM (unsaved changes)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

```
In [5]: #REGRESIE LINIARA MULTIPLA
import pandas as pd
import statsmodels.formula.api as smf

df = pd.read_csv('Publicitate.csv')
df
```

Out[5]:

	Nr_Crt	TV	Internet	Panou	Raiting
0	1	230.1	37.8	69.2	22.1
1	2	44.5	39.3	45.1	10.4
2	3	17.2	45.9	69.3	9.3
3	4	151.5	41.3	58.5	18.5
4	5	180.8	10.8	58.4	12.9

```
In [9]: X = pd.DataFrame(df, columns=['TV', 'Internet', 'Panou'])
y = df['Raiting']
results = smf.ols('y ~ TV + Internet + Panou', data=df).fit()
print("In urma regresiei liniare multiple avem urmatorii parametrii:\n")
print(results.params)
print(round(results.predict(df[:5])))
```

In urma regresiei liniare multiple avem urmatorii parametrii:

```
Intercept      0.658642
TV[T.17.2]     -12.022040
TV[T.180.8]    -0.823430
TV[T.230.1]     2.060803
TV[T.44.5]     -5.176824
Internet        0.155970
Panou           0.194868
dtype: float64
0      22.0
1      10.0
2       9.0
3      18.0
4      13.0
dtype: float64
```

## • Interpretare:

Valoarea parametrului de interceptare arată dacă cele trei variabile explicative, TV, Internet și Panou ar avea valoarea 0, atunci rating-ul va avea valoarea 0.65%. Altfel spus, dacă nu s-ar realiza nici o formă de publicitate pentru seriale atunci rating-ul s-ar poziționa sub 1%.

Se observă de asemenea, că cea mai ridicată valoare a rating-ului este de 22.1%. Prin urmare, se poate afirma că realizarea celor 3 forme de publicitate pentru serialele în curs de difuzare influențează rating-ul serialului.

## 13. REGRESIA LOGISTICĂ (pachet scikit-learn)

### • Specificații

În general, o regresie logistică binară descrie relația dintre variabila binară dependentă și una sau mai multe variabile independente.

Variabila dependentă binară are două rezultate posibile:

- „1” pentru adevărat / succes;
- '0' pentru fals / eșec

### • Descrierea problemei

Managerul televiziunii ArirangTV dorește să construiască un model de regresie logistică pentru a determina dacă actorii-candidații la casting vor putea juca în noul serial.

Exist două rezultate posibile pentru candidați: admis (reprezentat de valoarea „1”) și respins (reprezentat de valoarea „0”).

### • Informații necesare pentru rezolvare

Pentru realizarea acestor operații avem nevoie de următoarele informații :

- ✓ Variabila dependentă reprezintă dacă o persoană este internată;
- ✓ Cele 3 variabile independente și anume: studii în străinătate, poziția în Top 50 Korea, anii de experiență și numărul de filme în care au jucat.

### • Produs software / funcție / metodă de calcul folosită

- ✓ Produs software folosit: Anaconda Navigator, JupyterLab
- ✓ Limbaj: Python
- ✓ Funcția: `train_test_split`
- ✓ Metoda de calcul folosită:

Am importat pachetele necesare regresiei logistice, precum și fișierul CSV ce conține datele despre concurenți.

Am setat variabilele independente (reprezentate de X) și variabila dependentă (reprezentată de y).

Am aplicat `train_test_split` . Am setat dimensiunea testului la 0,25 și, prin urmare, testarea modelului se va baza pe 25% din setul de date, în timp ce formarea modelului se va baza pe 75% din setul de date. Am aplicat regresia logistică și apoi am creat matricea de confuzie, iar la final am afișat predicția și graficul rezultat din matricea de confuzie.

### • Codul utilizat

```
#REGRESIA LOGISTICA
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import seaborn as sn
import matplotlib.pyplot as plt

#Se organizeaza un concurs de casting pentru actorii ce vor juca in urmatorul serial.
df=pd.read_csv('Candidati.csv')
df
```

```
#Starea variabilei independente (reprezentate de X) și variabilei dependente (reprezentată de y)
X = df[['Studii Str', 'Top 50 Korea','Ani experienta']]
y = df['Admis']

#Am aplicat train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.25,random_state=0)

#Am aplicat regresia logistica
logistic_regression= LogisticRegression()
logistic_regression.fit(X_train,y_train)
y_pred=logistic_regression.predict(X_test)

#Matricea de confuzie pentru evaluarea performantei
confusion_matrix = pd.crosstab(y_test, y_pred, rownames=['Actual'], colnames=['Predictie'])
sn.heatmap(confusion_matrix, annot=True)

print('Precizie: ',metrics.accuracy_score(y_test, y_pred))
plt.show()
```

## • Rezultatele obținute

jupyter 13. Regresia Logistica Last Checkpoint: a day ago (autosaved) Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [31]: `#REGRESIA LOGISTICA`

```
import pandas as pd
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn import metrics
import seaborn as sn
import matplotlib.pyplot as plt
```

In [25]: `#Se organizeaza un concurs de casting pentru actorii ce vor juca in urmatorul serial.`

```
df=pd.read_csv('Candidati.csv')
df
```

Out[25]:

	Studii Str	Top 50 Korea	Ani experienta	Nr filme	Admis
0	1	1.7	5	12	1
1	0	2.3	3	13	1
2	0	2.7	2	9	1
3	0	3.7	5	8	0
4	1	3.8	6	11	0
5	1	3.6	7	5	0
6	1	3.9	8	7	0
7	0	3.5	5	7	0
8	0	2.1	7	19	1
9	0	4.1	3	6	0
10	0	5.0	1	3	0
11	0	4.9	1	5	0
12	0	4.7	6	11	0
13	1	1.5	3	12	1
14	1	1.2	4	10	1
15	1	3.3	5	9	0
16	1	3.1	4	9	0
17	0	4.2	4	7	0
18	0	4.5	5	5	0
19	0	4.8	2	5	0
20	1	1.3	4	15	1

In [26]: `#Starea variabilei independente (reprezentate de X) și variabilei dependente (reprezentată de y)`

```
X = df[['Studii Str', 'Top 50 Korea','Ani experienta']]
y = df['Admis']
```

In [27]: `#Am aplicat train_test_split`

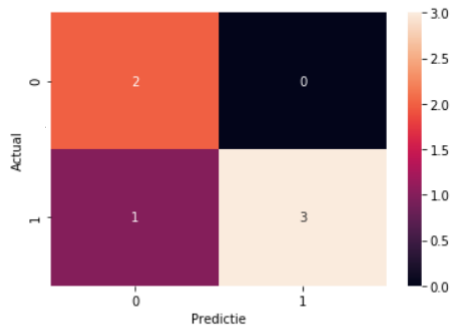
```
X_train,X_test,y_train,y_test = train_test_split(X,y,test_size=0.25,random_state=0)
```



```
In [28]: ► #Am aplicat regresia logistica
logistic_regression = LogisticRegression()
logistic_regression.fit(X_train,y_train)
y_pred=logistic_regression.predict(X_test)
```

```
In [29]: ► #Matricea de confuzie pentru evaluarea performantei
confusion_matrix = pd.crosstab(y_test, y_pred, rownames=['Actual'], colnames=['Predictie'])
sn.heatmap(confusion_matrix, annot=True)
```

Out[29]: <matplotlib.axes.\_subplots.AxesSubplot at 0x21f36d65fc8>



```
In [30]: ► print('Precizie: ',metrics.accuracy_score(y_test, y_pred))
plt.show()
```

Precizie: 0.8333333333333334

- **Interpretare:**

*Se poate observa că precizia este de 0.833, adică 83% pentru testul stabilit, deci variabilele independente influențează variabila dependentă.*

## 14. CLUSTERIZARE (pachet scikit-learn)

### • Descrierea problemei

Managerul televiziunii ArirangTV dorește să facă predicții referitoare la următorii câștigători de la Korean Drama Award și pentru acest lucru vrea să se ia în considerare venitul actorilor acumulat până în prezent din actorie, vârsta și numărul de filme realizate. După decernarea premiilor, managerul dorește să știe în ce procent au avut predicțiile sale dreptate.

### • Informații necesare pentru rezolvare

Pentru realizarea acestor operații avem nevoie de următoarele informații :

- ✓ Venitul actorului.
- ✓ Vârsta actorului.
- ✓ Numărul de filme realizate.

### • Produs software / funcție / metodă de calcul folosită

- ✓ Produs software folosit: Anaconda Navigator, JupyterLab, MS Excel
- ✓ Limbaj: Python
- ✓ Funcția: KMeans(), isna(), sum(), fillna()
- ✓ Metoda de calcul folosită:

În fișierul CSV importat, cu ajutorul funcției isna(), am identificat valorile lipsă, cu funcția sum() le-am numărat, iar funcția fillna() a înlocuit valorile lipsă cu media coloanei respective.

Am evaluat apoi câștigătorii premiilor Korean Drama Award în funcție de 3 criterii: vârstă, venit și numărul filmelor.

Cu funcția drop() am eliminat coloanele pe care le-am considerat irelevante pentru influența lor asupra câștigătorilor și anume coloanele: actori, top și naționalitate.

Apoi, am setat variabilele X și Y ca vectori de tip array din pachetul numpy.

Am apelat metoda Kmeans și am setat parametrul n\_clusters cu 2 adică, rezultatele se pot încadra în 2 categorii: câștigători și necâștigători.

În final, am evaluat rezultatele conform algoritmului de clusterizare.

### • Codul utilizat

```
#CLUSTERIZARE
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler

#Citirea din fisierul csv
df=pd.read_csv('Actori2.csv')
df.head()

#Identificarea valorilor lipsa ~ Acolo unde e True valoarea lipseste.
print(df.columns.values)
print(df.isna())

#Facem suma valorilor lipsa
print(df.isna().sum())
```

```
# Înlocuirea valorilor lipsă cu media coloanei, utilizând fillna()
df.fillna(df.mean(), inplace=True)
print(df.isna().sum())

# Evaluarea castigatorilor premiilor Korean Drama Award în funcție de Venit, Varsta si Nr filme
print("*****Venit - Castigator*****")
print(df[['Venit', 'Castigator']].groupby(['Venit'],
as_index=False).mean().sort_values(by='Castigator', ascending=False))
print('\n')
print("*****Varsta - Castigator*****")
print(df[['Varsta', 'Castigator']].groupby(['Varsta'],
as_index=False).mean().sort_values(by='Castigator', ascending=False))
print('\n')
print("*****Nr filme - Castigator*****")
print(df[["Nr filme", "Castigator"].groupby(['Nr filme'],
as_index=False).mean().sort_values(by='Castigator', ascending=False))

#Eliminam coloanele non-numerice care nu influențează castigarea premiului, dar si coloana Top
df = df.drop("Actori", axis=1)
df = df.drop("Nationalitate", axis=1)
df=df.drop("Top",axis=1)
df.head()


# Variabila X este un vector (array din pachetul numpy) identic cu setul df,
# din care a fost ștearsă coloana Castigator, iar y este un vector format din coloana Castigator
X = np.array(df.drop(['Castigator'], 1).astype(float))
Y = np.array(df['Castigator'])

#Aplearea metodei Kmeans și setarea parametrului n_clusters = 2
# (castigatori/necastigatori)
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)

# Evaluarea rezultatelor:
correct = 0
for i in range(len(X)):
    predict_me = np.array(X[i].astype(float))
    predict_me = predict_me.reshape(-1, len(predict_me))
    prediction = kmeans.predict(predict_me)
    print("Estimam ca actorul va fi", prediction[0], "Actorul este", Y[i])
    if prediction[0] == Y[i]:
        correct += 1

print('Am estimat corect, in medie:')
print(correct / len(X))
```

## • Rezultatele obținute

jupyter 14. Clusterizare Last Checkpoint: an hour ago (unsaved changes)  Logout

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3

In [78]: `#CLUSTERIZARE`

```
import pandas as pd
import numpy as np
from sklearn.cluster import KMeans
from sklearn.preprocessing import LabelEncoder
from sklearn.preprocessing import MinMaxScaler
```

In [79]: `#Citirea din fisierul csv`

```
df=pd.read_csv('Actori2.csv')
df.head()
```

Out[79]:

	Top	Actori	Venit	Varsta	Nr filme	Castigator	Nationalitate
0	1	Yoo Seung Ho	191500.0	26.0	52.0	1	Coreeana
1	2	Park Min Young	92000.0	32.0	50.0	0	Coreeana
2	3	Park Seo Joon	241200.0	28.0	35.0	1	Coreeana
3	4	Lee Jun Ki	35000.0	35.0	40.0	1	Coreeana
4	5	Park Jung Min	NaN	32.0	20.0	1	Coreeana

In [80]: `#Identificarea valorilor lipsa ~ Acolo unde e True valoarea lipseste.`

```
print(df.columns.values)
print(df.isna())

#Facem suma valorilor lipsa
print(df.isna().sum())
```

```
['Top' 'Actori' 'Venit' 'Varsta' 'Nr filme' 'Castigator' 'Nationalitate']
Top      Actori  Venit  Varsta  Nr filme  Castigator  Nationalitate
0  False  False  False  False  False  False  False
1  False  False  False  False  False  False  False
2  False  False  False  False  False  False  False
3  False  False  False  False  False  False  False
4  False  False  True  False  False  False  False
5  False  False  False  False  False  False  False
6  False  False  False  False  False  False  False
7  False  False  False  False  False  False  False
8  False  False  False  False  False  False  False
9  False  False  False  False  False  False  False
10 False  False  False  True  False  False  False
11 False  False  True  False  False  False  False
12 False  False  False  False  True  False  False
13 False  False  False  False  False  False  False
14 False  False  False  False  False  False  False
Top      0
Actori    0
Venit     2
Varsta    1
Nr filme  1
Castigator 0
Nationalitate 0
dtype: int64
```

In [82]: `Evaluarea castigatorilor premiilor Korean Drama Award în funcție de Venit, Varsta si Nr filme`

```
int("*****Venit - Castigator*****")
int(df[['Venit', 'Castigator']].groupby(['Venit'], as_index=False).mean().sort_values(by='Castigator', ascending=False))
int('\n')
int("*****Varsta - Castigator*****")
int(df[['Varsta', 'Castigator']].groupby(['Varsta'], as_index=False).mean().sort_values(by='Castigator', ascending=False))
int('\n')
int("*****Nr filme - Castigator*****")
int(df[['Nr filme', 'Castigator']].groupby(['Nr filme'], as_index=False).mean().sort_values(by='Castigator', ascending=False))
```

```
*****Venit - Castigator*****
Venit  Castigator
1  25000.000000    1
2  35000.000000    1
3  41376.000000    1
8  79248.000000    1
9  81933.769231    1
12 191500.000000    1
13 241200.000000    1
0  18000.000000    0
4  45612.000000    0
5  65370.000000    0
6  65464.000000    0
7  76535.000000    0
10 88834.000000    0
11 92000.000000    0

*****Varsta - Castigator*****
Varsta  Castigator
0  15.0    1.0
2  27.0    1.0
3  28.0    1.0
5  31.0    1.0
9  35.0    1.0
12 59.0    1.0
1  26.0    0.5
6  32.0    0.5
4  29.0    0.0
7  34.0    0.0
8  34.5    0.0
10 53.0    0.0
11 56.0    0.0

*****Nr filme - Castigator*****
Nr filme  Castigator
0  12.000000    1.0
2  20.000000    1.0
4  27.000000    1.0
5  35.000000    1.0
7  40.000000    1.0
8  44.000000    1.0
10 52.000000    1.0
9  50.000000    0.5
1  19.000000    0.0
3  21.000000    0.0
6  38.642857    0.0
11 55.000000    0.0
12 58.000000    0.0
```



```
In [83]: #Eliminam coloanele non-numerice care nu influențează castigarea premiului, dar si coloana Top
df = df.drop("Actori", axis=1)
df = df.drop("Nationalitate", axis=1)
df=df.drop("Top",axis=1)
df.head()
```

Out[83]:

	Venit	Varsta	Nr filme	Castigator
0	191500.000000	26.0	52.0	1
1	92000.000000	32.0	50.0	0
2	241200.000000	28.0	35.0	1
3	35000.000000	35.0	40.0	1
4	81933.769231	32.0	20.0	1

```
In [84]: # Variabila X este un vector (array din pachetul numpy) identic cu setul df,
# din care a fost ștersă coloana Castigator, iar y este un vector format din coloana Castigator
X = np.array(df.drop(['Castigator'], 1).astype(float))
Y = np.array(df['Castigator'])
```

```
In [85]: #Aplearea metodei Kmeans și setarea parametrului n_clusters = 2
# (castigatori/necastigatori)
kmeans = KMeans(n_clusters=2)
kmeans.fit(X)
```

```
Out[85]: KMeans(algorithm='auto', copy_X=True, init='k-means++', max_iter=300,
n_clusters=2, n_init=10, n_jobs=None, precompute_distances='auto',
random_state=None, tol=0.0001, verbose=0)
```

```
In [86]: #Evaluarea rezultatelor:
correct = 0
for i in range(len(X)):
    predict_me = np.array(X[i].astype(float))
    predict_me = predict_me.reshape(-1, len(predict_me))
    prediction = kmeans.predict(predict_me)
    print("Estimam ca actorul va fi", prediction[0], "Actorul este", Y[i])
    if prediction[0] == Y[i]:
        correct += 1

print('Am estimat corect, in medie:')
print (correct/len(X))
```

```
Estimam ca actorul va fi 1 Actorul este 1
Estimam ca actorul va fi 0 Actorul este 0
Estimam ca actorul va fi 1 Actorul este 1
Estimam ca actorul va fi 0 Actorul este 1
Estimam ca actorul va fi 0 Actorul este 1
Estimam ca actorul va fi 0 Actorul este 1
Estimam ca actorul va fi 0 Actorul este 0
Estimam ca actorul va fi 0 Actorul este 1
Estimam ca actorul va fi 0 Actorul este 0
Estimam ca actorul va fi 0 Actorul este 0
Estimam ca actorul va fi 0 Actorul este 0
Estimam ca actorul va fi 0 Actorul este 1
Estimam ca actorul va fi 0 Actorul este 0
Estimam ca actorul va fi 0 Actorul este 1
Estimam ca actorul va fi 0 Actorul este 0
Am estimat corect, in medie:
0.6
```

## • Interpretare

*În urma aplicării algoritmului de clusterizare și a predicțiilor făcute asupra setului de date se poate afirma că algoritmul a estimat corect 60% din rezultatele obținute de actori.*

***Sfârșitul celei de-a doua părți !***

## **BIBLIOGRAFIE**

### **1. Liste, Dicționare, Seturi, Tupluri, Funcții și Grafic:**

- Seminar 1 Python

### **2. Import CSV, Accesarea datelor cu loc și iloc, Tratarea valorilor lipsă, ștergerea coloanelor și a înregistrărilor:**

- Seminar 2 Python

### **3. Prelucrarea seturilor de date cu Merge:**

- Seminar 3 Python:
  - <https://hub.gke.mybinder.org/user/iintorsureanu-psw-sem-python-3-y7qn9kl1/lab>
- <https://www.youtube.com/watch?v=h4hOPGo4UVU>

### **4. Gruparea datelor și prelucrarea statistică:**

- Seminar 2 Python
- Seminar 3 Python:
  - <https://hub.gke.mybinder.org/user/iintorsureanu-psw-sem-python-3-y7qn9kl1/lab>
- <https://pandas.pydata.org/pandas-docs/stable/reference/api/pandas.DataFrame.groupby.html>

### **5. Conversia unui tip de dată:**

- Seminar 3 Python:
  - <https://hub.gke.mybinder.org/user/iintorsureanu-psw-sem-python-3-y7qn9kl1/lab>
- <https://datatofish.com/string-to-integer-dataframe/>

### **6. Regresia liniară multiplă (pachet statsmodels):**

- Seminar 4 Python:
  - <https://hub.gke.mybinder.org/user/iintorsureanu-psw-sem-python-4-glorhn8n/lab>

### **7. Regresia logistică (pachet scikit-learn):**

- Seminar 4 Python:
  - <https://hub.gke.mybinder.org/user/iintorsureanu-psw-sem-python-4-glorhn8n/lab>
- <https://datatofish.com/logistic-regression-python/>
- [https://www.youtube.com/watch?time\\_continue=2312&v=WSNihAEAtXo&feature=emb\\_logo](https://www.youtube.com/watch?time_continue=2312&v=WSNihAEAtXo&feature=emb_logo)

### **• Alte link-uri utile:**

