Big Data

Concepte Python

Python

- Unul dintre cele mai utilizate limbaje din sfera ML.

Tipuri de date

- Numerice - **Int (Integer)** sau **Float (Floating Point)**
- **Bool (Boolean)** - corespunde unei valori binare `{True, False}`
- **Str (String)** - pentru șiruri de caractere.

Observație: pentru a determina clasa (tipul) unui obiect, utilizăm funcția type(nume_obiect)

- **Exemplu:** `type(1.2)` returnează ‘float’

Operatorii aritmetici

Operatorii aritmetici din Python sunt:

1) +
2) -
3) *
4) / (diviziune reală)
5) ** (ridicare la putere)
6) % (modulo)
7) // (diviziune întreagă)

Operatorii de comparare

Compară valori de același tip și returnează un rezultat de tip boolean.

În Python, acești operatori sunt:
`{<, <=, >, >=, !=, ==}`

Exemple :

- `a = (12 == 13)` `a` primește valoarea False
- `b = ("hello"== "hello")` `b` primește valoarea True

Alte operații

Cod	Rezultat	Numele operației
`"hello"+"world"`	`"helloworld"`	Concatenare
`"hello"[0]`	`"h"`	Accesarea elementului corespunzător unui indice
`"hello"*3`	`"hellohellohello"`	Repetare (concatenare)
`"hello"[-1]`	`"o"`	Accesarea ultimului element
`"hello"[1:4]`	`"ell"`	Substring
`A=1`	`1`	Atribuire
`B=1.2`	`1.2`	Atribuire

Instrucțiuni condiționale

Instrucțiune

```
if condition:
        bloc
else:
        bloc
```

Exemplu

```
nota = 82
if nota >= 50:
    print("Succes")
else:
    print("Esec")
```

Instrucțiunea For

Instrucțiunea **for** se aplică pe o colecție de valori. De exemplu tupluri, liste etc..

Putem genera o colecție de valori succesive pentru indici cu funcția `range(start, stop, step)`

Exemple :

- `range(4)` → `0 1 2 3`
- `range(1,4)` → `1 2 3`
- `range(0,5,2)` → `0 2 4`

Instrucțiunea For

Instrucțiune

```
for indice in secventa:
    bloc de instructiuni
```

Exemplu

```
for i in range(10):
    if(i%2 == 0):
        print('Par')
    else:
        print('Impar')
```

Observații:

- Atenție la indentare!
- Putem întrerupe ciclarea cu ajutorul instrucțiunii `break`
- Putem itera direct prin colecțiile de valori

Instrucțiunea While

Instrucțiune

```
while conditie:
    bloc de instructiuni
```

Exemplu

```
i = 10
while (i >= 0):
    print('pozitiv')
    i = i-1
```

Observații:

- Atenție la indentare!
- Putem întrerupe ciclarea cu ajutorul instrucțiunii `break`

Funcții : Definire

Instrucțiune

```
def nume_functie(args):
    bloc de instructiuni
```

Exemplu

```
def minimum(a, b):
    if (a < b):
        out = a
    else:
        out = b
    return out
```

Observații:

- Atenție la indentare!
- Instrucțiunea `def` permite definirea unei funcții
- `args` sunt parametrii (argumentele) funcției
- Instrucțiunea `return` întoarce valoarea funcției și marchează finalul execuției funcției

Funcții : Apelare

Definiția funcției

```
def minimum(a, b):
    if (a < b):
        out = a
    else:
        out = b
    return out
```

Apelarea funcției	Rezultat
`print(minimum(10, 12))`	10
`print(minimum(a=10, b=12))`	10
`print(minimum(b=12, a=10))`	10
`print(minimum(12, 10))`	10

Structuri de Date: Tupluri

Definiția unui tuplu	`t = (2,6,8,10,15,26)`
Dimensiunea unui tuplu	`len(t)`
Acces prin indice	`a = t[0]`
Indici negativi	`e = t[-3:]`
Domeniu de indici	`b = t[2:5]`
Indice negativ	`d = t[-1]`
Modificare ?	`t[2] = 3` EROARE

Structuri de Date: Lista 1/2

Definiția unei liste	`l = [2,6,8,10,15,26]`
Dimensiunea unei liste	`len(l)`
Acces prin indice	`a = l[0]`
Indici negativi	`e = l[-3:]`
Domeniu de indici	`b = l[2:5]`
Indice negativ	`d = l[-1]`
Modificare	`l[2] = 3`

Structuri de Date: Lista 2/2

Cod	Rezultat	Comentariu

`L2 = [32,60,28,60]`
`L2.append(21)`
`print(L2)`
	`[32,60,28,60,21]`	Adaugă un element la finalul listei

`L2.insert(1,53)` `print(L2)`	`[32,53,60,28,60,21]`	Inserează un element la indicele 1
`del L2[3]` `print(L2)`	`[32,53,60,60,21]`	Suprimă elementul de la indicele 3
`a = L2.pop(1)` `print(a)`	`53`	Suprimarea + accesarea elementului de la indicele 1
`L2.reverse()` `print(L2)`	`[21,60,60,32]`	Inversarea ordinii elementelor
`L2.extend([34,55])` `print(L2)`	`[21,60,60,32,34,55]`	Adăugarea unei colecții la finalul listei

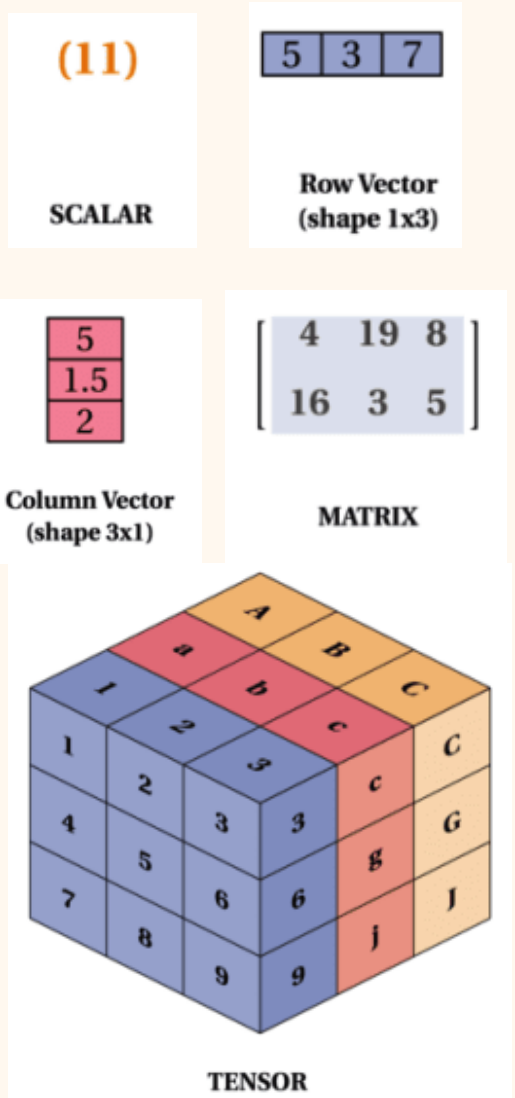
Structuri de Date: Dicționar

Definiția unui dicționar	`d = {'Petre':17, 'Paul':15, 'Maria':16}`
Lista elementelor	`print(d.items())`
Lista cheilor	`print(d.keys())`
Lista valorilor	`print(d.values())`
Modificarea unei valori	`d['Maria'] = 18`
Adaugarea unei perechi cheie-valoare	`d['Ana'] = 22`
Adăugarea unui bloc de elemente	`d.update({'Monica':36, 'Ion':49})`
Suprimarea prin intermediul cheii	`del d['Monica']`

Biblioteci de calcul științific

Numpy

- **Numpy** est un pachet Python specializat în prelucrarea tablourilor (array-urilor) multidimensionale.
- Se bazează pe structura de date `ndarray` – tablou multidimensional.
- Pachetul propune un număr mare de rutine pentru accesul rapid la date (ex. căutare, extragere), pentru prelucrări diverse (ex. sortare), pentru calcule (ex. calcul matematic, statistic etc.)
- Tablourile « numpy » sunt mai performante (rapiditate, gestiune a volumetriei) decât colecțiile obișnuite din Python (ex. liste)



Numpy

Cod	Rezultat
`data1 = [1, 2, 3, 4, 5]`	# listă
`arr1 = np.array(data1)`	# 1d array
`data2 = [range(1, 5), range(5, 9)]`	# listă de liste
`arr2 = np.array(data2)`	# 2d array (matrice)
`arr2.tolist()`	# listă
`np.zeros(10)`	# 1d array de zero-uri
`np.zeros((3, 6))`	# 2d array de zero-uri de forma 3x6
`np.linspace(0, 1, 5)`	# 1d array de 5 elemente echidistante între 0 și 1 (inclus)
`np.logspace(0, 3, 4)`	# 1d array de 4 elemente echidistante în spațiul log
`arr1.dtype`	# tipul valorilor din ndarray (float64)
`arr2.ndim`	# numărul de dimensiuni ale ndarray (2)
`arr2.shape`	# formatul ndarray-ului (2 linii, 4 coloane)
`arr2.size`	# numărul total de elemente (8)
`len(arr2)`	# dimensiunea ndarray-ului după prima axă (2)

Numpy

Cod	Rezultat
`Arr = np.arange(10).reshape((2, 5))`	[[0, 1, 2, 3, 4], [5, 6, 7, 8, 9]]
`arr[0]`	# 0 (elementul situat la indicele 0)
`arr[0, 3]` `arr[0][3] # sintaxă alternativă`	# 3 (elementul situat pe linia 0, coloana 3)
`arr[0, :]`	# 1d array ([1, 2, 3, 4]) (linia cu indicele 0)
`Arr[:, 1]`	# 1d array ([1, 6]) (coloana cu indicele 1)
`arr[:, :2]`	# [[0, 1], [5, 6]] (coloanele cu indici strict mai mici decât 2)
`arr2 = arr[:, 1:4]`	# coloanele între indicii 1 (inclusiv) și 4 (exclusiv)

Pandas

- **Pandas** est un pachet Python ce se bazează pe NumPy, specializat în prelucrarea datelor structurate în format tabelar.
- Propune 2 structuri de date performante:
  - `Series`, care este un vector numpy cu indici cu nume.
  - `DataFrame`, care este o colecție de `Series`, cu aceiași indici.
- Pachetul propune mai multe funcții și metode pentru efectuarea task-urilor de analiză de date (de exemplu transformări, agregări etc.).

Zone	One-Way Miles	Economy	Premium Economy^
1	0-600	8,000	12,000
2	601-1,200	12,000	18,000
3	1,201-2,400	18,000	27,000
4	2,401-3,600	25,000	37,000
5	3,601-4,800	30,000	45,000
6	4,801-5,800	36,000	54,000
7	5,801-7,000	42,000	63,000
8	7,001-8,400	48,000	72,000
9	8,401-9,600	56,000	84,000
10	9,601-15,000	64,000	96,000

Pandas : Crearea unui DataFrame

Metoda 1

```
df1 = pd.DataFrame({
    'name': ['John Smith', 'Jane Doe'],
    'address': ['13 Main St.', '46 Maple Ave.'],
    'age': [34, 28]})
```

Metoda 2

```
df2 = pd.DataFrame([
    ['John Smith', '123 Main St.', 34],
    ['Jane Doe', '456 Maple Ave.', 28],
    ['Joe Schmo', '9 Broadway', 51]
    ],columns =['name', 'address', 'age'])
```

Pandas : Încărcare, salvare și inspectare

Încărcarea/Salvarea unui DataFrame

Încărcarea unui fișier CSV într-un DataFrame

```
df = pd.read_csv('my-csv-file.csv’)
```

Salvarea unui DataFrame într-un fișier CSV

```
df.to_csv('new-csv-file.csv')
```

Inspectarea unui DataFrame

Primele 5 linii

```
df.head(5)
```

Statistici asupra coloanelor (numărul de linii, valori null, dtype)

```
df.info()
```

Pandas : Selectarea unei linii

Selectarea unei linii prin intermediul indicelui

```
march = df.iloc[2]
```

Selectarea mai multor linii în același timp, prin intermediul indicilor lor

```
jan_feb_march = df.iloc[:3]
feb_march_april = df.iloc[1:4]
may_june = df.iloc[-2:]
```

Selectarea unei linii printr-o operație booleană

```
january = df[df.month == 'January']
march_april = df[(df.month == 'March') | (df.month == 'April')]

january_february_march = df[df.month.isin(['January',
'February', 'March'])]
 # dataframe.column_name.isin(list_of_values)
```

Pandas : Adăugarea unei coloane

```
df = pd.DataFrame([
  [1, '3 inch screw', 0.5, 0.75],
  [2, '2 inch nail', 0.10, 0.25],
  [3, 'hammer', 3.00, 5.50],
  [4, 'screwdriver', 2.50, 3.00]],
  columns=['Product_ID', 'Description', 'Cost_to_Manufacture', 'Price'])
```

Adăugarea unei coloane specificând valorile

```
df['is_sold_in_bulk'] = ['Yes', 'Yes', 'No', 'No']
```

Adăugarea unei coloane repetând o singură valoare (broadcasting)

```
df['is_taxed'] = 'Yes'
```

Adăugarea unei coloane pe baza altor coloane

```
df['Revenue'] = df['Price'] - df['Cost_to_Manufacture']
```

Pandas : Efectuarea de operații pe coloane

```
df = pd.DataFrame([
  ['JOHN SMITH', 'john.smith@gmail.com'],
  ['Jane Doe', 'jdoe@yahoo.com'],
  ['joe schmo', 'joeschmo@hotmail.com']],
  columns=['Name', 'Email'])
```

Aplicarea unei operații (sau funcții) pe toate valorile unei coloane

```
df['Name'] = df.Name.apply(lower) #lower, upper
```

Aplicarea unei functii lambda pe toate valorile unei coloane

```
get_last_name = lambda x: x.split(" ")[-1]
df['last_name'] = df.Name.apply(get_last_name)
```

Pandas : Redenumirea coloanelor

Metoda 1: Modificarea atributului .columns al DataFrame-ului

```
df.columns = ['NewName_1', 'NewName_2', 'NewName_3', '...']
```

Metoda 2: Utilizarea metodei .rename() a DataFrame-ului

```
df.rename(columns={
    'OldName_1': 'NewName_1',
    'OldName_2': 'NewName_2'},
    inplace=True)
```

Pandas : Statistici Descriptive

Mean (Average)	`df.column.mean()`
Median	`df.column.median()`
Minimal Value	`df.column.min()`
Maximum Value	`df.column.max()`
Number of Values	`df.column.count()`
Number of Unique Values	`df.column.nunique()`
Standard Deviation	`df.column.std()`
Unique Values	`df.column.unique()`

Pandas : Fuziune/Concatenare

```
sales = pd.read_csv('sales.csv')
targets = pd.read_csv('targets.csv')
men_women = pd.read_csv('men_women_sales.csv’)
```

Fuziune

Metoda 1 : Fuziune simplă

```
sales_targets = pd.merge(sales, targets, how="inner")#how:
"inner"(default), "outer", "left", "right »
```

Metoda 2 : Înlănțuire de fuziuni

```
all_data = sales.merge(targets).merge(men_women)
```

Concatenare

```
bakery = pd.read_csv('bakery.csv')
ice_cream = pd.read_csv('ice_cream.csv')
menu = pd.concat([bakery, ice_cream], axis=0)#axis: axa de-a lungul
căreia se efectuează concatenarea. 0 pentru linii, 1 pentru coloane
```

Pandas : Valori lipsă

Găsirea valorilor lipsă în coloana height

```
df.height.isnull()          #True dacă null, False altfel
df.height.notnull()         #False dacă null, True altfel
df[df.height.notnull()]     #afișează doar liniile pe care height nu
este null
df.height.isnull().sum()    #numără valorile lipsă ale lui height
```

Găsirea valorilor lipsă în DataFrame-ul df

```
df.isnull()                 #DataFrame de valori booleene
df.isnull().sum()           #numără valorile lipsă din fiecare
coloană
```