## Pachetul pandas

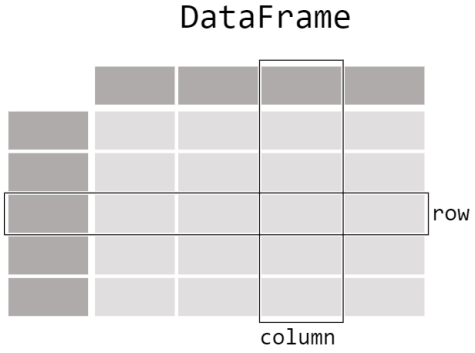
Pandas conține obiecte și funcții pentru prelucrarea datelor tabelare (asemănătoare cu foile de calcul sau bazele de date relaționale); oferă mecanisme avansate de indexare și funcții de procesare complexe. Pentru o prezentare mai detaliată v. acest link.

Pandas permite:

- Manipularea usoara a datelor lipsa
- Adaugarea si stergerea coloanelor
- Alinierea auyomata sau explicita a datelor
- Functionalitati de tip group by - operațiuni de împărțire-aplicare-combinare pe seturi de date, atât pentru agregarea, cât și pentru transformarea datelor
- Merge si join pe data sets
- Instrumente de intrare/iesire pentru încărcarea datelor din fișiere flate (CSV și delimitate), fișiere Excel, baze de date
- Functionalitati specific pentru serii de timp

## Reprezentarea tabelelor de date (DataFrame) in Pandas

Datele în Pandas sunt structurate ca tabele (obiectul **DataFrame**) și ca serii unidimensionale (obiectul **Series**).
Fiecare coloană din **DataFrame** este un obiect **Series**.




Selectarea unei singure coloane este asemănătoare cu selecția valorilor de dicționarului pe baza cheiilor.

Rândurile sunt considerate ca **axa 0** din DataFrame, iar coloanele ca **axa 1**. Unele metode au un argument axis pentru a diferenția între prelucrarea de rânduri și coloane (de ex. drop(), v. secțiunea 2.6).

Cu datele tabulare (DataFrame) este mai util din punct de vedere semantic să ne gândim la **index** (rândurile) și la **coloane**, mai degrabă decât la axa 0 și la axa 1.

Obiectele DataFrame conțin un **index** - implicit acesta conține valori întregi (0, 1, ...) asociate fiecărui rând. Prin intermediul indexului se pot accesa anumite rânduri. Atributul index permite accesarea indexului (ca obiect).

Coloanele au nume (etichete) de coloane (câmpuri, variabile).

Pentru a stoca manual datele într-un tabel, creați un DataFrame. Când utilizați un dicționar Python de liste, cheile de dicționar vor fi folosite ca antete de coloană, iar valorile din fiecare listă ca coloane ale DataFrame.

De exemplu, urmatoarea secventa de cod:

```
import pandas as pd
df = pd.DataFrame(
      {"Name": ["Braund, Mr. Owen Harris", "Allen, Mr. William Henry",
"Bonnell, Miss. Elizabeth"],
       "Age": [22, 35, 58],
       "Sex": ["male", "male", "female"],}
             )
df
```

Va genera urmatorul rezultat

| | Name | Age | Sex |
|---|---|---|---|
| **0** | Braund, Mr. Owen Harris | 22 | male |
| **1** | Allen, Mr. William Henry | 35 | male |
| **2** | Bonnell, Miss. Elizabeth | 58 | female |

Când selectați o singură coloană dintr-un **DataFrame**, rezultatul este o **Serie**. Pentru a selecta coloana, utilizați eticheta coloanei între paranteze drepte [ ].

Selectarea unei singure coloane este asemănătoare cu selecția valorilor de dicționar pe baza cheii:

```
ages = pd.Series([22, 35, 58], name="Age")
ages
```

```
0    22
1    35
2    58
Name: Age, dtype: int64
```

Citirea datelor csv într-un obiect DataFrame se face cu metoda **read_csv()**. Metode similar există pentru citirea din alte formate: **read_excel()**, **read_json()**, **read_sql()**. Pentru salvarea datelor în fișiere se folosesc metodele **to_csv()**, **to_excel()**, **to_json()**, **to_sql()**.

Metodele **head()** si **tail()** returnează primele, respectiv ultimele n (implicit 5) rânduri.

Exemplul: import date din Csv, afișarea unor informații despre DataFrame

```
import pandas as pd
#df = pandas.read_csv(path+'clienti_leasing20.csv')
df = pd.read_csv('clienti_leasing20.csv')
print(df.index)
print('-'*40)
print(df.columns)
print('-'*40)
print(df.head())
print('-'*40)
#print(df.tail())
```

```
RangeIndex(start=0, stop=20, step=1)
----------------------------------------
Index(['ID_CLIENT', 'NAME_CLIENT', 'JOB', 'SEX', 'CURRENCY', 'INCOME_PER_YEAR',
       'DATE', 'AGE'],
      dtype='object')
----------------------------------------
   ID_CLIENT           NAME_CLIENT       JOB SEX CURRENCY  INCOME_PER_YEAR  \
0    3488784      Hajdu Ors Attila  Engineer   m      EUR         23000.00
1    4057671      Harpa Constantin    Worker   m      ROL          1000.00
2    7539733  Horvath Zoltan Ignac    Worker   m      ROL          1635.51
3    3180923         Herta Nicolae  Engineer   m      EUR          7000.00
4    1673453            Hagi Nazim  Engineer   m      ROL         16000.00

         DATE  AGE
0  14-04-2006   35
1  31-08-2009   35
2    8/5/2007   25
3    9/3/2006   33
4  27-10-2009   44
----------------------------------------
```

## Selecția datelor, indexare

Pentru a selecta o singură coloană, utilizați paranteze drepte [ ] cu numele coloanei coloanei de interes. Fiecare coloană dintr-un DataFrame este o serie. Pe măsură ce este selectată o singură coloană, obiectul returnat este o serie Pandas

```
import pandas as pd
df = pd.read_csv('clienti_leasing20.csv')
ages = df["AGE"]
ages.head()
```

```
0    35
1    35
2    25
3    33
4    44
Name: AGE, dtype: int64
```

## Selectia anumitor rânduri și coloane dintr-un DataFrame

Pentru a accesa anumite date (rânduri și coloane) din DataFrame sunt disponibile două atribute:
**iloc**: localizare după index - accesarea datelor folosind indecși întregi ai rândurilor și coloanelor (asemănător cu elementele dintr-o matrice);
**loc**: accesarea datelor după numele coloanelor și etichetele rîndurilor (sau index).

*a.* Utilizarea **iloc** pentru a afișa anumite rânduri

```
import pandas as pd
df = pd.read_csv('clienti_leasing20.csv')
print(df.iloc[2], '\n', type(df.iloc[2]))  #afișare rândul 3 ==> obiect Series
```

```
ID_CLIENT                    7539733
NAME_CLIENT        Horvath Zoltan Ignac
JOB                              Worker
SEX                                   m
CURRENCY                            ROL
INCOME_PER_YEAR                1635.51
DATE                           8/5/2007
AGE                                  25
Name: 2, dtype: object
 <class 'pandas.core.series.Series'>
```

```
print(df.iloc[[1,3,5]], '\n', type(df.iloc[[1,3,5]])) #anumite rânduri (ca listă),
# ==> obiect DataFrame
```

```
   ID_CLIENT       NAME_CLIENT        JOB SEX CURRENCY  INCOME_PER_YEAR  \
1    4057671  Harpa Constantin     Worker   m      ROL           1000.0
3    3180923     Herta Nicolae   Engineer   m      EUR           7000.0
5    2213309       Handa Maria  Professor   f      ROL              0.0

         DATE  AGE
1  31-08-2009   35
3    9/3/2006   33
5    7/9/2010   54
 <class 'pandas.core.frame.DataFrame'>
```

Pentru a selecta anumite coloane se indică un al doilea element în lista aferentă **iloc**. De exemplu: valoarea din rândul 3, prima coloană

```
import pandas as pd
df = pd.read_csv('clienti_leasing20.csv')
print(df.iloc[2, 0])
```

```
7539733
```

Randurile 1, 3, 5 si coloanele 0. 2

```
print(df.iloc[[1,3,5], [0,2]])
```

```
   ID_CLIENT         JOB
1    4057671      Worker
3    3180923    Engineer
5    2213309   Professor
```

Notația **slice** poate fi folosită pentru a specifica rânduri/coloane multiple. Atributul **iloc** cu **slice** se
comportă conform mecanismului obișnuit (include limita inferioară, dar nu include limita superioară.

```
print(df.iloc[3:6])
```

```
   ID_CLIENT    NAME_CLIENT        JOB SEX CURRENCY  INCOME_PER_YEAR  \
3    3180923  Herta Nicolae   Engineer   m      EUR           7000.0
4    1673453     Hagi Nazim   Engineer   m      ROL          16000.0
5    2213309    Handa Maria  Professor   f      ROL              0.0

         DATE  AGE
3    9/3/2006   33
4  27-10-2009   44
5   7/9/2010    54
```

```
print(df.iloc[:5])
```

```
   ID_CLIENT           NAME_CLIENT       JOB SEX CURRENCY  INCOME_PER_YEAR
0    3488784      Hajdu Ors Attila  Engineer   m      EUR         23000.00
1    4057671      Harpa Constantin    Worker   m      ROL          1000.00
2    7539733  Horvath Zoltan Ignac    Worker   m      ROL          1635.51
3    3180923         Herta Nicolae  Engineer   m      EUR          7000.00
4    1673453            Hagi Nazim  Engineer   m      ROL         16000.00

         DATE  AGE
0  14-04-2006   35
1  31-08-2009   35
2    8/5/2007   25
3    9/3/2006   33
4  27-10-2009   44
```

Sau print(df.iloc[15:])

Rândurile 7 - 9, coloane 2 – 4: print(df.iloc[7:10, 2:5])
Primul rând, toate coloanele: print(df.iloc[0, :])

**Utilizare loc - Accesarea datelor pentru afișare rânduri si coloane**

```
import pandas as pd
df = pd.read_csv('clienti_leasing20.csv')
print(df.loc[ : ,'NAME_CLIENT'])
```

```
0               Hajdu Ors Attila
1               Harpa Constantin
2          Horvath Zoltan Ignac
3                 Herta Nicolae
4                    Hagi Nazim
5                   Handa Maria
6          Heredea Marius Petru
7               Hristea Stelian
8                 Haidu Gyongyi
9                  Hurjui Mihai
10           Herta Cornel Romeo
11            Hanes Adrian Paul
12                Herteg Editha
13             Hasnas Luminita
14                Hluscu Adrian
15                   Hanu Maria
16                    Huc Laura
17         Hodea Robert-Emanuel
18                Hincu Gabriel
19    Hreniuc Lacramioara Carmen
Name: NAME_CLIENT, dtype: object
```

**import pandas as pd**
df = pd.read_csv(path+'clienti_leasing20.csv')
print(df.loc[ : ,'NAME_CLIENT'])

*Când se selectează o singură coloană se pot folosi și următoarele expresii:*

*print(df['NAME_CLIENT']) # indexare de tip dicționar*
*print(df.NAME_CLIENT) # coloană ca atribut (dacă numele de coloană este string)*
print(df.loc[5, 'NAME_CLIENT']) *# rd. 5, col. NAME_CLIENT*
print(df.loc[[5,10],['NAME_CLIENT','JOB']]) *# listă rânduri, listă coloane*

Notația slice se poate folosi cu .loc (nume de coloane), dar **rezultatul va include limita superioară** din slice (!)

***Utilizare .loc cu slice***

```
import pandas as pd
df = pd.read_csv(path+'clienti_leasing20.csv')
print(df.loc[ :10, 'ID_CLIENT':'JOB'])      # ! limita superioară inclusă !
print(df.loc[0, :'JOB'])        # slice cu lim. sup.
print(df.loc[1, 'JOB':])        # slice cu lim. inf.
print(df.loc[[0,3], : ])        # slice vid (toate coloanele)
```

**Filtrarea anumitor rânduri dintr-un DataFrame**

Pentru a selecta rânduri pe baza unei expresii condiționate, utilizați o condiție intre parantezele de selecție [ ].

```
import pandas as pd
df = pd.read_csv('clienti_leasing20.csv')
above_35 = df[df["AGE"] > 35]
above_35.head()
```

| | ID_CLIENT | NAME_CLIENT | JOB | SEX | CURRENCY | INCOME_PER_YEAR | DATE | AGE |
|---|---|---|---|---|---|---|---|---|
| **4** | 1673453 | Hagi Nazim | Engineer | m | ROL | 16000.0 | 27-10-2009 | 44 |
| **5** | 2213309 | Handa Maria | Professor | f | ROL | 0.0 | 7/9/2010 | 54 |
| **7** | 3669330 | Hristea Stelian | Asistent medical | m | ROL | 4100.0 | 26-11-2010 | 43 |
| **8** | 1222690 | Haidu Gyongyi | Asistent medical | f | ROL | 1500.0 | 7/3/2009 | 41 |
| **9** | 3678467 | Hurjui Mihai | Engineer | m | EUR | 0.0 | 24-10-2006 | 44 |

Ieșirea expresiei condiționate (>, dar și ==, !=, <, <=,...) este de fapt o **serie Pandas de valori booleene** (fie adevărate, fie false) cu același număr de rânduri ca și DataFrame original.

O astfel de serie de valori booleene poate fi folosită pentru a filtra DataFrame-ul, punându-l între parantezele de selecție [ ]. Vor fi selectate numai rândurile pentru care valoarea este True.

```
import pandas as pd
df = pd.read_csv('clienti_leasing20.csv')
df["AGE"] > 35
```

```
0     False
1     False
2     False
3     False
4      True
5      True
6     False
7      True
8      True
9      True
10     True
11    False
12     True
13     True
14    False
15     True
16    False
17    False
18     True
19    False
Name: AGE, dtype: bool
```

***Similar cu expresia condiționată***, funcția condiționată **isin()** returnează un True pentru fiecare rând în care valorile sunt în lista furnizată. Pentru a filtra rândurile pe baza unei astfel de funcții, utilizați funcția condiționată din parantezele de selecție [ ].

În acest caz, condiția din parantezele de selecție df["AGE"].isin([35, 44]) verifică pentru ce rânduri coloana AGE este fie 35, fie 44.

```
Age = df[df["AGE"].isin([35, 44])]
Age.head()
```

| | ID_CLIENT | NAME_CLIENT | JOB | SEX | CURRENCY | INCOME_PER_YEAR | DATE | AGE |
|---|---|---|---|---|---|---|---|---|
| **0** | 3488784 | Hajdu Ors Attila | Engineer | m | EUR | 23000.0 | 14-04-2006 | 35 |
| **1** | 4057671 | Harpa Constantin | Worker | m | ROL | 1000.0 | 31-08-2009 | 35 |
| **4** | 1673453 | Hagi Nazim | Engineer | m | ROL | 16000.0 | 27-10-2009 | 44 |
| **9** | 3678467 | Hurjui Mihai | Engineer | m | EUR | 0.0 | 24-10-2006 | 44 |
| **19** | 2043893 | Hreniuc Lacramioara Carmen | Professor | f | ROL | 13123.5 | 11/1/2007 | 35 |

**Când se combină mai multe instrucțiuni condiționale:**

- fiecare condiție trebuie să fie înconjurată de paranteze ().
- nu puteți utiliza operatorii or/and, ci utilizați operatorul or **|** și operatorul și **&**.

Age = df[(df["AGE"] == 2) | (df["AGE"] == 3)]

**Este posibil să se selecteze (filtreze) date pe baza unor condiții aplicate asupra loc / iloc** (selectare condiționată / indexare booleană):

print(df.loc[(df['AGE']==35),['NAME_CLIENT']]) *# clienți cu vârsta = 35*

| | NAME_CLIENT |
|---|---|
| 0 | Hajdu Ors Attila |
| 1 | Harpa Constantin |
| 19 | Hreniuc Lacramioara Carmen |

Expresia df ['AGE']==35 generează un obiect Series - vector de valori bool, conținând True/False pentru fiecare rând, în funcție de condiție. Această serie este dată ca argument în df.loc pentru a selecta rândurile care au o valoare asociată True.

*print(df.loc[(df['AGE']==35), ['NAME_CLIENT']])*

print(df['AGE']==35)

**Pentru formularea unor condiții complexe se utilizează operatorii logici: | pt. SAU, & pt ȘI, ~ pt negație. Expresiile trebuiesc grupate în paranteze ().**

*Afișare clienți cu vârsta = 35, care sunt bărbați*
**import pandas as pd**
df = pd.read_csv(path+'clienti_leasing20.csv')
print(df.loc[(df['AGE']==35)&(df['SEX']=='m'),['NAME_CLIENT','JOB','SEX', 'AGE']])

| | NAME_CLIENT | JOB | SEX | AGE |
|---|---|---|---|---|
| 0 | Hajdu Ors Attila | Engineer | m | 35 |
| 1 | Harpa Constantin | Worker | m | 35 |

---

Afișare clienți care nu sunt ingineri

print(df.loc[df['JOB'] != 'Engineer', 'NAME_CLIENT':'INCOME_PER_YEAR'])

**Utilizarea metodelor pe șiruri în condiții**

Metodele de prelucrare a șirurilor de caractere se pot accesa prin atributul **.str** al obiectului Series. În general, acestea au aceleași nume ca metodele clasei string:

**str.len()**, **str.lower()**, **str.upper()**, **str.strip()**, **str.startswith()**, **str.endswith()** etc.

Afișarea numelui si a sexului clienților al căror nume se termină cu 'a'

```
print(df.loc[df['NAME_CLIENT'].str.endswith("a"),['NAME_CLIENT','SEX']])
```

Afișarea numelui si a sexului clienților al căror nume începe cu 'Ha'

```
print(df.loc[df['NAME_CLIENT'].str.startswith("Ha"),['NAME_CLIENT','SEX']])
```

Metoda **DataFrame.isin()** permite verificarea existenței valorilor într-o listă.

Afișare clienți care sunt ingineri sau profesori

```
print(df.loc[df['JOB'].isin(['Engineer', 'Professor']),['NAME_CLIENT','SEX ', 'JOB']])
```

**Modificarea datelor din DataFrame**

**Notația .loc / .iloc permite atribuirea pentru a modifica valorile datelor.**

Modificarea venitului pentru primul rând

```
df.loc[0,'INCOME_PER_YEAR'] = 30000
```

Modificare condiționată: crește veniturile mai mici decât 5000, dacă vârsta e mai mare decât 30.

```
df1.loc[(df1['INCOME_PER_YEAR']<5000)&(df1['AGE']>30),'INCOME_PER_YEAR']= 10000
```

**Calcularea indicatorilor statistici de bază**

Metoda **describe()** returnează un sumar statistic pentru toate coloanele numerice, sau pentru coloanele specificate.

Pentru a afișa statistici pentru toate coloanele se utilizează argumentul **include="all"** Observăm că pentru coloanele de tip object se calculează alți indicatori decât pt. coloanele numerice.

```
import pandas as pd
df=pd.read_csv('clienti_leasing20.csv',usecols=['NAME_CLIENT','JOB','SEX','CURRENCY','INCOME_PER_YEAR','DATE','AGE'])
print(df.describe())
#print(df.describe(include="all"))

print('\n***** describe() pt. coloana JOB *****')
```

```
print(df['JOB'].describe())

print('\n***** describe() pt. coloana AGE *****')
print(df['AGE'].describe())
```

**Statistici descriptive** : metodele **sum()**, **mean()**, **median()**, **nunique()**, **max()**, **min()**

```
df=pd.read_csv(path+'clienti_leasing20.csv',usecols=['NAME_CLIENT','JOB','SEX','CU
RRENCY','INCOME_PER_YEAR','DATE','AGE'])

print('Venit total', df['INCOME_PER_YEAR'].sum())
print('Venit mediu', df['INCOME_PER_YEAR'].mean())
print('Mediana venit', df['INCOME_PER_YEAR'].median())
print('Nr. valori unice',df['INCOME_PER_YEAR'].nunique())
print('Venit maxim', df['INCOME_PER_YEAR'].max())
print('Venit minim', df['INCOME_PER_YEAR'].min())
```

**Alte operații cu DataFrame**

Tipurile de date ale coloanelor sunt recunoscute automat de read_csv(). Acestea sunt memorate în atributul **.dtypes**.
În unele cazuri, este necesar să setăm manual tipul de date al coloanelor.

**Afișarea și modificarea tipului de date al coloanelor**

***Ce tip au coloanele AGE, DATE ?***

```
print(df.dtypes)
```

***Citire 'DATE' ca dată calendaristică***

```
df = pd.read_csv(path+'clienti_leasing20.csv', parse_dates = ['DATE'])
```

***Conversie tip AGE în float***

```
df.AGE = df.AGE.astype(float)
print(df.dtypes)
```

Înlocuirea valorilor lipsă (*missing values*) se face cu **fillna().**
**dropna()** se poate folosi și pentru a șterge rândurile care conțin valori lipsă.

```
In [86]: #Exemplul 21: Înlocuirea valorilor lipsă
import pandas as pd
df=pd.read_csv(path+'clienti_leasing20missing.csv',
usecols=['NAME_CLIENT','JOB','SEX','CURRENCY','INCOME_PER_YEAR','DATE'
,'AGE'])
```

print(df['AGE']) # *verificați dacă există valori lipsă!*
*print(df.loc[df['AGE'].isnull()]) # afișare rânduri cu valori lipsă pt. AGE*
print(df['AGE'].fillna('lipsa')) # *înlocuire cu un șir*

Ștergerea de rânduri și coloane se fae cu metoda **drop()**. Argumentul axis indică dacă se vor șterge rânduri ( axis=0) sau coloane ( axis=1).

drop() returnează un nou DataFrame. Dacă se utilizează argumentul **inplace=True**, se modifică obiectul curent.

***Ștergere coloană - axis = 1***
df1 = df.drop("ID_CLIENT", axis=1)

***Ștergere coloane (ca listă), salvare într-un nou fișier csv***
df3 = df.drop(["JOB", "SEX"], axis=1)
print(df3.head()) df3.to_csv('clienti_df.csv', index = **False**) print('-'*40)

***Ștergere coloană – modifică dataframe current (****inplace = True, drop() returnează None )*
df.drop("INCOME_PER_YEAR", axis=1, inplace = **True**) print(df.head())

***Ștergere rânduri (înregistrări)***

*Șterge rânduri 3, 5, 8*

```
df4 = df.drop([3,5,8],
axis=0)
```

Indexul implicit al rândurilor (nr. întreg) poate fi înlocuit cu valori din coloane ale DataFrame-ului, folosind metoda **set_index()**.

***Șterge clienți care sunt Ingineri.***
set_index() setează coloana JOB ca index. Astfel putem folosi valori din JOB pentru a selecta rânduri.
df5 = df.set_index("JOB") df5.drop("Engineer", axis=0, inplace = **True**)

**Modificări la importul datelor din csv**
La citirea din fișiere csv putem selecta anumite coloane (argumentul **usecols**) și putem exclude anumite rânduri (arg. **skiprows** / **skipfooter**).
Numărul de rânduri ce vor fi citite este specificat cu **nrows**.

df = pd.read_csv(path+'clienti_leasing20.csv', usecols = ['NAME_CLIENT','JOB'],
skiprows = [6,7,9])

***Restricționează nr. de rânduri citite (nrows)***
**import pandas as pd**
df=pd.read_csv(path+'clienti_leasing20.csv',nrows=6,usecols=['NAME_CLIENT','INCOME_PER_YEAR'])

**Sortarea înregistrărilor din DataFrame**

***sortare 1 col., crescător (implicit)***
df7 = df.sort_values(by='INCOME_PER_YEAR')
***sortare descrescatoare.***
df8=df.sort_values(by='AGE', ascending=**False**)
***sortare multiplă***
df9=df.sort_values(by=['JOB', 'AGE'])
***sortare multiplă cu direcție***
df10=df.sort_values(by=['JOB', 'INCOME_PER_YEAR'], ascending=[**True**, **False**])

13123.50

Metoda **rename()** redenumește coloane. Argumentul columns primește un dicționar cu numele curente (chei) și numele noi (valori).

*df = df.rename(columns={"ID_CLIENT": "Id"})*
*df=df.rename(columns={"JOB": "Position"})* df=df.rename(columns={"ID_CLIENT": "ID", "JOB": "Position"}) print(df.columns)

***Aplicarea unei funcții pe șiruri la numele coloanelor***
df=df.rename(columns=str.lower)

Index(['ID', 'NAME_CLIENT', 'Position', 'SEX', 'CURRENCY', 'INCOME_PER_YEAR ', 'DATE', 'AGE'], dtype='object')
---
Index(['id', 'name_client', 'position', 'sex', 'currency', 'income_per_year ', 'date', 'age'], dtype='object')

**Utilizarea modulului csv pentru citirea din fișiere csv**

Un obiect **csv.reader** este creat și utlizat pentru a accesa datele, pe baza unui fișier text (csv) deschis. Acest obiect poate fi iterat într-o structură **for**, pentru a procesa fiecare linie.
La deschiderea fișierului csv se folosește argumentul **newline=""** pentru a asigura tratarea corectă a delimitatorilor sfârșit de linie.

**import csv**

```
with open(path+'clienti_leasing20.csv', 'r', newline='') as f: reader = csv.reader(f)
   for row in reader:
      print (row)
```

***Citirea unei coloane din fișier .csv***

```
import csv
with open(path+'clienti_leasing20.csv', 'r', newline='') as f: reader = csv.reader(f)
   for row in reader:
      print (row[1])
```

***Citire coloane***

```
import csv
with open(path+'clienti_leasing20.csv', 'r', newline='') as f: reader = csv.reader(f)
   for row in reader:
       print (row[1],row[7])          # col. NAME_CLIENT și AGE
      print(row[0:3])                 # primele 3 col.
```

Funcția built-in **enumerate()** se utilizează pentru a genera un contor automat la iterarea cu **for ... in ...**. Este aplicabilă și pentru iterarea rândurilor într-un obiect reader.

***Utilizare enumerate() pentru a citi primele 10 înreg.***

```
import csv
with open(path+'clienti_leasing20.csv', 'r', newline='') as f: reader = csv.reader(f)
      for i, row in enumerate(reader): print(row)
         if(i >= 10):
            break
```

În același scop se poate folosi și un obiect **islice** din modulul **itertools** (v. doc. islice).

***Citirea primelor 10 înreg. cu islice***

```
import csv
from itertools import islice
with open(path+'clienti_leasing20.csv', 'r', newline='') as f: reader = csv.reader(f)
    for row in islice(reader, 10):
       print(row)
```

***Citirea coloanelor csv în liste***

```
id_client    =    []
name_client  =  []
sex = []
with open(path+'clienti_leasing20.csv', 'r', newline='') as f: reader = csv.reader(f)
for row in reader:
```

```
id_client.append(row[0])
name_client.append(row[1])
sex.append(row[3])
```

Obiectul **csv.DictReader** poate fi folosit în locul lui **reader** pentru a citi date într-un dicționar.

Scrierea într-un fișier csv se face cu obiectele **csv.writer** sau **csv.DictWriter**, folosind metoda **writerow()**.

```
import csv
with open(path+'angajati.csv', mode='w', newline='') as f: #creare fișier
    writer = csv.writer(f)
    writer.writerow(['Nume', 'Departament', 'Luna']) writer.writerow(['Cosmin Antonescu', 'Marketing', 'Noiembrie']) writer.writerow(['Eugenia Marin', 'Vanzari', 'Iulie'])
```