Pandas

Koristimo pandas biblioteku za manipulaciju podataka.

Glavni objekat u biblioteci je `DataFrame`, koji služi za rad sa tabelarnim podacima. Može se posmatrati kao matrica redova i kolona koja nudi puno dodatnih funkcionalnosti.

Učitavamo paket i podatke o pokemonima pozivom metode `read_csv()`.

In [98]:
```
import pandas as pd

df = pd.read_csv('pokemon_data.csv')
print(type(df))
```

```
<class 'pandas.core.frame.DataFrame'>
```

U `df` promenljivu smo učitali podatke i napravili smo `DataFrame` objekat.

Vizualizujemo podatke - prvih 5 redova - pozivom metode `head()`.

Napomena u python fajlovima dodati `print` iskaz: `print(df.head())`. U nastavku koda ne poziva print, ali ga je neophodno dodati kako bi se ispisala tabela.

In [99]:
```
df.head() # da vidimo prvih 5
```

Out[99]:

	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
0	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1	False
1	Ivysaur	Grass	Poison	60	62	63	80	80	60	1	False
2	Venusaur	Grass	Poison	80	82	83	100	100	80	1	False
3	VenusaurMega Venusaur	Grass	Poison	80	100	123	122	120	80	1	False
4	Charmander	Fire	NaN	39	52	43	60	50	65	1	False

Ispis svih kolona iz podataka. Pristupamo atributu `columns`.

In [100…
```
print(df.columns)
```

```
Index(['Name', 'Type 1', 'Type 2', 'HP', 'Attack', 'Defense', 'Sp. Atk',
       'Sp. Def', 'Speed', 'Generation', 'Legendary'],
      dtype='object')
```

Koliko ima ukupno redova u podacima:

In [101…
```
# 1. način
num_rows = len(df)
print(num_rows)

# 2. način
num_rows, num_columns = df.shape
print(num_rows)
```

```
800
800
```

Čitanje podataka

Da pristupimo tačno određenoj koloni (npr. kolona `Name`) koristimo `df['Name']` ili `df.Name`. Oba pristupa daju isti rezultat.

In [102…
```
print(df['Name'])
print('\n--------------------\n')
print(df.Name)
```

```
0                  Bulbasaur
1                    Ivysaur
2                   Venusaur
3      VenusaurMega Venusaur
4                 Charmander
               ...
795                  Diancie
796      DiancieMega Diancie
797      HoopaHoopa Confined
798       HoopaHoopa Unbound
799                Volcanion
Name: Name, Length: 800, dtype: object

--------------------

0                  Bulbasaur
1                    Ivysaur
2                   Venusaur
3      VenusaurMega Venusaur
4                 Charmander
               ...
795                  Diancie
796      DiancieMega Diancie
797      HoopaHoopa Confined
798       HoopaHoopa Unbound
799                Volcanion
Name: Name, Length: 800, dtype: object
```

Ipak, ako pristupimo koloni koja ima specijalne karaktere ili razmak (npr. kolona 'Type 1') onda prvi pristup funkcioniše, ali drugi ne. Primer:

In [103…
```
print(df['Type 1'])
print('\n--------------------\n')
# print(df.Type 1) # SyntaxError se dobije
```

```
0        Grass
1        Grass
2        Grass
3        Grass
4         Fire
        ...
795       Rock
796       Rock
797    Psychic
798    Psychic
799       Fire
Name: Type 1, Length: 800, dtype: object

--------------------
```

Dalje možemo da vidimo koliko pokemona ima u svakoj kategoriji `Type 1`, tako što pozovemo metodu `value_counts()`.

In [104…
```
type1_pokemon_count = df['Type 1'].value_counts()
print(type1_pokemon_count)
```

```
Type 1
Water       112
Normal       98
Grass        70
Bug          69
Psychic      57
Fire         52
Electric     44
Rock         44
Dragon       32
Ground       32
Ghost        32
Dark         31
Poison       28
Steel        27
Fighting     27
Ice          24
Fairy        17
Flying        4
Name: count, dtype: int64
```

Možemo da preuzmemo tačan broj pokemona koji pripada vodi, tako što čitamo vrednost `Water`:

In [105…
```
water_pokemon_count = type1_pokemon_count['Water']
print(water_pokemon_count)
```

```
112
```

Indeksiranje redova

Kada želimo da dobijom određeni red iz podataka (npr. 3. red), koristimo `iloc[3]` nad `DataFrame` objektom.

In [106…
```
thrid_row = df.iloc[3]
print(thrid_row)
```

```
Name          VenusaurMega Venusaur
Type 1                        Grass
Type 2                       Poison
HP                               80
Attack                          100
Defense                         123
Sp. Atk                         122
Sp. Def                         120
Speed                            80
Generation                        1
Legendary                     False
Name: 3, dtype: object
```

Kada želimo da dobijmo redove od 3. do 5. reda, koristimo sintaksu kao za *slicing* redova:

In [107…
```
rows = df.iloc[3:5]
rows.head()
```

Out[107]:

	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
3	VenusaurMega Venusaur	Grass	Poison	80	100	123	122	120	80	1	False
4	Charmander	Fire	NaN	39	52	43	60	50	65	1	False

Da iteriramo kroz sve redove postoji više načina. Prikazujemo dva učestala:

In [ ]:
```
# 1. način - već poznato iloc[] indeksiranje
for index in range(len(df)):
    row = df.iloc[index]
    print(row)
    print('---------')

# 2. način - poziv metode iterrows()
for index, row in df.iterrows():
```

```
    print(row)
    print('---------')
```

Ako želimo da iteriramo kroz tačno određene redove, npr. da iteriramo od 3. do 5. reda:

In [109…
```
for index in range(3, 5):
    row = df.iloc[index]
    print(row)
    print('---------')
```

```
Name          VenusaurMega Venusaur
Type 1                       Grass
Type 2                      Poison
HP                              80
Attack                         100
Defense                        123
Sp. Atk                        122
Sp. Def                        120
Speed                           80
Generation                       1
Legendary                    False
Name: 3, dtype: object
---------
Name          Charmander
Type 1              Fire
Type 2               NaN
HP                    39
Attack                52
Defense               43
Sp. Atk               60
Sp. Def               50
Speed                 65
Generation             1
Legendary          False
Name: 4, dtype: object
---------
```

Čitanje redova prema uslovu

Umesto po indeksu, možemo da izdvojimo redove po nekom uslovu. Želimo redove svih vodenih pokemona (videli smo gore da ukupno takvih pokemona ima 112).

In [110…
```
# 1. način
water_pokemon = df[df['Type 1'] == 'Water']

# 2. način
water_pokemon = df.loc[df['Type 1'] == 'Water']
water_pokemon.head() # da vidimo prvih 5
```

Out[110]:

	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
9	Squirtle	Water	NaN	44	48	65	50	64	43	1	False
10	Wartortle	Water	NaN	59	63	80	65	80	58	1	False
11	Blastoise	Water	NaN	79	83	100	85	105	78	1	False
12	BlastoiseMega Blastoise	Water	NaN	79	103	120	135	115	78	1	False
59	Psyduck	Water	NaN	50	52	48	65	50	55	1	False

Možemo napraviti komplikovaniji uslov, da dobijemo vodene pokemone i koji imaju zdravlje `HP` iznad 70 poena:

In [111…
```
water_pokemon_hp_gt_70 = df[(df['Type 1'] == 'Water') & (df['HP'] > 70)]
water_pokemon_hp_gt_70.head() # da vidimo prvih 5
```

Out[111]:

	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
11	Blastoise	Water	NaN	79	83	100	85	105	78	1	False
12	BlastoiseMega Blastoise	Water	NaN	79	103	120	135	115	78	1	False
60	Golduck	Water	NaN	80	82	78	95	80	85	1	False
67	Poliwrath	Water	Fighting	90	95	95	70	90	70	1	False
79	Tentacruel	Water	Poison	80	70	65	80	120	100	1	False

Napomena: naprednije indeksiranje radićemo u narednim terminima vežbi.

Sortiranje vrednosti

Možemo sortirati podatke.

Sortiramo prema `Type 1` rastuće, a prema `HP` (*health points*) opadajuce:

In [112…
```
df_sorted = df.sort_values(['Type 1', 'HP'], ascending=[1,0])
df_sorted.head(100) # prvih 100 redova
```

Out[112]:

	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary
520	Yanmega	Bug	Flying	86	76	86	116	56	95	4	False
698	Volcarona	Bug	Fire	85	60	65	135	105	100	5	False
231	Heracross	Bug	Fighting	80	125	75	40	95	85	2	False
232	HeracrossMega Heracross	Bug	Fighting	80	185	115	40	105	75	2	False
678	Accelgor	Bug	NaN	80	70	40	100	60	145	5	False
…	…	…	…	…	…	…	…	…	…	…	…
246	Houndour	Dark	Fire	45	60	30	80	50	65	2	False
685	Pawniard	Dark	Steel	45	85	70	40	40	60	5	False
568	Purrloin	Dark	NaN	41	50	37	50	37	66	5	False
631	Zorua	Dark	NaN	40	65	40	80	40	65	5	False
284	Poochyena	Dark	NaN	35	55	35	30	30	35	3	False

100 rows × 11 columns

Pisanje novih podataka

Možemo dodati nove kolone. Pravimo kolonu `Total` koja procenjuje koliko vredi svaki pokemon prema nekoj formuli.

In [113…
```
df['Total'] = df['HP'] + df['Attack'] + df['Defense'] + df['Sp. Atk'] + df['Sp. Def'] + df['Spe
df.head()
```

Out[113]:

	Name	Type 1	Type 2	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary	Total
0	Bulbasaur	Grass	Poison	45	49	49	65	65	45	1	False	318
1	Ivysaur	Grass	Poison	60	62	63	80	80	60	1	False	405
2	Venusaur	Grass	Poison	80	82	83	100	100	80	1	False	525
3	VenusaurMega Venusaur	Grass	Poison	80	100	123	122	120	80	1	False	625
4	Charmander	Fire	NaN	39	52	43	60	50	65	1	False	309

Brisanje podataka

Mozemo brisati redove ili kolone (ukoliko smatramo da za to ima potrebe) pozivom metode `drop`.

In [114…
```
# brisanje prva dva reda
df = df.drop([0, 1])

# brisanje kolone Type 2
df = df.drop(columns=['Type 2'])

df.head() # da vidimo prvih 5
```

Out[114]:

	Name	Type 1	HP	Attack	Defense	Sp. Atk	Sp. Def	Speed	Generation	Legendary	Total
2	Venusaur	Grass	80	82	83	100	100	80	1	False	525
3	VenusaurMega Venusaur	Grass	80	100	123	122	120	80	1	False	625
4	Charmander	Fire	39	52	43	60	50	65	1	False	309
5	Charmeleon	Fire	58	64	58	80	65	80	1	False	405
6	Charizard	Fire	78	84	78	109	85	100	1	False	534

Čuvanje podataka

Finalno, sačuvamo naše modifikovane podatke pozivom metode `to_csv()`.

In [115…
```
df.to_csv('modified_pokemon_data.csv', index=False, sep=',')
```