

Leksioni 3

Endri Raco

24 April, 2024



- 1 Hyrje në Analizën e të Dhënave
- 2 Statistikat Përshkruese
- 3 Vizualizimi i të Dhënave në pandas
- 4 Manipulimi i të dhënave
- 5 Zgjedhja e rreshtave dhe shtyllave
- 6 Filtrimi dhe përditësimi i të dhënave
- 7 Vlerat që mungojnë (Missing values)



- 8 Konvertimi i tipit të të dhënave
- 9 Vlerat unike
- 10 Sortimi i të dhënave
- 11 Analiza e avancuar me Python



Section 1

Hyrje në Analizën e të Dhënave



Çfarë është pandas?

- Pandas është një bibliotekë Python që ofron një paketë të fuqishme për analizën e të dhënave.
- Është një mjet i lehtë për t'u përdorur, por shumë i fuqishëm për analizën e të dhënave.



Instalimi i pandas

```
mamba install -c conda-forge pandas
```



- Pandas është një paketë “nivel i lartë”, që përdor shumë biblioteka të tjera si NumPy, matplotlib, dhe SciPy.
- Një nga veçoritë më të dobishme të pandas është aftësia për të ndërvepruar me shumë formate të të dhënave.



- Pandas mund të lexojë dhe shkruajë të dhëna nga shumë formate:
- CSV
- JSON
- HTML
- MS Excel



- Stata
- SAS
- SQL (Postgresql, MySQL, Oracle, MariaDB, etj.)



Strukturat e të Dhënave në Pandas

- Në pandas, të dhënat në formë tabulare ruhen në objekte të tipit DataFrame me rreshta dhe kolona të etiketuara.

Jupyter format **Standard Python format**

Column labels

	YEARMODA	TEMP	MAX	MIN
0	20160601	65.5	73.6	54.7
1	20160602	65.8	80.8	55.0
2	20160603	68.4	77.9	55.6
3	20160604	57.5	70.9	47.3
4	20160605	51.4	58.3	43.2
5	20160606	52.2	59.7	42.8
6	20160607	56.9	65.1	45.9
7	20160608	54.2	60.4	47.5
8	20160609	49.4	54.1	45.7
9	20160610	49.5	55.9	43.0

Data

Index

Pandas Series

Pandas DataFrame
`pandas.core.frame.DataFrame`

Standard Python format

Index

0	65.5
1	65.8
2	68.4
3	57.5
4	51.4
5	52.2
6	56.9
7	54.2
8	49.4
9	49.5

Data

Name: TEMP, dtype: float64

Column label Data type

Pandas Series
`pandas.core.series.Series`

pandas DataFrame

- **DataFrame** është një strukturë e të dhënave 2-dimensionale e përdorur për ruajtjen dhe manipulimin e të dhënave në formë tabelare (të dhëna me rreshta dhe kolona) në Python.
- **DataFrame** mund të krahasohet me një spreadsheet të programueshëm, ku mund të ruani, organizoni dhe analizoni të dhënat me lehtësi.



pandas Series

- **Series** është një strukturë e të dhënave 1-dimensionale që përdoret për ruajtjen dhe manipulimin e një vargu vlerash.
- **Series** është e ngjashme një listë, por më e zgjuar. Një rresht ose një kolonë në një DataFrame të pandas është në fakt një Series e pandas.



Indekse

- Të dy strukturat kanë indekse (indices) që ju lejojnë të qasni të dhënat lehtësisht.



Strukturat e të Dhënave në Pandas

Jupyter format

	YEARMODA	TEMP	MAX	MIN
0	20160601	65.5	73.6	54.7
1	20160602	65.8	80.8	55.0
2	20160603	68.4	77.9	55.6
3	20160604	57.5	70.9	47.3
4	20160605	51.4	58.3	43.2
5	20160606	52.2	59.7	42.8
6	20160607	56.9	65.1	45.9
7	20160608	54.2	60.4	47.5
8	20160609	49.4	54.1	45.7
9	20160610	49.5	55.9	43.0

Standard Python format

	YEARMODA	TEMP	MAX	MIN
0	20160601	65.5	73.6	54.7
1	20160602	65.8	80.8	55.0
2	20160603	68.4	77.9	55.6
3	20160604	57.5	70.9	47.3
4	20160605	51.4	58.3	43.2
5	20160606	52.2	59.7	42.8
6	20160607	56.9	65.1	45.9
7	20160608	54.2	60.4	47.5
8	20160609	49.4	54.1	45.7
9	20160610	49.5	55.9	43.0

Pandas DataFrame

`pandas.core.frame.DataFrame`

Standard Python format

```
0    65.5
1    65.8
2    68.4
3    57.5
4    51.4
5    52.2
6    56.9
7    54.2
8    49.4
9    49.5
Name: TEMP, dtype: float64
```

Pandas Series

`pandas.core.series.Series`

- Krijojmë një folder **data** në folderin tonë të punës
- Shkarkojmë në folderin data skedarin nga linku:

<https://github.com/endri81/instatgis/blob/master/data/albania-Meteo-metadata.txt>



- Për të lexuar të dhënat nga një skedar CSV, përdorim funksionin `read_csv()` nga `pandas`.
- Shembull: Leximi i të dhënave nga një skedar CSV me të dhënat e motit nga Kumpula, Helsinki:

```
import pandas as pd  
data = pd.read_csv("data/albania-Meteo-metadata.txt")
```



- Ky funksion lexon skedarin CSV dhe ruan përmbajtjen në një DataFrame të quajtur data.
- Mund të përdorim metodën head() për të shfaqur disa nga rreshtat e parë të DataFrame.

```
data.head(10)  # Kthen 10 rreshtat e parë
```



Leximi i të Dhënave Tabulare

# Data file contents: Daily temperatures (mean		min	max) for Tirana	Albania
0	# for June 1-30	2016	NaN	NaN
1	# Data source: https://www.ncdc.noaa.gov/cdo-w...	NaN	NaN	NaN
2	# Data processing: Extracted temperatures from... converted to		NaN	NaN
3	# comma-separated format	NaN	NaN	NaN
4	#	NaN	NaN	NaN
5	# Endri Raco - 24.04.2024	NaN	NaN	NaN
6	YEAR	TEMP	MAX	MIN
7	20160601	18.4	23.6	13.1
8	20160602	19.7	26.2	13.6
9	20160603	20.0	NaN	14.0



- Gjatë analizës së një DataFrame, mund të hasim disa probleme si:
 - **Vlerat e çuditshme:** Si NaN (“not a number”), që mund të tregojë një problem me leximin e skedarit.
 - **Vlerat jo të pritura:** Indeksi tregon 36 rreshta, ndërsa duhet të ketë vetëm 30.
 - **Metadata:** Rreshtat e parë përmbajnë informacione që nuk duam t'i procesojmë.



Për të zgjidhur këto probleme:

- Përdorni opsionin `skiprows` për të anashkaluar rreshtat me metadata. Për shembull, për të lexuar vetëm të dhënat relevante:

```
reg_data = pd.read_csv("data/albania-Meteo-metadata.txt", skiprows=1)
```



- Kontrolloni DataFrame-in pas leximit të të dhënave. Përdorni funksionin `.head()` për të parë rreshtat e parë dhe `.tail()` për rreshtat e fundit:

```
reg_data.head()  
reg_data.tail()
```



Analiza e të dhënave me pandas

	YEAR	TEMP	MAX	MIN
0	20160601	18.4	23.6	13.1
1	20160602	19.7	26.2	13.6
2	20160603	20.0	NaN	14.0
3	20160604	14.8	21.2	8.7
4	20160605	11.3	15.1	6.8
5	20160606	11.7	16.2	6.5
6	20160607	14.1	19.0	7.8
7	20160608	12.6	NaN	9.0
8	20160609	10.3	13.1	8.1



- Verifikoni llojin e të dhënave për të konfirmuar që është një DataFrame i pandas:

```
type(reg_data)
```



Për të lexuar vetëm disa kolona specifike nga një skedar CSV:

- Përdorni opsionin `usecols` për të lexuar vetëm kolonat e dëshiruara.
- Për shembull:

```
temp_data = pd.read_csv("data/albania-Meteo-metadata.txt", skiprows=1, usecols=(0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 11, 12, 13, 14, 15, 16, 17, 18, 19, 20, 21, 22, 23, 24, 25, 26, 27, 28, 29, 30, 31, 32, 33, 34, 35, 36, 37, 38, 39, 40, 41, 42, 43, 44, 45, 46, 47, 48, 49, 50, 51, 52, 53, 54, 55, 56, 57, 58, 59, 60, 61, 62, 63, 64, 65, 66, 67, 68, 69, 70, 71, 72, 73, 74, 75, 76, 77, 78, 79, 80, 81, 82, 83, 84, 85, 86, 87, 88, 89, 90, 91, 92, 93, 94, 95, 96, 97, 98, 99))
temp_data.head()
```



Hapat e parë pas ngarkimit të të dhënave:

- Kontrolloni madhësinë e DataFrame për të kuptuar se sa rreshta dhe kolona përmban.
- Përdorni funksionin `len()` për të marrë numrin e rreshtave:

```
len(reg_data)
```



- Përdorni atributin `shape` për të marrë një përmbledhje të shpejtë të madhësisë së të dhënave:

```
reg_data.shape
```

- Kjo tregon numrin e rreshtave dhe kolonave në DataFrame.



Kontrolloni emrat e kolonave:

- Për të parë emrat e kolonave në DataFrame, përdorni `data.columns.values`:

```
reg_data.columns.values
```

- Kjo do të shfaqë të gjitha etiketat e kolonave.



Indikatorët e rreshtave:

- Atributi `index` tregon se si janë indeksuar rreshtat:

```
reg_data.index
```

- Indeksi zakonisht fillon nga 0, përfundon në 30, dhe rritet me 1. Por, në pandas, rreshtat mund të indeksohen edhe me karaktere ose data.



Llojet e të dhënave për çdo kolonë:

- Përdorni `data.dtypes` për të parë llojet e të dhënave në çdo kolonë:

```
reg_data.dtypes
```

YEAR është një vlerë e tipit integer (int64), ndërsa kolonat e tjera janë float64.



Sintaksa bazë për zgjedhjen e kolonave:

- Për të zgjedhur një ose më shumë kolona nga një DataFrame, përdorni sintaksën `dataframe[value]`, ku `value` mund të jetë emri i një kolone ose një listë emrash kolonash.
- Për shembull, për të zgjedhur kolonat “YEARMODA” dhe “TEMP”:

```
selection = reg_data[["YEAR", "TEMP"]]
selection
```



Kontrollimi i tipit të nën-pjesës së zgjedhur:

- Pasi të keni zgjedhur kolonat, mund të kontrolloni tipin e kësaj nën-pjesë:

```
type(selection)
```

- Kjo nën-pjesë është ende një DataFrame i pandas dhe mund të përdorni të gjitha metodat dhe atributet e lidhura me një DataFrame.



Kontrollimi i formës (shape) të zgjedhjes:

- Për të marrë madhësinë e nën-pjesës së zgjedhur, përdorni **shape**:
`selection.shape`



Qasja e një kolone të vetme:

- Për të zgjedhur një kolonë të vetme, përdorni emrin e kolonës brenda kllapave katrore:

```
reg_data["TEMP"]
```



Kontrollimi i tipit të një kolone të vetme:

- Për të parë tipin e një kolone të vetme, përdorni `type()`:

```
type(reg_data["TEMP"])
```

Çdo kolonë dhe çdo rresht në një DataFrame është në fakt një Series i pandas, një strukturë të dhënash 1-dimensionale.



Sintaksë alternative për zgjedhjen e kolonave:

- Ju mund të përdorni një sintaksë alternative për të zgjedhur një kolonë:

```
reg_data.TEMP
```

- Kjo funksionon vetëm nëse emri i kolonës është një emër i vlefshëm për një variabël Python, dhe nuk përmban hapësira.
- Sintaksa `data["column"]` punon për çdo emër kolone, ndaj rekomandohet të përdorni këtë qasje.



Section 2

Statistikat Përshkruese



Metodat e zakonshme për statistikat përshkruese:

- pandas DataFrames dhe Series ofrojnë metoda për të marrë statistikat përshkruese, duke përfshirë `mean()`, `median()`, `min()`, `max()`, dhe `std()` (devijimin standard).



Marrja e vlerës mesatare:

- Për të kontrolluar vlerën mesatare për një kolonë të vetme (Series):

```
reg_data["TEMP"].mean()
```



- Për të marrë vlerën mesatare për të gjitha kolonat në një DataFrame:

```
reg_data.mean()
```



Marrja e statistikave përshkruese për të gjitha kolonat:

- Metoda `describe()` ofron një përmbledhje të shpejtë të statistikave kryesore për të gjitha atributet në DataFrame:

```
reg_data.describe()
```



Section 3

Vizualizimi i të Dhënave në pandas



Grafikët bazë në pandas:

- pandas ka metoda të integruara për vizualizimin e të dhënave, duke përdorur bibliotekën **Matplotlib**.
- Për të krijuar një grafik të thjeshtë që tregon temperaturat:

```
reg_data[["TEMP", "MAX", "MIN"]].plot()
```

- Ky grafik tregon të dhënat për temperatura mesatare, maksimale dhe minimale.



Krijimi i një Series:

- Ju mund të krijoni një Series nga një listë e numrave. Kjo mund të jetë e dobishme për të punuar me të dhënat më shpejt dhe më lehtë:

```
number_series = pd.Series([4, 5, 6, 7.0])  
print(number_series)
```



Krijimi i pandas Series nga Listat

- pandas automatikisht konverton tipet e të dhënave, duke përdorur **float64** kur është e nevojshme.
- Për të vendosur një indeks të personalizuar për Series:

```
number_series = pd.Series([4, 5, 6, 7.0], index=["a", "b",  
print(number_series)
```



Krijimi i pandas DataFrames nga Lista

Krijimi i një DataFrame nga disa lista:

- Përdorni një fjalor Python për të krijuar një DataFrame nga disa lista:

```
# Emrat e stacioneve të motit
stacionet = ["Tirana", "Durrës", "Shkodra", "Elbasan"]

# Koordinatat e gjerësisë gjeografike të stacioneve të motit
gjeresia = [41.33, 41.32, 42.07, 41.11]

# Koordinatat e gjatësisë gjeografike të stacioneve të motit
gjatesia = [19.82, 19.45, 19.52, 20.07]

# Krijimi i DataFrame nga listat
new_data = pd.DataFrame(data={"Emri i Stacionit": stacionet,
                               "Gjerësia": gjeresia,
                               "Gjatësia": gjatesia})
new_data
```

Krijimi i një DataFrame bosh

Punimi me DataFrames bosh:

- Ndonjëherë do të filloni me një DataFrame bosh dhe do të shtoni të dhënat më vonë:

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



Section 4

Manipulimi i të dhënave



Krijimi i Kolonave të Reja në DataFrame

Krijimi i kolonave të reja:

- Një nga gjërat më të zakonshme për të bërë në pandas është krijimi i kolonave të reja bazuar në kalkulime midis kolonave të ndryshme.
- Për të krijuar një kolonë të re, thjesht specifikoni emrin e kolonës dhe caktoni një vlerë të paracaktuar.
- Për shembull, për të krijuar një kolonë “DIFF” me vlerën 0.0:

```
reg_data["DIFF"] = 0.0
```



Kontrollimi i tipit të të dhënave për kolonën e re:

- Mund të kontrolloni tipin e të dhënave të kolonës së re për të konfirmuar që pandas e ka njohur si float:

```
reg_data["DIFF"].dtypes
```



Përditësimi i kolonës “DIFF” me kalkulime

Përditësimi i kolonës “DIFF” me kalkulime:

- Për të llogaritur diferencën midis kolonave “MAX” dhe “MIN”, mund të përdorni një operacion matematikor dhe të përditësoni kolonën “DIFF”:

```
reg_data["DIFF"] = reg_data["MAX"] - reg_data["MIN"]  
reg_data.head()
```



Krijimi i një kolone për të kthyer Celsius në Fahrenheit

Krijimi i një kolone për të kthyer Celsius në Fahrenheit:

- Për të kthyer vlerat nga Celsius në Fahrenheit dhe t'i ruani në një kolonë të re, mund të përdorni formulën $F = (C \times 9/5) + 32$:

```
reg_data["TEMP_FAHRENHEIT"] = (reg_data["TEMP"] * (9/5)) +  
reg_data.head()
```



Section 5

Zgjedhja e rreshtave dhe shtyllave



- Shpesh të zgjidhni rreshta dhe kolona të veçanta në një DataFrame pandas.
- Këtu janë disa mënyra të ndryshme për të zgjedhur nënsete të një DataFrame, të shpjeguara me shembuj dhe në formatin R Markdown.



Zgjedhja e disa rreshtave:

- Për të zgjedhur një nëngrup të veçantë të rreshtave nga një DataFrame, mund të përdorni indeksimin për të marrë pjesë të DataFrame.
- Këtu është një shembull që zgjedh pesë rreshtat e parë dhe i ruan në një variabël të quajtur selection:

```
selection = reg_data.iloc[0:5] # Zgjidhni pesë rreshtat e parë
selection # shfaq përzgjedhjen
```

- Në këtë rast, kemi zgjedhur pesë rreshtat e parë (indekset 0-4) duke përdorur indeksimin e thjeshtë.



Zgjedhja e disa rreshtave dhe kolonave

- Për të zgjedhur një subset të rreshtave dhe kolonave, mund të përdorni **loc** që zgjedh të dhënat bazuar në emrat e kolonave dhe rreshtave.
- Këtu është një shembull që zgjedh vlerat e kolonës “TEMP” nga rreshtat 0-5:

```
selection = reg_data.loc[0:5, "TEMP"]  
selection
```

- Në këtë rast, marrim gjashtë rreshtat e parë (indekset 0-5) duke përdorur emrat e rreshtave.



Zgjedhja e shumëfishtë e kolonave me loc:

- Mund të zgjidhni më shumë se një kolonë duke përdorur loc me një listë kolonash.
- Këtu është një shembull që zgjedh kolonat “TEMP” dhe “TEMP_FAHRENHEIT” nga rreshtat 0-5:

```
# Përdorim të saktë të loc me lista të kolonave  
selection = reg_data.loc[0:5, ["TEMP", "TEMP_FAHRENHEIT"]] #  
selection
```



Zgjedhja e një rreshti të vetëm

- Mund të zgjidhni gjithashtu një rresht të vetëm nga një pozicion i caktuar duke përdorur indeksimin `.loc[]`.
- Këtu zgjedhim të gjitha vlerat e të dhënave duke përdorur indeksin 4 (rreshti i 5-të):
- Zgjidhni një rresht duke përdorur indeksin

```
# Select one row using index  
row = reg_data.loc[4]  
row
```



Zgjedhja e një rreshti të vetëm

- Indeksimi `.loc[]` kthen vlerat nga ajo pozicion si një `pd.Series` ku indekset në fakt janë emrat e kolonave të atyre variablave.
- Prandaj, mund të hyjni në vlerën e një kolone të veçantë duke u referuar në indeksin e saj duke përdorur formatin e mëposhtëm (të dy duhet të funksionojnë):



Zgjedhja e një rreshti të vetëm

- Printoni një atribut nga rreshti i zgjedhur

```
row["TEMP"]
```



Zgjedhja e një vlere të vetme bazuar në rresht dhe kolonë

- Në disa raste është mjaftueshëm të hysh në një vlerë të vetme në një `DataFrame`.
- Në këtë rast, mund të përdorim `DataFrame.at` në vend të `DataFrame.loc`.



Zgjedhja e një vlere të vetme bazuar në rresht dhe kolonë

- Zgjidhni temperaturën (kolonën TEMP) në rreshtin e parë (indeksi 0) të DataFrame tonë.

```
reg_data.at[0, "TEMP"]
```



Zgjedhjet nga pozicioni integer (opsionale)

- `.loc` dhe `.at` bazohen në *etiketat e aksit*, emrat e kolonave dhe rreshtave.
- Etiketat e aksit mund të jenë gjëra të tjera përveç vlerave “tradicionale” të indeksit (p.sh., 0, 1, ...).
- Për shembull, datat kohore shpesh përdoren si indeksi i rreshtave për rreshtat e listuara sipas dates dhe kohës së të dhënave.



Zgjedhjet nga pozicioni integer (opsionale)

- `.iloc` është një operator tjetër indeksimi që bazohet në *vlerat e integer* indeksit.
- Duke përdorur `.iloc`, është e mundur të referohemi gjithashtu në kolonat bazuar në vlerën e tyre të indeksit.
- Për shembull, `reg_data.iloc[0,0]` do të kthejë 20160601 në DataFrame-in tonë shembull.



Zgjedhjet nga pozicioni integer (opsionale)

- Për shembull, mund të zgjidhni nga një grup rreshtash tek kolonat YEAR dhe TEMP bazuar në indeksin e tyre.

```
reg_data.iloc[0:5:, 0:2]
```



Zgjedhjet nga pozicioni integer (opsionale)

- Për të aksesuar vlerën në rreshtin e parë dhe kolonën e dytë (TEMP), sintaksa për `iloc` do të ishte:

```
reg_data.iloc[0, 1]
```



Zgjedhjet nga pozicioni integer (opsionale)

- Gjithashtu mund të qaseni në rreshta individuale duke përdorur `iloc`.
- Le të shohim rreshtin e fundit të të dhënave:

```
reg_data.iloc[-1]
```



Section 6

Filtrimi dhe përditësimi i të dhënave



Filtrimi dhe përditësimi i të dhënave

- Një veçori shumë e dobishme në pandas është aftësia për të filtruar dhe zgjedhur rreshtat në bazë të një deklarate me kusht.
- Në vijim është një shembull se si të zgjidhni rreshtat kur temperatura në Celsius ka qenë më e lartë se 15 gradë dhe t'i ruani ato në variabël **temp_ngroht** (temperature të ngrohta).
- pandas kontrollon nëse kushti është i vërtetë ose fals për çdo rresht, dhe kthen ato rreshta ku kushti është i vërtetë:

```
# Kontrolllo kushtin
```

```
reg_data["TEMP"] > 15
```

```
# Zgjidh rreshtat me temperaturë Celsius më të lartë se 15 gradë
```

```
temp_ngroht = reg_data.loc[reg_data["TEMP"] > 15]
```

```
temp_ngroht
```



- Është gjithashtu e mundur të kombinoni disa kriterë njëkohësisht.
- Këtu, zgjedhim temperaturat mbi 15 gradë që u regjistruan në gjysmën e dytë të qershorit në vitin 2016 (pra. `YEAR >= 20160615`).
- Kombinimi i kriterëve të shumta mund të bëhet me operatorin `&` (DHE) ose operatorin `|` (OSE).
- Është shpesh e dobishme të ndani kushtet e ndryshme duke përdorur paranteza `()`.



```
# Zgjidh rreshtat me temperaturë Celsius më të lartë se 15 gradë  
temp_ngroht = reg_data.loc[(reg_data["TEMP"] > 15) & (reg_data["YEAR"] > 2015)]  
temp_ngroht
```

- Tani kemi një nën-set të DataFrame tonë me vetëm rreshtat ku TEMP është mbi 15 dhe datat në kolonën YEAR fillojnë nga 15 qershori.



- Vini re se vlerat e indeksit (numrat në të majtë) tregojnë ende pozicionet nga DataFrame origjinale.
- Është e mundur të rivendosni indeksin duke përdorur funksionin **reset_index()**, i cili mund të jetë i dobishëm në disa raste për të qenë në gjendje të shkëputni të dhënat në një mënyrë të ngjashme me atë më lart.



- Në mënyrë parazgjedhëse **reset_index()** do të krijojë një kolonë të re quajtur **index** për të mbajtur gjurmën e indeksit të mëparshëm, që mund të jetë e dobishme në disa raste.
- Ky nuk është rasti këtu, kështu që mund ta injorojmë këtë veprim duke kaluar parametrin `drop=True`.




```
# Rivendos indeksin
```

```
temp_ngroht = temp_ngroht.reset_index(drop=True)
```

```
temp_ngroht
```

Siç mund të shihni, tani vlerat e indeksit shkojnë nga 0 në 12 tani.



Section 7

Vlerat që mungojnë (Missing values)



Vlerat që mungojnë (Missing values)

- Siç mund të keni vënë re deri më tani, kemi disa vlera të humbura në kolonat e temperaturës minimale, maksimale, dhe diferencës (MIN, MAX, dhe DIFF).
- Këto vlera të humbura shënohen si **NaN** (jo një numër).



Vlerat që mungojnë (Missing values)

- Të dhënat e munguara në tabelë është një situatë e zakonshme dhe zakonisht duam t'i trajtojmë ato
- Procedurat e zakonshme për të trajtuar vlerat NaN janë ose t'i largoni ato nga DataFrame ose t'i mbushni ato me një vlerë tjetër.
- Në pandas të dyja këto opsione janë të lehta për tu bërë.



Vlerat që mungojnë (Missing values)

- Le të shohim së pari si mund të largojmë vlerat e munguara (pra, të pastroni të dhënat) duke përdorur funksionin **.dropna()**.
- Brenda funksionit mund të kaloni një listë të kolonave nga të cilat vlerat duhet të gjeni NaN duke përdorur parametrin subset.
- Outputi do të heqë çdo rresht që përmban vlera NaN nga grupi i kolonave të dhëna në parametrin subset.



Vlerat që mungojnë (Missing values)

```
# Largo vlerat NaN bazuar në kolonën MIN  
temp_ngroht_clean = temp_ngroht.dropna(subset=["MIN"])  
temp_ngroht_clean
```

Sic mund të shihni nga tabela e mësipërme (dhe ndryshimi në vlerat e indeksit), tani kemi një DataFrame pa vlerat NaN.



Vlerat që mungojnë (Missing values)

- Vini re se zëvendësuam variablin fillestarë 'temp_ngroht' me versionin ku nuk ka të dhëna missing.
- Funkzioni `.dropna()` mund të aplikohet edhe “inplace” që do të thotë që funksioni përditëson objektin DataFrame dhe kthen `None`:

```
temp_ngroht.dropna(subset=['MIN'], inplace=True)
```



Vlerat që mungojnë (Missing values)

- Një opsion tjetër është të mbushni vlerat e munguara me një vlerë duke përdorur funksionin **fillna()**.
- Këtu mund të mbushni vlerat e munguara me vlerën -9999.
- Vini re që këtë herë nuk i jepni parametrin subset.



Vlerat që mungojnë (Missing values)

```
# Mbush vlerat NaN  
fill_data = reg_data.fillna(-9999)  
fill_data.head(5)
```

Si rezultat tani kemi një DataFrame ku vlerat e munguara janë mbushur me vlerën -9999.



Vlerat që mungojnë (Missing values)

- Në shumë raste, mbushja e të dhënave me një vlerë specifike është e rrezikshme sepse ju përfundoni duke ndryshuar të dhënat reale, e cila mund të ndikojë në rezultatet e analizës suaj.
- Për shembull, në rastin e mësipërm do të kishim ndryshuar dramatikisht kolonat e diferencës së temperaturës sepse vlerat -9999 nuk janë një diferencë temperaturash aktuale!



Section 8

Konvertimi i tipit të të dhënave



- Ka raste kur do të duhet të kthejmë të dhënat e ruajtura brenda një Series në një lloj tjetër të të dhënave, për shembull, nga float në integer.
- Kujtoni, që kemi bërë konvertimin e llojeve të të dhënave duke përdorur funksionet e ndërtuara në Python si `int()` ose `str()`.
- Për vlerat në pandas DataFrames dhe Series, mund të përdorim metodën `astype()`.



Kujdes me konvertimet nga float në integer.

- Konvertimi thjesht heq pjesën pas presjes dhjetore, kështu që të gjitha vlerat janë rrumbullakosur poshtë në numrin e plotë më të afërt.
- Për shembull, 99.99 do të shkurtohet në 99 si një integer, kur duhet të rrumbullohet lart në 100.



- Lidhja e funksioneve **round** dhe konvertimi i tipeve e zgjidh këtë situatë, pasi komanda `.round(0).astype(int)` fillimisht rrumbullon vlerat me zero presje dhe pastaj i konverton ato në integer.



Konvertimi i tipit të të dhënave

```
print("Vlerat origjinale:")  
reg_data["TEMP"].head()
```

```
print("Vlerat e shkurtra në integer:")  
reg_data["TEMP"].astype(int).head()
```

```
print("Vlerat e rumbulluara në integer:")  
reg_data["TEMP"].round(0).astype(int).head()
```



Section 9

Vlerat unike



- Ndonjëherë është e dobishme të gjeni vlerat unike që keni në shtyllë.
- Mund ta bëjmë këtë duke përdorur metodën **unique()**:



```
# Merrni vlerat unike të celsiusit  
unik = reg_data["TEMP"].unique()  
unik
```

Si rezultat, marrim një varg vlerash unike në atë kolonë.

- Ndonjëherë nëse keni një listë të gjatë vlerash unike, nuk i shihni domosdoshmërisht të gjitha vlerat unike direkt pasi IPython/Jupyter mund t'i fshehë ato.
- Megjithatë, është e mundur të shihni të gjitha këto vlera duke i printuar ato si një listë.



Vlerat unike

```
# vlerat unike si listë  
list(unik)
```



Sa ditë me temperaturë mesatare unike kemi pasur në qershor 2016

Mund ta verifikojmë këtë!

```
# Numri i vlerave unike  
temperatura_unike = len(unik)  
print(f"Ka pasur {temperatura_unike} ditë me temperature mesatare unike")
```



Section 10

Sortimi i të dhënave



- Shpeshherë është e dobishme të jemi në gjendje të sortojmë të dhënat (në rritje/zbritje) bazuar në vlera në një kolonë.
- Kjo mund të bëhet lehtësisht me pandas duke përdorur funksionin `sort_values(by='EmriKolonësTuaj')`.
- Le të sortojmë së pari vlerat në rend rritës bazuar në kolonën **TEMP**:



Sortimi i të dhënave

```
# Sortoni DataFrame sipas temperaturës, në rritje  
reg_data.sort_values(by="TEMP")
```



Sortimi i të dhënave

Sigurisht, është gjithashtu e mundur t'i sortoni ato në rend zbritës me parametrin `ascending=False`:

```
# Sortoni DataFrame sipas temperaturës, në zbritje  
data.sort_values(by="TEMP", ascending=False)
```



Shkrimi i të dhënave në një file

- Është e rëndësishme të jeni në gjendje të shkruani të dhënat që keni analizuar në një skedar në kompjuterin tuaj.
- Kjo është shumë kollaj në pandas pasi panda supporton shumë formate



Shkrimi i të dhënave në një file

- Formati më tipik output është pa dyshim një skedar CSV.
- Funkzioni `to_csv()` mund të përdoret për të ruajtur lehtësisht të dhënat tuaja në formatin CSV.
- Le të bëjmë save së pari të dhënat nga DataFrame ynë në një skedar të quajtur `TiranaMeteo.csv`.



Shkrimi i të dhënave në një file

```
# përcaktoni emrin e skedarit të daljes  
output_fp = "TiranaMeteo.csv"  
  
# Ruaj DataFrame në csv  
reg_data.to_csv(output_fp, sep="," )
```

Tani kemi të dhënat nga DataFrame ynë të ruajtura në një skedar



Shkrimi i të dhënave në një file

- Siç mund të shihni, vlera e parë në skedarin e të dhënave tani përmban vlerën e indeksit të rreshtave.
- Ka gjithashtu një numër të madh të presjeve dhjetore të pranishme në kolonat e reja që krijuam.
- Le të merremi me këto dhe të ruajmë vlerat e temperaturës nga DataFrame `warm_temps` pa indeksin dhe vetëm me 1 decimal për numrat me presje dhjetore.



Shkrimi i të dhënave në një file

```
# përcaktoni emrin e skedarit të daljes
```

```
output_fp2 = "TiranaMeteo2.csv"
```

```
# Ruaj DataFrame në csv
```

```
temp_ngroht.to_csv(output_fp2, sep="," , index=False, float_for
```



Shkrimi i të dhënave në një file

- Neglizhimi i indeksit mund të bëhet me parametrin **index=False**.
- Përcaktimi se sa presje dhjetore duhet të shkruhen mund të bëhet me parametrin **float_format** ku teksti **%.1f** udhëzon panda-n të përdorë 1 decimal në të gjitha kolonat kur shkruan të dhënat në një skedar.
- Ndryshimi i vlerës 1 në 2 do të shkruante 2 presje dhjetore, dhe kështu me radhë



Section 11

Analiza e avancuar me Python



- Shkarkojmë skedarin **rar** nga linku:

<https://github.com/endri81/instatgis/blob/master/data/029440.rar>

Ekstraktojmë përmbajtjen në folderin tonë **data**



- Në këtë fazë duhet të keni një dosje të re quajtur data që përmban të dhënat me skedarin **029440.txt** në të.
- Mund ta konfirmoni këtë duke shfaqur përmbajtjen e dosjes data:

```
ls data
```



Rreth të dhënave

- Të dhënat input janë të ndara me numër të ndryshëm të hapësirave (dmth, me gjerësi të caktuar të fiksuar).
- Rreshtat dhe kolonat e para të të dhënave duket si më poshtë:

USAF	WBAN	YR	MODA	HRMN	DIR	SPD	GUS	CLG	SKC	L	M	H	VS	B	MW	MW
029440	99999	1906	0101	0600	090	7	***	***	OVC	*	*	*	0.0	**	**	**
029440	99999	1906	0101	1300	***	0	***	***	OVC	*	*	*	0.0	**	**	**
029440	99999	1906	0101	2000	***	0	***	***	OVC	*	*	*	0.0	**	**	**
029440	99999	1906	0102	0600	***	0	***	***	CLR	*	*	*	0.0	**	**	**



- Ne do të zhvillojmë rrjedhën tonë të analizës duke përdorur të dhënat për një stacion.
- Më pas, do të përsërisim të njëjtën proces për të gjithë stacionet.



Për të filluar, le të importojmë pandas:

```
import pandas as pd
```



- Në këtë pikë, ne tashmë mund të hedhim një shikim të shpejtë në dosjen e të dhënave 029440.txt
- Ne mund të vëmë re të paktën dy gjëra që duhet të kemi parasysh kur lexojmë të dhënat:



```
import pandas as pd

# Definoni rrugën relative të skedarit
file_path = "data/029440.txt"

# Lexoni skedarin duke përdorur Pandas
data = pd.read_csv(
    file_path,
    sep='\s+',
    na_values=["*", "**", "***", "****", "*****", "*****"]
)
```



Lexojmë të dhënat edhe njëherë

- Do lexojmë përsëri të dhënat duke mbajtur vetëm disa shtylla
- Duke përdorur `usecols`, ne lexojmë vetëm kolonat që janë të nevojshme për analizën tonë.

Lexoni skedarin duke përdorur Pandas

```
data = pd.read_csv(  
    file_path,  
    sep='\s+',  
    usecols=["USAF", "YR--MODAHRMN", "DIR", "SPD", "GUS", "TEM",  
    na_values=["*", "**", "***", "****", "*****", "*****"]  
)
```

Kontrolli i pjesës së parë të të dhënave të lexuara

```
data.head()
```



- Disa nga emrat e kolonave janë të vështirë për t'u interpretuar.
- Me funksionin **rename**, mund të ndryshojmë emrat e kolonave që të jenë më të kuptueshme.



Krijo një fjalor për emrat e rinj të kolonave

```
new_names = {"YR--MODAHRMN": "TIME", "SPD": "SPEED", "GUS": "C"
```

Riemërto kolonat duke përdorur dictionary-n e krijuar

```
data = data.rename(columns=new_names)
```



Kontrollo emrat e rinj të kolonave

```
data.columns
```



- Pasi kemi lexuar të dhënat, është e rëndësishme të kontrollojmë formën e tyre dhe tipet e kolonave.
- Këtu, ne kontrollojmë rreshtat e parë dhe të fundit, si dhe disa statistika përshkruese.



```
# Kontrolllo formën e të dhënave  
data.shape
```

Kontrollo rreshtat e parë dhe të fundit

```
data.head()  
data.tail()
```



```
data.dtypes
```

Konvertimi i Temperaturave

- Temperaturat janë në Fahrenheit, por ne duam t'i konvertojmë ato në Celsius.
- Për ta bërë këtë, ne krijojmë një funksion dhe e aplikojmë atë në kolonën që përmban temperaturat në Fahrenheit.



Konvertimi i Temperaturave

```
# Funkcioni për konvertimin nga Fahrenheit në Celsius  
def fahr_to_celsius(temp_f):  
    return (temp_f - 32) / 1.8
```



Krijoni një kolonë të re për temperaturat në Celsius

```
data["TEMP_C"] = data["TEMP_F"].apply(fahr_to_celsius)

# Kontrolloni rreshtat e parë pas konvertimit
data.head()
```



Testimi i funksionit

```
# Testimi i funksionit  
fahr_to_celsius(32) # Rezultati duhet të jetë 0.0
```



- Për të iteruar mbi rreshtat, përdorim metodën **iterrows()**.
- Kjo metodë na lejon të kalojmë mbi çdo rresht të një DataFrame-i.
- Më poshtë, tregojmë një shembull të përdorimit të kësaj metode.



Iterimi mbi rreshtat

```
# Për të iteruar mbi rreshtat
for idx, row in data.iterrows():
    # Printimi i vlerës së indeksit
    print(f"Indeksi: {idx}")

    # Printimi i temperaturës në Fahrenheit
    print(f"Temp F: {row['TEMP_F']}")

break # Thyerja e ciklit pas rreshtit të parë për testim
```



Iterimi mbi rreshtat

Krijojmë një kolonë të re për temperaturat në Celsius dhe përdorim funksionin e mëparshëm për të bërë konvertimin.

```
# Krijimi i një kolone të re për Celsius  
data["TEMP_C"] = 0.0
```

```
# Iterimi mbi rreshtat dhe konvertimi në Celsius  
for idx, row in data.iterrows():  
    # Konvertimi i Fahrenheit në Celsius  
    celsius = fahr_to_celsius(row["TEMP_F"])  
  
    # Përditësimi i kolonës TEMP_C  
    data.at[idx, "TEMP_C"] = celsius
```



- Metoda **apply()** në pandas lejon aplikimin e funksioneve në kolonat e caktuara.
- Këtu, do të përdorim `apply()` për të aplikuar funksionin e konvertimit nga Fahrenheit në Celsius.



```
# Aplikimi i funksionit në TEMP_F  
data["TEMP_F"].apply(fahr_to_celsius)  
  
# Ruajtja e rezultateve në TEMP_C  
data["TEMP_C"] = data["TEMP_F"].apply(fahr_to_celsius)
```

