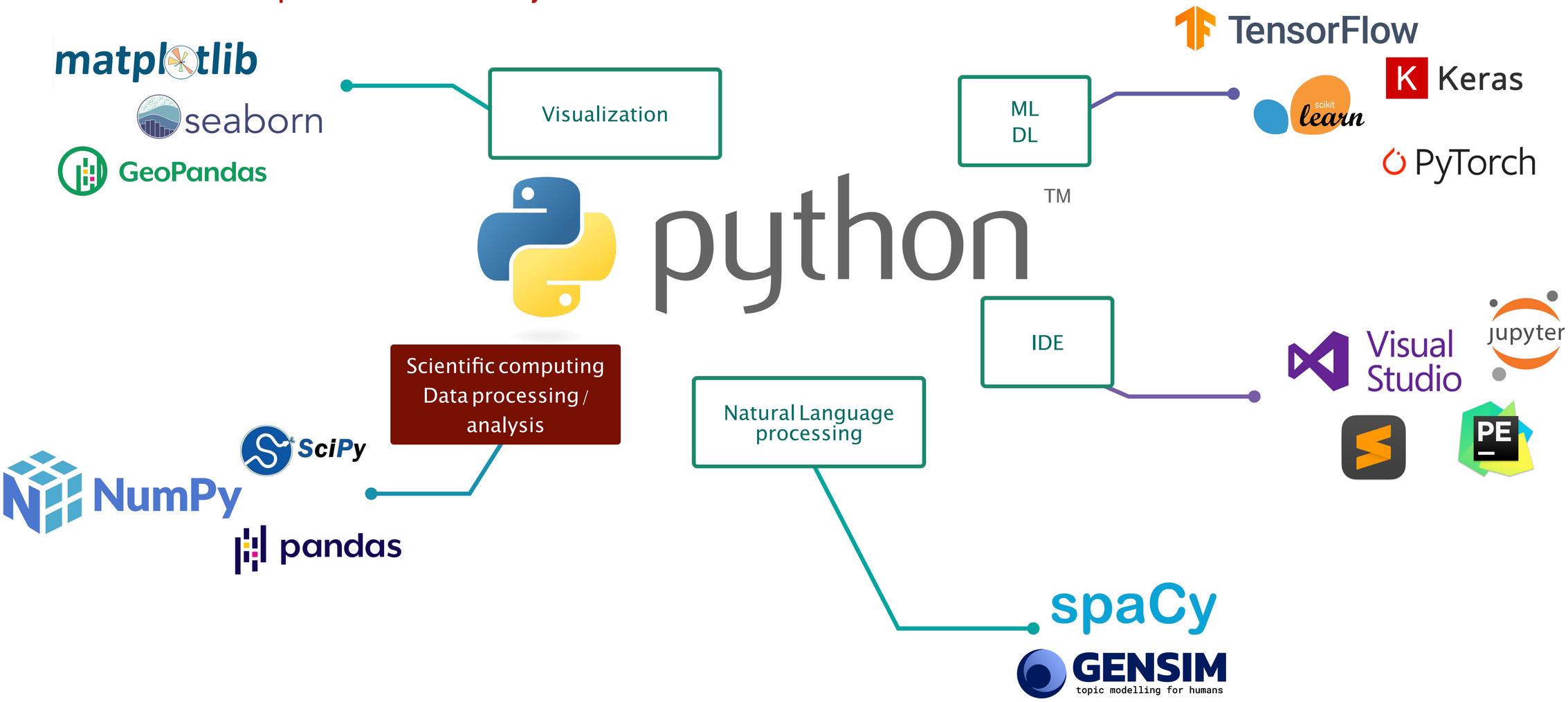
PIAS
PHENIKAA INSTITUTE FOR ADVANCED STUDY

PHENIKAA UNIVERSITY
Realizing your potential

Tien-Lam Pham
Anh-Tuan Nguyen
Phenikaa School of Computing

1. INTRODUCTION

- Pandas is a software library written for the Python programming language for data manipulation and analysis



1. INTRODUCTION

1 Field

1 Record

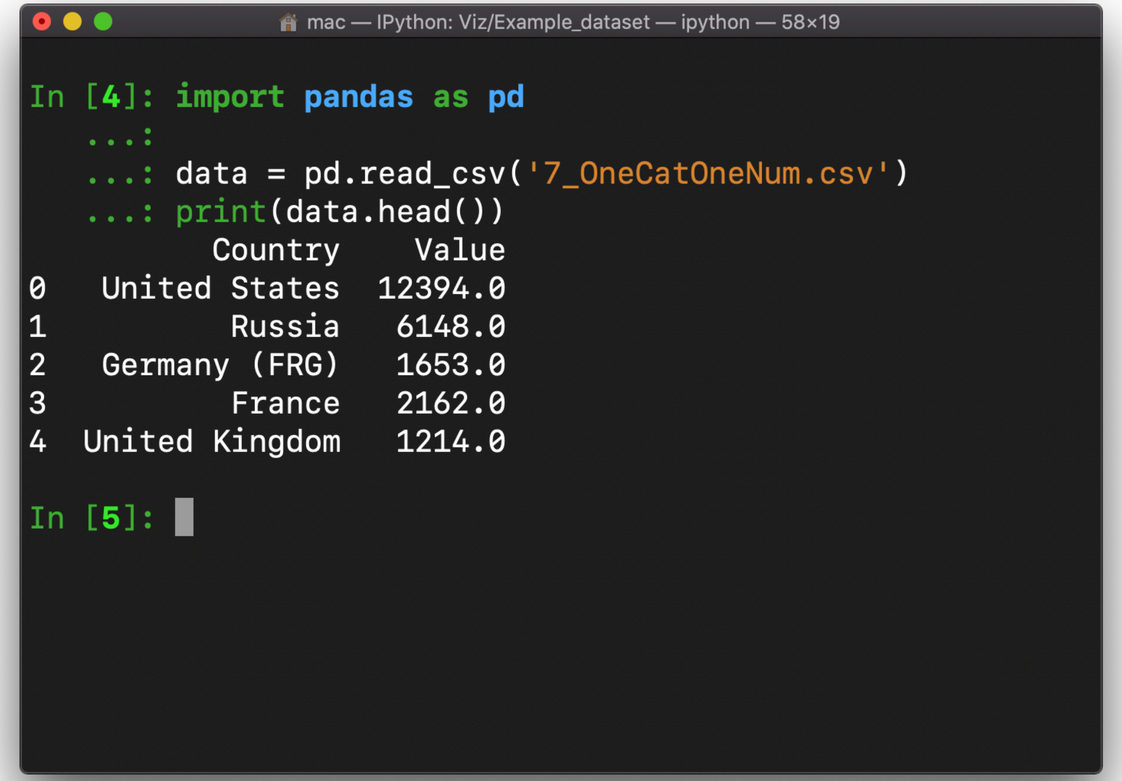
Country	Continent	LifeExp	Pop	GdpPercap
Afghanistan	Asia	43.828	31889923	974.5803384
Albania	Europe	76.423	3600523	5937.029526
Algeria	Africa	72.301	33333216	6223.367465
Angola	Africa	42.731	12420476	4797.231267
Argentina	Americas	75.32	40301927	12779.37964
Australia	Oceania	81.235	20434176	34435.36744
Austria	Europe	79.829	8199783	36126.4927
Bahrain	Asia	75.635	708573	29796.04834
Bangladesh	Asia	64.062	150448339	1391.253792
Belgium	Europe	79.441	10392226	33692.60508
Benin	Africa	56.728	8078314	1441.284873

Field label

Data body

2. DATA IMPORTING

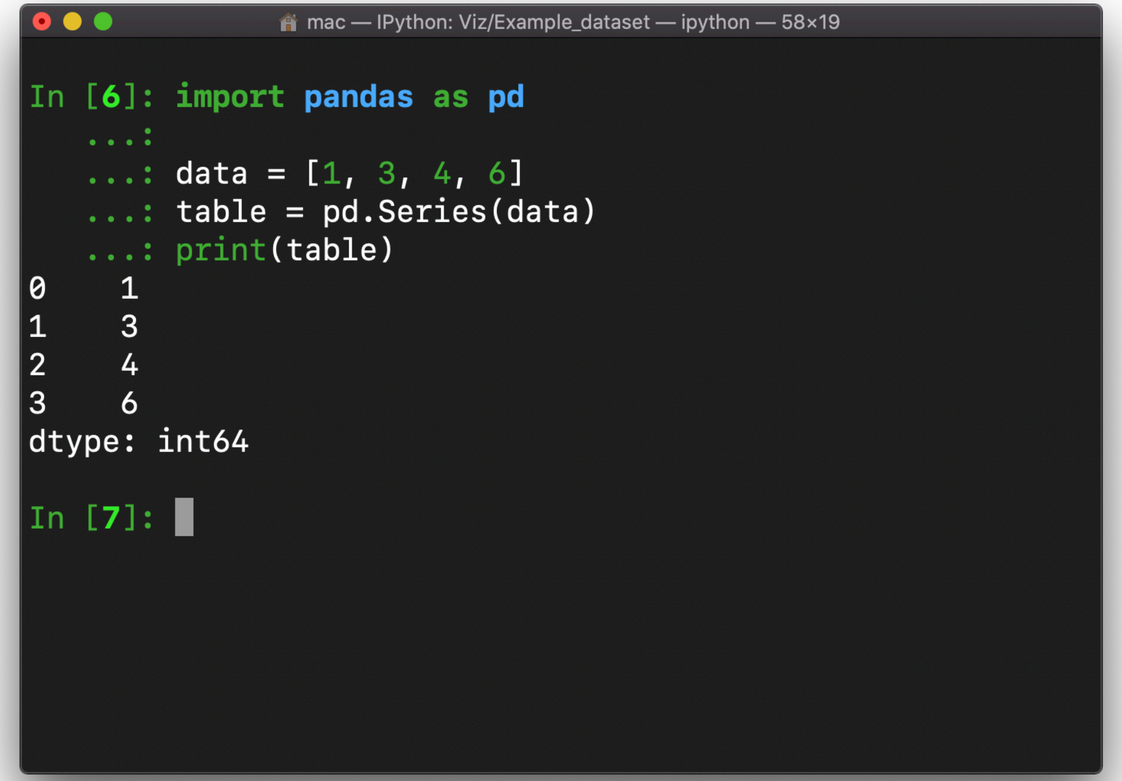
- read_csv
- read_table
- read_exel
- read_sql
- read_json
- read_html



2. DATA IMPORTING

- Series

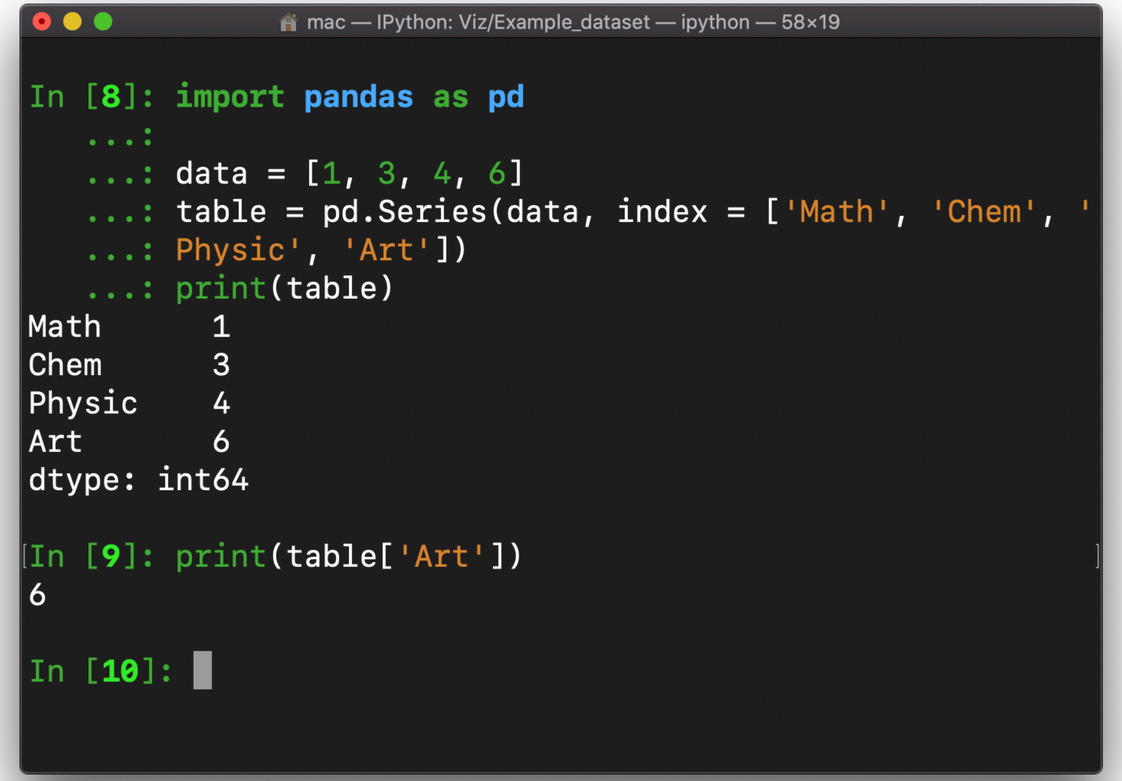
It is like a column in a table



2. DATA IMPORTING

- Series

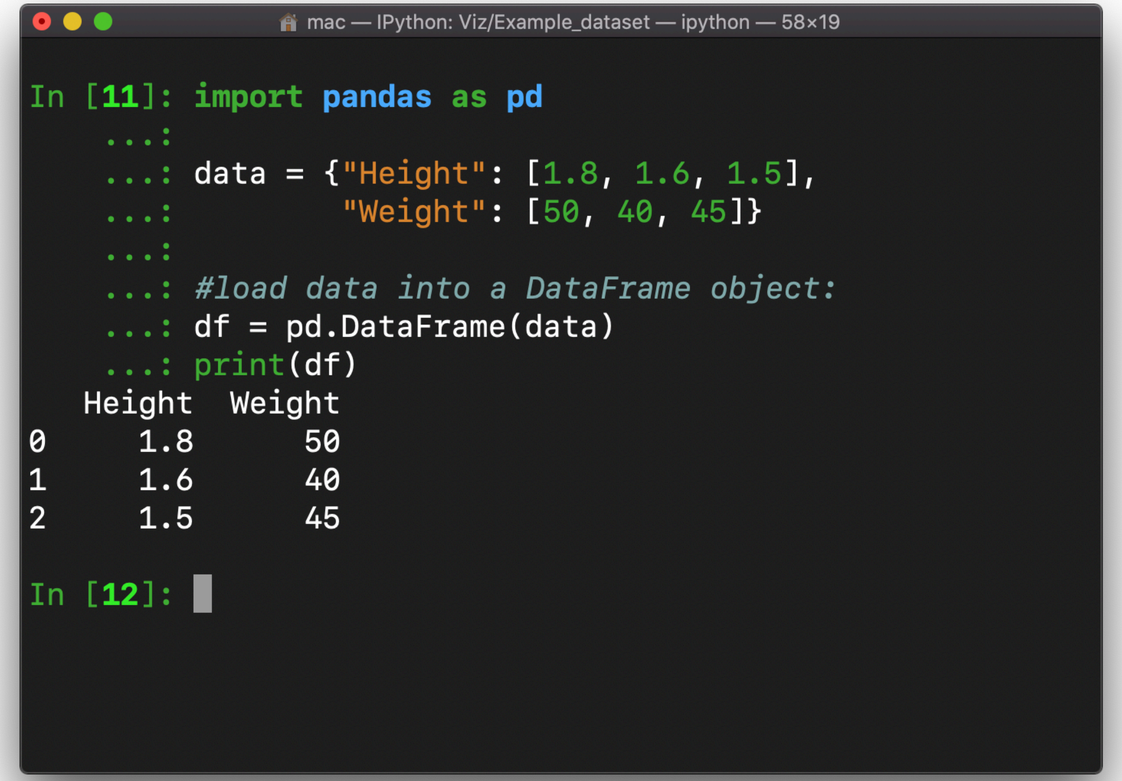
index argument?



2. DATA IMPORTING

- Dataframe

2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.



2. DATA IMPORTING

Dataframe

loc vs iloc ?

- loc gets rows (and/or columns) with particular **labels**.
- iloc gets rows (and/or columns) at integer **locations**.

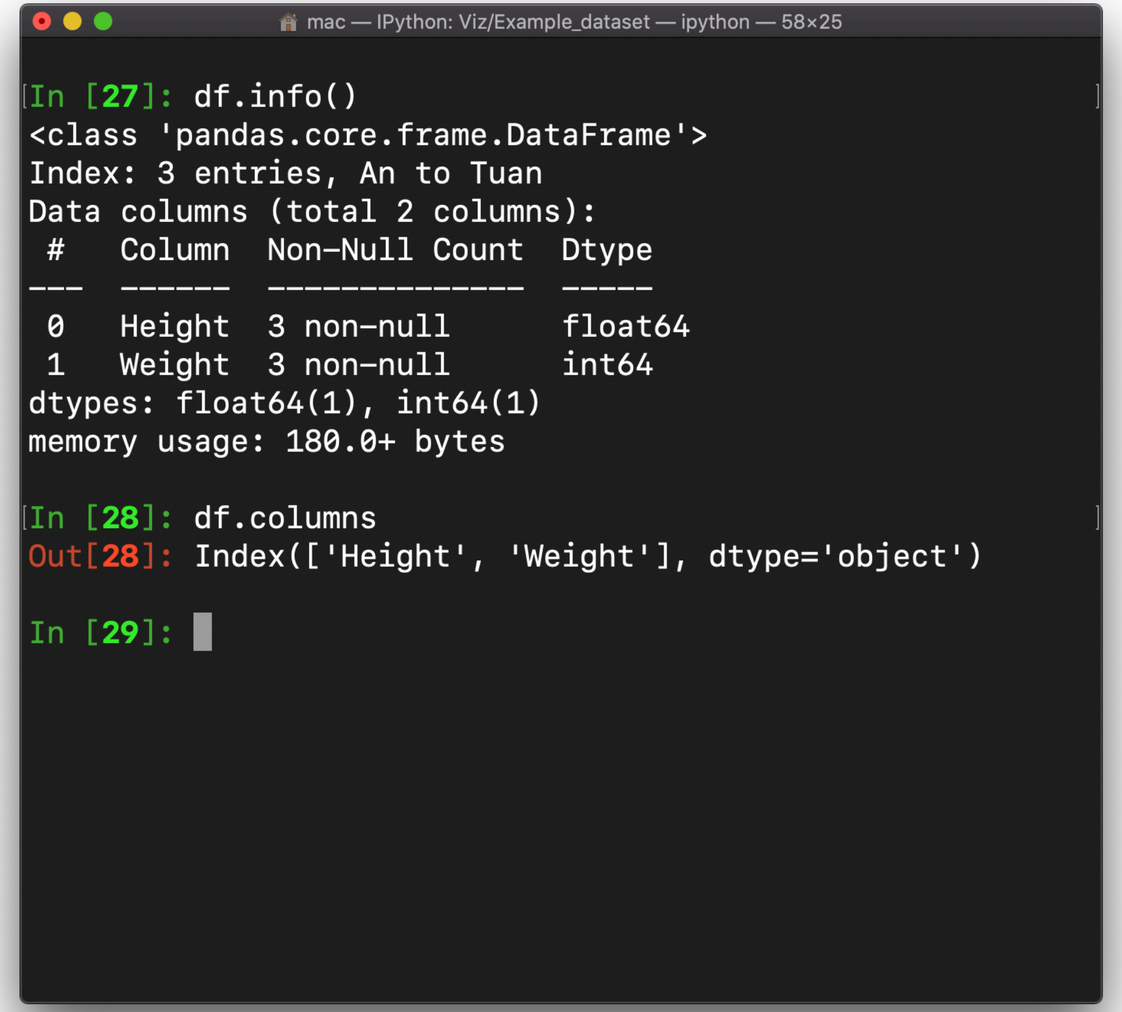
```
mac — IPython: Viz/Example_dataset — ipython — 58×25
In [15]: import pandas as pd
    ...:
    ...: data = {"Height": [1.8, 1.6, 1.5],
    ...:         "Weight": [50, 40, 45]}
    ...:
    ...: #load data into a DataFrame object:
    ...: df = pd.DataFrame(data, index = ['An', 'Tu', 'Tua
    ...: n'])
    ...: print(df)
       Height  Weight
An        1.8      50
Tu        1.6      40
Tuan      1.5      45

In [16]: df.loc['Tu']
Out[16]:
Height     1.6
Weight    40.0
Name: Tu, dtype: float64

In [17]: df.iloc[1]
Out[17]:
Height     1.6
Weight    40.0
Name: Tu, dtype: float64
```

2. DATA IMPORTING

- Dataframe

Common attributes

head, tail, info, columns



3. CLEANING DATA

- Data needs to be cleaned
  - Empty cells
  - Data in wrong format
  - Wrong data
  - Duplicates
  - Outline data

mac — IPython: Viz/Example_dataset — ipython — 65×25

```
In [31]: data
Out[31]:
                Country     1990  ...     2016     2017
0         United States  10732.0  ...  10304.0  12394.0
1                Russia      NaN  ...   6937.0   6148.0
2         Germany (FRG)   1816.0  ...   2535.0   1653.0
3                France   1712.0  ...   2278.0   2162.0
4        United Kingdom   1867.0  ...   1365.0   1214.0
5                 China    941.0  ...   2192.0   1131.0
6          Soviet Union   9777.0  ...      NaN      NaN
7           Netherlands    416.0  ...    471.0   1167.0
8                 Italy    206.0  ...    670.0    660.0
9                Israel     85.0  ...   1433.0   1263.0
10              Ukraine      NaN  ...    535.0    240.0
11                Spain    108.0  ...    507.0    814.0
12               Sweden    231.0  ...    267.0     83.0
13          Switzerland    404.0  ...    186.0    186.0
14               Canada    101.0  ...    144.0     87.0
15          South Korea     72.0  ...    585.0    587.0
16              Belarus      NaN  ...    152.0     23.0
17               Norway     85.0  ...    150.0    134.0
18               Turkey      NaN  ...    311.0    244.0
19         South Africa      NaN  ...     65.0     74.0
20       Czech Republic      NaN  ...    133.0    110.0
```

3. CLEANING DATA

PHENIKAA UNIVERSITY
Realizing your potential

Common atributes

df.dropna()	remove rows that contain empty cells
df.fillna()	replace empty cells with a value
df.sort_value()	sort data by specific column
df.groupby()	Splitting the data into groups/Applying a function to each group/ Combining the results into a data structure.
df.join()/merge()/concate()	combining together Series or DataFrame
df.rename()	Rename Specific Columns

3. CLEANING DATA

- df.dropna()

mac — IPython: Viz/Example_dataset — ipython — 57×28

```
In [52]: data = pd.read_csv('sub_data.csv')

In [53]: data
Out[53]:
           Country     1990     1991     1992
0    United States  10732.0  12515.0  14132.0
1           Russia      NaN      NaN   2605.0
2    Germany (FRG)   1816.0   2482.0   1437.0
3           France   1712.0   1111.0   1185.0
4   United Kingdom   1867.0   1523.0   1180.0
5            China    941.0   1315.0    703.0
6     Soviet Union   9777.0   5667.0      NaN
7      Netherlands    416.0    455.0    365.0
8            Italy    206.0    349.0    248.0
9           Israel     85.0    152.0    324.0
10         Ukraine      NaN      NaN    165.0
11           Spain    108.0    100.0     65.0
12          Sweden    231.0    156.0    149.0
13     Switzerland    404.0    432.0    396.0
14          Canada    101.0    121.0    143.0
15     South Korea     72.0    118.0      NaN
16         Belarus      NaN      NaN      NaN
17          Norway     85.0    170.0     75.0
18          Turkey      NaN      NaN      NaN
19    South Africa      NaN     54.0     75.0

In [54]:
```

mac — IPython: Viz/Example_dataset — ipython — 57×28

```
In [56]: data.dropna(inplace=True)

In [57]: data
Out[57]:
           Country     1990     1991     1992
0    United States  10732.0  12515.0  14132.0
2    Germany (FRG)   1816.0   2482.0   1437.0
3           France   1712.0   1111.0   1185.0
4   United Kingdom   1867.0   1523.0   1180.0
5            China    941.0   1315.0    703.0
7      Netherlands    416.0    455.0    365.0
8            Italy    206.0    349.0    248.0
9           Israel     85.0    152.0    324.0
11           Spain    108.0    100.0     65.0
12          Sweden    231.0    156.0    149.0
13     Switzerland    404.0    432.0    396.0
14          Canada    101.0    121.0    143.0
17          Norway     85.0    170.0     75.0

In [58]:
```

3. CLEANING DATA

- df.fillna()

mac — IPython: Viz/Example_dataset — ipython — 57×28

```
In [52]: data = pd.read_csv('sub_data.csv')

In [53]: data
Out[53]:
           Country     1990     1991     1992
0    United States  10732.0  12515.0  14132.0
1           Russia      NaN      NaN   2605.0
2    Germany (FRG)   1816.0   2482.0   1437.0
3           France   1712.0   1111.0   1185.0
4   United Kingdom   1867.0   1523.0   1180.0
5            China    941.0   1315.0    703.0
6     Soviet Union   9777.0   5667.0      NaN
7      Netherlands    416.0    455.0    365.0
8            Italy    206.0    349.0    248.0
9           Israel     85.0    152.0    324.0
10         Ukraine      NaN      NaN    165.0
11           Spain    108.0    100.0     65.0
12          Sweden    231.0    156.0    149.0
13     Switzerland    404.0    432.0    396.0
14          Canada    101.0    121.0    143.0
15     South Korea     72.0    118.0      NaN
16         Belarus      NaN      NaN      NaN
17          Norway     85.0    170.0     75.0
18          Turkey      NaN      NaN      NaN
19    South Africa      NaN     54.0     75.0

In [54]:
```

mac — IPython: Viz/Example_dataset — ipython — 57×28

```
In [59]: data = pd.read_csv('sub_data.csv')

In [60]: data.fillna(0, inplace=True)

In [61]: data
Out[61]:
           Country     1990     1991     1992
0    United States  10732.0  12515.0  14132.0
1           Russia      0.0      0.0   2605.0
2    Germany (FRG)   1816.0   2482.0   1437.0
3           France   1712.0   1111.0   1185.0
4   United Kingdom   1867.0   1523.0   1180.0
5            China    941.0   1315.0    703.0
6     Soviet Union   9777.0   5667.0      0.0
7      Netherlands    416.0    455.0    365.0
8            Italy    206.0    349.0    248.0
9           Israel     85.0    152.0    324.0
10         Ukraine      0.0      0.0    165.0
11           Spain    108.0    100.0     65.0
12          Sweden    231.0    156.0    149.0
13     Switzerland    404.0    432.0    396.0
14          Canada    101.0    121.0    143.0
15     South Korea     72.0    118.0      0.0
16         Belarus      0.0      0.0      0.0
17          Norway     85.0    170.0     75.0
18          Turkey      0.0      0.0      0.0
19    South Africa      0.0     54.0     75.0
```

3. CLEANING DATA

- df.groupby()

```
In [72]: data = pd.read_csv('10_OneNumSevCatSubgroupsSevO
    ...: bs.csv')

In [73]: data
Out[73]:
     total_bill   tip     sex smoker   day    time  size
0         16.99  1.01  Female     No   Sun  Dinner     2
1         10.34  1.66    Male     No   Sun  Dinner     3
2         21.01  3.50    Male     No   Sun  Dinner     3
3         23.68  3.31    Male     No   Sun  Dinner     2
4         24.59  3.61  Female     No   Sun  Dinner     4
..          ...   ...     ...    ...   ...     ...   ...
239       29.03  5.92    Male     No   Sat  Dinner     3
240       27.18  2.00  Female    Yes   Sat  Dinner     2
241       22.67  2.00    Male    Yes   Sat  Dinner     2
242       17.82  1.75    Male     No   Sat  Dinner     2
243       18.78  3.00  Female     No  Thur  Dinner     2

[244 rows x 7 columns]

In [74]: grouped = data.groupby('sex')

In [75]: grouped.count()
Out[75]:
        total_bill  tip  smoker  day  time  size
sex
Female          87   87      87   87    87    87
Male           157  157     157  157   157   157

In [76]:
```

3. CLEANING DATA

- df.rename()

```
In [84]: import pandas as pd
    ...:
    ...: data = {"Height": [1.8, 1.6, 1.5],
    ...:         "Weight": [50, 40, 45]}
    ...:
    ...: #load data into a DataFrame object:
    ...: df = pd.DataFrame(data)
    ...:
    ...: print(df)
   Height  Weight
0     1.8      50
1     1.6      40
2     1.5      45

In [85]: df.rename(columns={'Height': 'Chieu_Cao'}, inplace=True)

In [86]: df
Out[86]:
   Chieu_Cao  Weight
0        1.8      50
1        1.6      40
2        1.5      45

In [87]:
```

3. DATA STATISTIC

PHENIKAA UNIVERSITY
Realizing your potential

- Common atributes

df.info()
df.describe()
df.mean()
df.median()
df.std()
df.max()/ min()

3. ASSIGNMENT