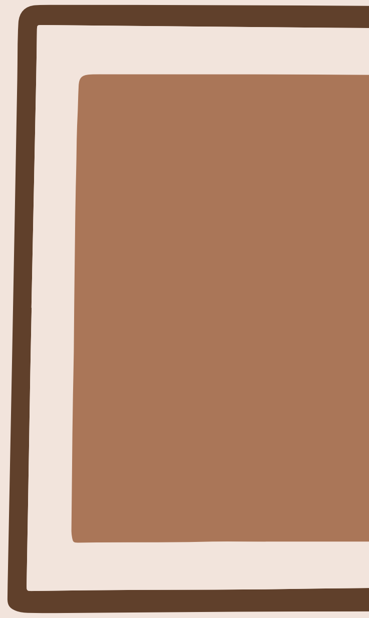


MyEduSolve

Kampus Merdeka INDONESIA JAYA

Group 3

IT SPECIALIST: PYTHON

Part 4

HELLO, ALL!
THIS IS GROUP 3

- Agung Prayogi
- Yesaya Arya Danar Kristuadji
- Dian Maulida
- Dwi Fitriawati Fajrin
- Axel Eldrian Hadiwibowo

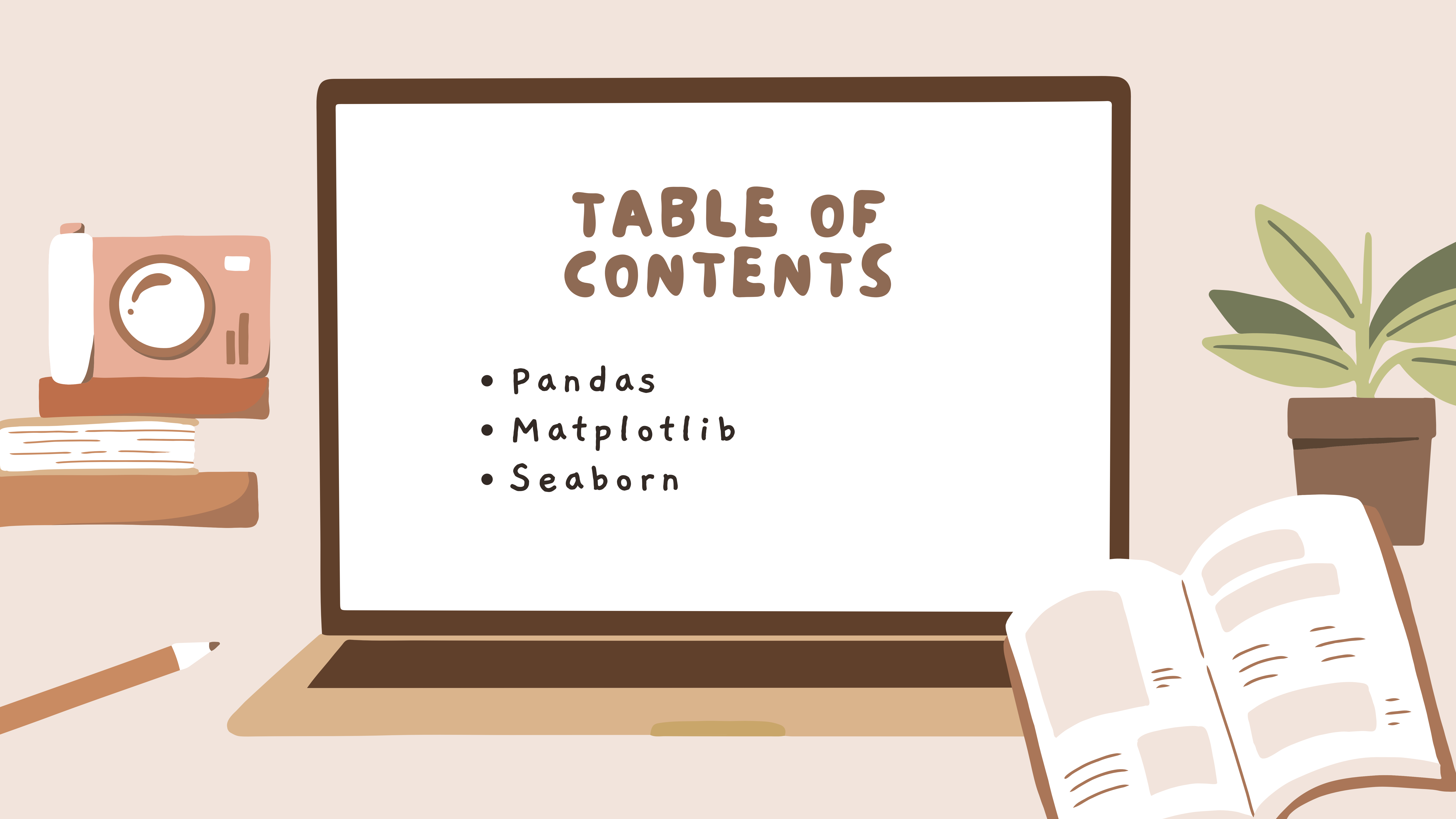


TABLE OF CONTENTS

- Pandas
- Matplotlib
- Seaborn

pandas

WHAT IS PANDAS?

Pandas is a library in Python that provides easy-to-use data structures and data analysis. Pandas is commonly used to create tables, change data dimensions, check data, and so on.

The basic data structure in Pandas is called DataFrame, which makes it easy for us to read a file with many types of formats such as .txt, .csv, .tsv files and can also process data using operations such as join, distinct, group by, aggregation, and other techniques found in SQL.

HOW TO INSTALL PANDAS?

pip

```
pip install pandas
```

anaconda

```
conda install pandas
```

dont forget to import module:

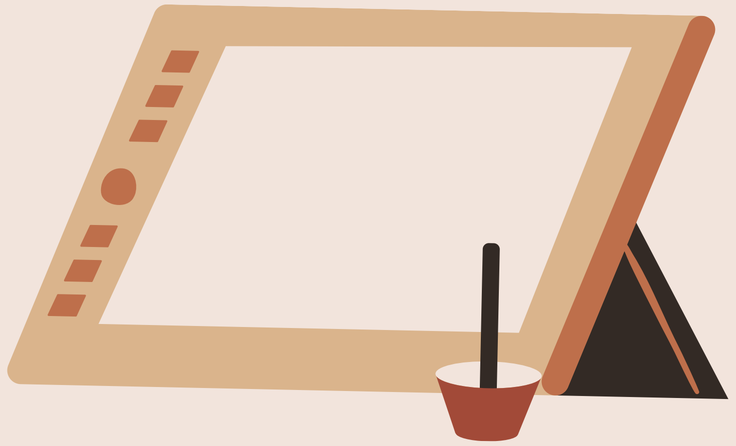
```
import pandas as pd
import numpy as np
```

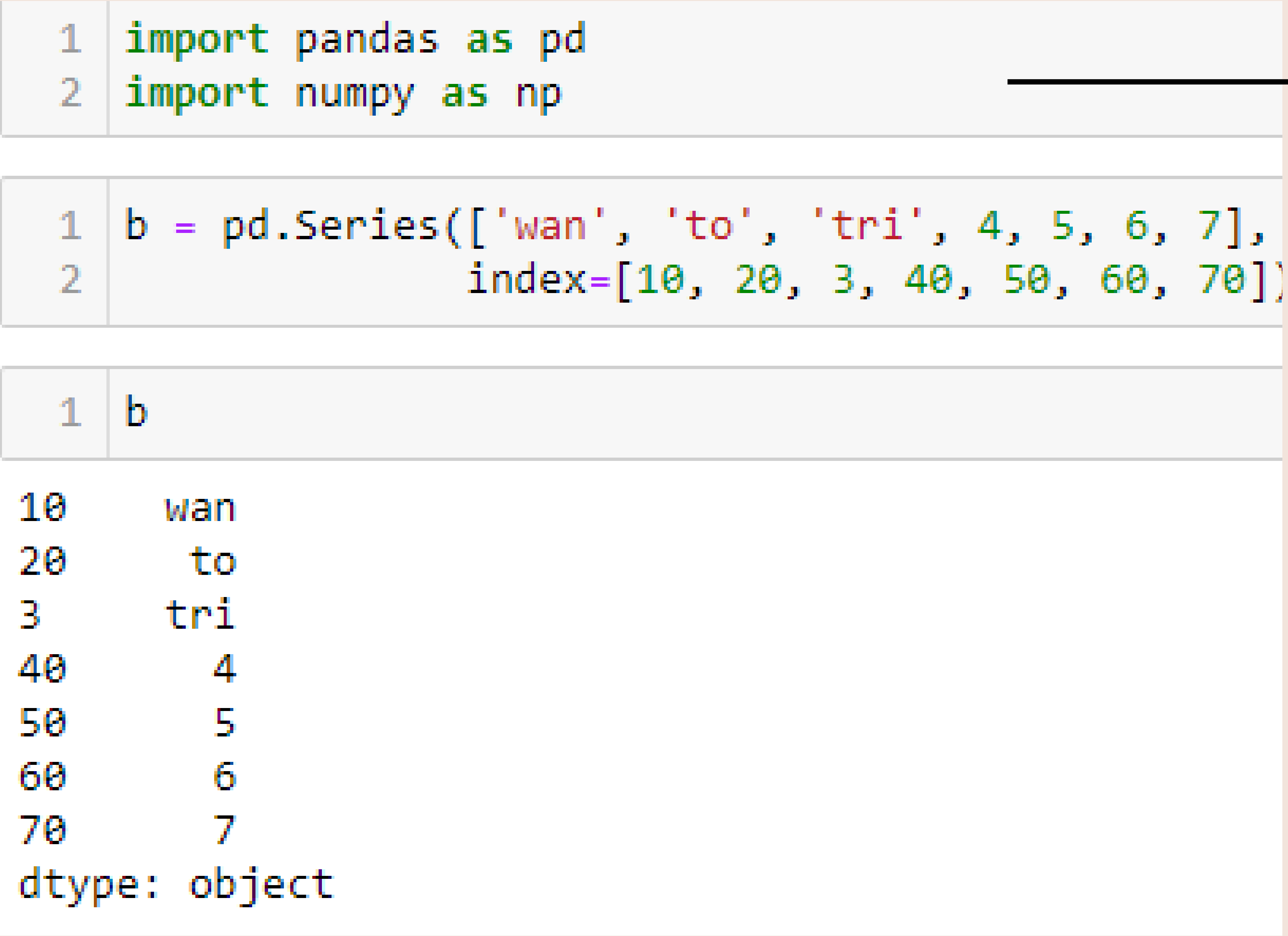
PANDAS

DATA SERIES

Pandas Series is a one-dimensional labeled array capable of holding data of any type (integer, string, float, python objects, etc.). The axis labels are collectively called index.

EXAMPLE





```
import pandas as pd
import numpy as np
```

```
b = pd.Series(['wan', 'to', 'tri', 4, 5, 6, 7],
              index=[10, 20, 3, 40, 50, 60, 70])
```

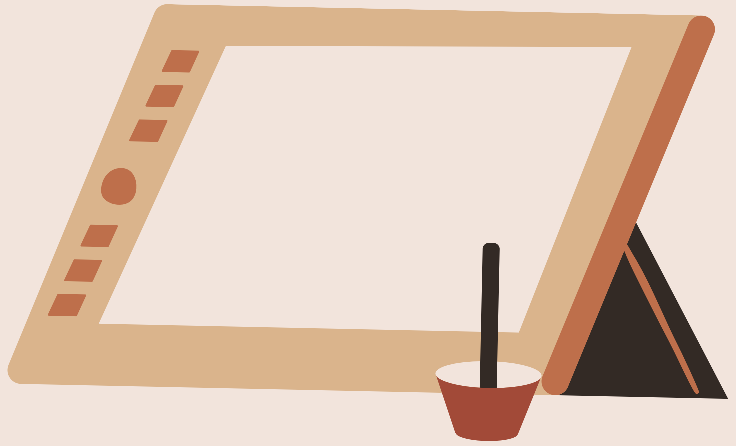
```
b
```

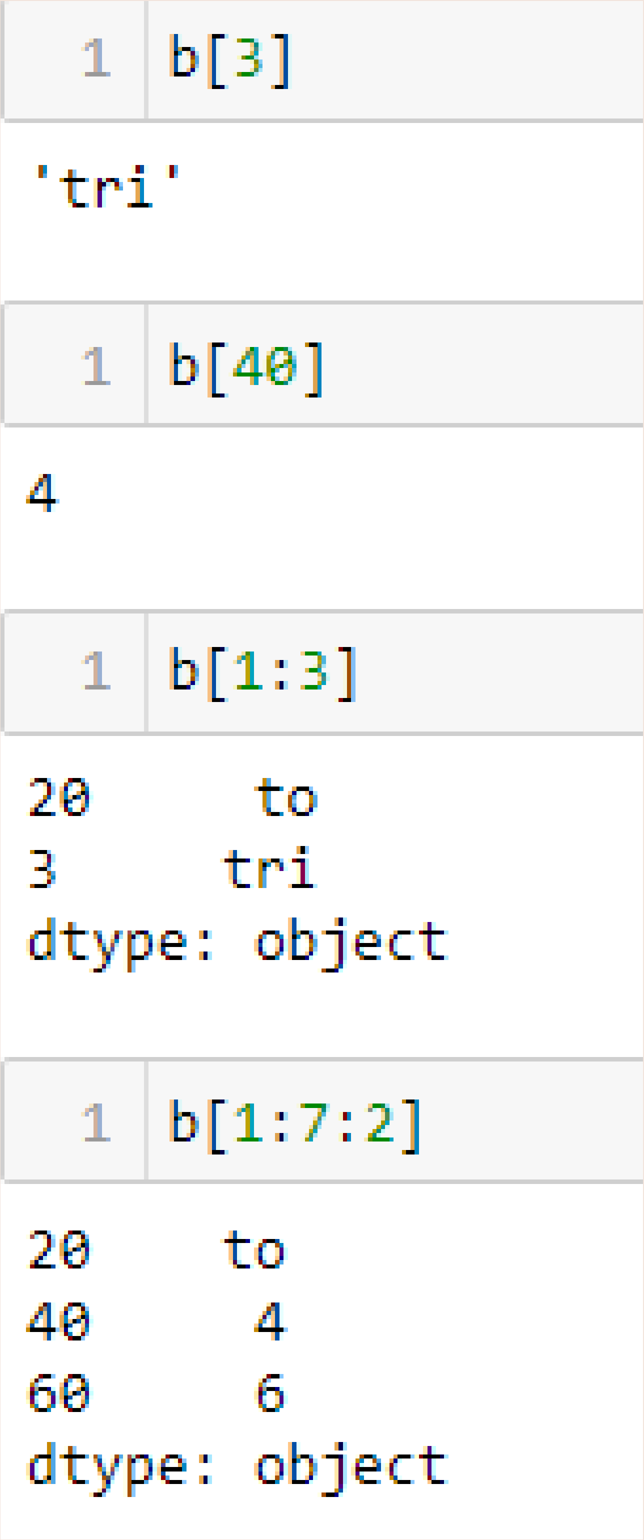
```
10    wan
20     to
3     tri
40      4
50      5
60      6
70      7
dtype: object
```

Import pandas and numpy

The list we made contains string and integer, and we created the label (explicit index) of each element, the label must be as much as the element

EXAMPLE





```
b[3]
```

```
'tri'
```

```
b[40]
```

```
4
```

```
b[1:3]
```

```
20     to
3     tri
dtype: object
```

```
b[1:7:2]
```

```
20    to
40     4
60     6
dtype: object
```

→ Call the element by its label

→ For slicing 2 and 3 parameters, we use the implicit index, the default index by python that begin with 0

LOC i LOC

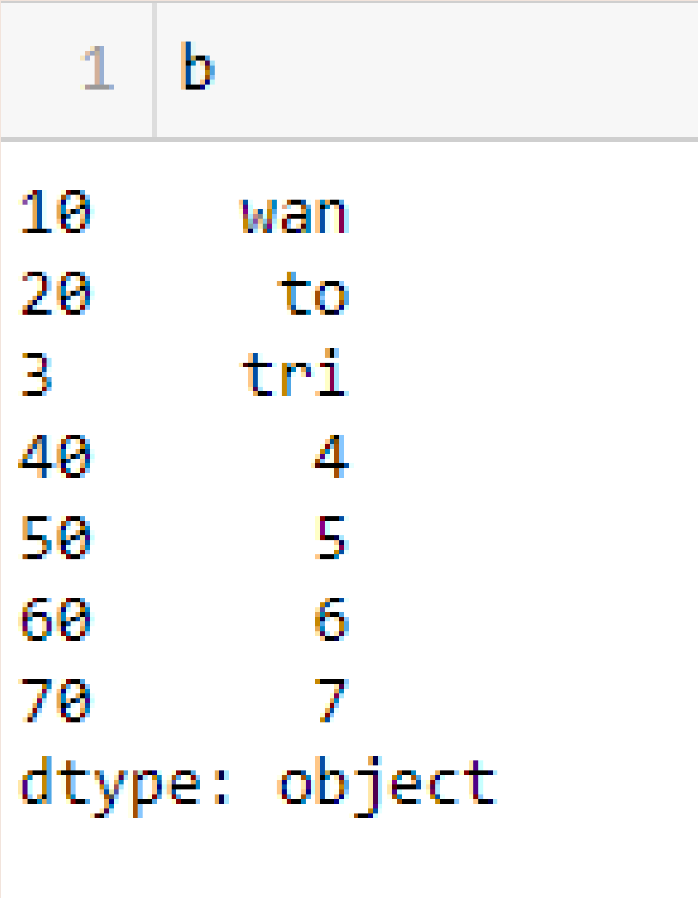
LOC

loc is label-based, which means that you have to specify rows and columns based on their row and column labels

ILOC

iloc is integer position-based, so you have to specify rows and columns by their integer position values (0-based integer position)

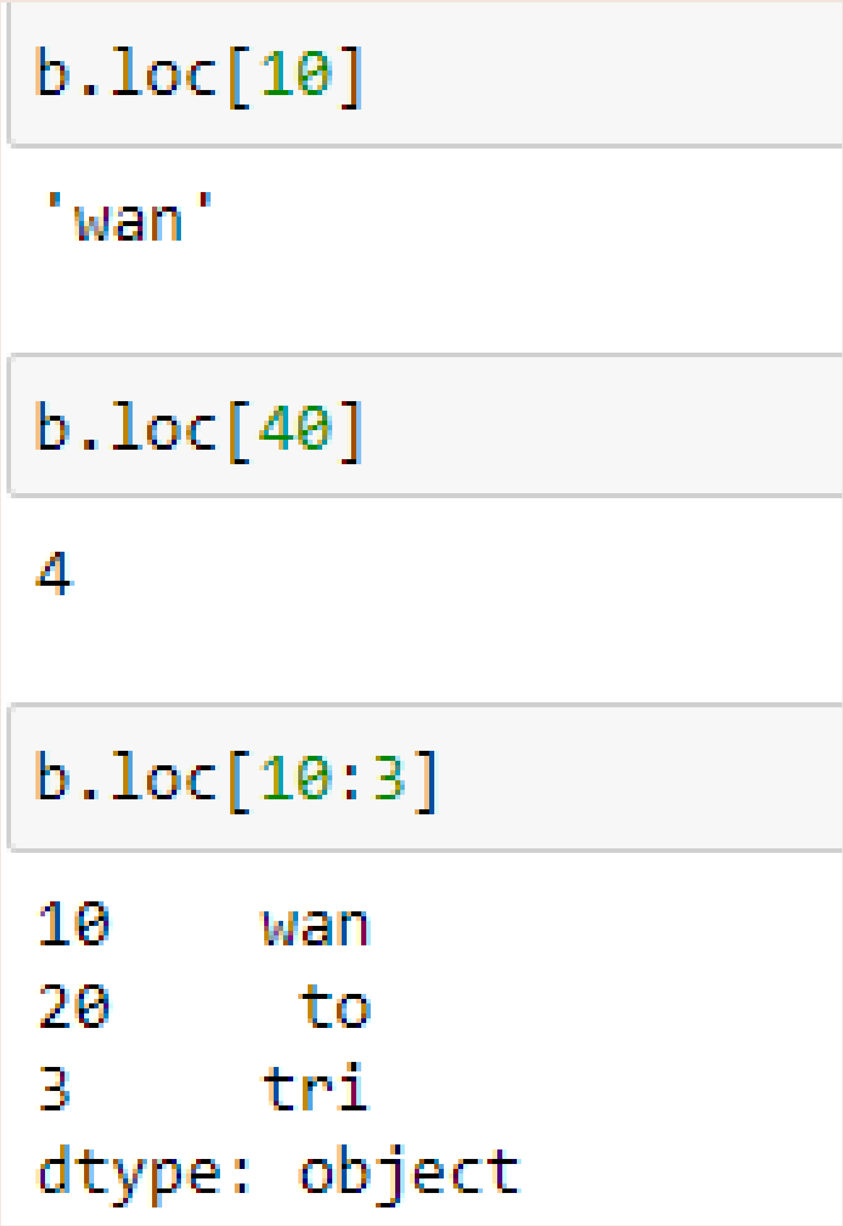
EXAMPLE USING LOC



```
b
```

```
10     wan
20      to
3      tri
40       4
50       5
60       6
70       7
dtype: object
```

Data series that we had before



```
b.loc[10]
```

```
'wan'
```

```
b.loc[40]
```

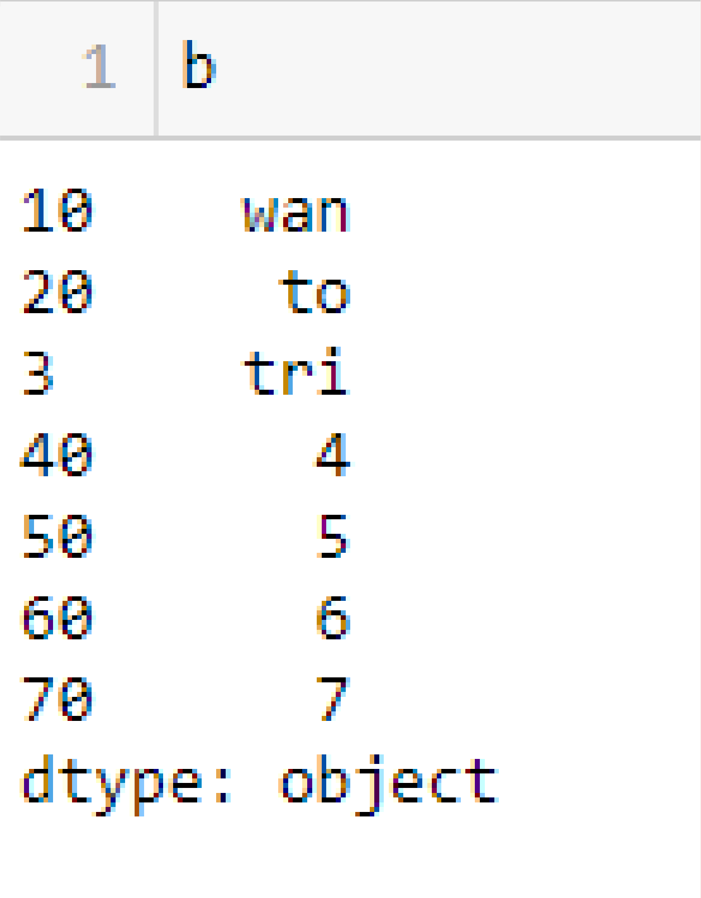
```
4
```

```
b.loc[10:3]
```

```
10     wan
20      to
3      tri
dtype: object
```

Using loc, we call the lement by its label we custom before, and for slicing, the index is exclusive, so its 'stop' will be called

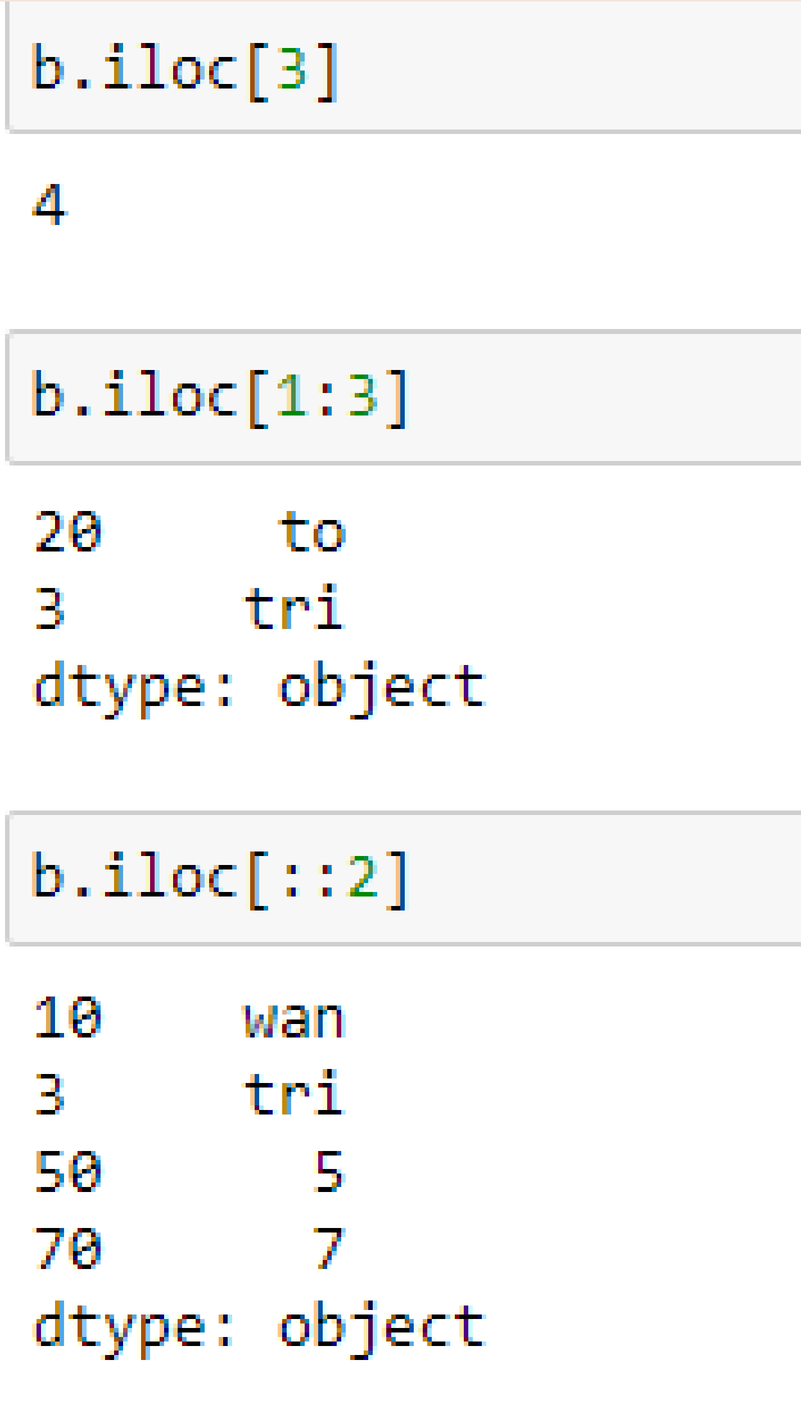
EXAMPLE USING iLOC



```
b
```

```
10     wan
20      to
3      tri
40       4
50       5
60       6
70       7
dtype: object
```

Data series that we had before



```
b.iloc[3]
```

```
4
```

```
b.iloc[1:3]
```

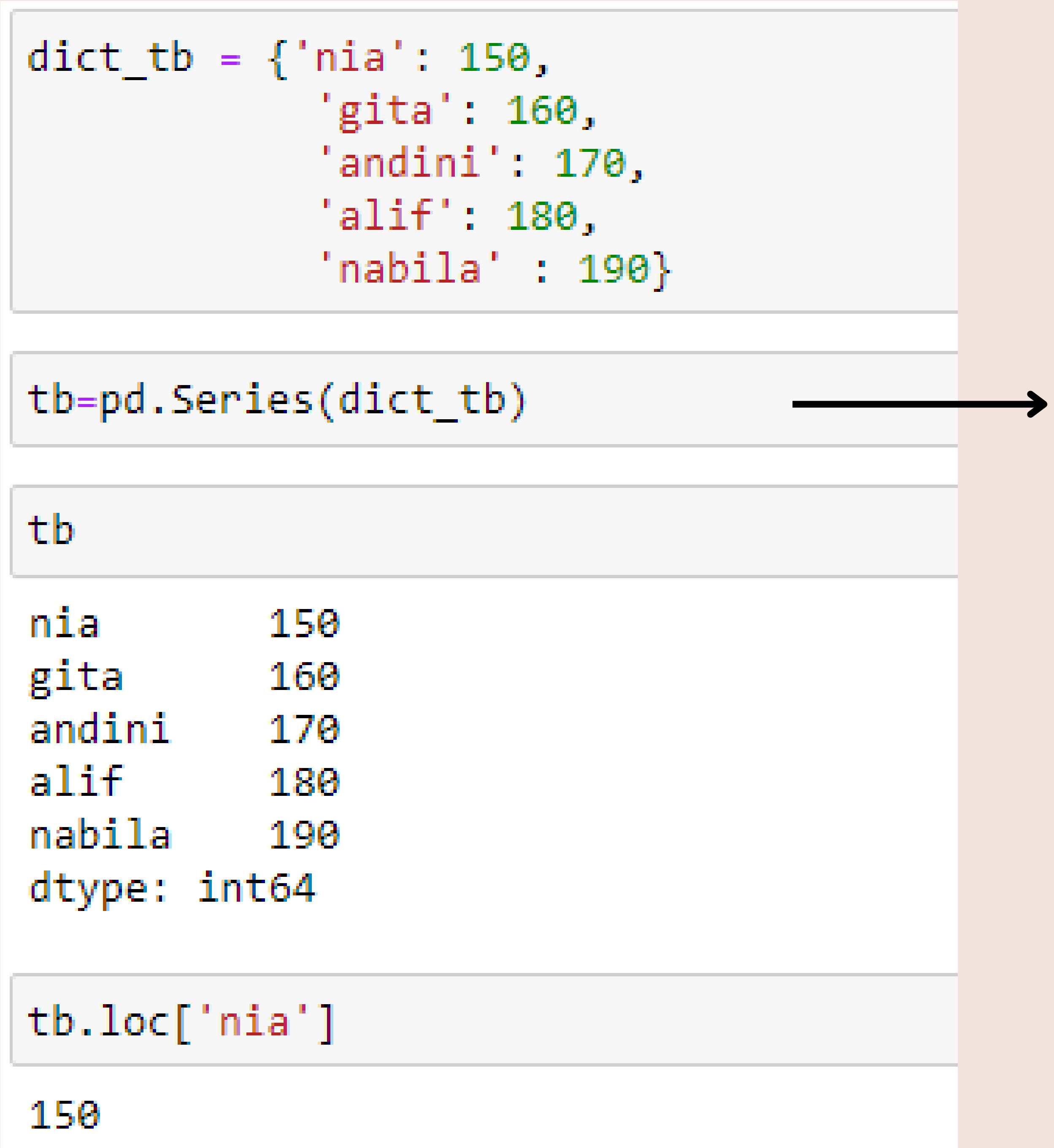
```
20      to
3      tri
dtype: object
```

```
b.iloc[::2]
```

```
10     wan
3      tri
50       5
70       7
dtype: object
```

Using iloc, we call the element by its implicit index, the default index by python that starts by 0, and for slicing, the index is inclusive, so its 'stop' will not be called

MAKE DATA SERiES FROM DiCTiONARY



```
dict_tb = {'nia': 150,
           'gita': 160,
           'andini': 170,
           'alif': 180,
           'nabila' : 190}
```

```
tb=pd.Series(dict_tb)
```

```
tb
```

```
nia       150
gita      160
andini    170
alif      180
nabila    190
dtype: int64
```

```
tb.loc['nia']
```

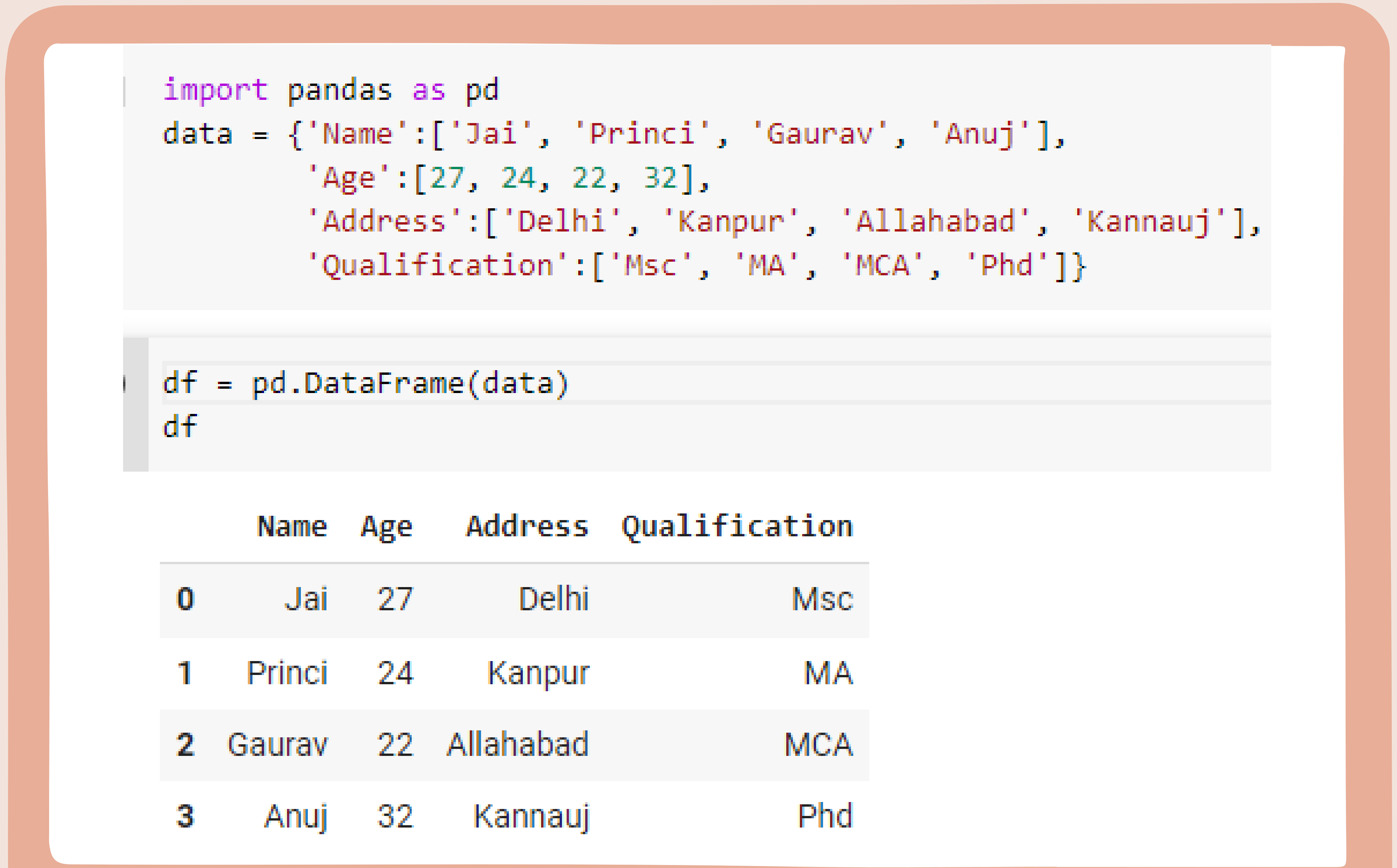
```
150
```

The transformation from dictionary to data series

The 'key' from dictionary would be the explicit index

DATAFRAME

A Pandas DataFrame is a 2 dimensional data structure, like a 2 dimensional array, or a table with rows and columns.



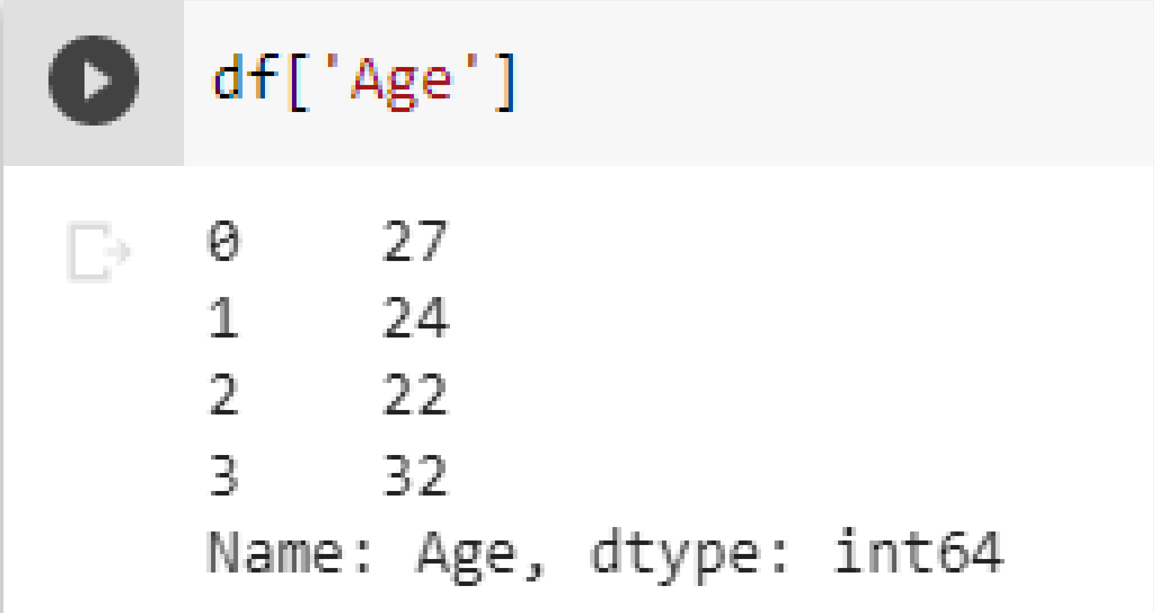
```
import pandas as pd
data = {'Name':['Jai', 'Princi', 'Gaurav', 'Anuj'],
        'Age':[27, 24, 22, 32],
        'Address':['Delhi', 'Kanpur', 'Allahabad', 'Kannauj'],
        'Qualification':['Msc', 'MA', 'MCA', 'Phd']}
```

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

```
     Name  Age    Address  Qualification
0     Jai   27      Delhi            Msc
1  Princi   24     Kanpur             MA
2  Gaurav   22  Allahabad            MCA
3    Anuj   32    Kannauj            Phd
```

INDEXING A DATAFRAME USING INDEXING OPERATOR []

Indexing operator is used to refer to the square brackets following an object. In this indexing operator to refer to df[].



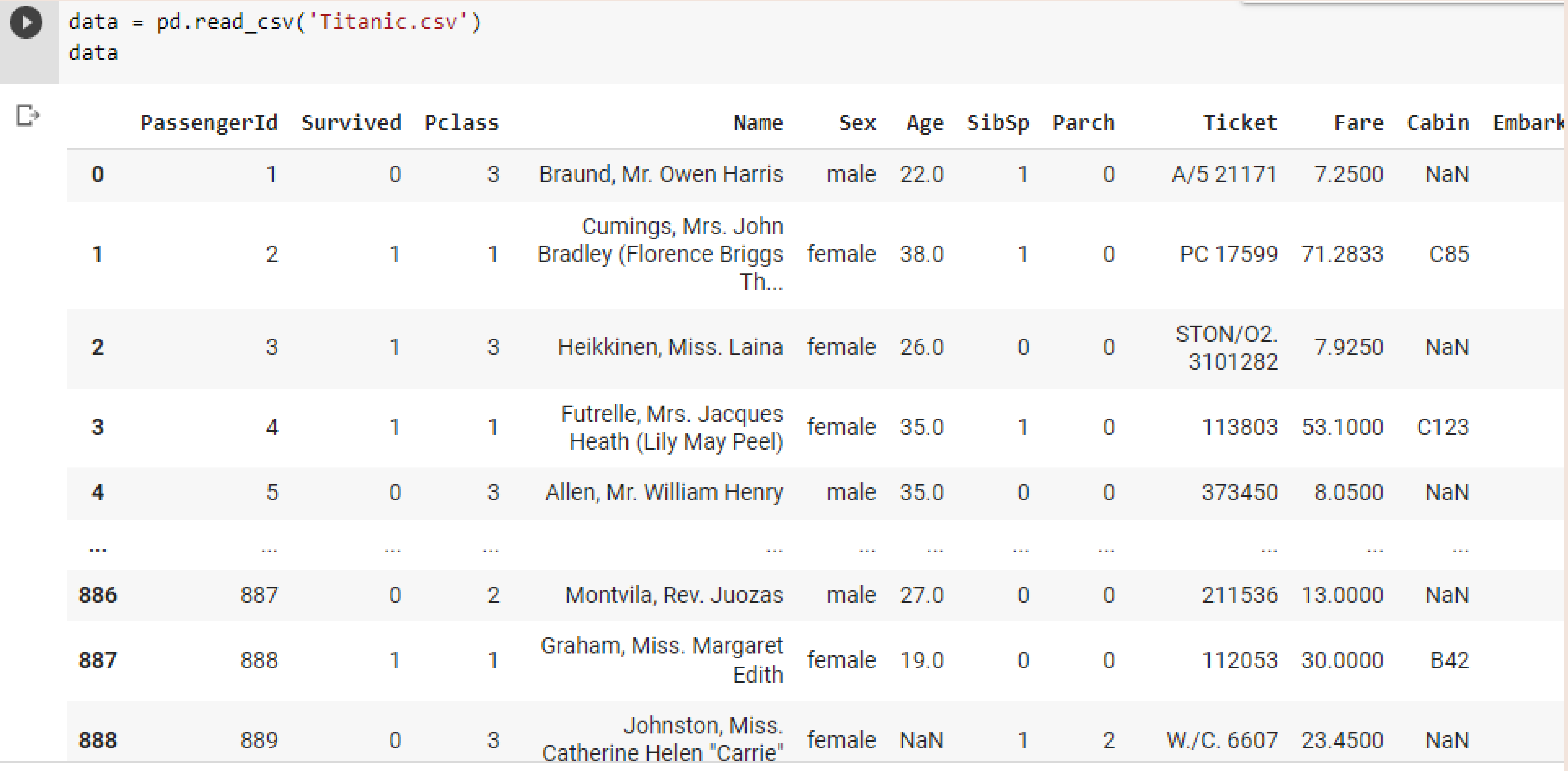
```
df['Age']
```

```
0    27
1    24
2    22
3    32
Name: Age, dtype: int64
```

PANDAS IMPORT CSV FILE

The pandas function read_csv() reads in values, where the delimiter is a comma character.
You can export a file into a csv file in any modern office suite including Google Sheets.

EXAMPLE OF IMPORT CSV FILE



```
data = pd.read_csv('Titanic.csv')
data
```

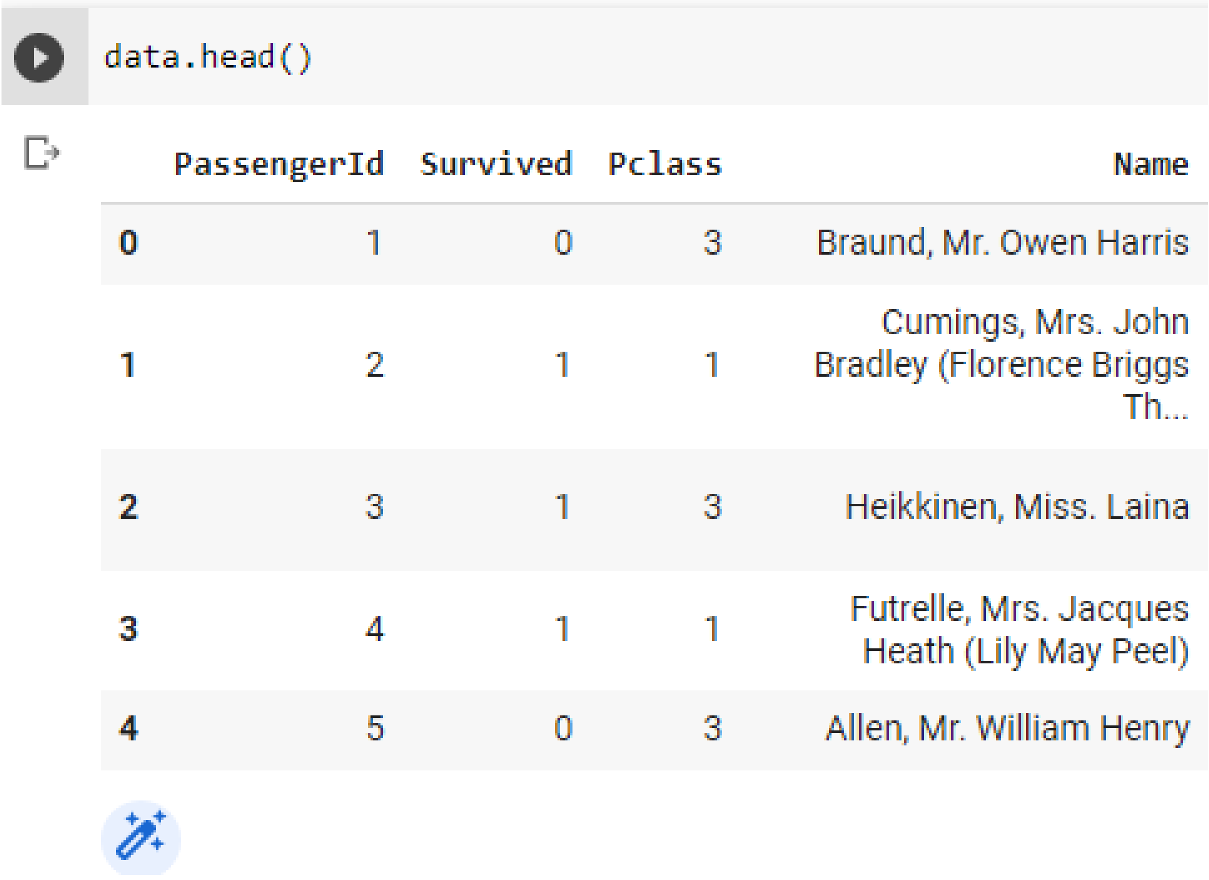
```
     PassengerId  Survived  Pclass                          Name     Sex   Age  SibSp  Parch           Ticket     Fare  Cabin  Embark
  0            1         0       3       Braund, Mr. Owen Harris    male  22.0      1      0        A/5 21171   7.2500    NaN
                                              Cumings, Mrs. John
  1            2         1       1   Bradley (Florence Briggs      female  38.0      1      0         PC 17599  71.2833    C85
                                                             Th...
                                                                                                   STON/O2.
  2            3         1       3        Heikkinen, Miss. Laina  female  26.0      0      0         3101282   7.9250    NaN
                                         Futrelle, Mrs. Jacques
  3            4         1       1        Heath (Lily May Peel)  female  35.0      1      0           113803  53.1000   C123
  4            5         0       3      Allen, Mr. William Henry    male  35.0      0      0           373450   8.0500    NaN
...          ...       ...     ...                           ...     ...   ...    ...    ...              ...      ...    ...
886          887         0       2         Montvila, Rev. Juozas    male  27.0      0      0           211536  13.0000    NaN
                                        Graham, Miss. Margaret
887          888         1       1                         Edith  female  19.0      0      0           112053  30.0000    B42
                                              Johnston, Miss.
888          889         0       3  Catherine Helen "Carrie"      female   NaN      1      2       W./C. 6607  23.4500    NaN
```

HEAD() FUNCTION

DEFINITION

This function returns the first n rows for the object based on position. It is useful for quickly testing if your object has the right type of data in it.

EXAMPLE



```
data.head()
```

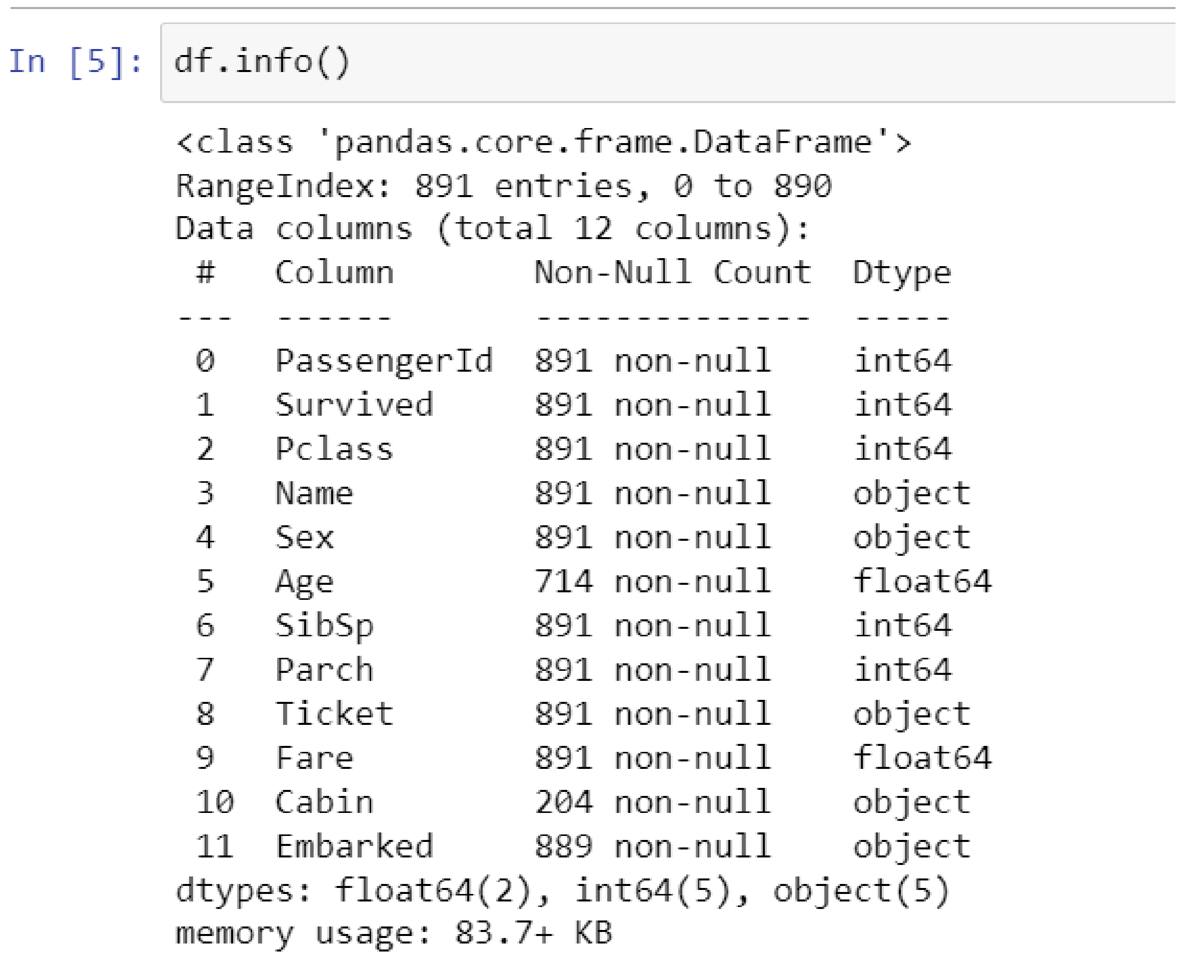
	PassengerId	Survived	Pclass	Name
0	1	0	3	Braund, Mr. Owen Harris
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...
2	3	1	3	Heikkinen, Miss. Laina
3	4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)
4	5	0	3	Allen, Mr. William Henry

info() FUNCTION

DEFINITION

This function can see the column name, the number of Non-Null values and the type of data in each column in the 'Titanic' data.

EXAMPLE



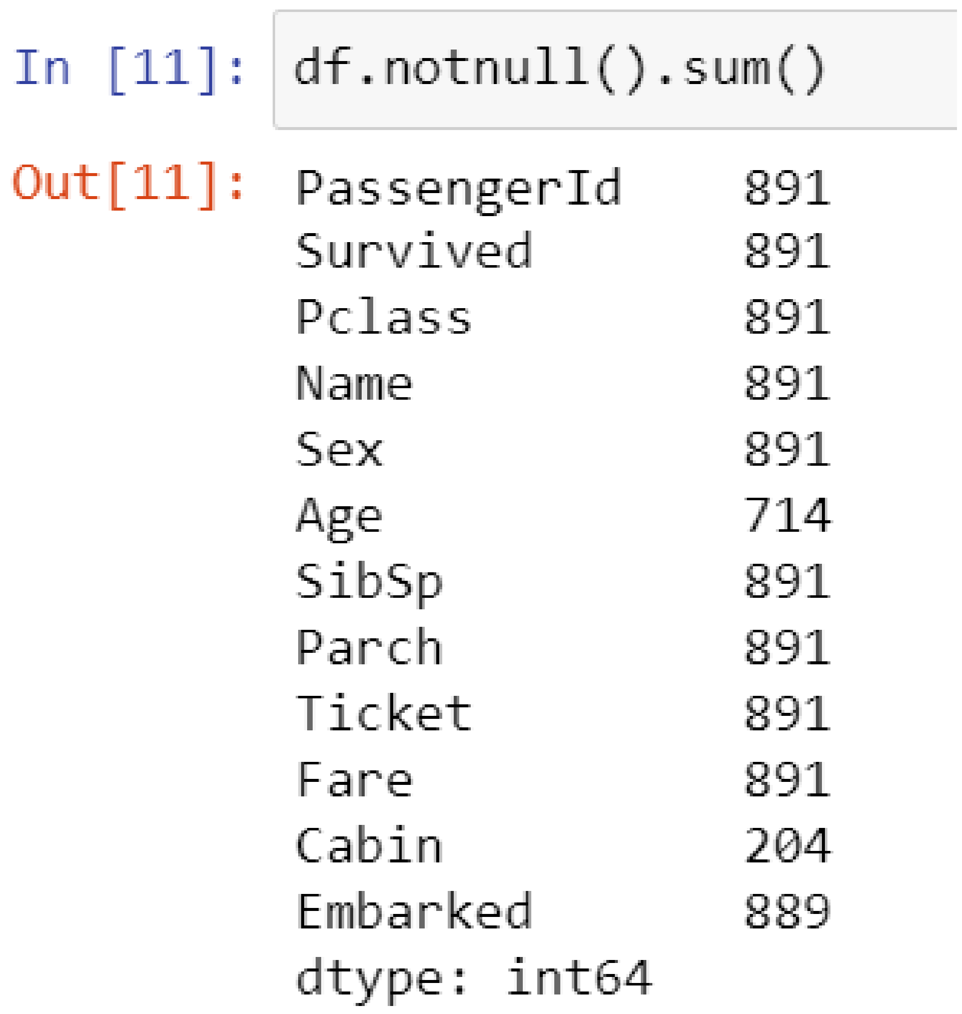
```
In [5]: df.info()

<class 'pandas.core.frame.DataFrame'>
RangeIndex: 891 entries, 0 to 890
Data columns (total 12 columns):
 #   Column       Non-Null Count  Dtype  
---  ------       --------------  -----  
 0   PassengerId  891 non-null    int64  
 1   Survived     891 non-null    int64  
 2   Pclass       891 non-null    int64  
 3   Name         891 non-null    object 
 4   Sex          891 non-null    object 
 5   Age          714 non-null    float64
 6   SibSp        891 non-null    int64  
 7   Parch        891 non-null    int64  
 8   Ticket       891 non-null    object 
 9   Fare         891 non-null    float64
 10  Cabin        204 non-null    object 
 11  Embarked     889 non-null    object 
dtypes: float64(2), int64(5), object(5)
memory usage: 83.7+ KB
```

NOTNULL().SUM() FUNCTION

DEFINITION

This function to see the amount of data filled in from each column.

EXAMPLE



```
In [11]: df.notnull().sum()
```

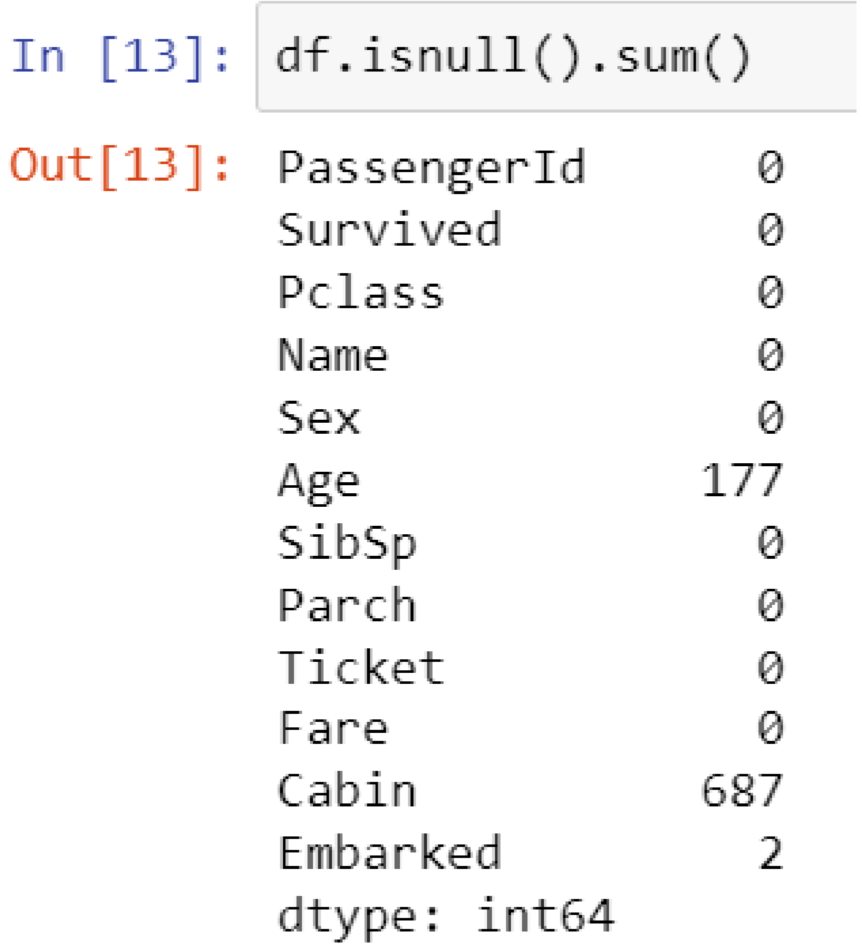
```
Out[11]: PassengerId    891
         Survived       891
         Pclass         891
         Name           891
         Sex            891
         Age            714
         SibSp          891
         Parch          891
         Ticket         891
         Fare           891
         Cabin          204
         Embarked       889
         dtype: int64
```

ISNULL().SUM() FUNCTION

DEFINITION

This function to see the empty value of each Titanic data column.

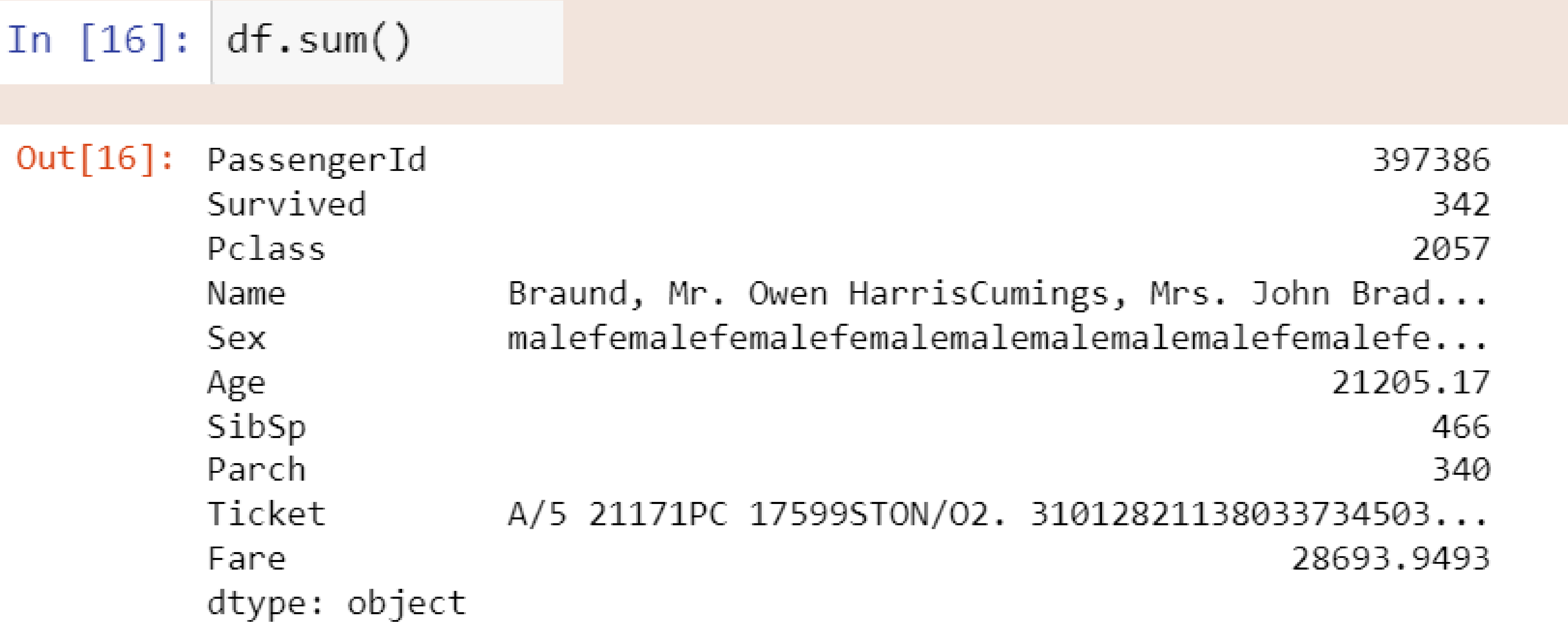
EXAMPLE



```
In [13]:  df.isnull().sum()

Out[13]:  PassengerId      0
          Survived         0
          Pclass           0
          Name             0
          Sex              0
          Age            177
          SibSp            0
          Parch            0
          Ticket           0
          Fare             0
          Cabin          687
          Embarked         2
          dtype: int64
```

Sum() Function

This function counts up all the data in each column.

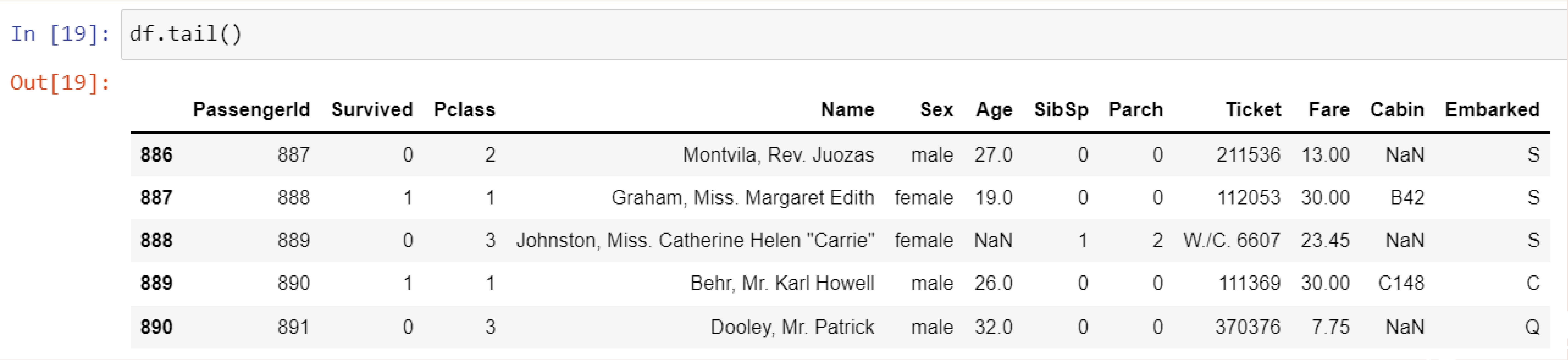


```
In [16]: df.sum()
```

```
Out[16]: PassengerId                                               397386
         Survived                                                     342
         Pclass                                                      2057
         Name           Braund, Mr. Owen HarrisCumings, Mrs. John Brad...
         Sex            malefemalefemalefemalemalemalemalemalefemalefe...
         Age                                                     21205.17
         SibSp                                                        466
         Parch                                                        340
         Ticket         A/5 21171PC 17599STON/O2. 31012821138033734503...
         Fare                                                  28693.9493
         dtype: object
```

TAIL() FUNCTION

This function to see the entire contents of the Titanic data starts from the bottom to the top.



```
In [19]: df.tail()
```

Out[19]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
886	887	0	2	Montvila, Rev. Juozas	male	27.0	0	0	211536	13.00	NaN	S
887	888	1	1	Graham, Miss. Margaret Edith	female	19.0	0	0	112053	30.00	B42	S
888	889	0	3	Johnston, Miss. Catherine Helen "Carrie"	female	NaN	1	2	W./C. 6607	23.45	NaN	S
889	890	1	1	Behr, Mr. Karl Howell	male	26.0	0	0	111369	30.00	C148	C
890	891	0	3	Dooley, Mr. Patrick	male	32.0	0	0	370376	7.75	NaN	Q

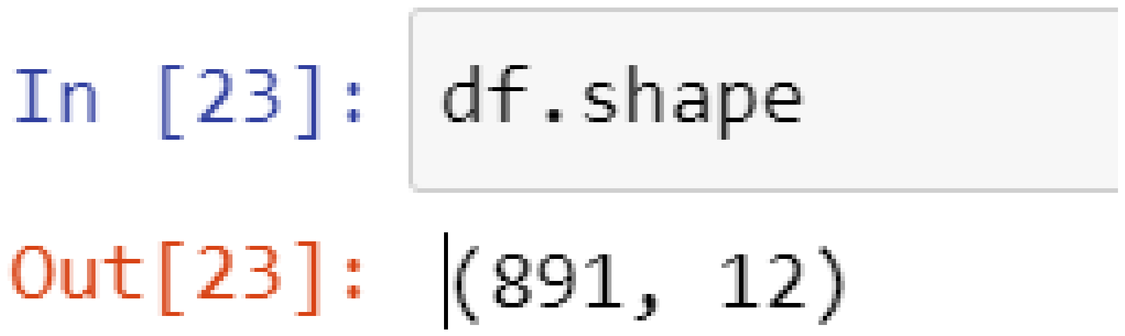
SHAPE() FUNCTION

DEFINITION

Shape function to see the count of rows and columns in the data.

(row, column)

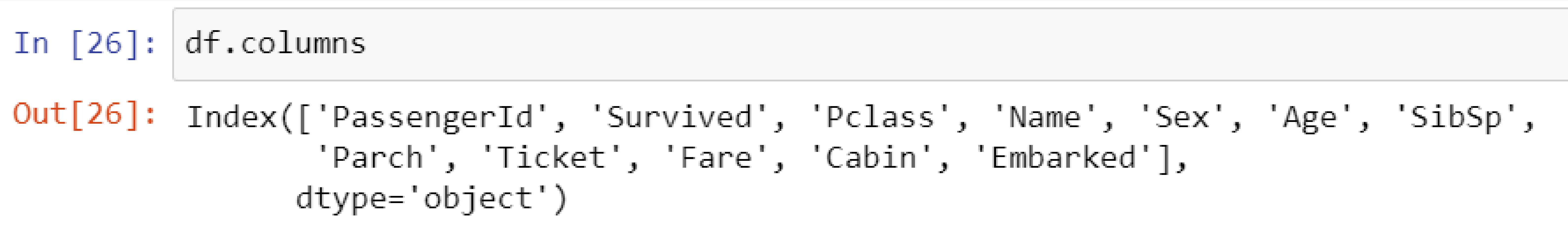
EXAMPLE



```
In [23]: df.shape

Out[23]: (891, 12)
```

COLUMN() FUNCTION

Column function to see the column names in the data.



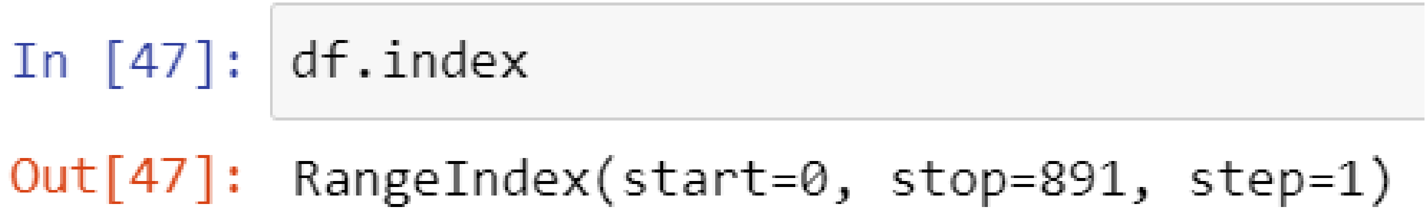
```
In [26]: df.columns

Out[26]: Index(['PassengerId', 'Survived', 'Pclass', 'Name', 'Sex', 'Age', 'SibSp',
                'Parch', 'Ticket', 'Fare', 'Cabin', 'Embarked'],
               dtype='object')
```

INDEX() FUNCTION

DEFINITION

Index function to see the count of rows in the data.

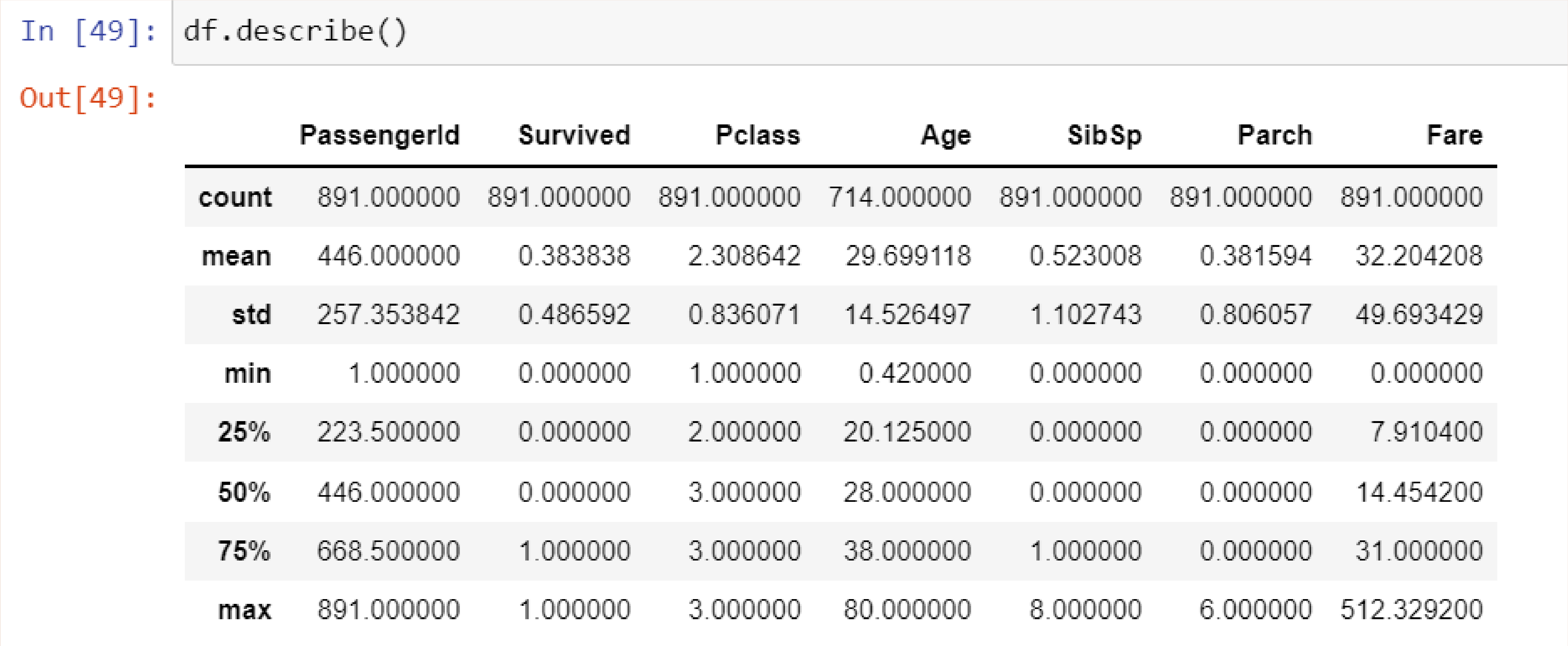
EXAMPLE



```
In [47]: df.index

Out[47]: RangeIndex(start=0, stop=891, step=1)
```

DESCRIBE() FUNCTION

Describe the function to see statistics from each column in the data. Statistics include count, mean, std, min, and max.



```
In [49]: df.describe()
```

Out[49]:

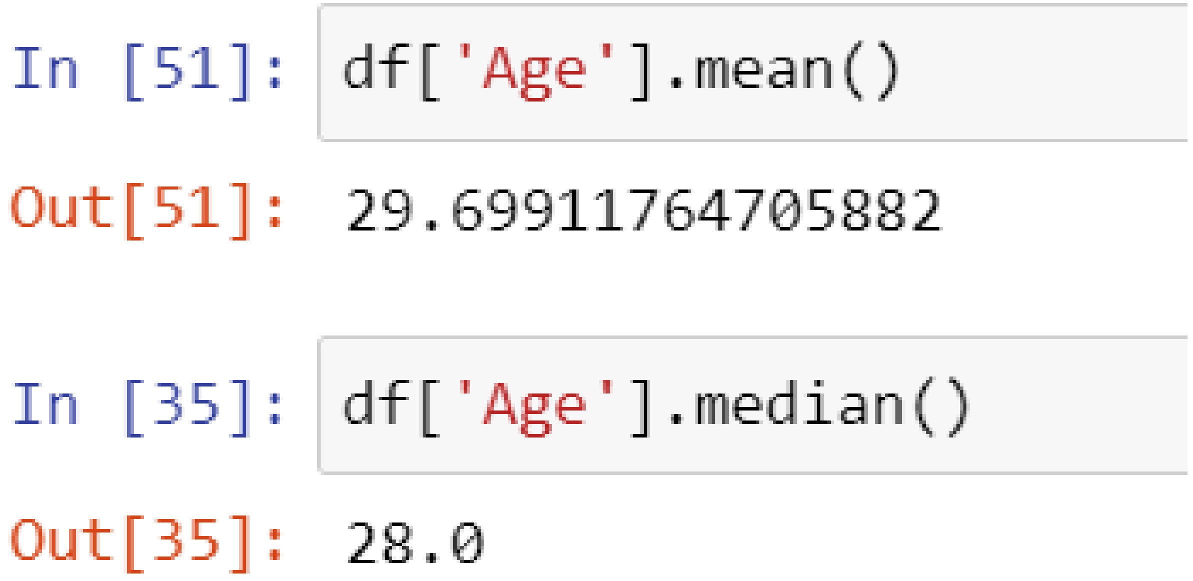
	PassengerId	Survived	Pclass	Age	SibSp	Parch	Fare
count	891.000000	891.000000	891.000000	714.000000	891.000000	891.000000	891.000000
mean	446.000000	0.383838	2.308642	29.699118	0.523008	0.381594	32.204208
std	257.353842	0.486592	0.836071	14.526497	1.102743	0.806057	49.693429
min	1.000000	0.000000	1.000000	0.420000	0.000000	0.000000	0.000000
25%	223.500000	0.000000	2.000000	20.125000	0.000000	0.000000	7.910400
50%	446.000000	0.000000	3.000000	28.000000	0.000000	0.000000	14.454200
75%	668.500000	1.000000	3.000000	38.000000	1.000000	0.000000	31.000000
max	891.000000	1.000000	3.000000	80.000000	8.000000	6.000000	512.329200

MEAN() AND MEDIAN() FUNCTION

DEFINITION

Mean and median functions to see the average and central value of data.
We want to see the mean and median values of the age column.

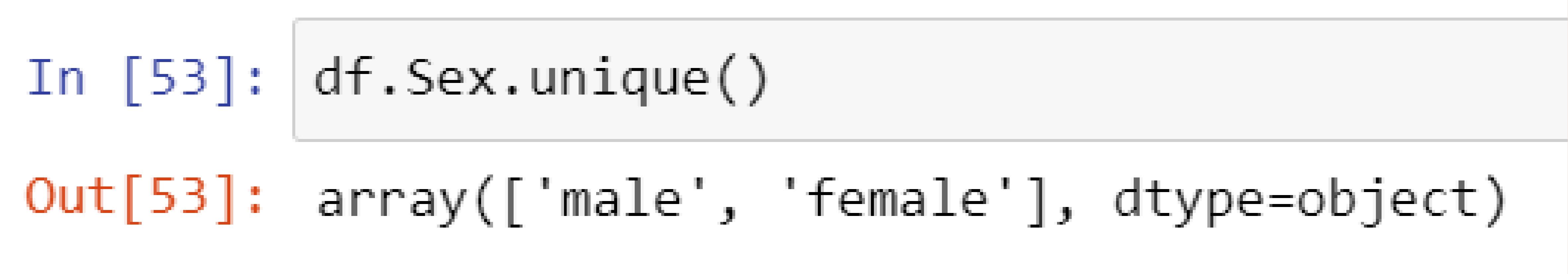
EXAMPLE



```
In [51]:  df['Age'].mean()

Out[51]:  29.69911764705882

In [35]:  df['Age'].median()

Out[35]:  28.0
```

UNIQUE() FUNCTION

Unique function to see the unique value (different value) of data.
We want to see type of sex in data 'Titanic'



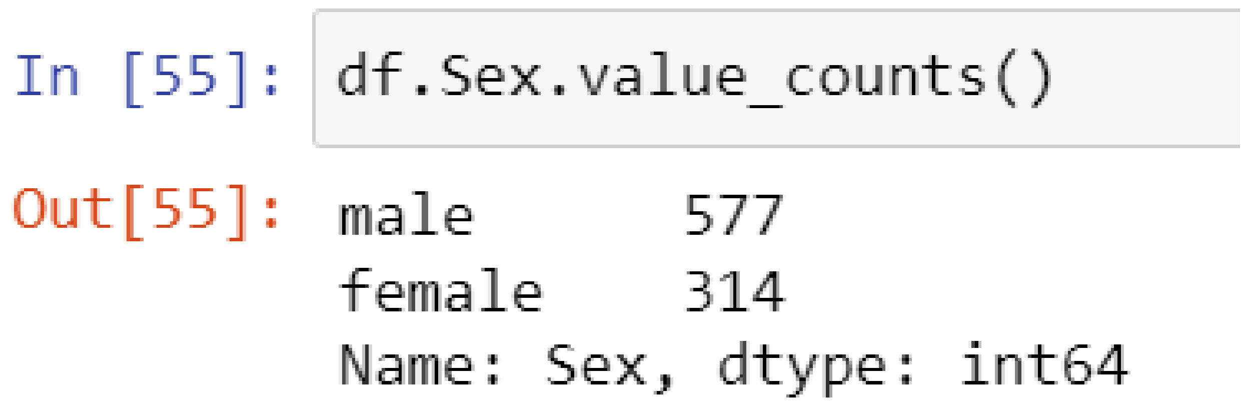
```
In [53]: df.Sex.unique()

Out[53]: array(['male', 'female'], dtype=object)
```

VALUE_COUNTS() FUNCTION

DEFINITION

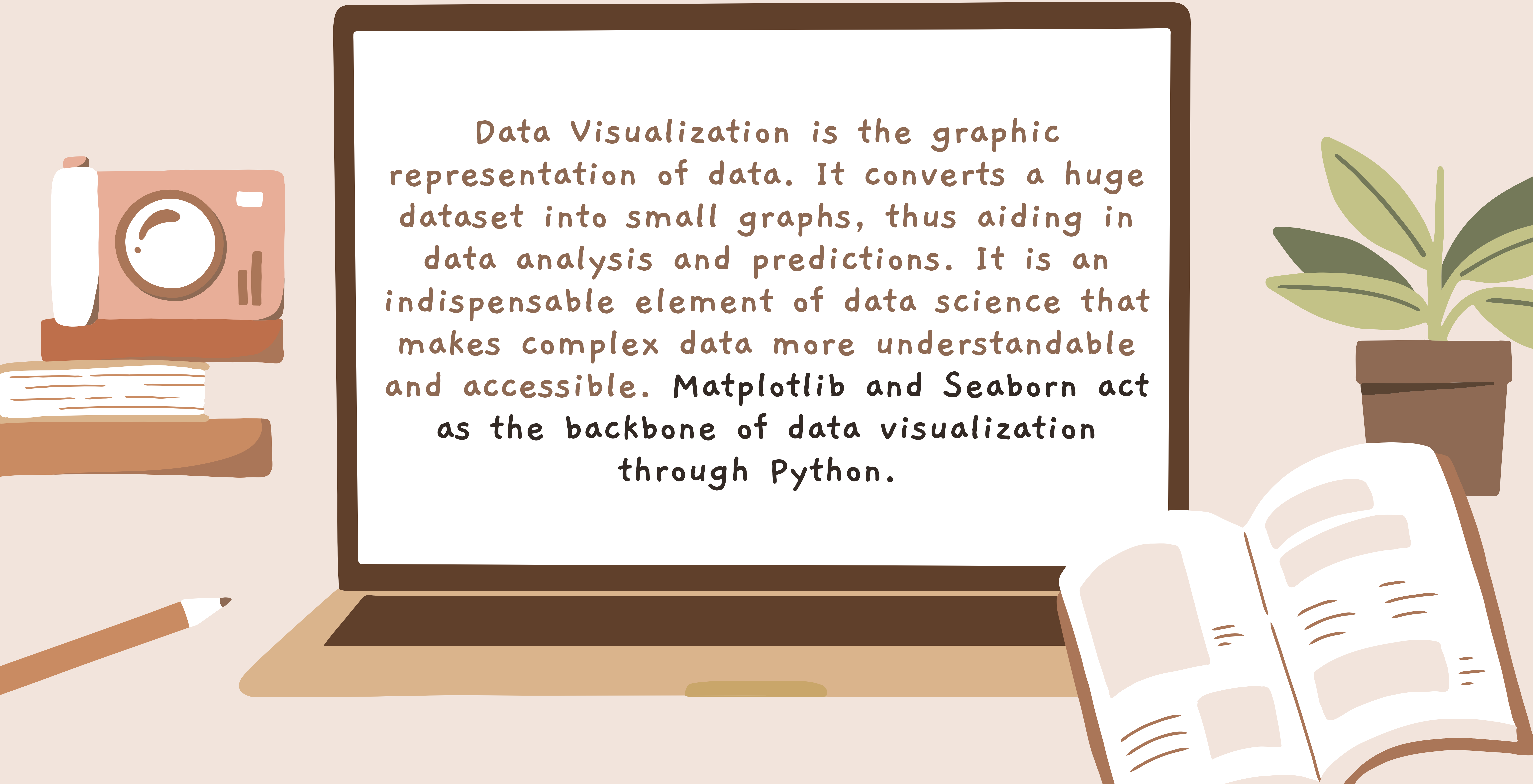
This function to see the amount of each unique value in the data.
We want to see the amount of type sex in the data 'Titanic'.

EXAMPLE

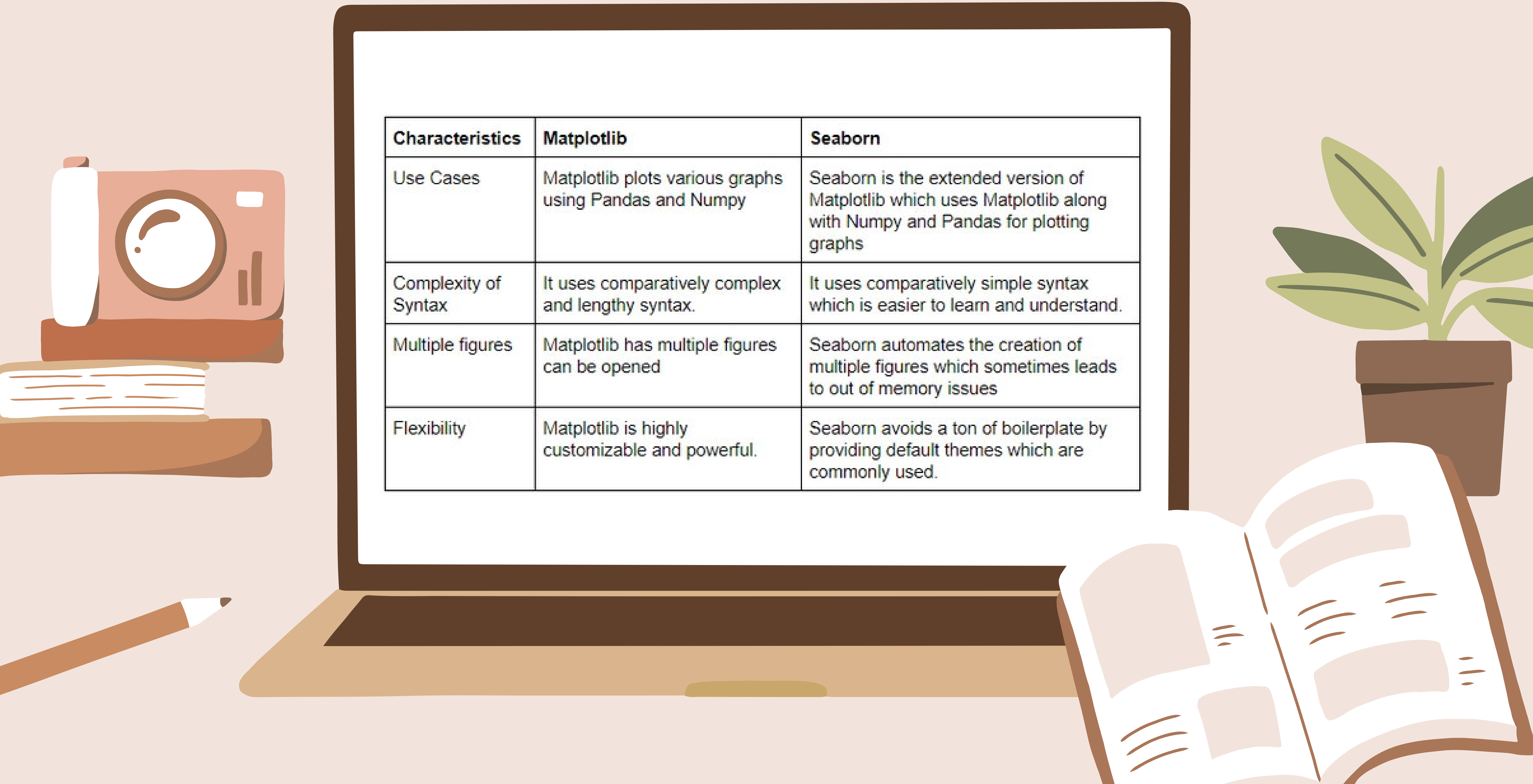


```
In [55]:  df.Sex.value_counts()

Out[55]:  male      577
          female    314
          Name: Sex, dtype: int64
```

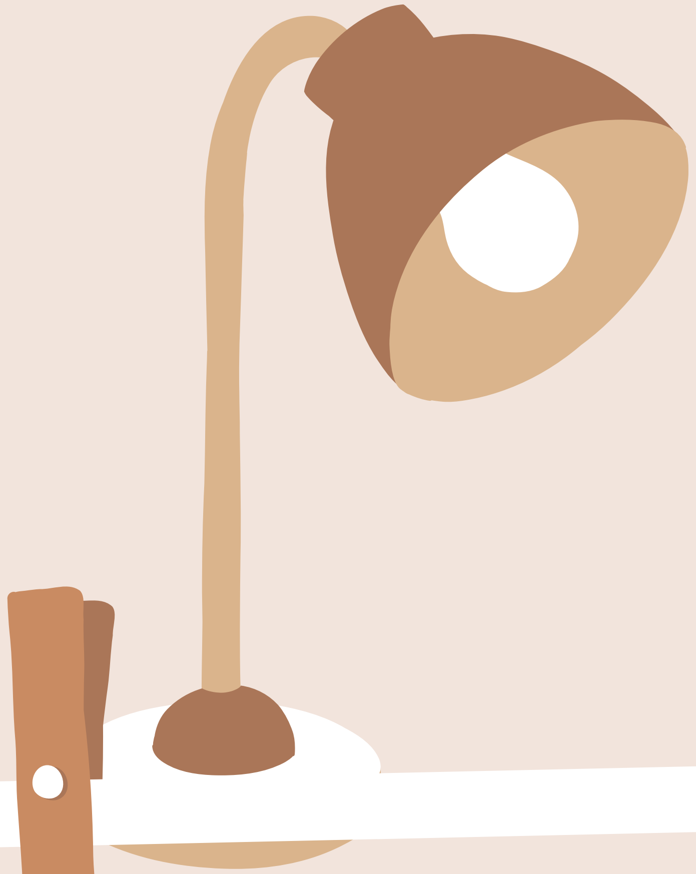
matplotlib AND seaborn



Data Visualization is the graphic representation of data. It converts a huge dataset into small graphs, thus aiding in data analysis and predictions. It is an indispensable element of data science that makes complex data more understandable and accessible. Matplotlib and Seaborn act as the backbone of data visualization through Python.



Characteristics	Matplotlib	Seaborn
Use Cases	Matplotlib plots various graphs using Pandas and Numpy	Seaborn is the extended version of Matplotlib which uses Matplotlib along with Numpy and Pandas for plotting graphs
Complexity of Syntax	It uses comparatively complex and lengthy syntax.	It uses comparatively simple syntax which is easier to learn and understand.
Multiple figures	Matplotlib has multiple figures can be opened	Seaborn automates the creation of multiple figures which sometimes leads to out of memory issues
Flexibility	Matplotlib is highly customizable and powerful.	Seaborn avoids a ton of boilerplate by providing default themes which are commonly used.



MATPLOTLIB

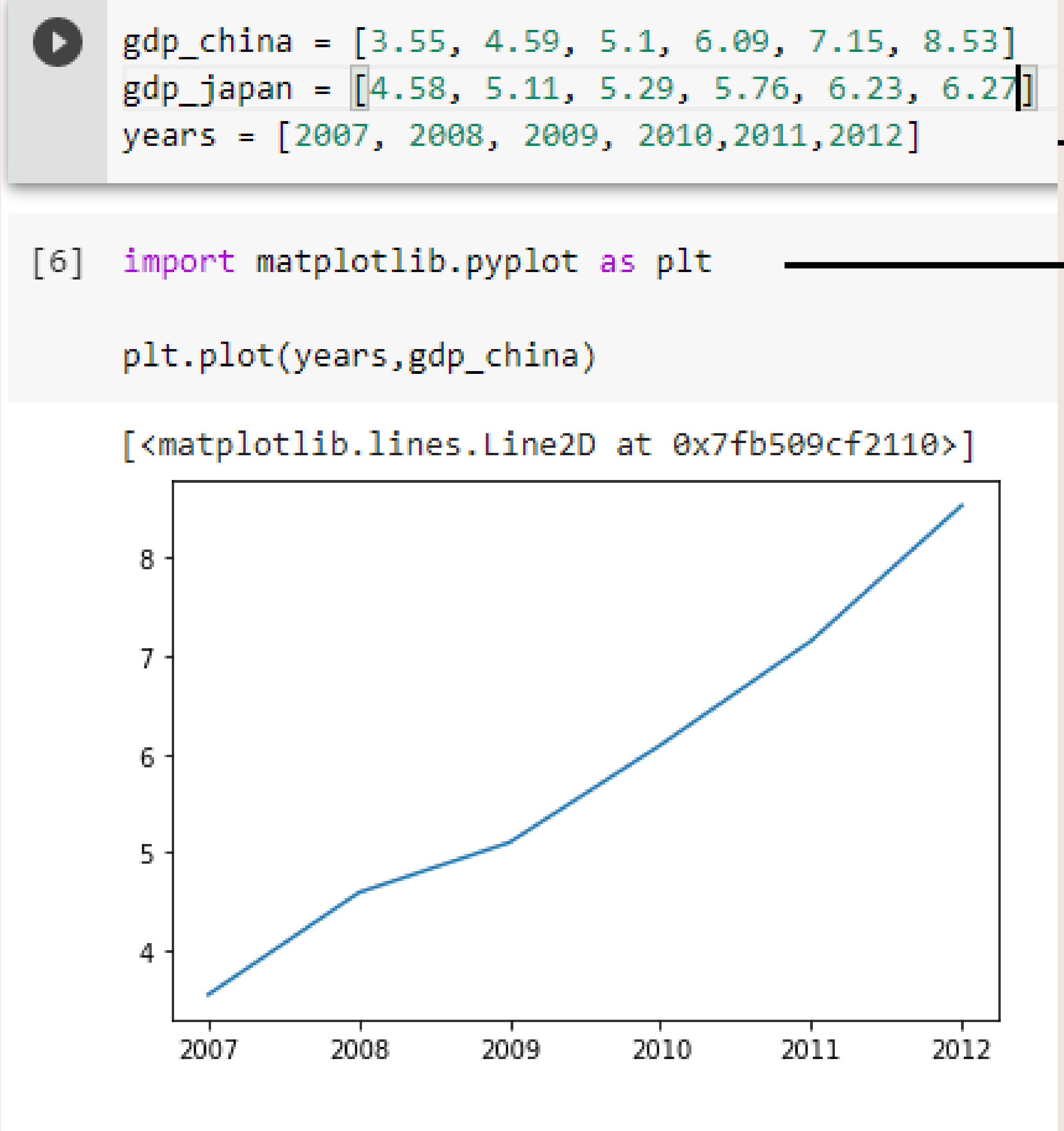
DEFINITION

Matplotlib is a comprehensive library for creating static, animated, and interactive visualizations in Python. Matplotlib makes easy things easy and hard things possible.

ADVANTAGE

- Customize visual style and layout.
- Export to many file formats.
- Embed in JupyterLab and Graphical User Interfaces.

EXAMPLE LINE CHART OF MATPLOTLIB



```
gdp_china = [3.55, 4.59, 5.1, 6.09, 7.15, 8.53]
gdp_japan = [4.58, 5.11, 5.29, 5.76, 6.23, 6.27]
years = [2007, 2008, 2009, 2010,2011,2012]
```

```
[6] import matplotlib.pyplot as plt

    plt.plot(years,gdp_china)
```

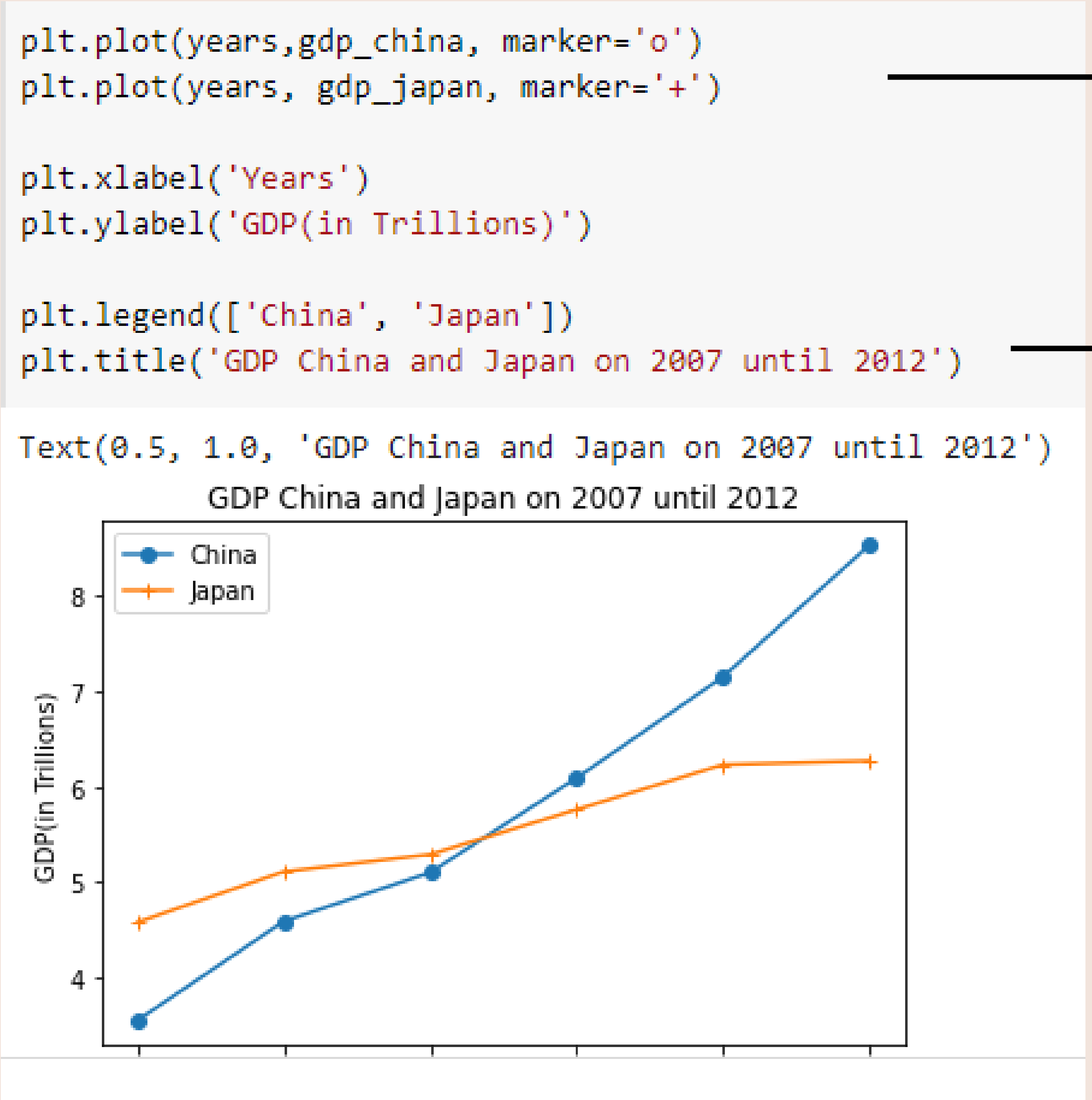
```
[<matplotlib.lines.Line2D at 0x7fb509cf2110>]
```

Data to visualize

Matplotlib module to make visualize data

EXAMPLE LINE CHART OF MATPLOTLIB

Make label for x and y axis ←



```
plt.plot(years,gdp_china, marker='o')
plt.plot(years, gdp_japan, marker='+')

plt.xlabel('Years')
plt.ylabel('GDP(in Trillions)')

plt.legend(['China', 'Japan'])
plt.title('GDP China and Japan on 2007 until 2012')
```

```
Text(0.5, 1.0, 'GDP China and Japan on 2007 until 2012')
```

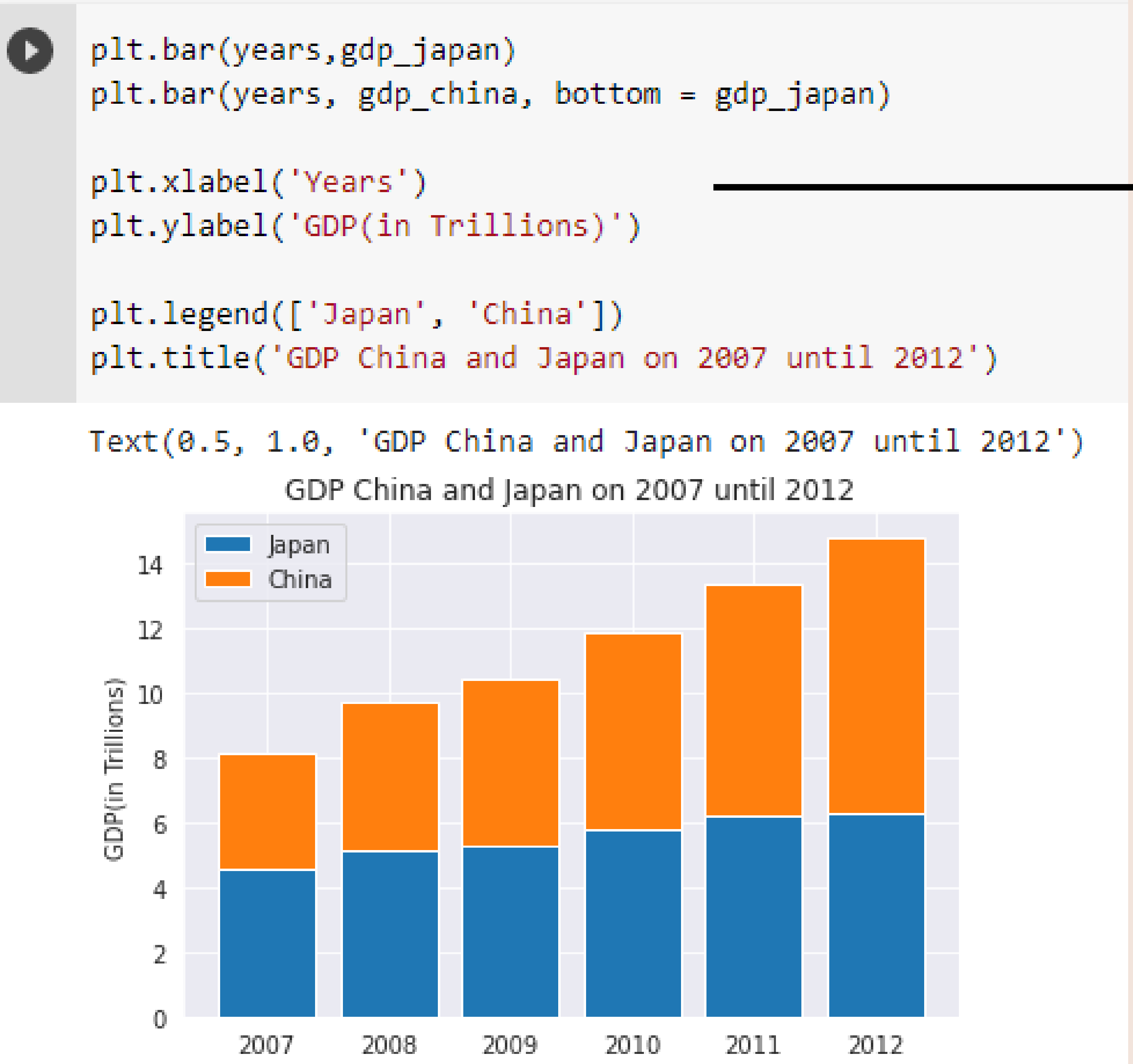
→ Plot multi data on Figure

→ Legend and title to explain the graph

EXAMPLE STACKED BAR CHART OF MATPLOTLIB

Plot multi data on Figure

Legend and title to explain the graph



```
plt.bar(years,gdp_japan)
plt.bar(years, gdp_china, bottom = gdp_japan)

plt.xlabel('Years')
plt.ylabel('GDP(in Trillions)')

plt.legend(['Japan', 'China'])
plt.title('GDP China and Japan on 2007 until 2012')
```

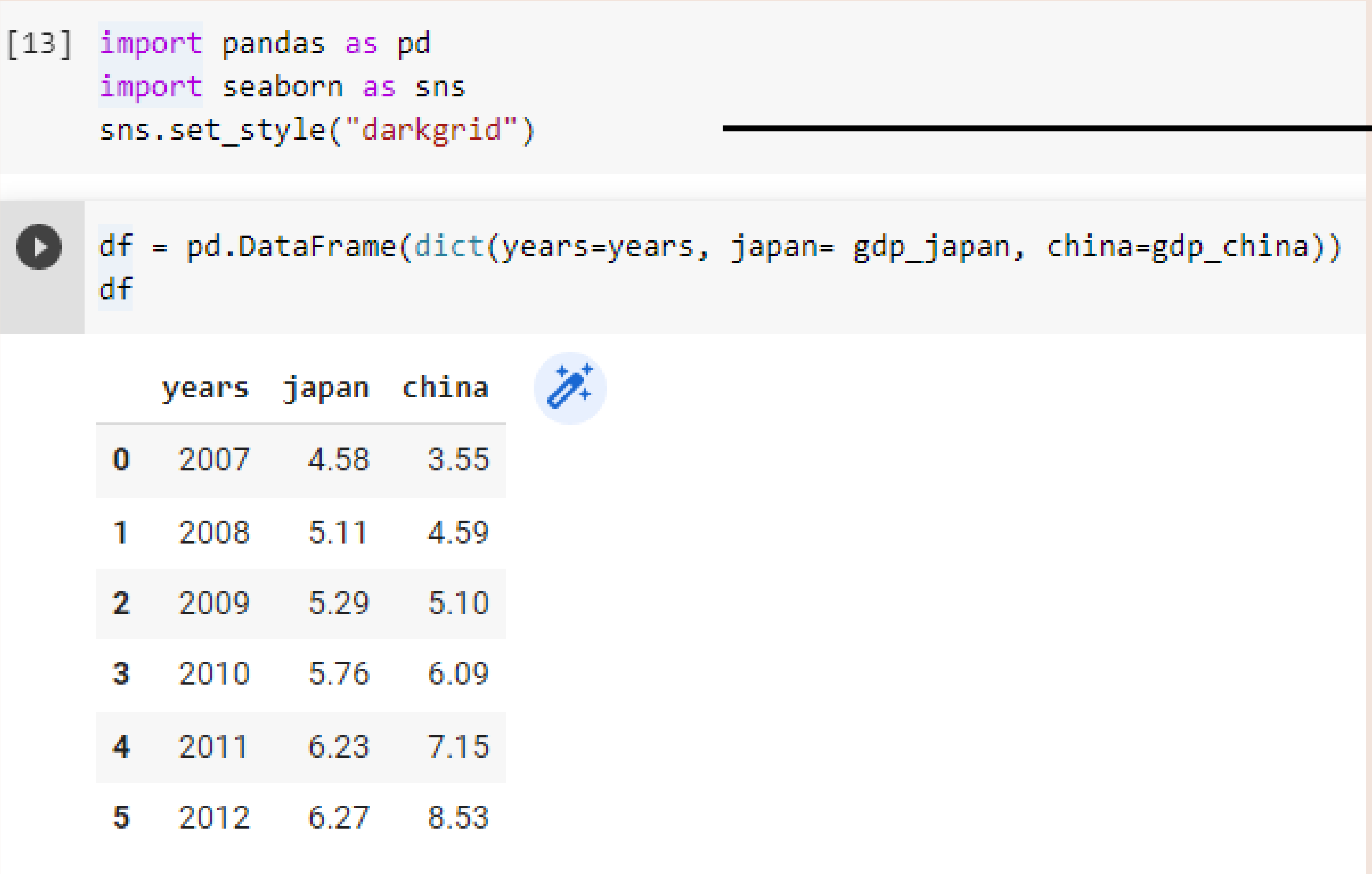
```
Text(0.5, 1.0, 'GDP China and Japan on 2007 until 2012')
```

Make label for x and y axis

SEABORN

Seaborn is a Python data visualization library based on matplotlib. It provides a high-level interface for drawing attractive and informative statistical graphics.

EXAMPLE OF IMPORT SEABORN



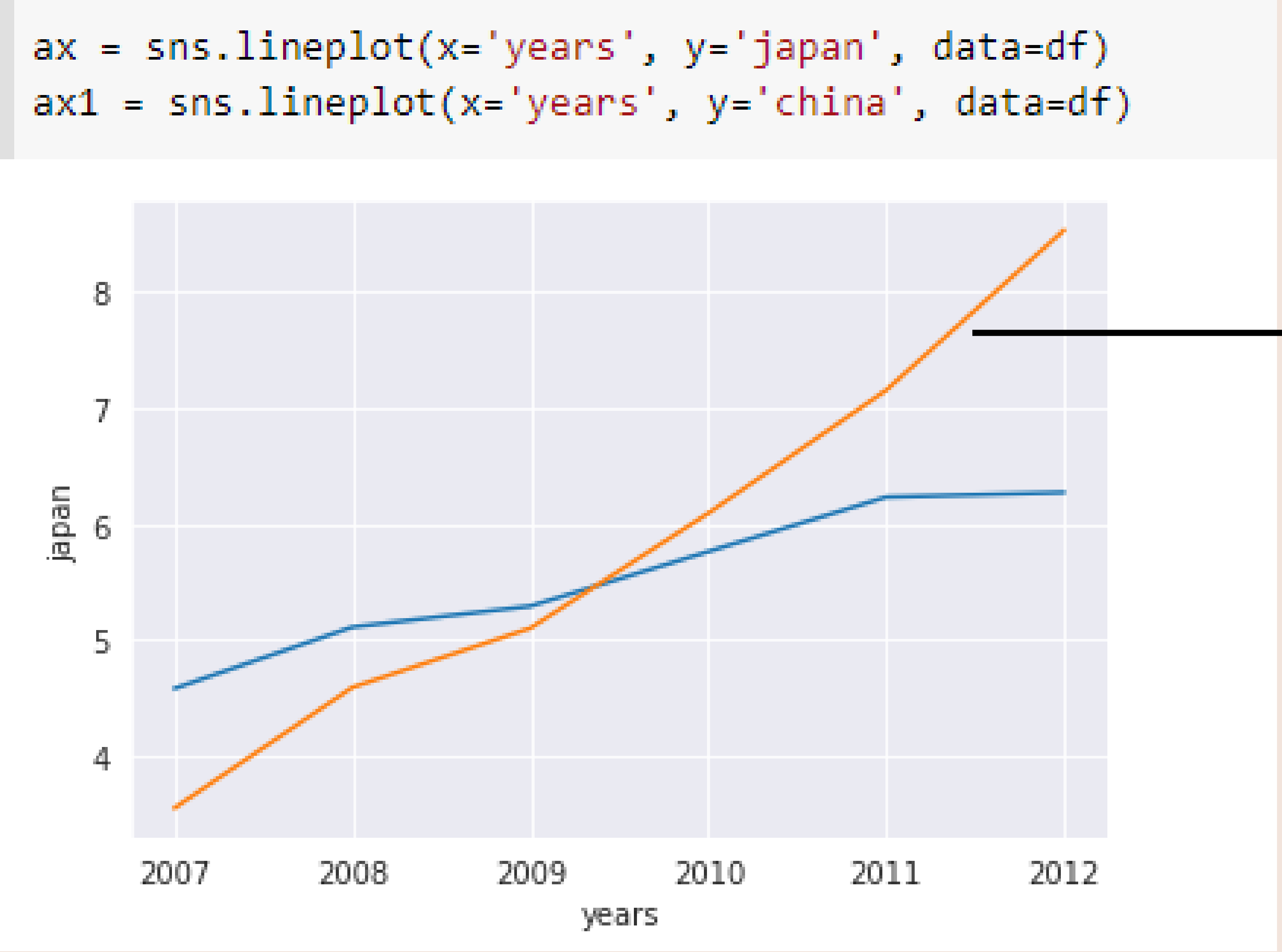
```
[13] import pandas as pd
     import seaborn as sns
     sns.set_style("darkgrid")
```

```
df = pd.DataFrame(dict(years=years, japan= gdp_japan, china=gdp_china))
df
```

	years	japan	china
0	2007	4.58	3.55
1	2008	5.11	4.59
2	2009	5.29	5.10
3	2010	5.76	6.09
4	2011	6.23	7.15
5	2012	6.27	8.53

Set grid style on graph to be dark mode

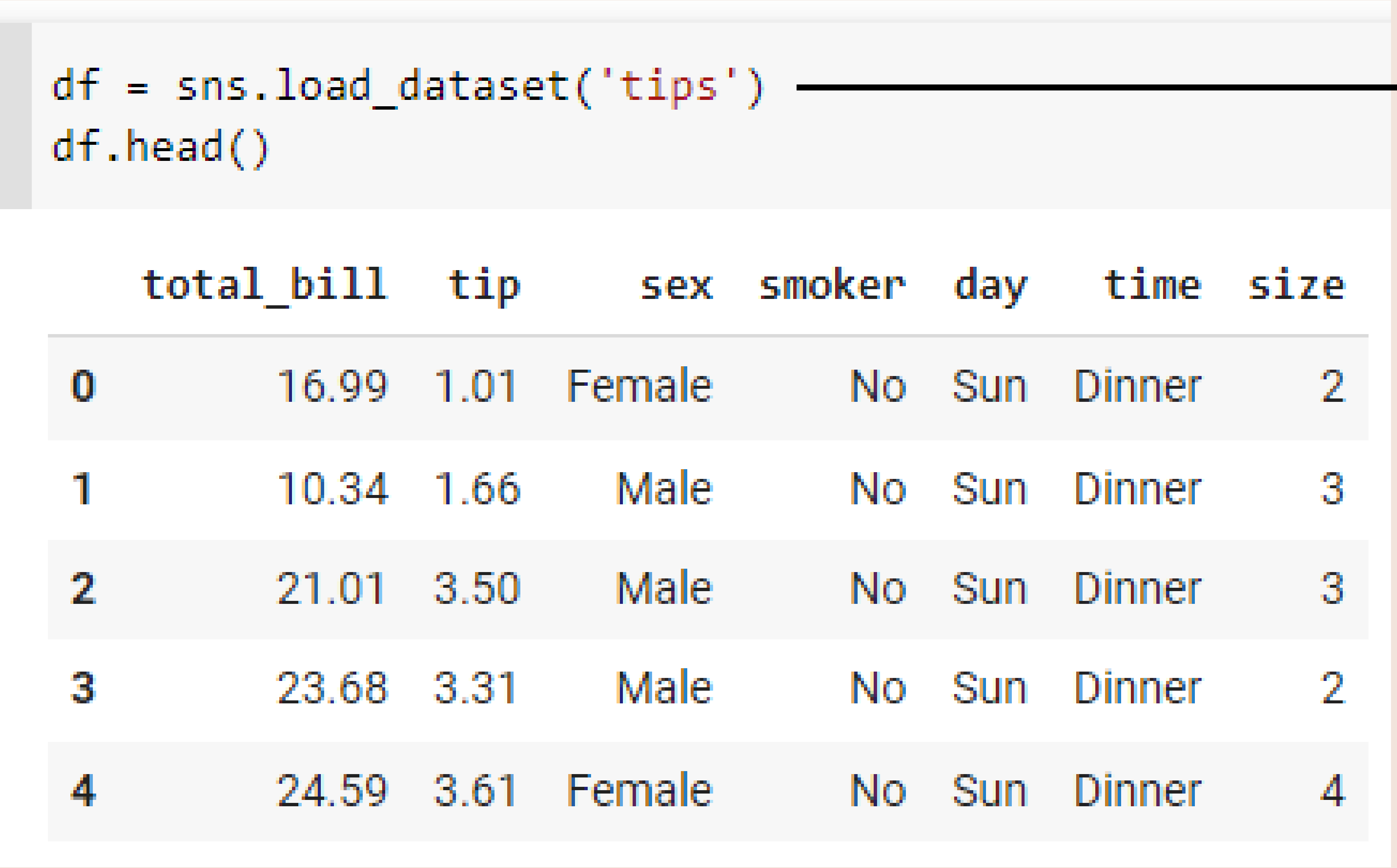
EXAMPLE LINE PLOT OF SEABORN



```
ax = sns.lineplot(x='years', y='japan', data=df)
ax1 = sns.lineplot(x='years', y='china', data=df)
```

Dark style grid because we set it before

EXAMPLE LOAD DEFAULT DATASET OF SEABORN

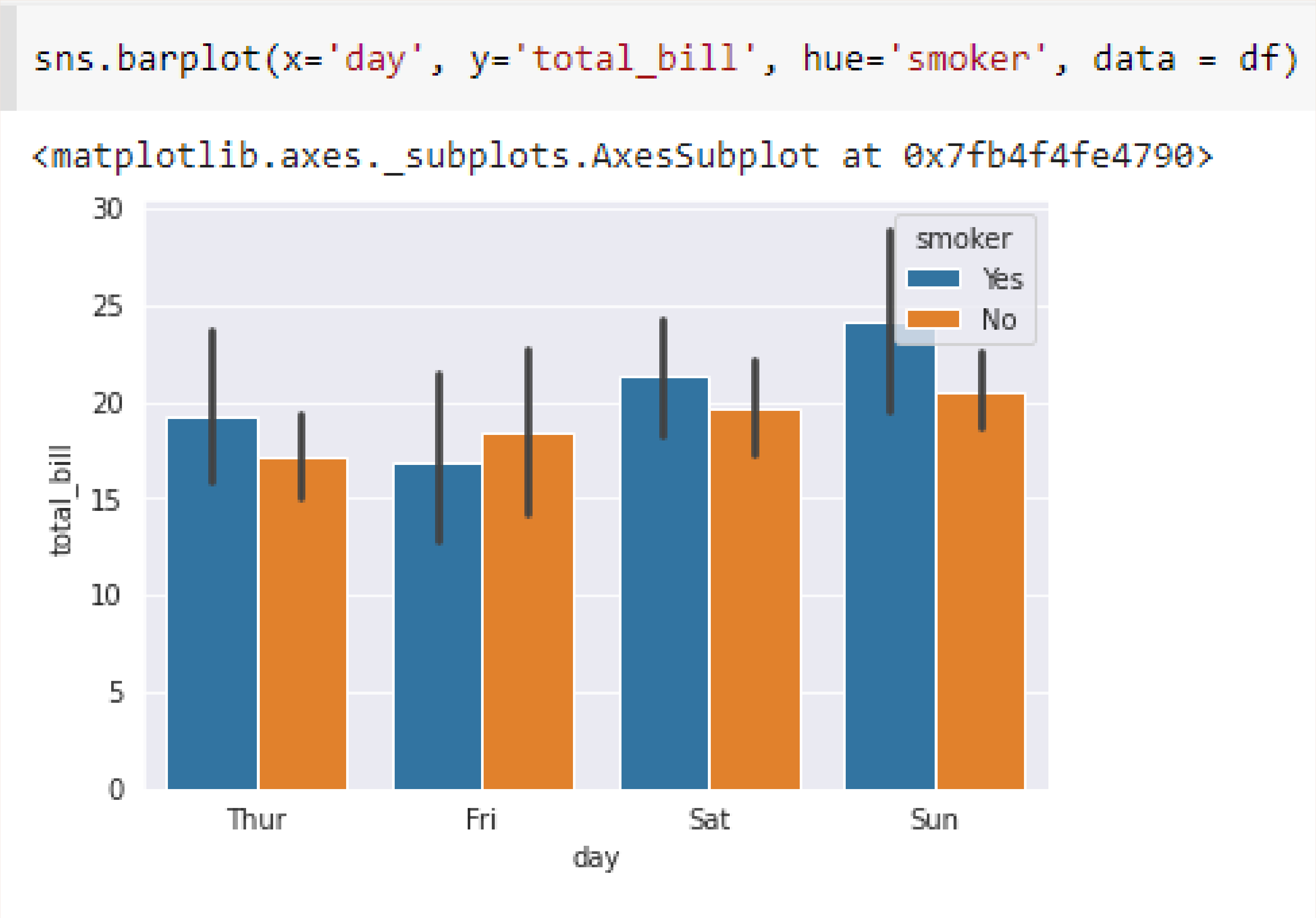


```python
df = sns.load_dataset('tips')
df.head()
```

	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

Tips are dataset that has been provided by the developer

EXAMPLE BAR CHART ON SEABORN



```
sns.barplot(x='day', y='total_bill', hue='smoker', data = df)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x7fb4f4fe4790>
```

→ Differentiated according to smoking or not

https://github.com/agung15-debug/python-week-4

THANK YOU