SINGAPORE UNIVERSITY
OF SOCIAL SCIENCES

Aug 2024

Data Types and Structure Seminar 3

ICT233 Data Programming

Veronica Hu huhe001@suss.edu.sg

RECAP

S2 Key Learning Objectives

1) Appreciate the concept of databases and how they are used for storage, filtering, and extraction of data.
2) Know the data concept and differences between SQL and NoSQL databases.
3) Understand how to structure database query languages to store and retrieve information.
4) Apply the concept of data modeling and using Object-Relational Mapping to store data objects in a database.
5) Know the basic CRUD (Create, Read, Update, Delete) operations for data management and apply such operations on a database.

SEMINAR OVERVIEW

Data Types and Structure – LEARNING OBJECTIVES

1) Appreciate the features and usage possibilities of the Pandas library as a data analytics package.
2) Understand the basic usage of the Python Pandas library, including loading files, counting data, and determining item structure and types in the data.
3) Learn basic data manipulation using Pandas, such as row and column selection, and itemized or vector operations on Pandas DataFrame.
4) Conduct operations on Pandas DataFrame, including subsetting, slicing, and indexing.
5) Present and visualize data in Pandas DataFrame using charting and plotting libraries like Matplotlib and Seaborn.

ETL Process

Seminar 3

Extract

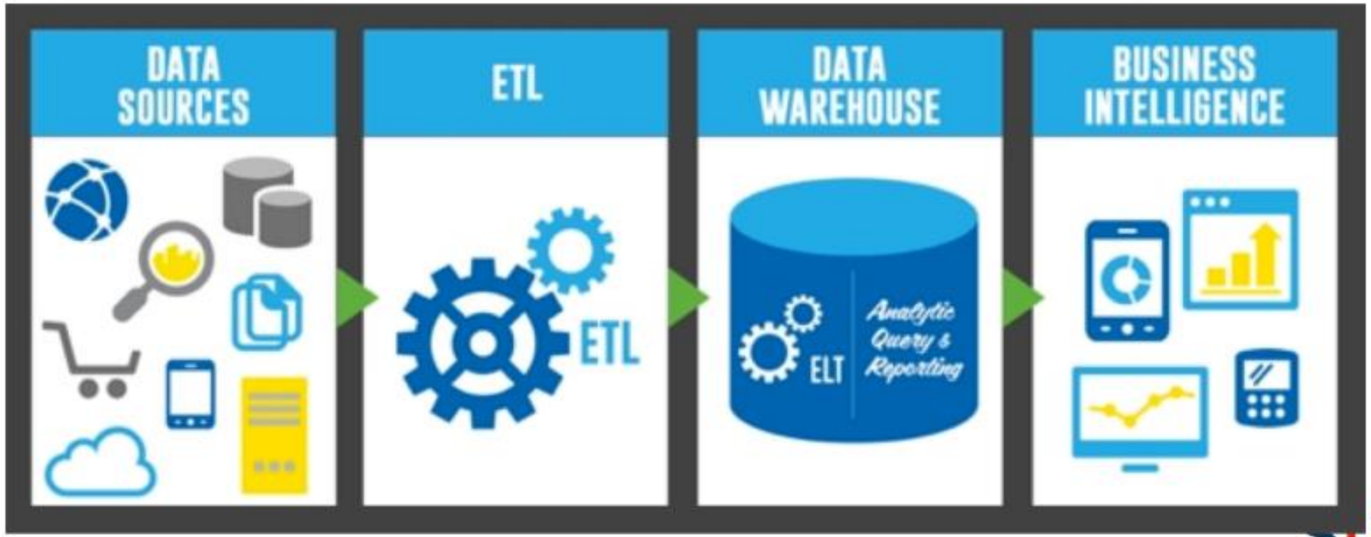
- One or more source systems containing customer, financial, or product data (CRM, Accounting system, Warehouse, MES)
- Files types - Flat files, XML, Oracle, IBM DB2, SQL Server,, IBM Websphere MQ, ODBC, JDBC, Hadoop Distributed File System (HDFS), Hive/HCatalog, JSON, Mainframe (IBM z/OS), Salesforce.com, SAP/R3

Transform

- Applying business rules, cleansing, and validating the data.
- Aggregation, Copy, Join, Sort, Merge, Partition, Filter, Reformat, Lookup
- Mathematical: +, -, x, /, Abs, IsValidNumber, Mod, Pow, Rand, Round, Sqrt, ToNumber, Truncate, Average, Min, Max
- Logical: And, Or, Not, IfThenElse, RegEx, Variables
- Text: Concatenate, CharacterLengthOf, LengthOf, Pad, Replace, ToLower, ToText, ToUpper, Translate, Trim, Hash
- Date: DateAdd, DateDiff, DateLastDay, DatePart, IsValidDate
- Format: ASCII, EBCDIC, Unicode

Load

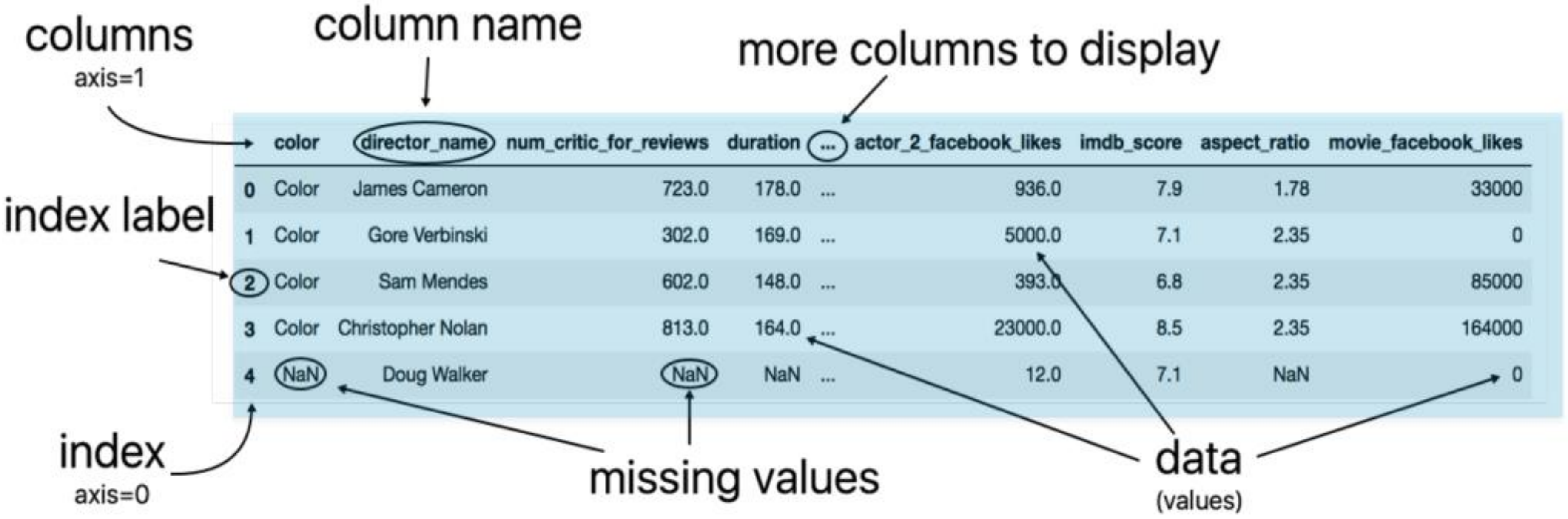
- Load the results into one or more target systems such as a data warehouse, datamart, or business intelligence reporting system.
- Output: Flat files, XML, Oracle, IBM DB2, SQL Server, Teradata, Sybase, Vertica, Netezza, Greenplum, ODBC, JDBC, Hadoop Distributed File System (HDFS), Hive/HCatalog, Mainframe (IBM z/OS), Salesforce.com, Tableau, QlikView



Chapter 1: Pandas Dataframe Basics

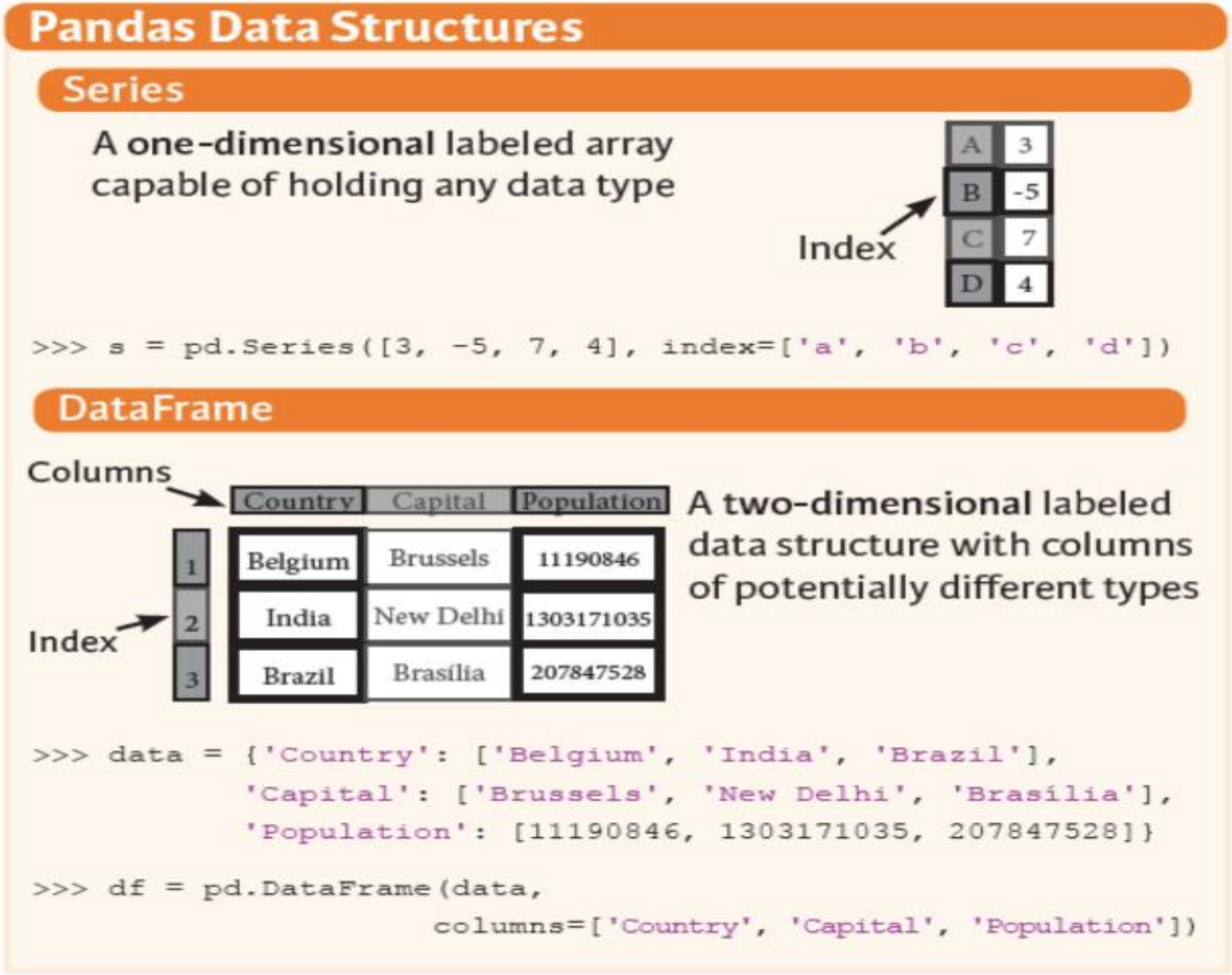
1.1 Introduction

- Pandas – Opensource Python libraries with ”spreadsheet” like functions
- New Data Types
  - Series – ie Single column
  - DataFrame – ie a collection of Series



	color	director_name	num_critic_for_reviews	duration	...	actor_2_facebook_likes	imdb_score	aspect_ratio	movie_facebook_likes
0	Color	James Cameron	723.0	178.0	...	936.0	7.9	1.78	33000
1	Color	Gore Verbinski	302.0	169.0	...	5000.0	7.1	2.35	0
2	Color	Sam Mendes	602.0	148.0	...	393.0	6.8	2.35	85000
3	Color	Christopher Nolan	813.0	164.0	...	23000.0	8.5	2.35	164000
4	NaN	Doug Walker	NaN	NaN	...	12.0	7.1	NaN	0

Chapter 1: Pandas Dataframe Basics



Pandas Data Structures

Series

A one-dimensional labeled array capable of holding any data type

Index

A	3
B	-5
C	7
D	4

```
>>> s = pd.Series([3, -5, 7, 4], index=['a', 'b', 'c', 'd'])
```

DataFrame

Columns

Index

	Country	Capital	Population
1	Belgium	Brussels	11190846
2	India	New Delhi	1303171035
3	Brazil	Brasília	207847528

A two-dimensional labeled data structure with columns of potentially different types

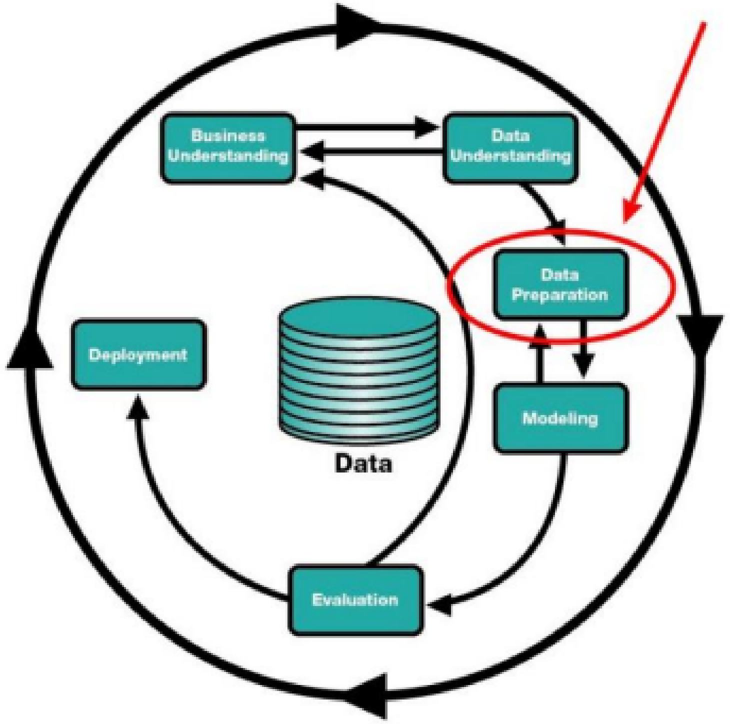
```
>>> data = {'Country': ['Belgium', 'India', 'Brazil'],
            'Capital': ['Brussels', 'New Delhi', 'Brasília'],
            'Population': [11190846, 1303171035, 207847528]}

>>> df = pd.DataFrame(data,
                      columns=['Country', 'Capital', 'Population'])
```

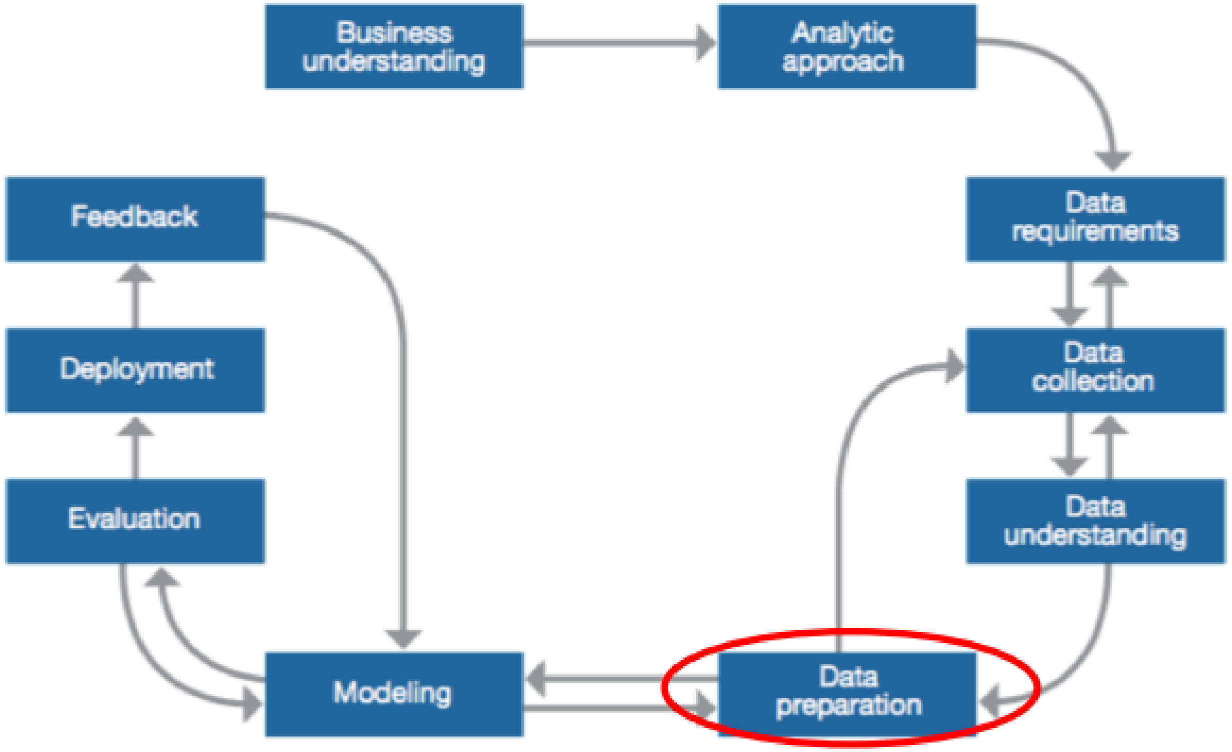
Pandas Cheat Sheet: Data Science and Data Wrangling in Python - KDnuggets

Chapter 1: Pandas Dataframe Basics

Data preparation, Cleansing, Pre-processing, Wrangling

- CRISP-DM model - https://en.wikipedia.org/wiki/Cross-industry_standard_process_for_data_mining
- Foundational Methodology for Data Science - IBM Analytics White Paper 2015

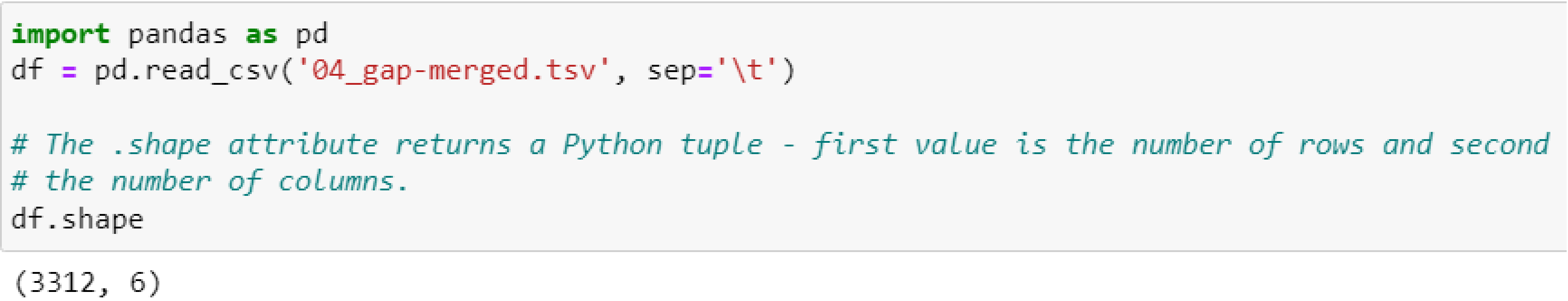




Chapter 1: Pandas Dataframe Basics

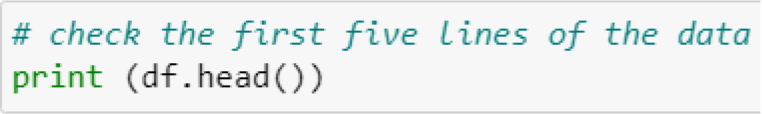
1.1 Introduction

- Loading and Examining the data

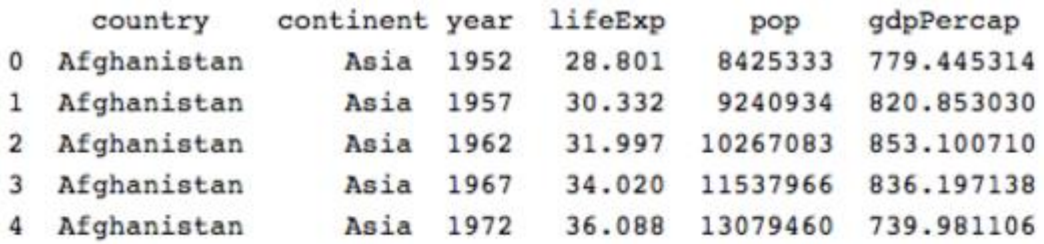


```
import pandas as pd
df = pd.read_csv('04_gap-merged.tsv', sep='\t')

# The .shape attribute returns a Python tuple - first value is the number of rows and second
# the number of columns.
df.shape
```

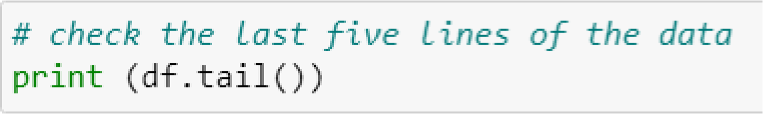
```
(3312, 6)
```



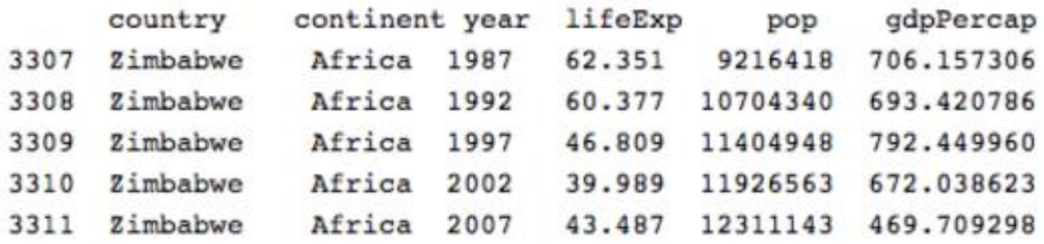
```
# check the first five lines of the data
print (df.head())
```



```
       country continent  year  lifeExp       pop   gdpPercap
0  Afghanistan      Asia  1952   28.801   8425333  779.445314
1  Afghanistan      Asia  1957   30.332   9240934  820.853030
2  Afghanistan      Asia  1962   31.997  10267083  853.100710
3  Afghanistan      Asia  1967   34.020  11537966  836.197138
4  Afghanistan      Asia  1972   36.088  13079460  739.981106
```



```
# check the last five lines of the data
print (df.tail())
```

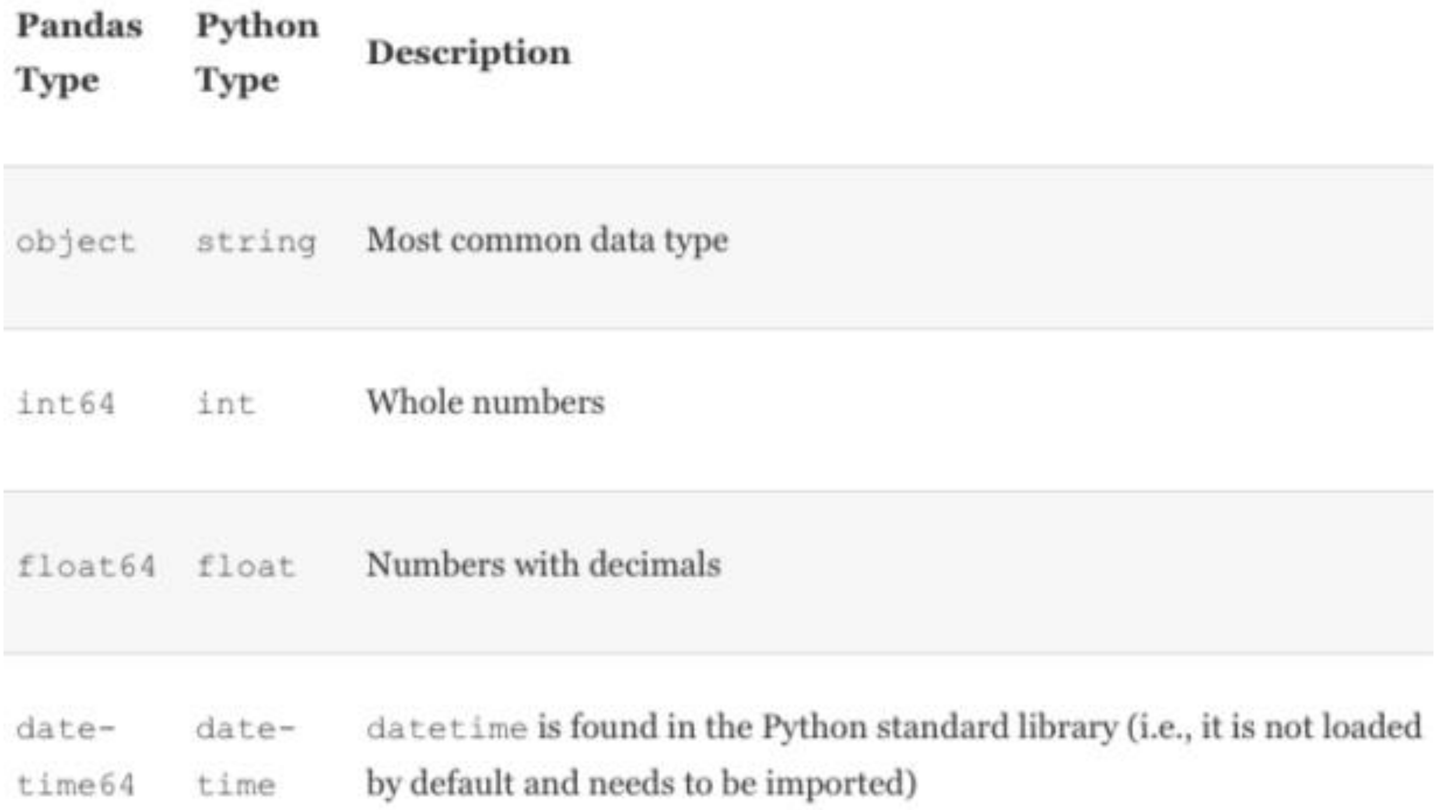


```
       country continent  year  lifeExp       pop   gdpPercap
3307  Zimbabwe    Africa  1987   62.351   9216418  706.157306
3308  Zimbabwe    Africa  1992   60.377  10704340  693.420786
3309  Zimbabwe    Africa  1997   46.809  11404948  792.449960
3310  Zimbabwe    Africa  2002   39.989  11926563  672.038623
3311  Zimbabwe    Africa  2007   43.487  12311143  469.709298
```

Chapter 1: Pandas Dataframe Basics

1.1 Introduction

- PD Data Types
  - Table 1.1 Pandas Types Versus Python Types

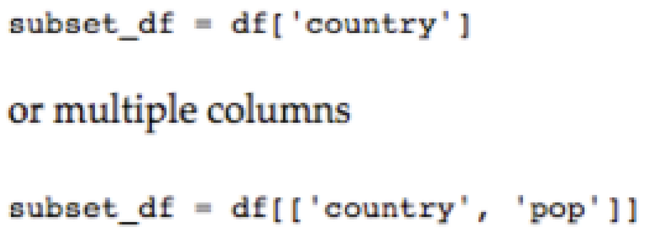


Pandas Type	Python Type	Description
object	string	Most common data type
int64	int	Whole numbers
float64	float	Numbers with decimals
datetime64	datetime	datetime is found in the Python standard library (i.e., it is not loaded by default and needs to be imported)

Chapter 1: Pandas Dataframe Basics

1.1 Introduction

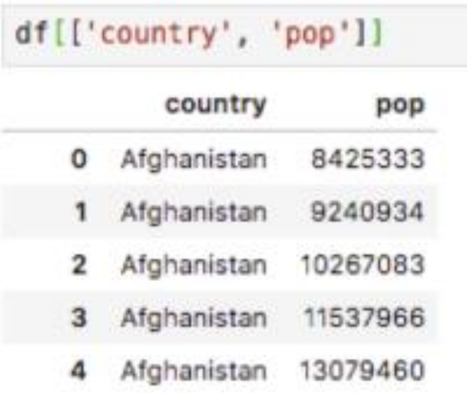
- Sub-setting Columns
  - By Name



```
subset_df = df['country']
```

or multiple columns

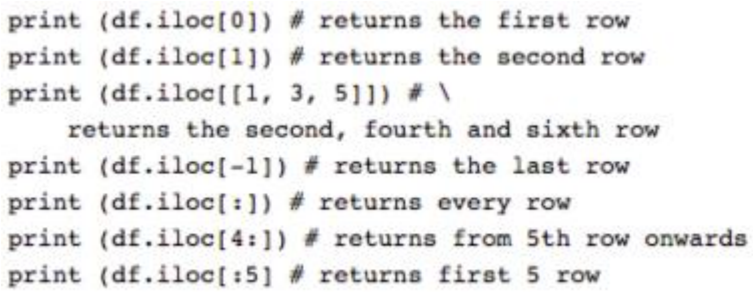
```
subset_df = df[['country', 'pop']]
```



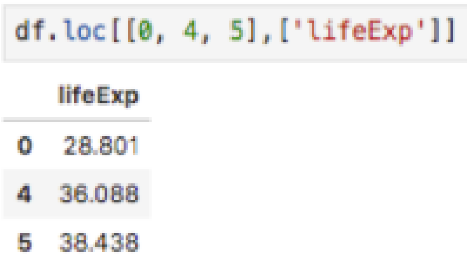
```
df[['country', 'pop']]
```

	country	pop
0	Afghanistan	8425333
1	Afghanistan	9240934
2	Afghanistan	10267083
3	Afghanistan	11537966
4	Afghanistan	13079460

- Sub-setting Rows
  - By Index, Name
  - Use df.loc[: , [columns]] to subset the column(s).



```
print (df.iloc[0]) # returns the first row
print (df.iloc[1]) # returns the second row
print (df.iloc[[1, 3, 5]]) # \
    returns the second, fourth and sixth row
print (df.iloc[-1]) # returns the last row
print (df.iloc[:]) # returns every row
print (df.iloc[4:]) # returns from 5th row onwards
print (df.iloc[:5] # returns first 5 row
```



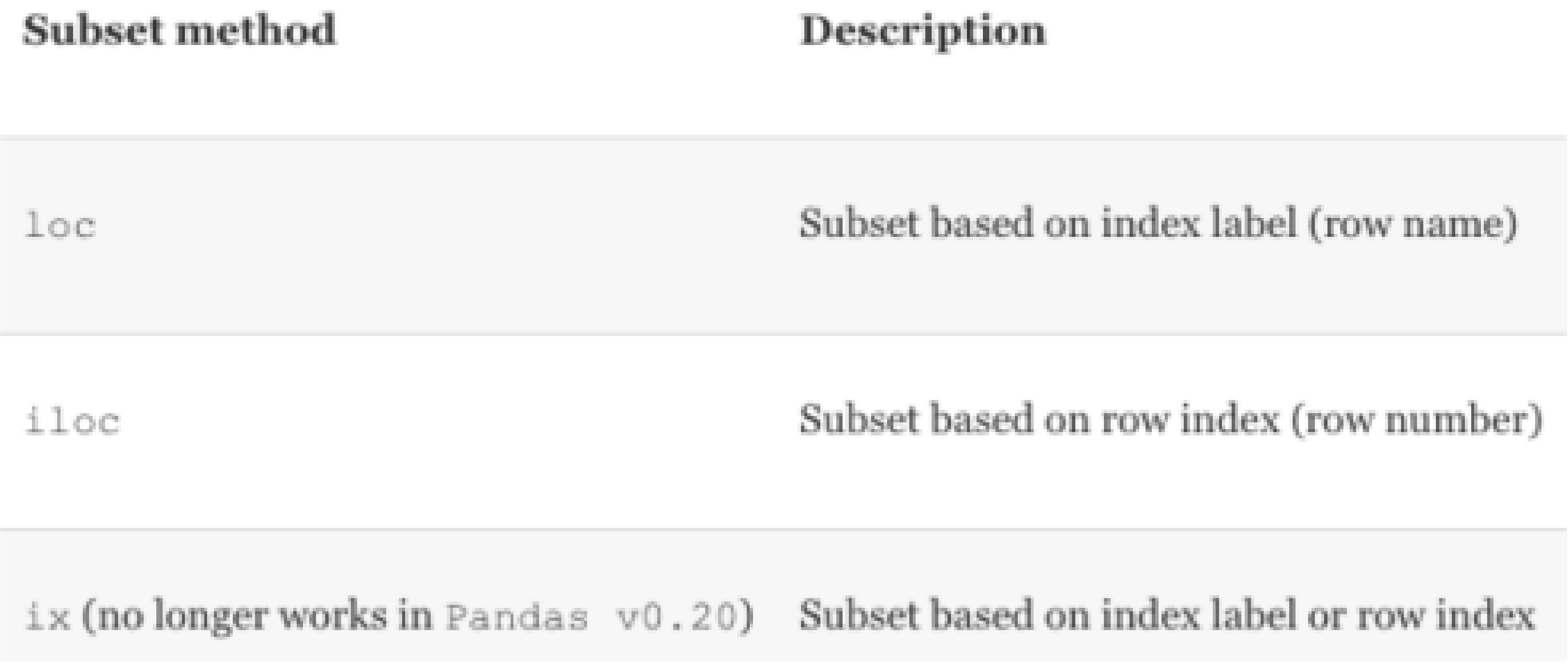
```
df.loc[[0, 4, 5],['lifeExp']]
```

	lifeExp
0	28.801
4	36.088
5	38.438

Chapter 1: Pandas Dataframe Basics

1.1 Introduction

- Sub-setting Rows
  - By Index, Name
  - Use df.loc[: , [columns]] to subset the column(s).

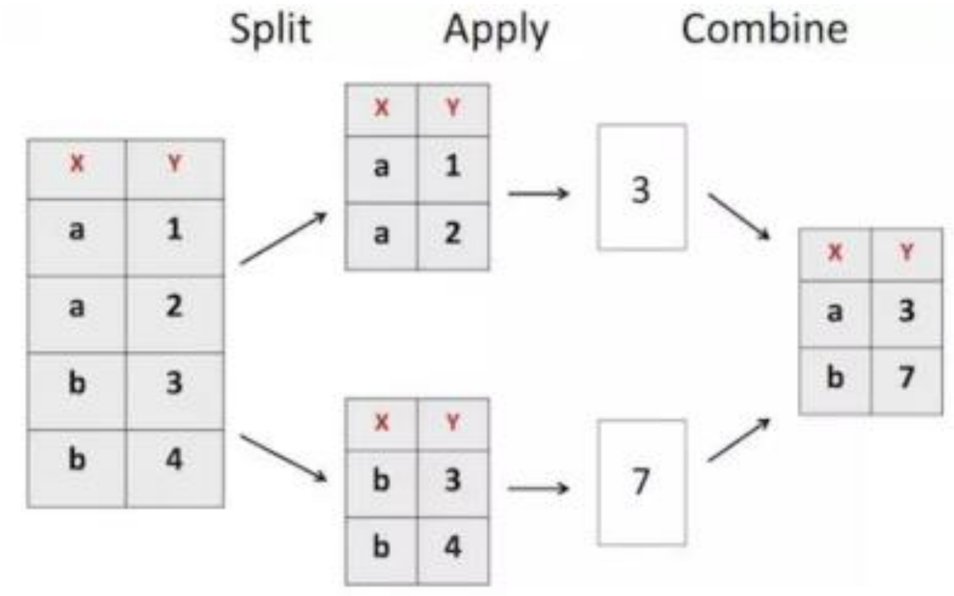


Subset method	Description
`loc`	Subset based on index label (row name)
`iloc`	Subset based on row index (row number)
`ix` (no longer works in `Pandas v0.20`)	Subset based on index label or row index

Chapter 1: Pandas Dataframe Basics

1.1 Introduction

- Grouped and Aggregated Calculations



X	Y
a	1
a	2
b	3
b	4

X	Y
a	1
a	2

X	Y
b	3
b	4

X	Y
a	3
b	7

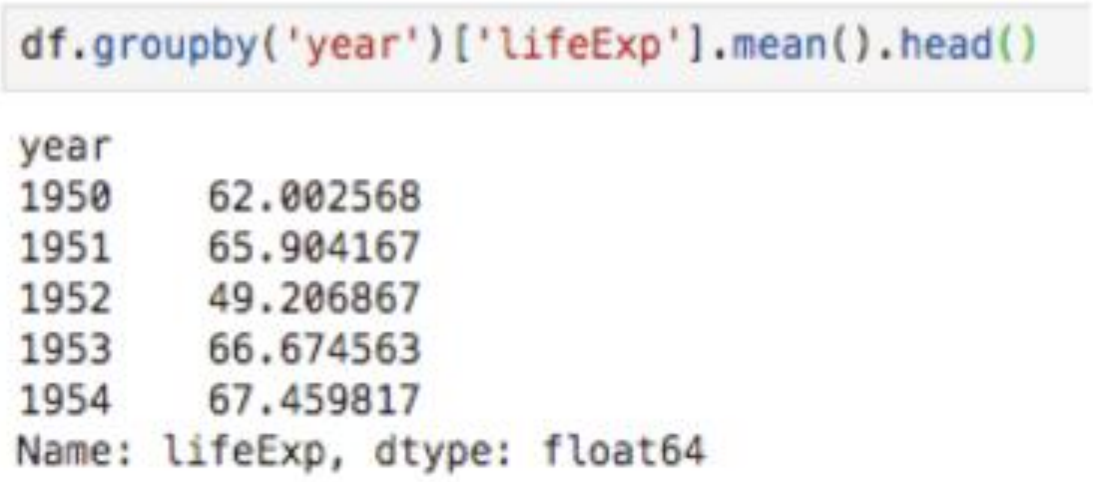
```
df.groupby('x').sum()
```

Select sum(y) as total_y from tb group by x

Chapter 1: Pandas Dataframe Basics

1.1 Introduction

- Grouped and Aggregated Calculations
  - For each year, what is the life expectancy average for all countries, or the total population for all countries?



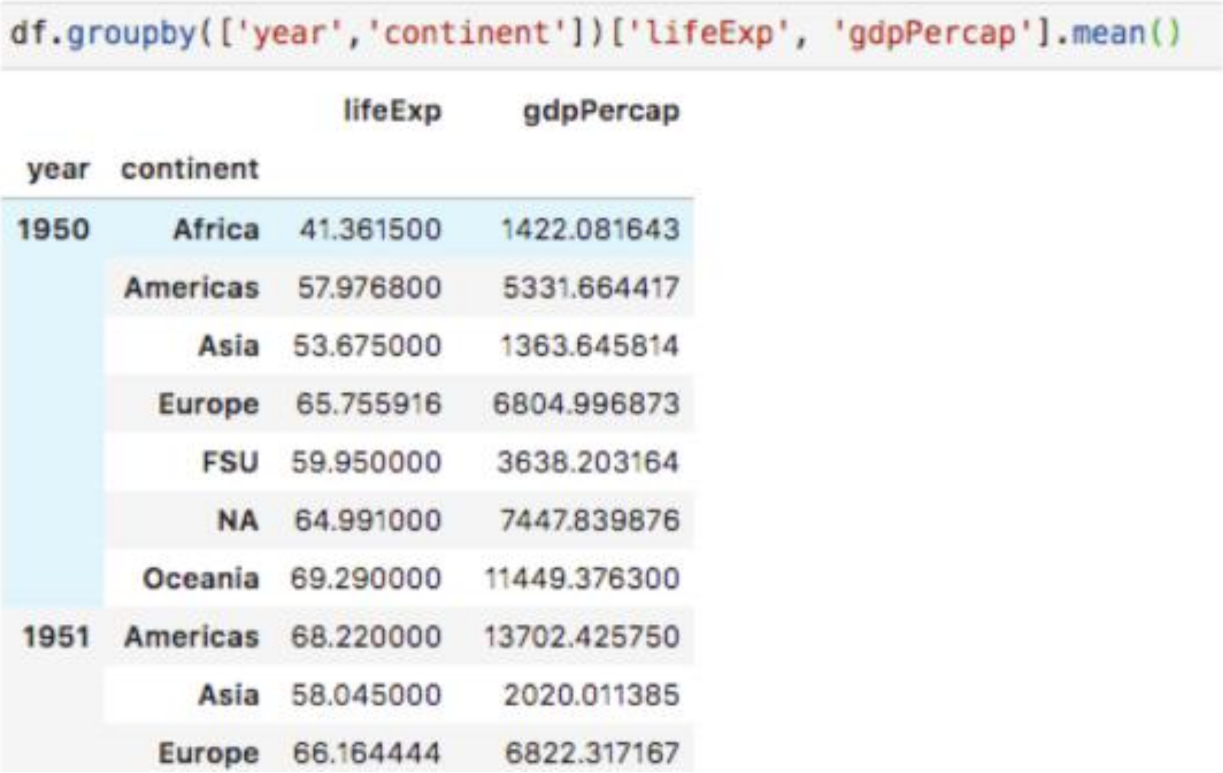
```
df.groupby('year')['lifeExp'].mean().head()
```

```
year
1950    62.002568
1951    65.904167
1952    49.206867
1953    66.674563
1954    67.459817
Name: lifeExp, dtype: float64
```

Chapter 1: Pandas Dataframe Basics

1.1 Introduction

- Grouped and Aggregated Calculations
  - For each year, what is the life expectancy average for all countries, by continent?



```
df.groupby(['year','continent'])['lifeExp', 'gdpPercap'].mean()
```

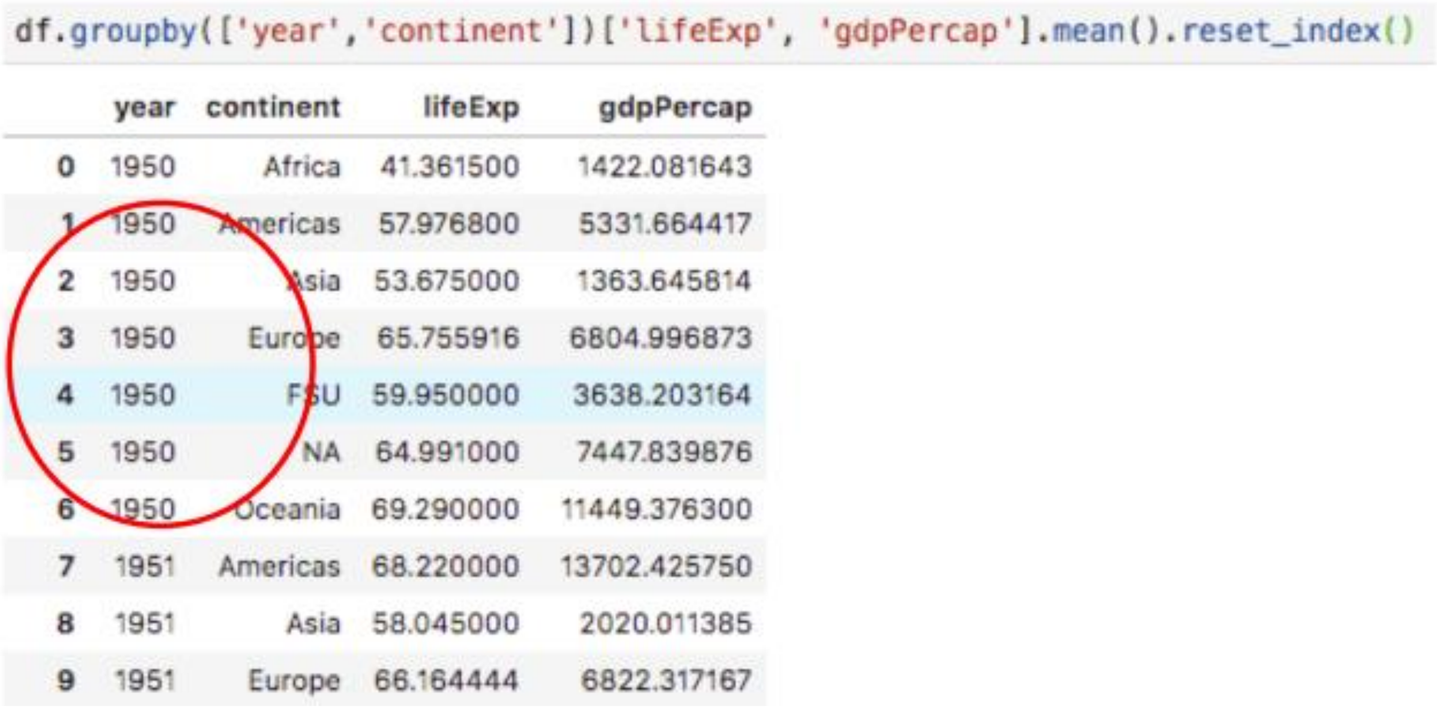
```
                   lifeExp     gdpPercap
year  continent
1950     Africa  41.361500   1422.081643
       Americas  57.976800   5331.664417
           Asia  53.675000   1363.645814
         Europe  65.755916   6804.996873
            FSU  59.950000   3638.203164
             NA  64.991000   7447.839876
        Oceania  69.290000  11449.376300
1951   Americas  68.220000  13702.425750
           Asia  58.045000   2020.011385
         Europe  66.164444   6822.317167
```

Chapter 1: Pandas Dataframe Basics

1.1 Introduction

- Grouped and Aggregated Calculations
  - For each year, what is the life expectancy average for all countries, by continent?

Flatten the dataframe, using reset_index method



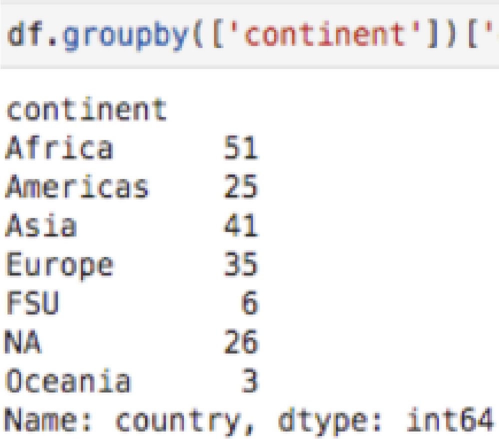
```
df.groupby(['year','continent'])['lifeExp', 'gdpPercap'].mean().reset_index()
```

	year	continent	lifeExp	gdpPercap
0	1950	Africa	41.361500	1422.081643
1	1950	Americas	57.976800	5331.664417
2	1950	Asia	53.675000	1363.645814
3	1950	Europe	65.755916	6804.996873
4	1950	FSU	59.950000	3638.203164
5	1950	NA	64.991000	7447.839876
6	1950	Oceania	69.290000	11449.376300
7	1951	Americas	68.220000	13702.425750
8	1951	Asia	58.045000	2020.011385
9	1951	Europe	66.164444	6822.317167

Chapter 1: Pandas Dataframe Basics

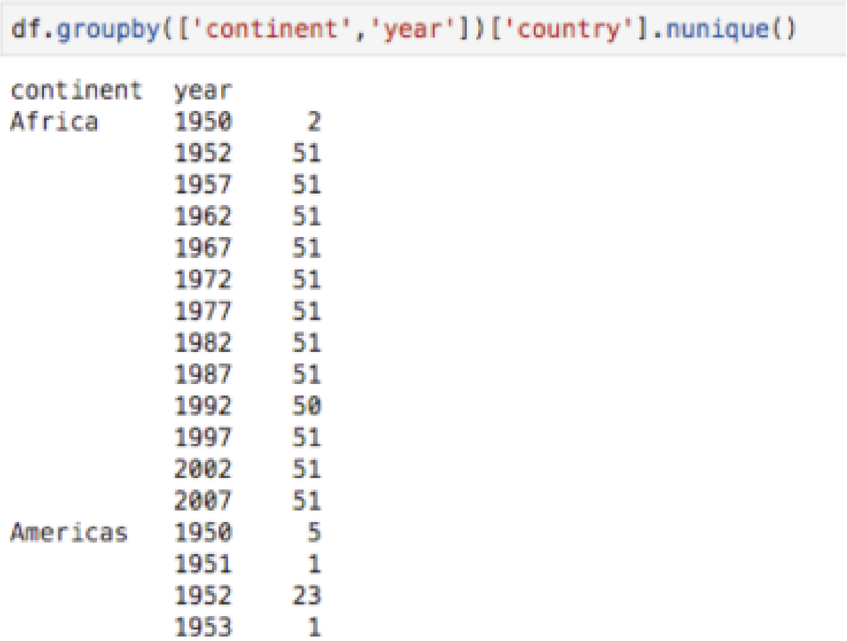
1.1 Introduction

- Grouped and Aggregated Calculations
  - Grouped Frequency Counts



```
df.groupby(['continent'])['country'].nunique()
```

```
continent
Africa      51
Americas    25
Asia        41
Europe      35
FSU          6
NA          26
Oceania      3
Name: country, dtype: int64
```



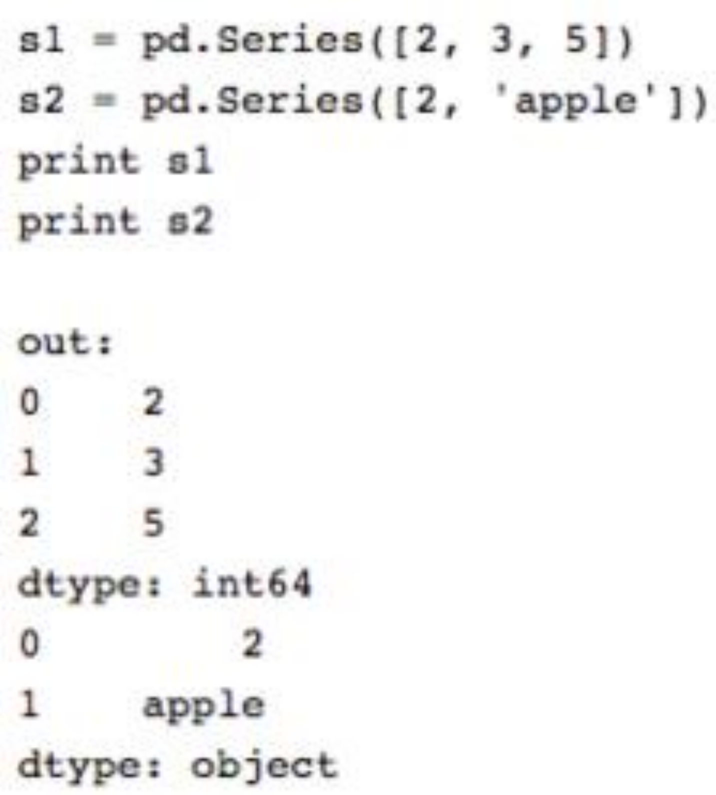
```
df.groupby(['continent','year'])['country'].nunique()
```

```
continent  year
Africa     1950     2
           1952    51
           1957    51
           1962    51
           1967    51
           1972    51
           1977    51
           1982    51
           1987    51
           1992    50
           1997    51
           2002    51
           2007    51
Americas   1950     5
           1951     1
           1952    23
           1953     1
```

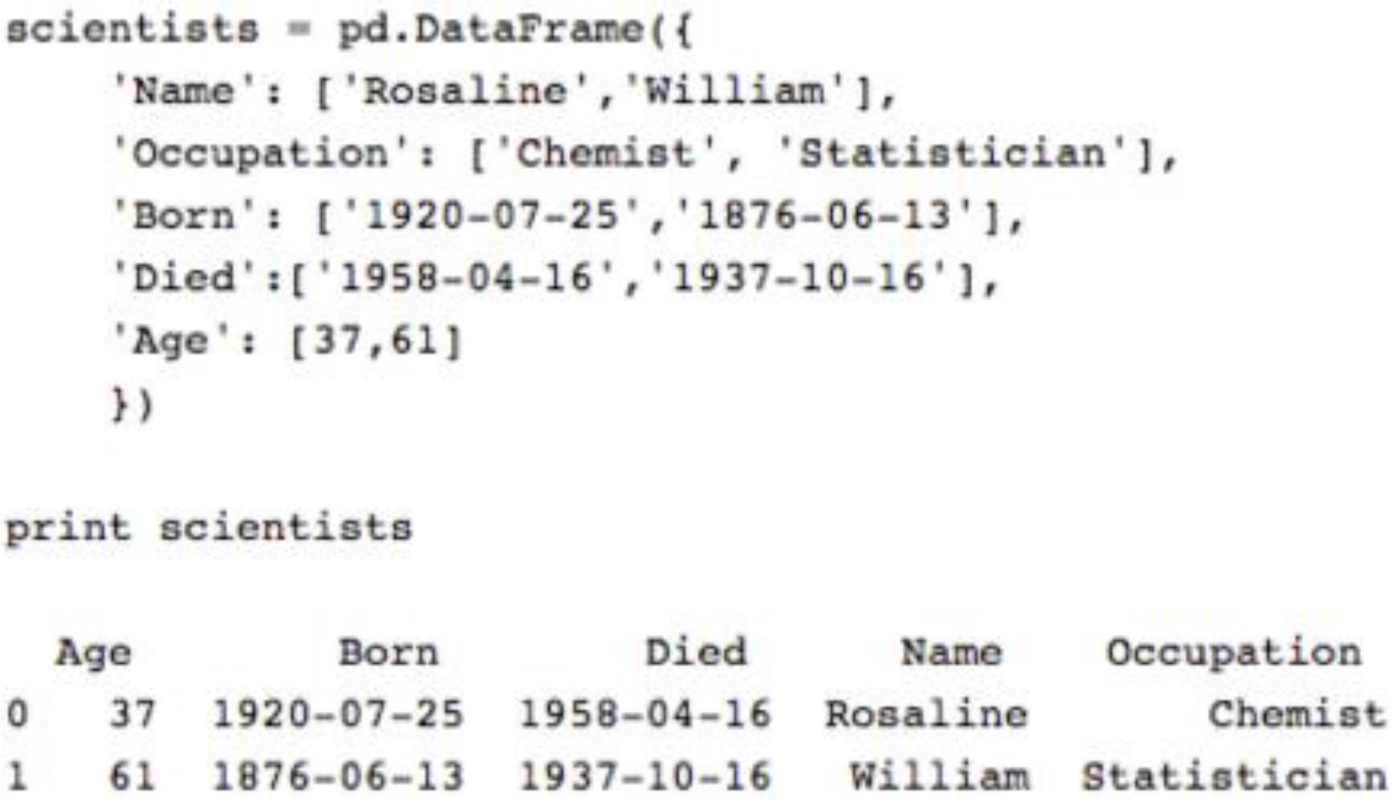
Chapter 1: Pandas Dataframe Basics

2.1 Dataframe Operations

- Creating a Pandas Series



```
s1 = pd.Series([2, 3, 5])
s2 = pd.Series([2, 'apple'])
print s1
print s2

out:
0     2
1     3
2     5
dtype: int64
0         2
1     apple
dtype: object
```

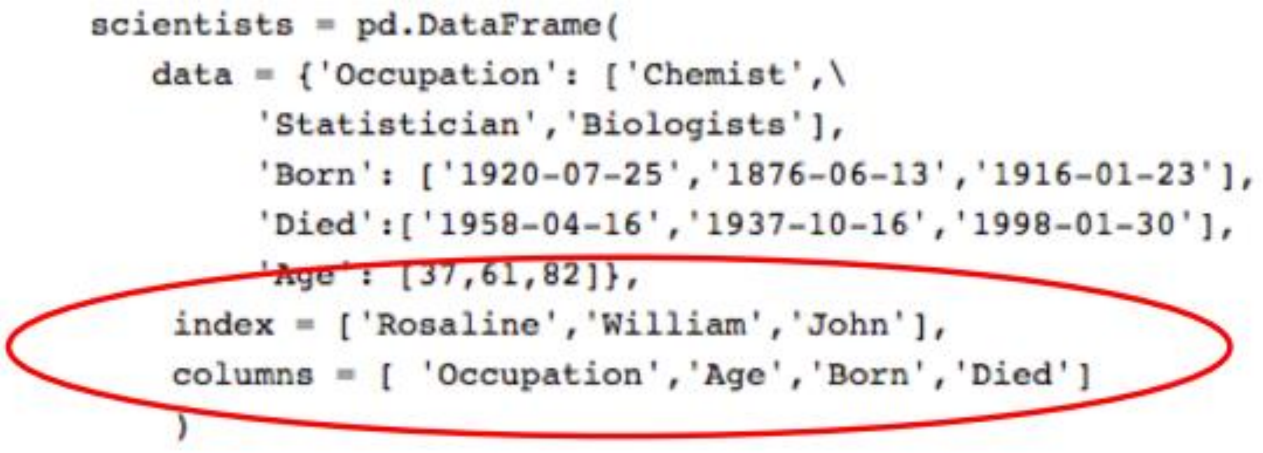


```
scientists = pd.DataFrame({
     'Name': ['Rosaline','William'],
     'Occupation': ['Chemist', 'Statistician'],
     'Born': ['1920-07-25','1876-06-13'],
     'Died':['1958-04-16','1937-10-16'],
     'Age': [37,61]
     })

print scientists

   Age        Born        Died      Name   Occupation
0   37  1920-07-25  1958-04-16  Rosaline      Chemist
1   61  1876-06-13  1937-10-16   William Statistician
```

Chapter 1: Pandas Dataframe Basics

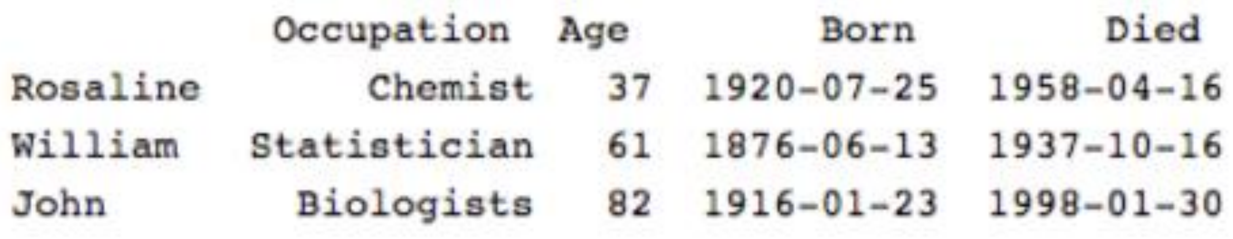
2.1 Dataframe Operations

- Creating a Pandas Series – additional parameters



```
scientists = pd.DataFrame(
   data = {'Occupation': ['Chemist',\
        'Statistician','Biologists'],
        'Born': ['1920-07-25','1876-06-13','1916-01-23'],
        'Died':['1958-04-16','1937-10-16','1998-01-30'],
        'Age': [37,61,82]},
    index = ['Rosaline','William','John'],
    columns = [ 'Occupation','Age','Born','Died']
    )
```

print (scientists)

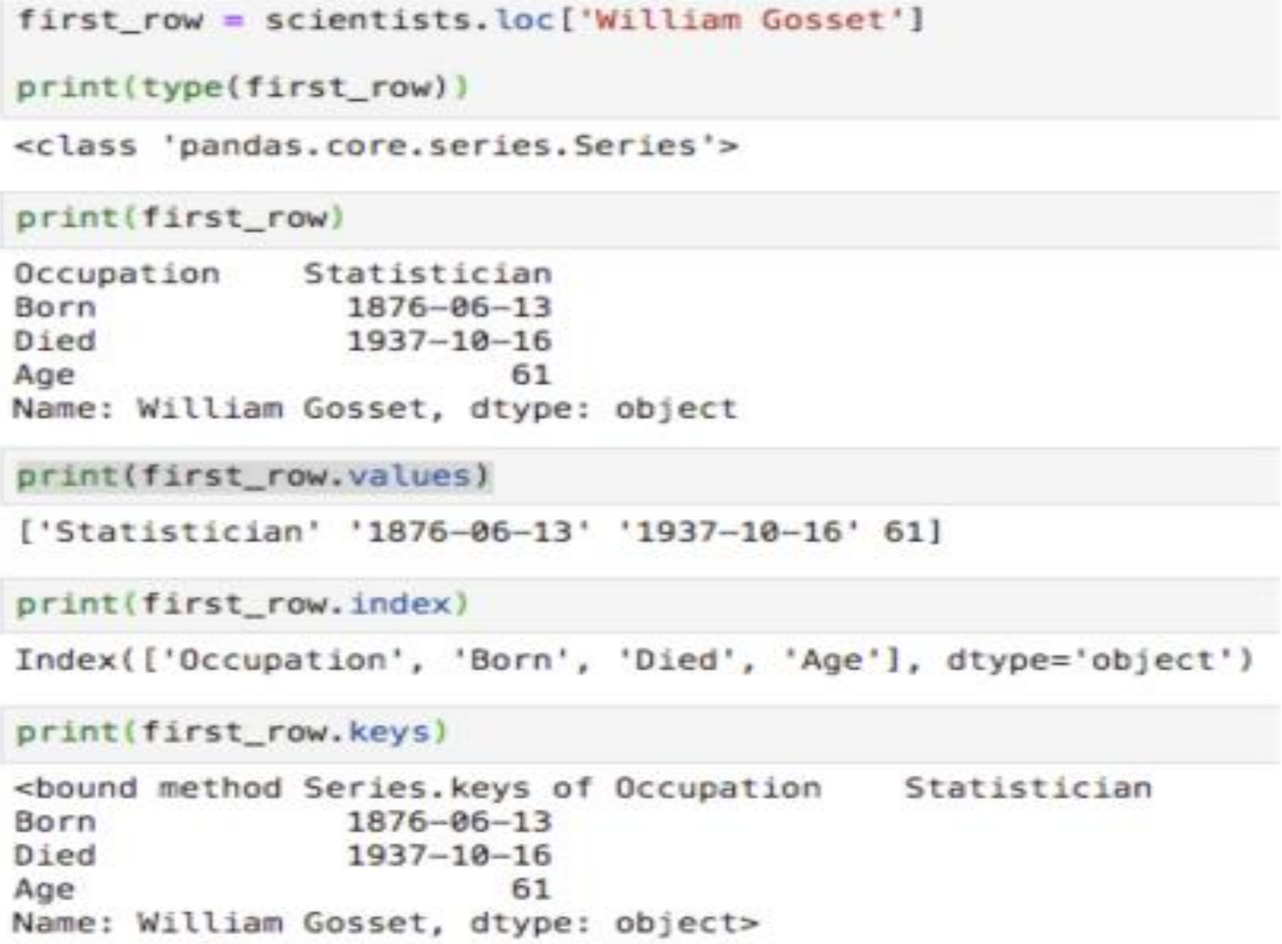


```
             Occupation  Age        Born        Died
Rosaline        Chemist   37  1920-07-25  1958-04-16
William    Statistician   61  1876-06-13  1937-10-16
John         Biologists   82  1916-01-23  1998-01-30
```

Chapter 1: Pandas Dataframe Basics

2.1 Dataframe Operations

- Pandas Series – Examining it



```
first_row = scientists.loc['William Gosset']

print(type(first_row))
```

```
<class 'pandas.core.series.Series'>
```

```
print(first_row)
```

```
Occupation    Statistician
Born            1876-06-13
Died            1937-10-16
Age                     61
Name: William Gosset, dtype: object
```

```
print(first_row.values)
```

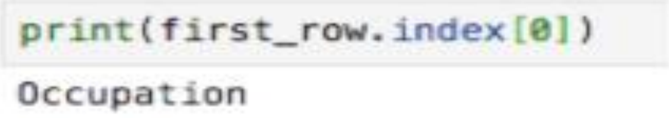
```
['Statistician' '1876-06-13' '1937-10-16' 61]
```

```
print(first_row.index)
```

```
Index(['Occupation', 'Born', 'Died', 'Age'], dtype='object')
```

```
print(first_row.keys)
```

```
<bound method Series.keys of Occupation    Statistician
Born            1876-06-13
Died            1937-10-16
Age                     61
Name: William Gosset, dtype: object>
```



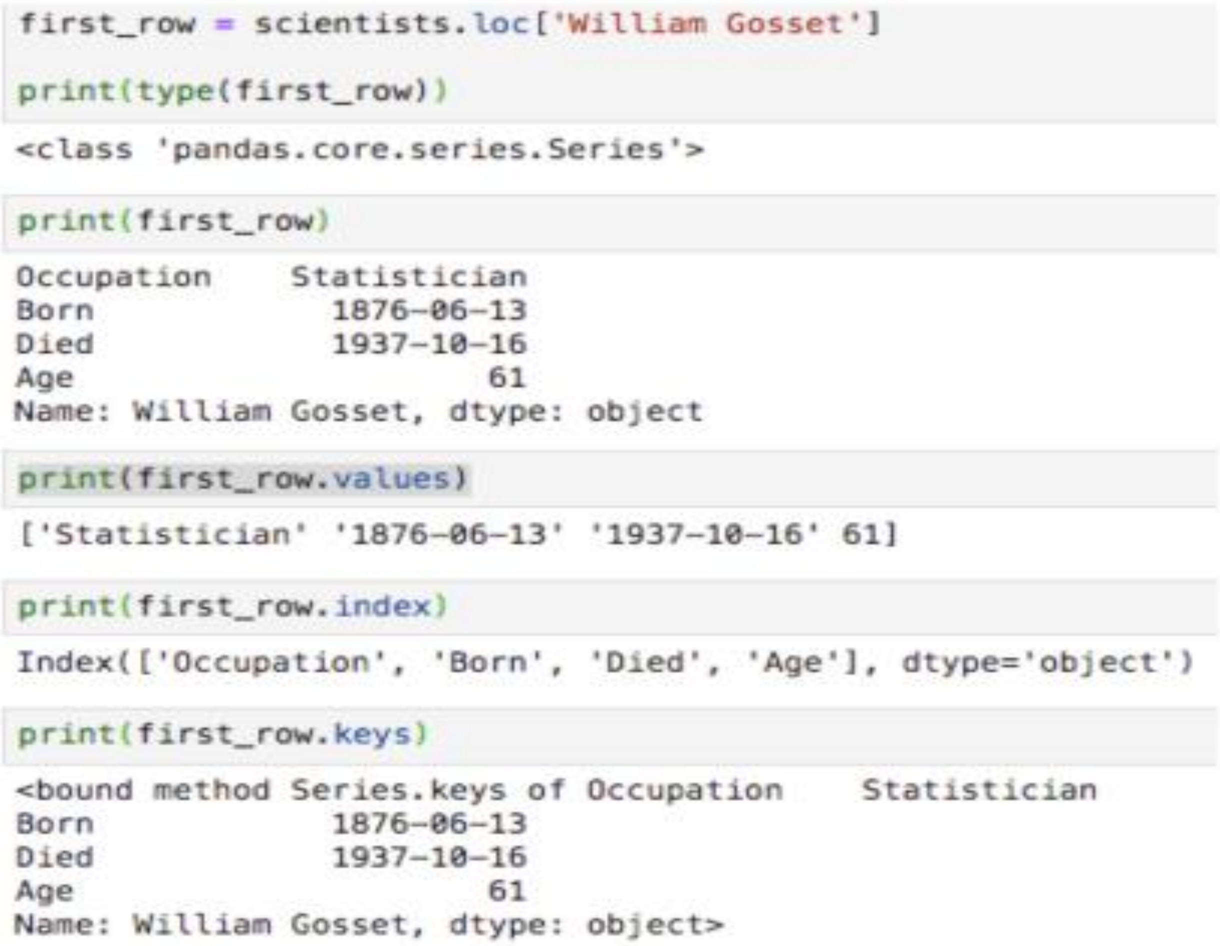
```
print(first_row.index[0])
```

```
Occupation
```

Chapter 1: Pandas Dataframe Basics

2.1 Dataframe Operations

- Pandas Series – Examining its attributes



```
first_row = scientists.loc['William Gosset']

print(type(first_row))
```

```
<class 'pandas.core.series.Series'>
```

```
print(first_row)
```

```
Occupation    Statistician
Born            1876-06-13
Died            1937-10-16
Age                     61
Name: William Gosset, dtype: object
```

```
print(first_row.values)
```

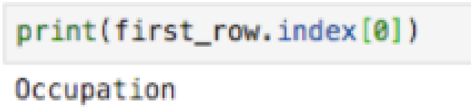
```
['Statistician' '1876-06-13' '1937-10-16' 61]
```

```
print(first_row.index)
```

```
Index(['Occupation', 'Born', 'Died', 'Age'], dtype='object')
```

```
print(first_row.keys)
```

```
<bound method Series.keys of Occupation    Statistician
Born            1876-06-13
Died            1937-10-16
Age                     61
Name: William Gosset, dtype: object>
```



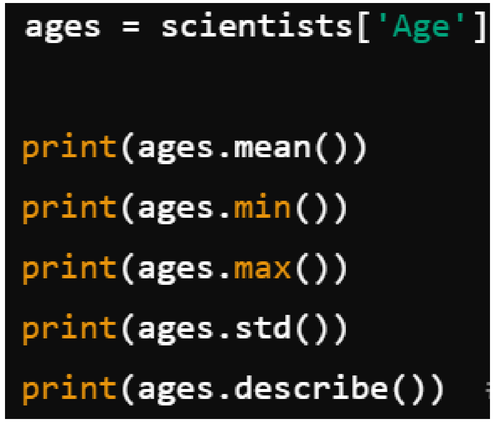
```
print(first_row.index[0])
```

```
Occupation
```

Chapter 1: Pandas Dataframe Basics

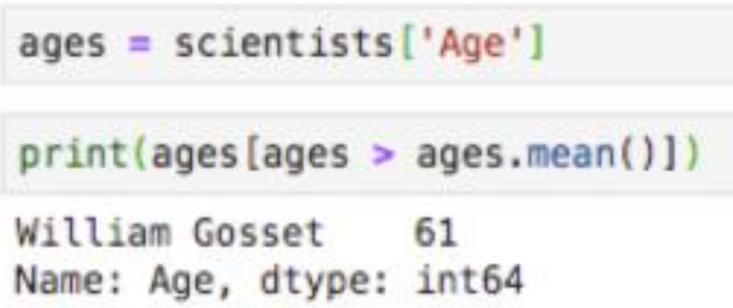
2.1 Dataframe Operations

- Pandas Series – Methods



```
ages = scientists['Age']

print(ages.mean())
print(ages.min())
print(ages.max())
print(ages.std())
print(ages.describe())
```

- Boolean Subsetting: Series

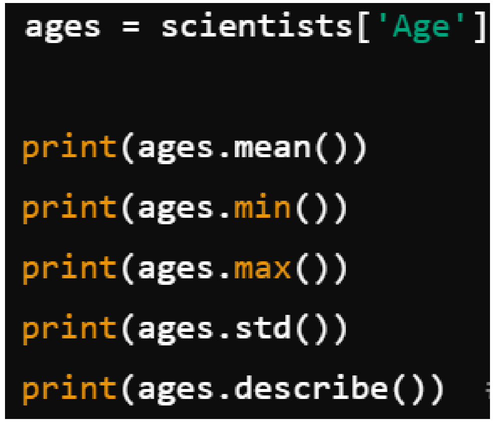


```
ages = scientists['Age']

print(ages[ages > ages.mean()])

William Gosset    61
Name: Age, dtype: int64
```

Chapter 1: Pandas Dataframe Basics

2.1 Dataframe Operations

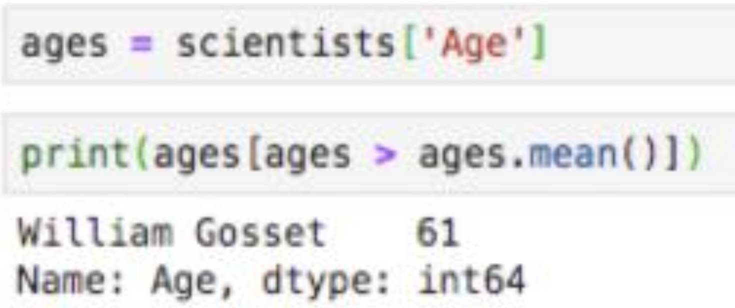
- Pandas Series – Methods



```
ages = scientists['Age']


print(ages.mean())
print(ages.min())
print(ages.max())
print(ages.std())
print(ages.describe())
```

# Calculate a summary of statistics

- Boolean Subsetting: Series



```
ages = scientists['Age']
```

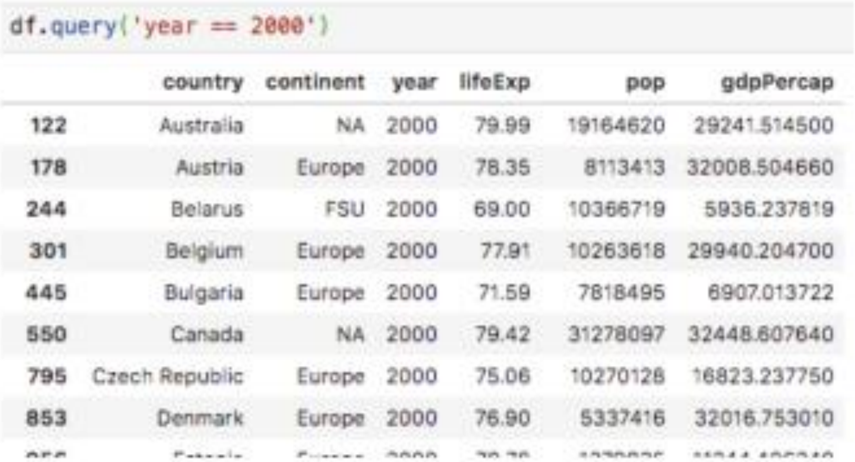
```
print(ages[ages > ages.mean()])
```

```
William Gosset    61
Name: Age, dtype: int64
```

Chapter 1: Pandas Dataframe Basics

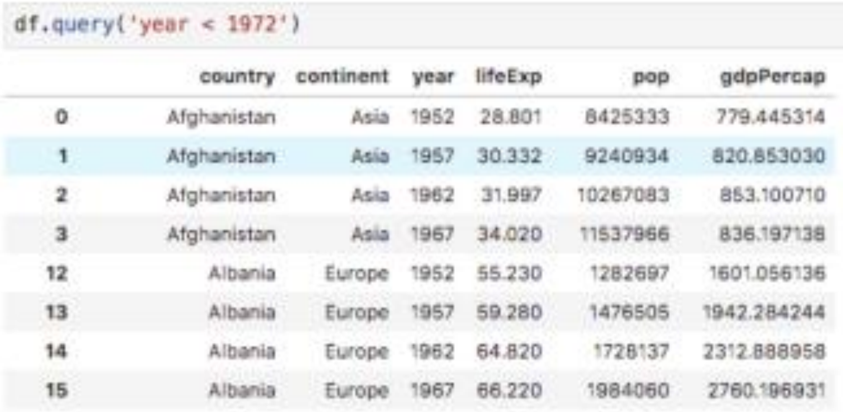
2.1 Dataframe Operations

- Querying and filter Series



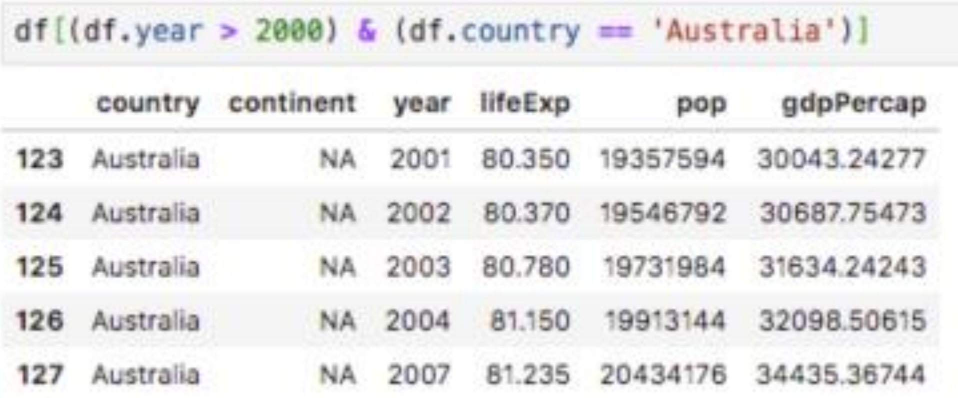
```
df.query('year == 2000')
```

	country	continent	year	lifeExp	pop	gdpPercap
122	Australia	NA	2000	79.99	19164620	29241.514500
178	Austria	Europe	2000	78.35	8113413	32008.504660
244	Belarus	FSU	2000	69.00	10366719	5936.237819
301	Belgium	Europe	2000	77.91	10263618	29940.204700
445	Bulgaria	Europe	2000	71.59	7818495	6907.013722
550	Canada	NA	2000	79.42	31278097	32448.607640
795	Czech Republic	Europe	2000	75.06	10270128	16823.237750
853	Denmark	Europe	2000	76.90	5337416	32016.753010



```
df.query('year < 1972')
```

	country	continent	year	lifeExp	pop	gdpPercap
0	Afghanistan	Asia	1952	28.801	8425333	779.445314
1	Afghanistan	Asia	1957	30.332	9240934	820.853030
2	Afghanistan	Asia	1962	31.997	10267083	853.100710
3	Afghanistan	Asia	1967	34.020	11537966	836.197138
12	Albania	Europe	1952	55.230	1282697	1601.056136
13	Albania	Europe	1957	59.280	1476505	1942.284244
14	Albania	Europe	1962	64.820	1728137	2312.888958
15	Albania	Europe	1967	66.220	1984060	2760.196931



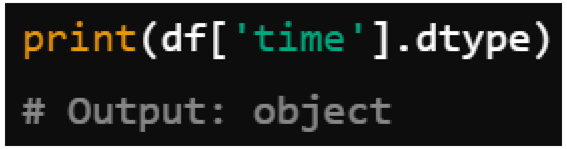
```
df[(df.year > 2000) & (df.country == 'Australia')]
```

	country	continent	year	lifeExp	pop	gdpPercap
123	Australia	NA	2001	80.350	19357594	30043.24277
124	Australia	NA	2002	80.370	19546792	30687.75473
125	Australia	NA	2003	80.780	19731984	31634.24243
126	Australia	NA	2004	81.150	19913144	32098.50615
127	Australia	NA	2007	81.235	20434176	34435.36744

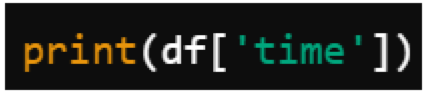
Chapter 1: Pandas Dataframe Basics

2.1 Dataframe Operations

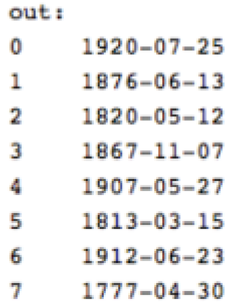
- Querying and filter Series
- Updating Series & Dataframes
  - Changing datatypes
  - Creating new columns



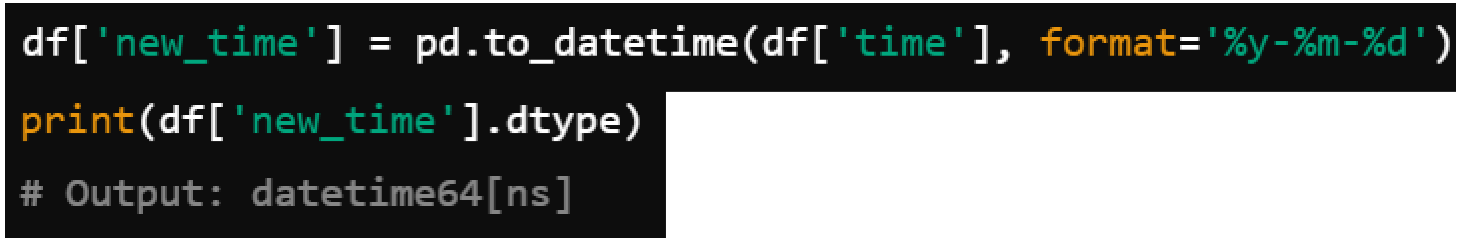
```
print(df['time'].dtype)
# Output: object
```



```
print(df['time'])
```



```
out:
0      1920-07-25
1      1876-06-13
2      1820-05-12
3      1867-11-07
4      1907-05-27
5      1813-03-15
6      1912-06-23
7      1777-04-30
```

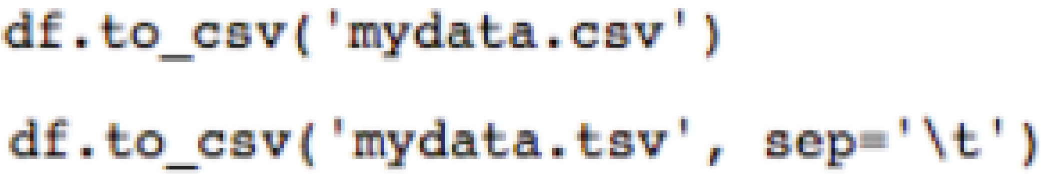


```
df['new_time'] = pd.to_datetime(df['time'], format='%y-%m-%d')
print(df['new_time'].dtype)
# Output: datetime64[ns]
```

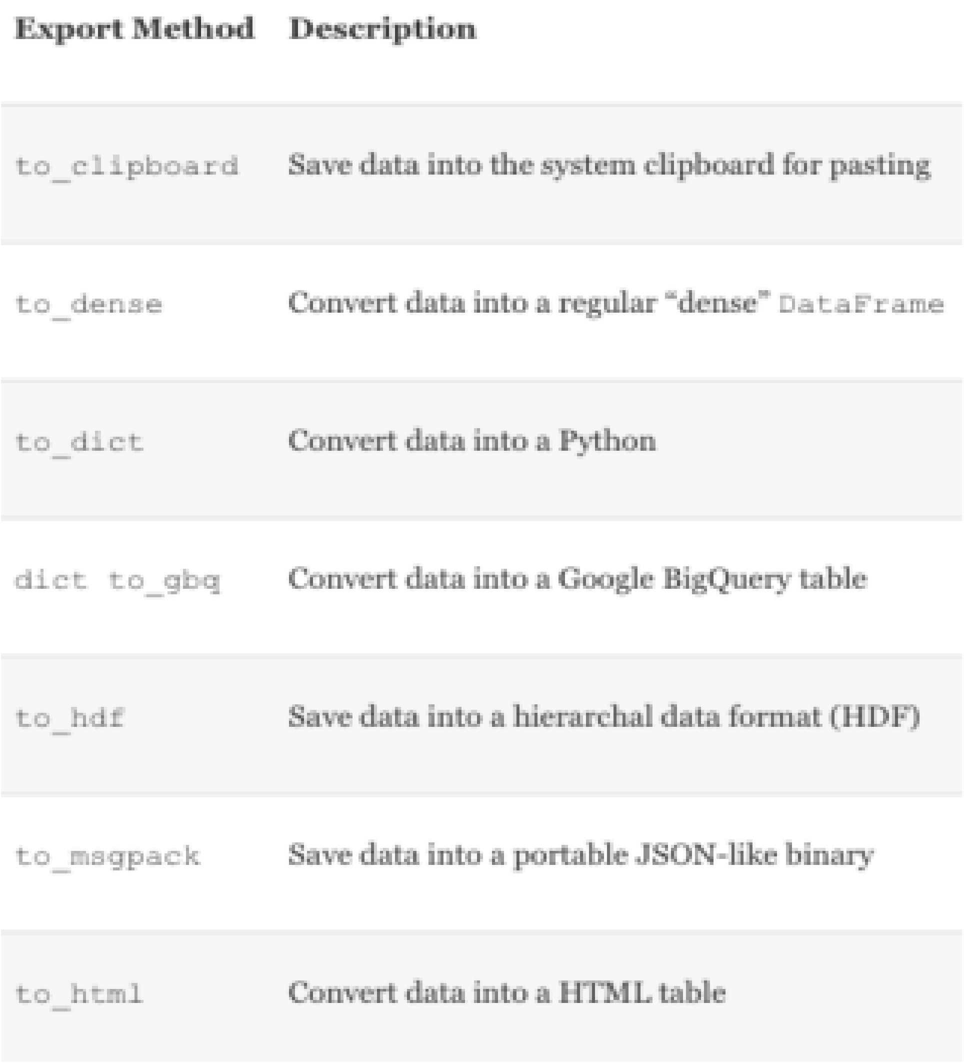
Chapter 1: Pandas Dataframe Basics

2.1 Dataframe Operations

- Exporting DataFrames



```
df.to_csv('mydata.csv')

df.to_csv('mydata.tsv', sep='\t')
```



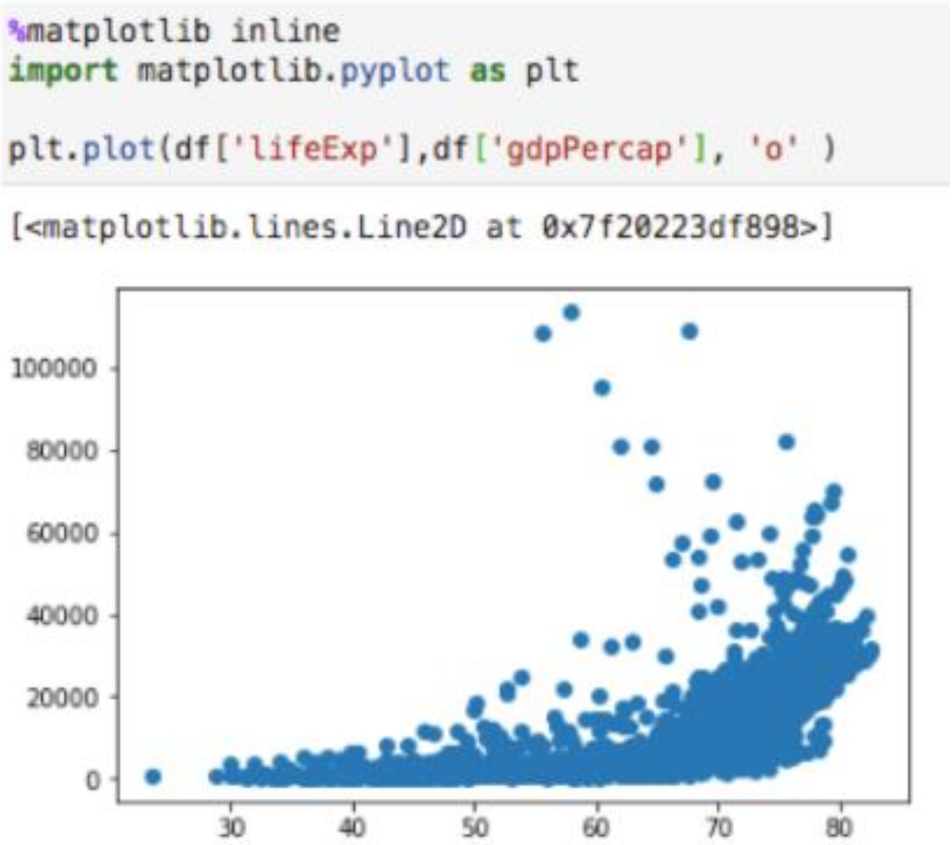
Export Method	Description
`to_clipboard`	Save data into the system clipboard for pasting
`to_dense`	Convert data into a regular "dense" `DataFrame`
`to_dict`	Convert data into a Python
`dict to_gbq`	Convert data into a Google BigQuery table
`to_hdf`	Save data into a hierarchal data format (HDF)
`to_msgpack`	Save data into a portable JSON-like binary
`to_html`	Convert data into a HTML table

Chap2: Table 2.4 DataFrame Export Methods

Chapter 1: Pandas Dataframe Basics

3.1 Introduction to Plotting

- Matplotlib – Python’s fundamental plotting library



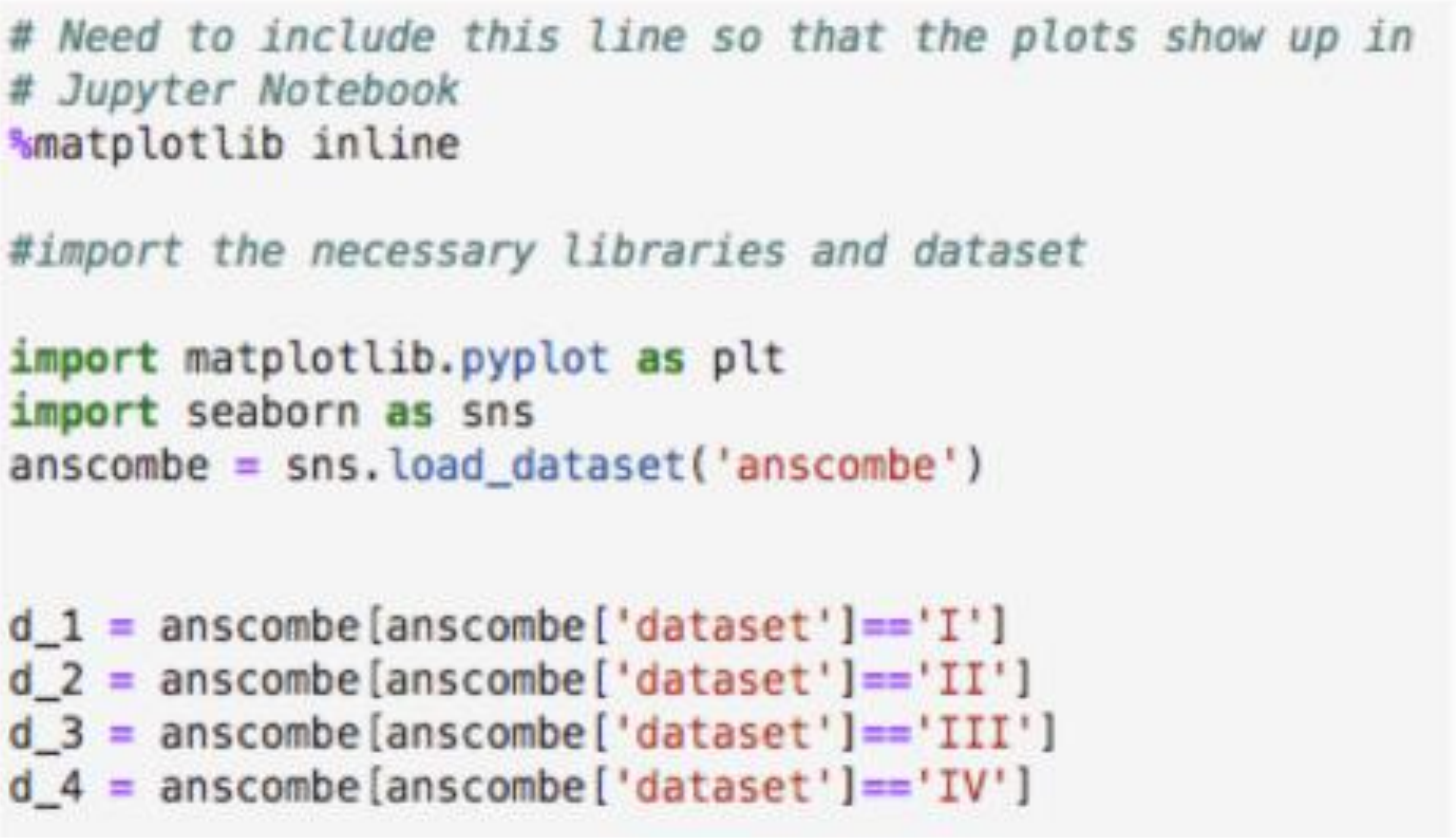
```
%matplotlib inline
import matplotlib.pyplot as plt

plt.plot(df['lifeExp'],df['gdpPercap'], 'o' )
```

```
[<matplotlib.lines.Line2D at 0x7f20223df898>]
```

Chapter 1: Pandas Dataframe Basics

3.1 Introduction to Plotting

- Matplotlib – Multiple Plots

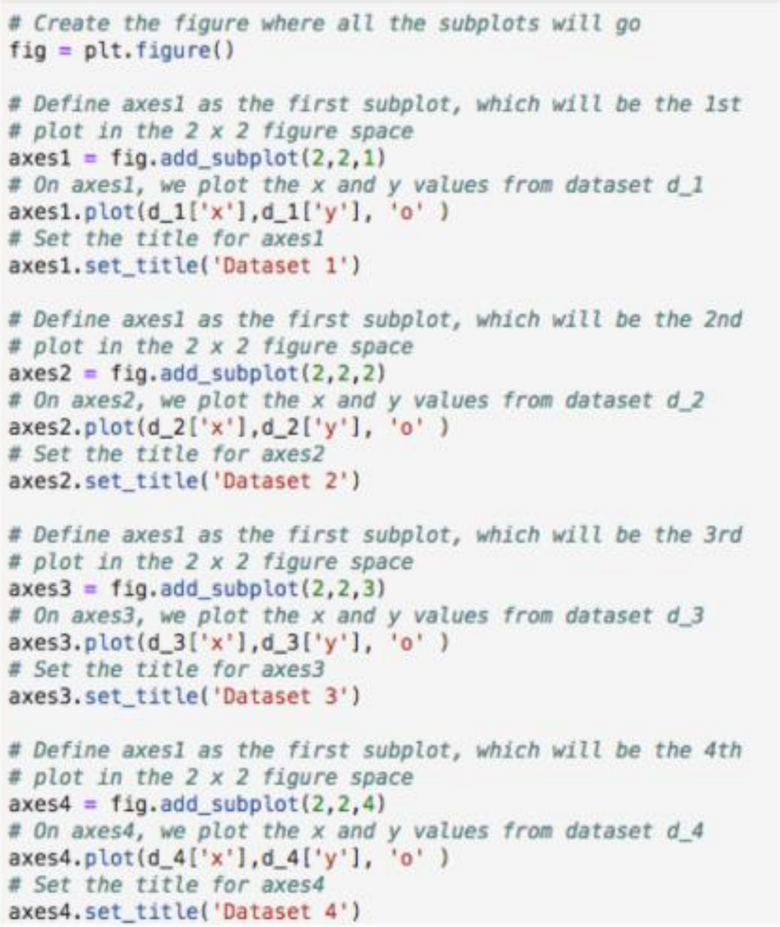


```
# Need to include this line so that the plots show up in
# Jupyter Notebook
%matplotlib inline

#import the necessary libraries and dataset

import matplotlib.pyplot as plt
import seaborn as sns
anscombe = sns.load_dataset('anscombe')


d_1 = anscombe[anscombe['dataset']=='I']
d_2 = anscombe[anscombe['dataset']=='II']
d_3 = anscombe[anscombe['dataset']=='III']
d_4 = anscombe[anscombe['dataset']=='IV']
```

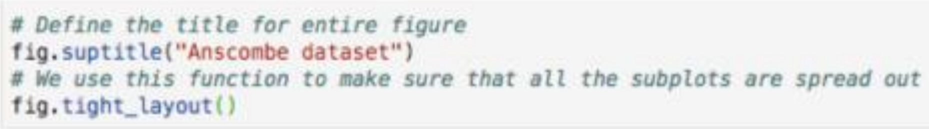
Chapter 1: Pandas Dataframe Basics

3.1 Introduction to Plotting

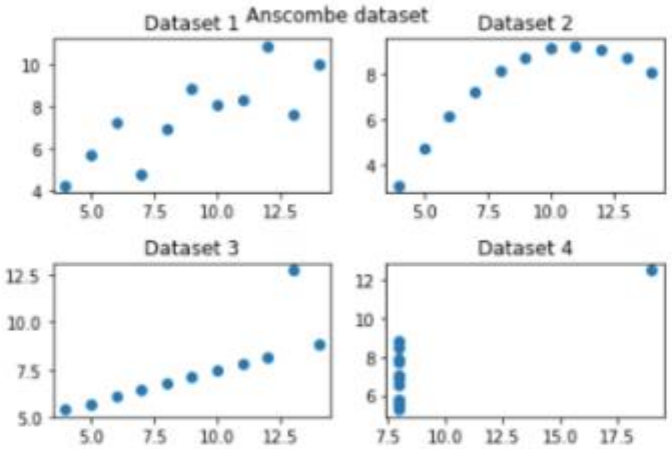
- Matplotlib – Multiple Plots



```
# Create the figure where all the subplots will go
fig = plt.figure()

# Define axes1 as the first subplot, which will be the 1st
# plot in the 2 x 2 figure space
axes1 = fig.add_subplot(2,2,1)
# On axes1, we plot the x and y values from dataset d_1
axes1.plot(d_1['x'],d_1['y'], 'o' )
# Set the title for axes1
axes1.set_title('Dataset 1')

# Define axes1 as the first subplot, which will be the 2nd
# plot in the 2 x 2 figure space
axes2 = fig.add_subplot(2,2,2)
# On axes2, we plot the x and y values from dataset d_2
axes2.plot(d_2['x'],d_2['y'], 'o' )
# Set the title for axes2
axes2.set_title('Dataset 2')

# Define axes1 as the first subplot, which will be the 3rd
# plot in the 2 x 2 figure space
axes3 = fig.add_subplot(2,2,3)
# On axes3, we plot the x and y values from dataset d_3
axes3.plot(d_3['x'],d_3['y'], 'o' )
# Set the title for axes3
axes3.set_title('Dataset 3')

# Define axes1 as the first subplot, which will be the 4th
# plot in the 2 x 2 figure space
axes4 = fig.add_subplot(2,2,4)
# On axes4, we plot the x and y values from dataset d_4
axes4.plot(d_4['x'],d_4['y'], 'o' )
# Set the title for axes4
axes4.set_title('Dataset 4')
```



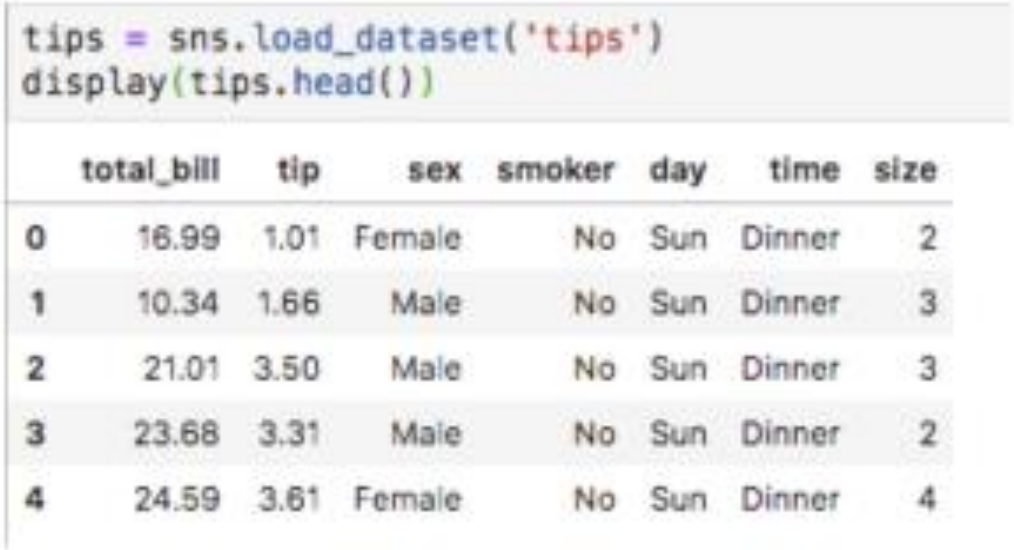
```
# Define the title for entire figure
fig.suptitle("Anscombe dataset")
# We use this function to make sure that all the subplots are spread out
fig.tight_layout()
```



Chapter 1: Pandas Dataframe Basics

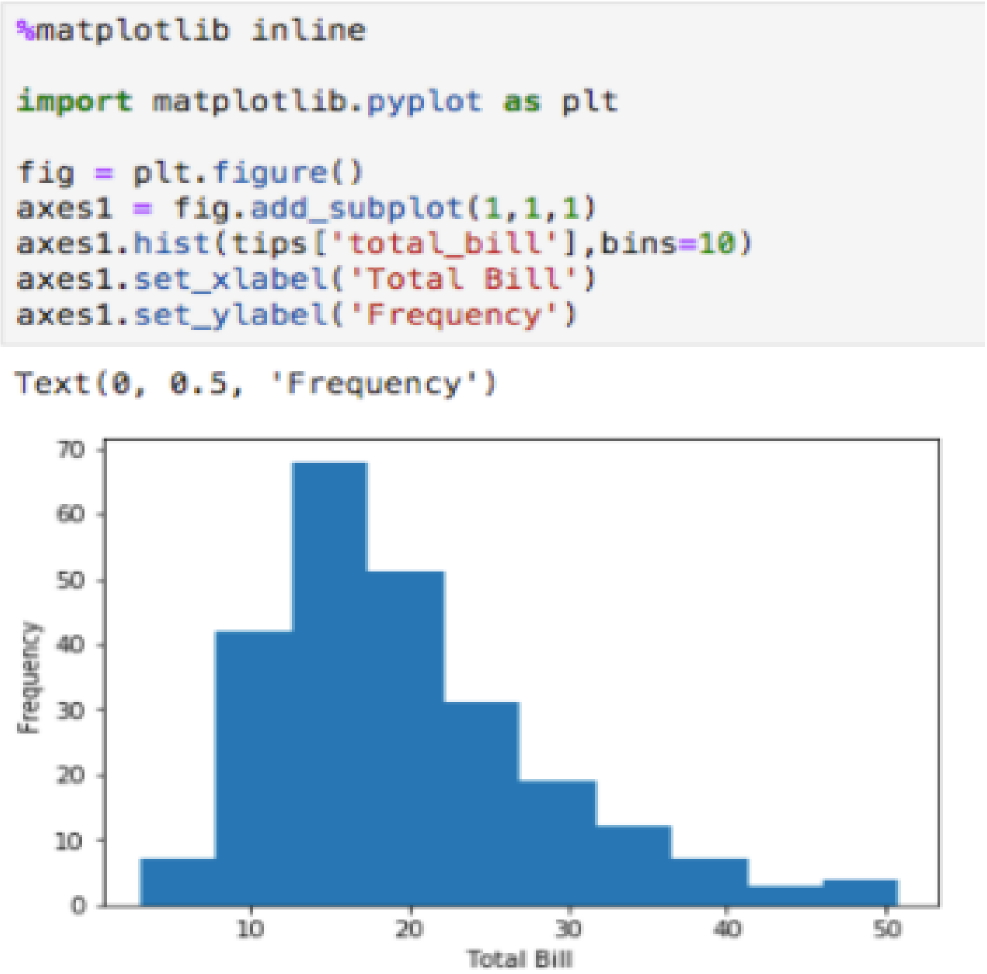
3.2 Statistical Graphics for Different Types of Data

- Univariate



```
tips = sns.load_dataset('tips')
display(tips.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



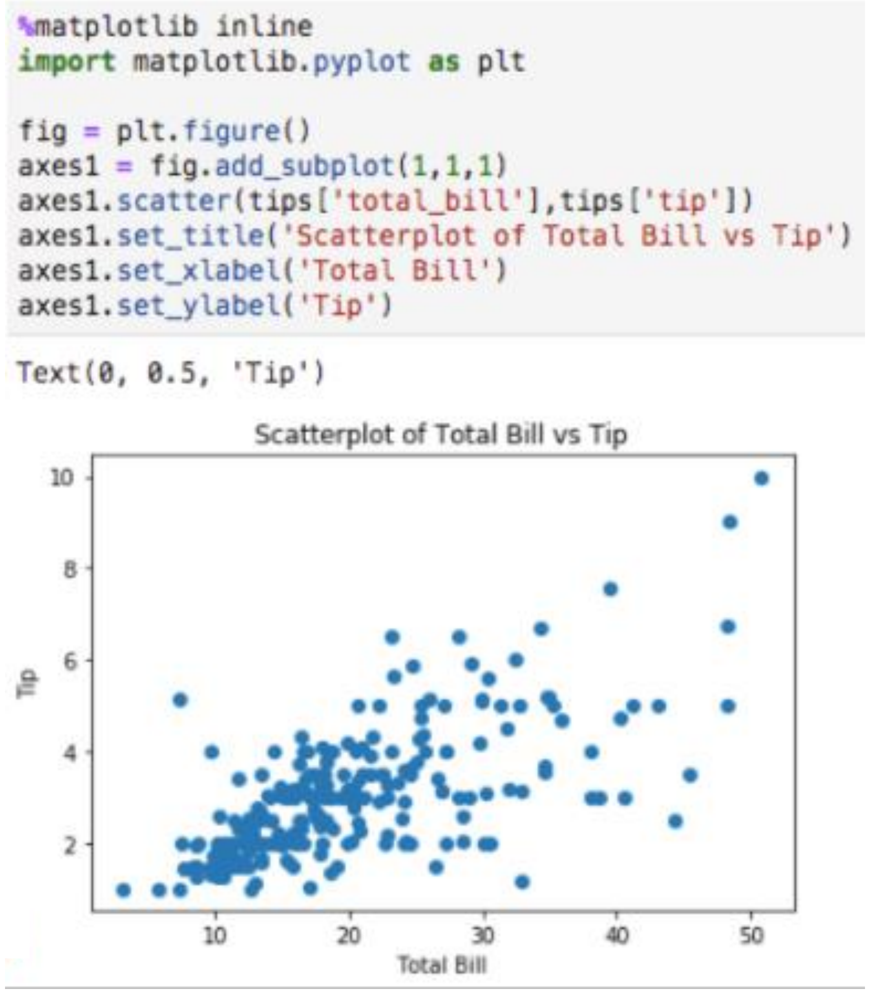
```
%matplotlib inline

import matplotlib.pyplot as plt

fig = plt.figure()
axes1 = fig.add_subplot(1,1,1)
axes1.hist(tips['total_bill'],bins=10)
axes1.set_xlabel('Total Bill')
axes1.set_ylabel('Frequency')
```

```
Text(0, 0.5, 'Frequency')
```

Chapter 1: Pandas Dataframe Basics

3.2 Statistical Graphics for Different Types of Data

- Bivariate



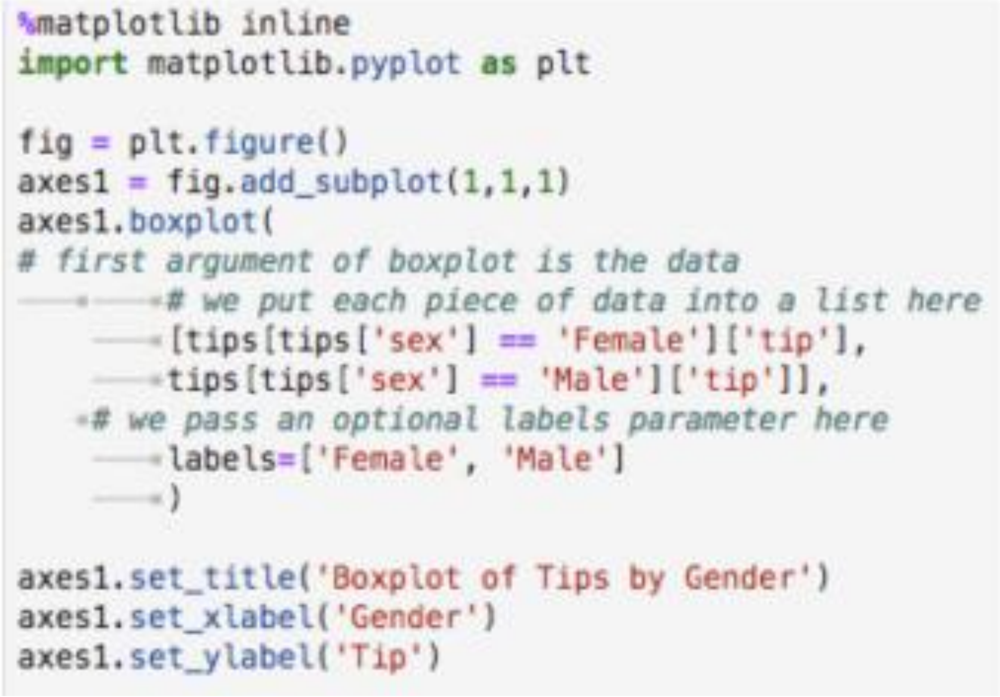
```
%matplotlib inline
import matplotlib.pyplot as plt

fig = plt.figure()
axes1 = fig.add_subplot(1,1,1)
axes1.scatter(tips['total_bill'],tips['tip'])
axes1.set_title('Scatterplot of Total Bill vs Tip')
axes1.set_xlabel('Total Bill')
axes1.set_ylabel('Tip')
```

```
Text(0, 0.5, 'Tip')
```

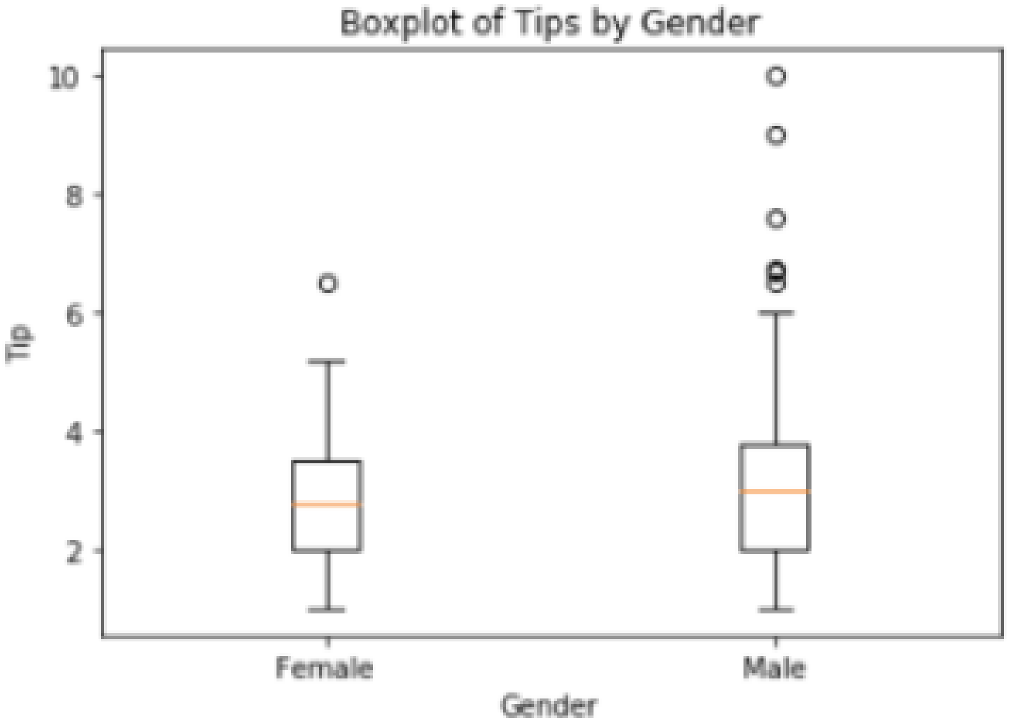
Chapter 1: Pandas Dataframe Basics

3.2 Statistical Graphics for Different Types of Data

- Discrete variable against variable (eg Gender vs Tips given)



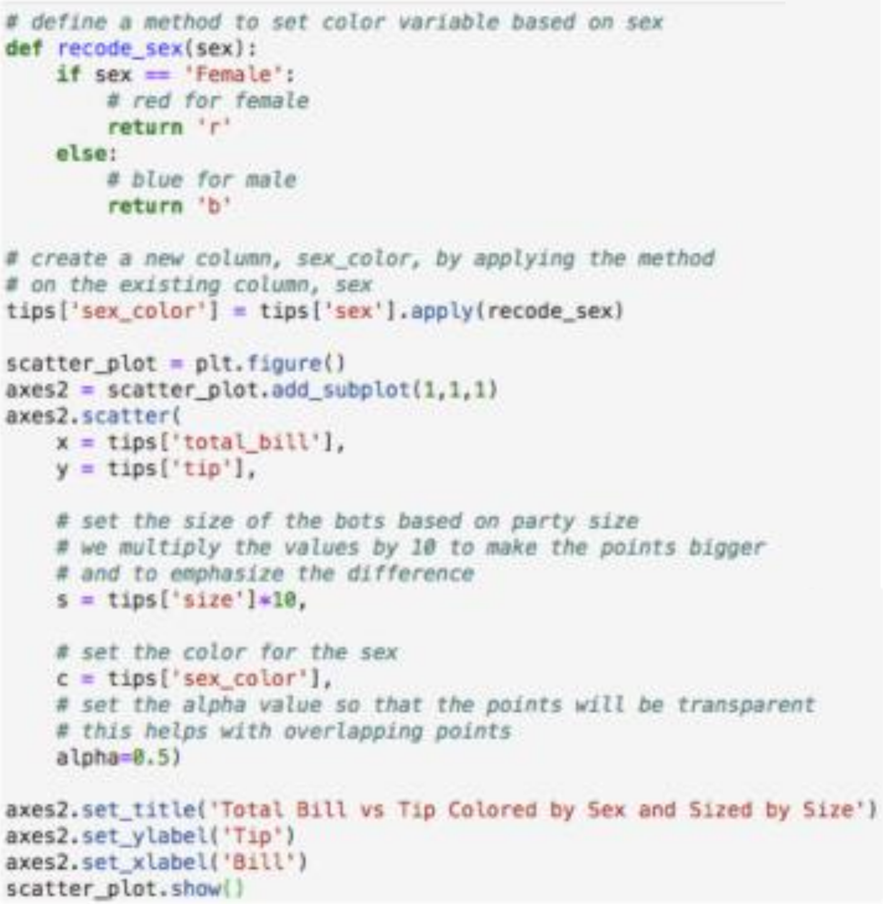
```
%matplotlib inline
import matplotlib.pyplot as plt

fig = plt.figure()
axes1 = fig.add_subplot(1,1,1)
axes1.boxplot(
# first argument of boxplot is the data
        # we put each piece of data into a list here
        [tips[tips['sex'] == 'Female']['tip'],
        tips[tips['sex'] == 'Male']['tip']],
    # we pass an optional labels parameter here
        labels=['Female', 'Male']
        )

axes1.set_title('Boxplot of Tips by Gender')
axes1.set_xlabel('Gender')
axes1.set_ylabel('Tip')
```



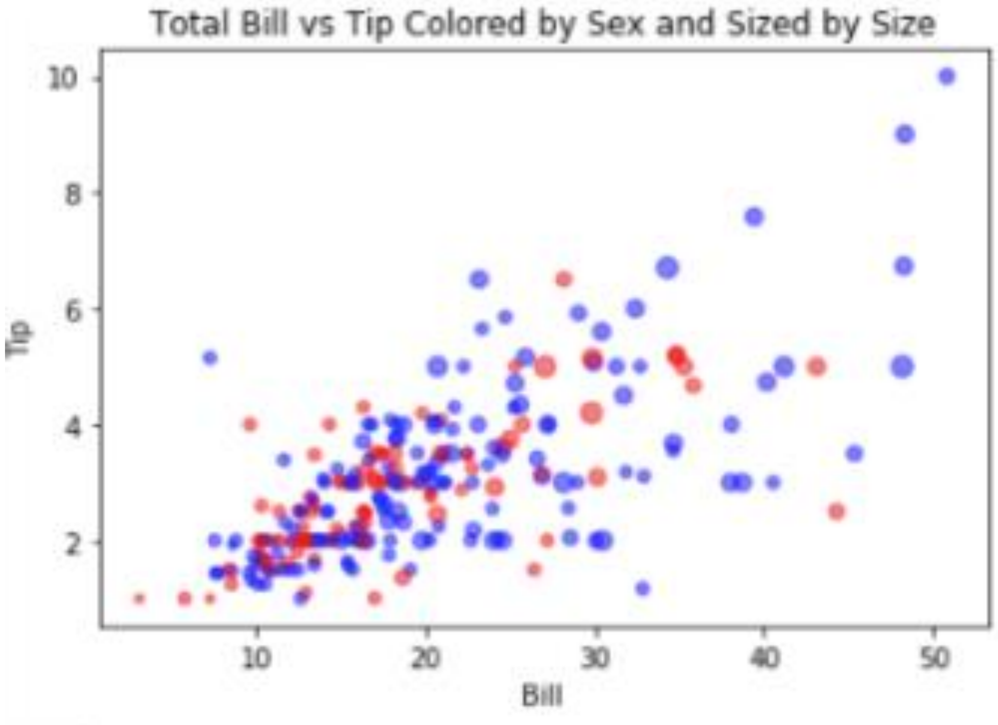
Chapter 1: Pandas Dataframe Basics

3.2 Statistical Graphics for Different Types of Data

- Multivariate - Scatterplot



```
# define a method to set color variable based on sex
def recode_sex(sex):
    if sex == 'Female':
        # red for female
        return 'r'
    else:
        # blue for male
        return 'b'

# create a new column, sex_color, by applying the method
# on the existing column, sex
tips['sex_color'] = tips['sex'].apply(recode_sex)

scatter_plot = plt.figure()
axes2 = scatter_plot.add_subplot(1,1,1)
axes2.scatter(
    x = tips['total_bill'],
    y = tips['tip'],

    # set the size of the bots based on party size
    # we multiply the values by 10 to make the points bigger
    # and to emphasize the difference
    s = tips['size']*10,

    # set the color for the sex
    c = tips['sex_color'],
    # set the alpha value so that the points will be transparent
    # this helps with overlapping points
    alpha=0.5)

axes2.set_title('Total Bill vs Tip Colored by Sex and Sized by Size')
axes2.set_ylabel('Tip')
axes2.set_xlabel('Bill')
scatter_plot.show()
```



SUMMARY

DATA TYPES & STRUCTURE – LEARNING OBJECTIVES

1) Appreciate the features and usage possibilities of the Pandas library as a data analytics package.
2) Understand the basic usage of the Python Pandas library, including loading files, counting data, and determining item structure and types in the data.
3) Learn the basic manipulation of data using Pandas, such as row and column selection, and itemized or vector operations on Pandas DataFrame.
4) Conduct operations on Pandas DataFrame, including subsetting, slicing, and indexing.
5) Present and visualize data in Pandas DataFrame using charting and plotting libraries like Matplotlib and Seaborn.

THANK YOU