



multiverse

Data Science Unit 2

# EDA using Python





## Before we start...

- Make sure you are comfortable
- Have water and maybe a strong coffee handy
- If you need a break... take it!
- If you need a stretch – please go ahead!
- Please mute yourselves if you are not talking
- Have your video on at all times

...and let's get started!





# In this session we will...

- Define what pandas is and how it relates to data science
- Manipulate pandas Dataframes and Series
- Filter and sort data using pandas
- Manipulate DataFrame columns
- Know how to handle null and missing values



multiverse

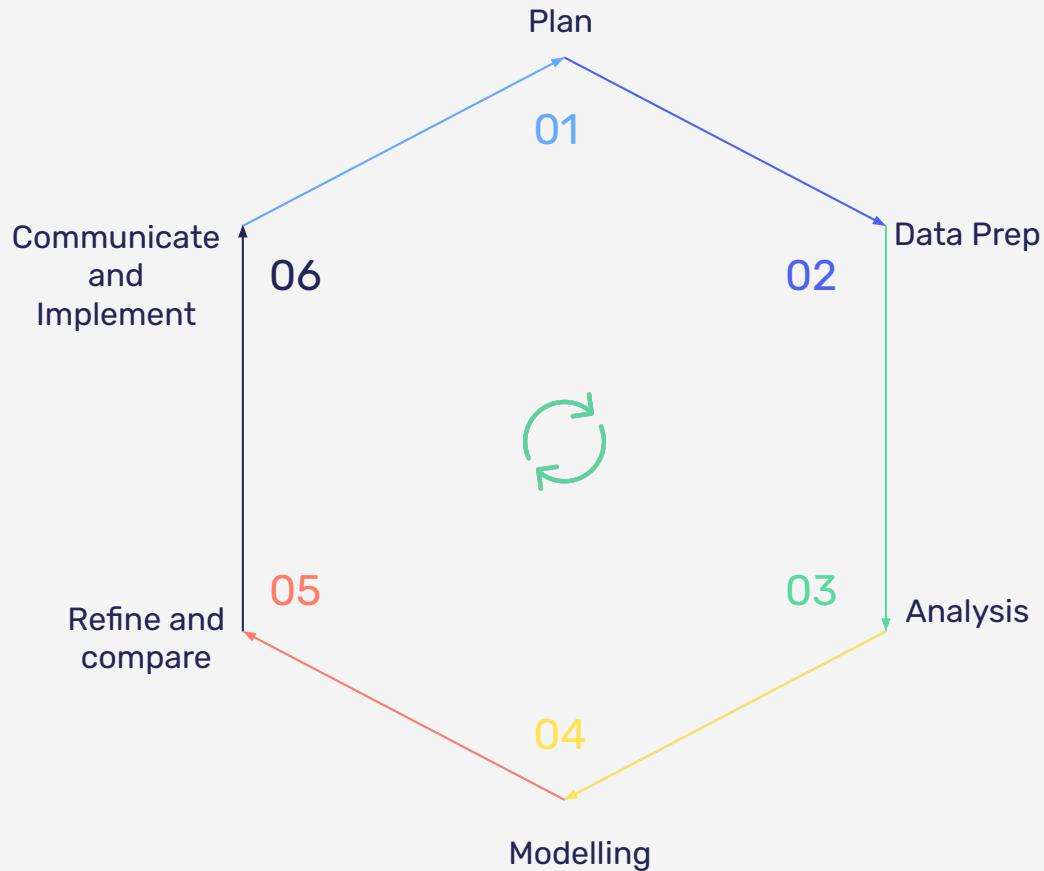


# Exploratory Data Analysis in Python



## Intro

# Recall the Data Analytics Lifecycle





# Libraries

Python Libraries...

- A library is a piece of reusable code
- Each library is centered around a single topic
- A library saves the user time, because it summarises common actions in less code





# Numpy

Compute the mean for **num = [1, 2, 5, 10]**

In Pure Python

```
sum=0  
  
for i in num:  
  
    sum=sum+i  
  
sum/len(num)
```

Using a Library

```
import numpy as np  
np.mean(num)
```

multiverse

# Pandas library





# Pandas

- It's Python gold standard for data analysis
- It Provides built-in data structures which simplify the manipulation and analysis of data sets.

[pandas] is derived from the term "panel data", an econometrics term for data sets that include observations over multiple time periods for the same individuals. — Wikipedia

multiverse



# Importing Pandas



# Importing



```
import pandas as pd
```

multiverse

# The pandas data structures



# Pandas data types

Pandas data types:

- **Series:** one-dimensional arrays holding any type of data (i.e. a column)
- **Dataframes:** two-dimensional array (i.e. a table)



# Series & Dataframes



	Apples
0	3
1	2
2	0
3	1

+

	Oranges
0	0
1	3
2	7
3	2

=

	Apples	Oranges
0	3	0
1	2	3
2	0	7
3	1	2

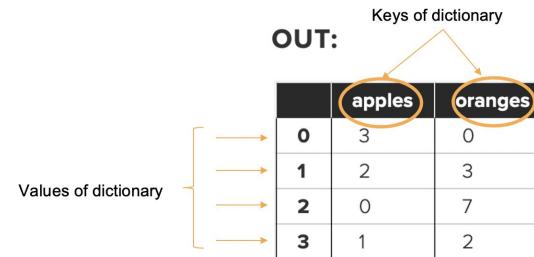


# Series & Dataframes



```
data = {  
    'apples': [3, 2, 0, 1],  
    'oranges': [0, 3, 7, 2]  
}
```

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



# Importing a data file as a dataframe using pandas





## Reading & writing datasets



```
df=pd.read_csv('filename.csv')
```



multiverse

# Dataframe slicing





# Creating subset of dataframe

This is our dummy **df**:

	A	B	C	D
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15
4	16	17	18	19

Select columns using labels

To select a single column.

`df['A']`

or

`df.A`

or

`df.loc[:, 'A']`

Output:

0	0
1	4
2	8
3	12
4	16

To select multiple columns.

`df[['A', 'C']]`

or

`df.loc[:, ['A', 'C']]`

Output:

0	0	2
1	4	6
2	8	10
3	12	14
4	16	18

# Knowledge check

df:

Quarter	Sold
0	Q1 100
1	Q2 120
2	Q3 90
3	Q4 150

What is the **VALUE** and  
**TYPE** of each of the following?

1. df['Quarter']
2. df[ ['Quarter'] ]
3. df['Sold'] < 110
4. df[ df['Sold'] < 110 ]





# Answers



	Quarter	Sold
0	Q1	100
1	Q2	120
2	Q3	90
3	Q4	150

```
df['Quarter']  
0    Q1  
1    Q2  
2    Q3  
3    Q4  
Name: Quarter, dtype: object
```

The output is a Series

```
df[['Quarter']]  
      Quarter  
0      Q1  
1      Q2  
2      Q3  
3      Q4
```

The output is a Dataframe

```
df['Sold'] < 110  
0    True  
1   False  
2    True  
3   False  
Name: Sold, dtype: bool
```

```
df[df['Sold']<110]  
      Quarter  Sold  
0      Q1     100  
2      Q3      90
```



# Dataframe subsets by row

This is our dummy **df**:

	A	B	C	D
0	0	1	2	3
1	4	5	6	7
2	8	9	10	11
3	12	13	14	15
4	16	17	18	19

## Select rows using labels

Select a row by its label

```
df.loc[0]
```

Output:

A	0
B	1
C	2
D	3

Select multiple rows by label.

```
df.loc[[0,1]]
```

Output:

	A	B	C	D
0	0	1	2	3
1	4	5	6	7

Access values by row/column label

```
df.loc[0, 'D']
```

Output:

3
---

# Most important dataframe operations



# Knowledge check

	apples	oranges
June	3	0
Robert	2	3
Lily	0	7
David	1	2



```
purchases.loc['June']
```

Output???

OUT:

```
apples      3
oranges     0
Name: June, dtype: int64
```



2 minutes



# Useful Functions

Function Name	What it does
.head()	Prints first n rows (default 5)
.tail()	Prints last n rows (default 5)
.describe()	Prints summary statistics for each column
.index	Prints indices of dataframe
.columns	Prints column names of dataframe
.dtypes	Prints data types for each column
.shape	Prints number of rows and columns
.info()	Concise summary (use to check for nulls)
.values	Gives values as an array

# Section 1

exploratory-data-analysis-appentice.ipynb





multiverse



# Filtering and Sorting





# Filtering

movies\_df

Title	Rank	Genre	Description	Director
Guardians of the Galaxy	1	Action,Adventure,Sci-Fi	A group of intergalactic criminals are forced ...	James Gunn
Prometheus	2	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott
Split	3	Horror,Thriller	Three girls are kidnapped by a man with a diag...	M. Night Shyamalan
Sing	4	Animation,Comedy,Family	In a city of humanoid animals, a hustling thea...	Christophe Lourdelet
Suicide Squad	5	Action,Adventure,Fantasy	A secret government agency recruits some of th...	David Ayer

Task: Filter our movies DataFrame to show only films directed by Ridley Scott

```
movies_df[movies_df['director'] == "Ridley Scott"]
```

OUT:

Title	rank	genre	description	director	
Prometheus	2	Adventure,Mystery,Sci-Fi	Following clues to the origin of mankind, a te...	Ridley Scott	Noo Rap... Log... Mars... Gree... Mich... Fa...
The Martian	103	Adventure,Drama,Sci-Fi	An astronaut becomes stranded on Mars after hi...	Ridley Scott	Matt Dam... Jess... Cha... Krist... Wiig
Robin Hood	388	Action,Adventure,Drama	In 12th century England, Robin and his band o...	Ridley Scott	Russ... Crov... Cate... Bian... Matt... Mac
American Gangster	471	Biography,Crime,Drama	In 1970s America, a detective works to bring d...	Ridley Scott	Den... Was... Russ... Crov... Chiv... Ej...



# Filtering



```
movies_df[movies_df['rating'] >= 8.6].head(3)
```

OUT:

description	director	actors	year	runtime	rating	
A team of explorers travel through a wormhole ...	Christopher Nolan	Matthew McConaughey, Anne Hathaway, Jessica Ch...	2014	169	8.6	1
When the menace known as the Joker wreaks havo...	Christopher Nolan	Christian Bale, Heath Ledger, Aaron Eckhart,Mi...	2008	152	9.0	1
A thief, who steals corporate secrets through ...	Christopher Nolan	Leonardo DiCaprio, Joseph Gordon-Levitt, Ellen...	2010	148	8.8	1





# Filtering

Find: all movies that were released between **2005 and 2010**, have a **rating above 8.0**, but made **below the 25th percentile in revenue**.



```
movies_df[  
    ((movies_df['year'] >= 2005) & (movies_df['year'] <= 2010))  
    & (movies_df['rating'] > 8.0)  
    & (movies_df['revenue_millions'] < movies_df['revenue_millions'].quantile(0.25))  
]
```

OUT:

actors	year	runtime	rating	votes	revenue_millions
Aamir Khan, Madhavan, Mona Singh, Sharman Joshi	2009	170	8.4	238789	6.52
Ulrich Mühe, Martina Gedeck,Sebastian Koch, Ul...	2006	137	8.5	278103	11.28
Lubna Azabal, Mélissa Désormeaux- Poulin, Maxim...	2010	131	8.2	92863	6.86
Darsheel Safary, Aamir Khan, Tanay Chheda, Sac...	2007	165	8.5	102697	1.20



# Sorting



```
df.sort_values('state')
```

df =

	name	age	state	point
0	Alice	24	NY	64
1	Bob	42	CA	92
2	Charlie	18	CA	70
3	Dave	68	TX	70
4	Ellen	24	CA	88
5	Frank	30	NY	57

	name	age	state	point
1	Bob	42	CA	92
2	Charlie	18	CA	70
4	Ellen	24	CA	88
0	Alice	24	NY	64
5	Frank	30	NY	57
3	Dave	68	TX	70

```
df.sort_values('state', ascending=False)
```

	name	age	state	point
3	Dave	68	TX	70
0	Alice	24	NY	64
5	Frank	30	NY	57
1	Bob	42	CA	92
2	Charlie	18	CA	70
4	Ellen	24	CA	88

multiverse



# Sections 2-4

exploratory-data-analysis-apprentice.





multiverse



# Handling Missing Values





# Missing Values

```
movies_df.isnull()
```

OUT:

Title	rank	genre	description	director	actors	year
<b>Guardians of the Galaxy</b>	False	False	False	False	False	False
<b>Prometheus</b>	False	False	False	False	False	False
<b>Split</b>	False	False	False	False	False	False
<b>Sing</b>	False	False	False	False	False	False
<b>Suicide Squad</b>	False	False	False	False	False	False



```
movies_df.isnull().sum()
```

OUT:

```
rank          0
genre         0
description   0
director      0
actors        0
year          0
runtime       0
rating        0
votes          0
revenue_millions 128
metascore     64
dtype: int64
```

multiverse

# Split-Apply-Combine The groupby function



10 minutes



# Split-apply-combine

DataFrame

	Gender	Height
0	m	172
1	f	171
2	f	169
3	m	173
4	f	170
5	m	175
6	m	178

How can we calculate the mean height of each gender category?



Split

	Gender	Height
0	m	172
1	m	173
2	m	175
3	m	178

Apply

	Gender	Height
	m	174.5

Combine

	Gender	Height
	f	170.0
	m	174.5

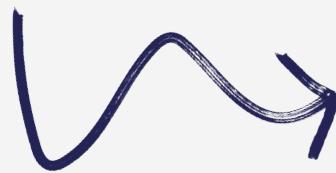
	Gender	Height
0	f	171
1	f	169
2	f	170

	Gender	Height
	f	170.0



# groupby

	Gender	Height
0	m	172
1	f	171
2	f	169
3	m	173
4	f	170
5	m	175
6	m	178



```
df_sample.groupby('Gender').mean()
```

Gender	Height
f	170.0
m	174.5



# groupby



```
Table_name.groupby(['Group'])['Feature'].aggregation()
```

- `Table_name` : this would be the name of the DataFrame, the source of the data you are working on.
- `groupby` : the group by in Python is for sorting data based on different criteria. In this case, the condition is `Group` .
- `Feature` : the part of the data or feature you want to be **inserted** in the computation.
- `aggregation()` : the specific **function name** or aggregation you wish to execute with this operation.



## Section 5

exploratory-data-analysis-appentice.ipynb



multiverse



# The apply function



# Using the apply function

df =

	A	B
0	4	9
1	4	9
2	4	9

```
>>> df.apply(np.sqrt)
      A      B
0  2.0  3.0
1  2.0  3.0
2  2.0  3.0
```

```
>>> df.apply(np.sum, axis=0)
A    12
B    27
dtype: int64
```

```
>>> df.apply(np.sum, axis=1)
0    13
1    13
2    13
dtype: int64
```

np refers to the NumPy library that has inbuilt functions allowing you to perform math operations



# Using the apply function

df =

	height	width
0	40.0	10
1	20.0	9
2	3.4	4

```
def calculate_area(row):  
    return row['height'] * row['width']
```

```
df.apply(calculate_area, axis=1)
```

```
0    400.0  
1    180.0  
2    13.6  
dtype: float64
```



# The lambda function

Regular function



```
def add3(x):  
    return x=x+3
```

Lambda function



```
lambda x: x+3
```

## Regular functions:

- created using the **def** keyword
- can have any number of arguments and any number of expressions
- are generally used for large blocks of code.

## Lambda functions:

- defined using the keyword **lambda**
- can have any number of arguments but only **one** expression
- are generally used for one-line expressions.



# Using the lambda function

```
df['age']=df.apply(lambda x: x['age']+3, axis=1)
```

	id	name	age	income
0	1	Jeremy	20	4000
1	2	Frank	25	7000
2	3	Janet	15	200
3	4	Ryan	10	0
4	5	Mary	30	10000

df =

Output:

	id	name	age	income
0	1	Jeremy	23	4000
1	2	Frank	28	7000
2	3	Janet	18	200
3	4	Ryan	13	0
4	5	Mary	33	10000



# Using the lambda function

**df =**

	<b>id</b>	<b>name</b>	<b>age</b>	<b>income</b>
<b>0</b>	1	Jeremy	20	4000
<b>1</b>	2	Frank	25	7000
<b>2</b>	3	Janet	15	200
<b>3</b>	4	Ryan	10	0
<b>4</b>	5	Mary	30	10000

**Output:**

	<b>id</b>	<b>name</b>	<b>age</b>	<b>income</b>	<b>catagory</b>
<b>0</b>	1	Jeremy	23	4800	Adult
<b>1</b>	2	Frank	28	8400	Adult
<b>2</b>	3	Janet	18	240	Adult
<b>3</b>	4	Ryan	13	0	child
<b>4</b>	5	Mary	33	12000	Adult

```
df['category']=df['age'].apply(lambda x: 'Adult' if x>=18 else 'Child')
```



## Cheat sheets:



- <https://www.dataquest.io/blog/pandas-cheat-sheet/>
- <http://datacamp-community-prod.s3.amazonaws.com/dbed353d-2757-4617-8206-8767ab379ab3>



## Additional practise



<https://pynative.com/python-pandas-exercise/>



multiverse



# Summary



# Summary

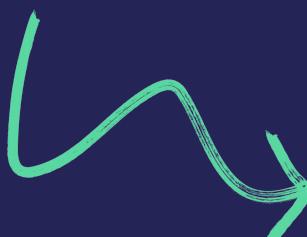
The most important things to familiarize yourself with are the basics:

- Manipulating Dataframes and Series
- Filtering columns and rows
- Handling missing values
- Split-apply-combine (this one takes some practice!)

The background features abstract teal brushstrokes on a dark navy blue surface. A large, horizontal brushstroke near the bottom center has a textured, layered appearance. Another smaller, curved brushstroke is located in the upper right corner.

multiverse

# Thank you

A thick, teal brushstroke arrow points from the bottom left towards the text on the right.

**Get in touch**  
[info@multiverse.io](mailto:info@multiverse.io)