# Introduction to Python

Professor: Horacio Larreguy
TA: Eduardo Zago

ITAM Applied Research 2 / Economics Research Seminar,
11/01/2023

## General Perspective

## Downloading Python and Setting up the Environment

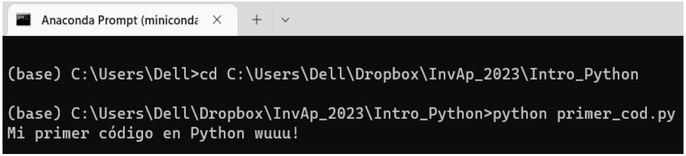
- To start using Python, we must follow a number of steps:
    1. Download Python.exe in your computer (notice it depends which operating system you have)
    2. Once that is installed, we must set up an Environment. The easiest one is downloading Mini Conda
    3. After the download is complete, run the installer and click through the setup steps leaving all the pre-selected installation defaults.
    4. Once complete, check that Miniconda was installed correctly by opening a Command Prompt (CMD or PowerShell for Windows) or a Terminal (for Mac) and entering the command 'conda list'

## Python Code From the Terminal

- Once this is downloaded, go to *Anaconda Prompt (miniconda3)* and we can start using Python from the terminal.
- Notice the file in the DropBox Folder I sent you. We'll use the `print()` function:

```
# Changing the Directory (check yours):
cd \Dropbox\InvAp_2023\Intro_Python
# Call the py file
python primer_cod.py
```

Figure: Running Code from the Terminal (Output)



```
(base) C:\Users\Dell>cd C:\Users\Dell\Dropbox\InvAp_2023\Intro_Python

(base) C:\Users\Dell\Dropbox\InvAp_2023\Intro_Python>python primer_cod.py
Mi primer código en Python wuuu!
```

## Jupyter Notebook

- Although running code from the terminal is useful, we will start using a more friendly code compiler: Jupyter Notebook

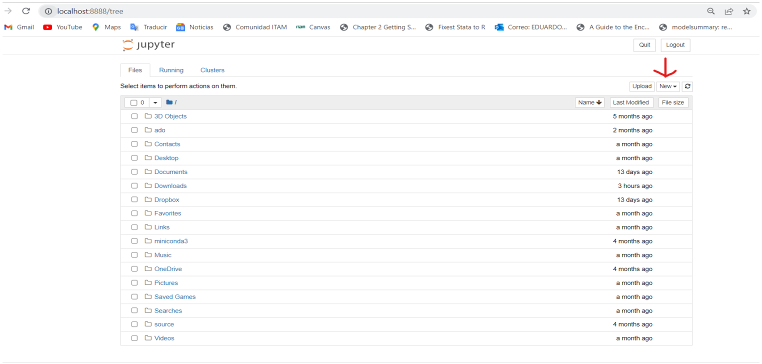
```
# Install Jupyter Notebook using pip:
pip install jupyter
```

```
# Run it in your terminal:
jupyter notebook
```

- You will get the following page browser:

# Jupyter Notebook

Figure: Jupyter Notebook's Main Menu



localhost:8888/tree

Gmail YouTube Maps Traducir Noticias Comunidad ITAM Canvas Chapter 2 Getting S... Fixest Stata to R Correo: EDUARDO... A Guide to the Enc... modelsummary: re...

jupyter Quit Logout

Files Running Clusters

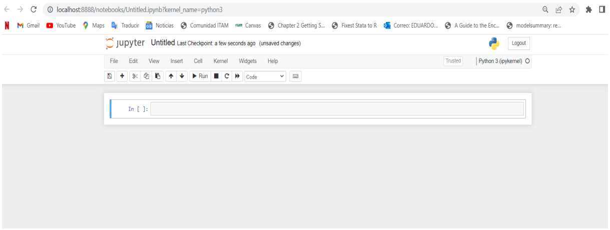
Select items to perform actions on them. Upload New

| Name | Last Modified | File size |
|---|---|---|
| 3D Objects | 5 months ago | |
| ado | 2 months ago | |
| Contacts | a month ago | |
| Desktop | a month ago | |
| Documents | 13 days ago | |
| Downloads | 3 hours ago | |
| Dropbox | 13 days ago | |
| Favorites | a month ago | |
| Links | a month ago | |
| miniconda3 | 4 months ago | |
| Music | a month ago | |
| OneDrive | 4 months ago | |
| Pictures | a month ago | |
| Saved Games | a month ago | |
| Searches | a month ago | |
| source | 4 months ago | |
| Videos | a month ago | |

- To start writing code, we must do so in an .ipynb file. To create one select *New > Python 3 (ipykernel)* on your desired folder.

# Jupyter Notebook

Figure: Python 3 Notebook



- Now we can start writing code and running it in a more friendly interface. For the whole tutorial, we will be using Python 3 Notebooks to run our code.
- More references: Visual Studio Code, CodeCademy: How to install Conda

## Installing Packages

- Finally, to correctly set up the Environment, we need to download the necessary packages. In this case we will be using the function `!pip`, as follows:

```
# Installing packages using pip:
!pip install matplotlib
!pip install seaborn
```

- We must load them to the kernel to use their functions:

```
# Load the packages to the session:
import matplotlib.pyplot as plt
import seaborn as sns
```

## pandas Package

- pandas is a fast, flexible and easy to use open source data analysis and manipulation tool.
- It is a DataFrame object for data manipulation with integrated indexing, tools for reading and writing data, flexible reshaping, merging, and intelligent data alignment.
- This is the go to package for data manipulation and cleaning in Python, such as dplyr is for R.
- To install it and load it use the following command:

```
# Install it
!pip install pandas

# Load the packages to the session:
import pandas as pd
```

## NumPy Package

- NumPy is a Python library used for working with arrays.
- It also has functions for working in domain of linear algebra, fourier transform, and matrices.
- NumPy arrays are stored at one continuous place in memory unlike lists, so processes can access and manipulate them very efficiently.
- To install it and load it use the following command:

```
# Install it
!pip install numpy

# Load the packages to the session:
import numpy as np
```

## Calculator and Locals

- Python can be used to perform operations as in a calculator.

```python
# Sum
2 + 2
### Returns: 4
```

- Or save locals and perform operations on these variables:

```python
# Define locals:
a = 5
b = 10
# Perform operation, save it in another local
c = a + b
# Print it
print(c)
### Returns: 15
```

## Local Strings

- It will prove very useful to learn how to work with local strings, here are a few examples:

```
# Define local strings:
b1 = 'Hola'
b2 = 'Cómo estas?'

# Paste them and save it in another local
c1 = b1 + ' ' + b2
print(c1)
### Returns: 'Hola Cómo estas?'
```

```
# Useful when we want to work with relative paths:
base = '../../data/baseline/'
path_in = base + 'in/'
path_out = base + 'out/'

print(path_in, path_out)
### Returns: ../../data/baseline/in/ ../../data/baseline/out/
```

## Local Strings

- To import data, we might want to do it in an automated way, thus using the function `f''` is very useful:

```
# Local variables
country1 = 'Argentina'
country2 = 'Mexico'

# Concatenated paths
base = f'../../data/baseline/{country1}/'
base2 = f'../../data/baseline/{country2}/'
path_in = base + 'in/'
path_out = base2 + 'out/'

print(path_in, path_out)
### Returns: ../../data/baseline/Argentina/in/
### ../../data/baseline/Mexico/out/
```

## Creating Lists

- There are several objects that store information in Python, the most important ones being: data frames, dictionaries and lists.
- Lists help store 1-dimensional vectors of relevant information and are the easiest ones to loop on.
- There are several ways of defining a list:

```python
# Define a list from scratch:
a = [1, 2, 3, 8, 9, 10, 20]

# Define an integer list for a certain range
b = range(3, 10) # From 3 to 9, increments of 1
c = range(3, 10, 2) # Increments of 2

# From a data frame column using pandas:
list_frame = df['ids'].tolist()
```

## Manipulating Lists

- We can manipulate lists by adding elements, removing, concatenating with other lists, etc.

```python
# Extracting specific elements:
a[0] # First element
a[-1] # Last element

# Subsetting lists
a[1:n] # from position 1 to n-1 [,)

# Getting the list's length
len(a)

# Appending new elements at the end of a list:
a.append('Hola')

# Removing elements by name from the list:
a.remove('Hola')
```

## Functions

- The Python function syntaxis is the following:

```
def function1(input1, input2, ..., inputn):
     output1 = f(input1, ..., inputn)
     output2 = f(output1, input1, ..., inputn)
     .
     .
     .
     outputn = f(output1, ..., output(n-1), input1, ..., inputn)
 return output1, ..., outputn
```

- To call the function, you must provide the same amount of names as there are outputs:

```
df1, df2, ..., dfn = function1(input1, ..., inputn)
```

## Function Examples:

```python
# Prints and stores the output in a
def my_second_function(word1, word2):
    word = word1 + ' ' + word2
    print(word)
return word
a = my_second_function('Hola', 'cómo estas?')
# Input: Two Strings, Output: 1 String
```

```python
def get_path(country):
    base = f'../../data/{country}/baseline/'
    path_in = base + 'in/'
    path_out = base + 'out/'
return base, path_in, path_out

base, path_in, path_out = get_path('Argentina')
print(base, path_in, path_out)

### Returns: ../../data/Argentina/baseline/
### ../../data/Argentina/baseline/in/
### ../../data/Argentina/baseline/out/
```

## For Loops

- For loops in Python are relatively easier to understand than in R. Their syntaxis is much more friendly.

```
# Prints from 1 to 10
for i in range(1, 10):
    print(i)
```

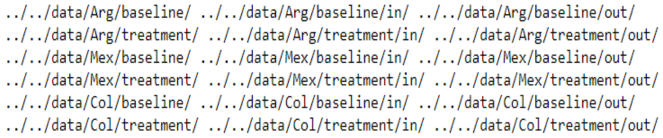
```
# We could loop through lists and use functions inside the loop:
list_loop = ['Arg', 'Mex', 'Col']
for country in list_loop:
    base, path_in, path_out = get_path(country)
    print(base, path_in, path_out)

### Returns: ../../data/Arg/baseline/ ../../data/Arg/baseline/in/
### ../../data/Arg/baseline/out/
### ../../data/Mex/baseline/ ../../data/Mex/baseline/in/
### ../../data/Mex/baseline/out/
### ../../data/Col/baseline/ ../../data/Col/baseline/in/
### ../../data/Col/baseline/out/
```

## Nested For Loops

- And more importantly, we could easily do nested for loops.

```
list_loop = ['Arg', 'Mex', 'Col']
type_of_data = ['baseline', 'treatment']

def get_path2(country, type_d):
    base = f'../../data/{country}/{type_d}/'
    path_in = base + 'in/'
    path_out = base + 'out/'
return base, path_in, path_out

for country in list_loop:
    for type_d in type_of_data:
        base, path_in, path_out = get_path2(country, type_d)
        print(base, path_in, path_out)
```

Figure: Output Loop 1

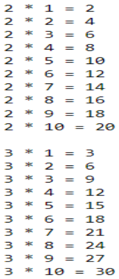


```
../../data/Arg/baseline/ ../../data/Arg/baseline/in/ ../../data/Arg/baseline/out/
../../data/Arg/treatment/ ../../data/Arg/treatment/in/ ../../data/Arg/treatment/out/
../../data/Mex/baseline/ ../../data/Mex/baseline/in/ ../../data/Mex/baseline/out/
../../data/Mex/treatment/ ../../data/Mex/treatment/in/ ../../data/Mex/treatment/out/
../../data/Col/baseline/ ../../data/Col/baseline/in/ ../../data/Col/baseline/out/
../../data/Col/treatment/ ../../data/Col/treatment/in/ ../../data/Col/treatment/out/
```

## Nested For Loops Examples

```
# More Nested Loops:
for i in range(2, 4):
    # Printing inside the outer loop
    # Running inner loop from 1 to 10
    for j in range(1, 11):

        # Printing inside the inner loop
        print(i, "*", j, "=", i*j)
    # Printing inside the outer loop
    print()
```

Figure: Output Loop 2

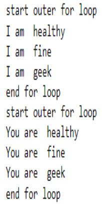


```
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
2 * 10 = 20

3 * 1 = 3
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
3 * 7 = 21
3 * 8 = 24
3 * 9 = 27
3 * 10 = 30
```

## Nested For Loops Examples

```python
# Initialize list1 and list2 with some strings
list1 = ['I am ', 'You are ']
list2 = ['healthy', 'fine', 'geek']
list2_size = len(list2) # length list 2

# Running outer for loop to iterate through a list1.
for item in list1:
    print("start outer for loop ") # Printing outside inner loop
    i = 0 # Initialize counter i with 0
    # Running inner While loop to iterate through list2.
    while(i < list2_size):
        print(item, list2[i]) # Printing inside inner loop
        i = i+1 # Incrementing the value of i
    # Printing outside inner loop
    print("end for loop ")
### See the example code for the output
```

Figure: Output Loop 2



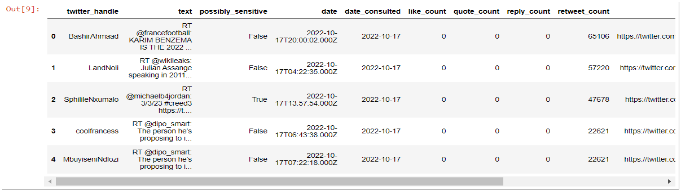
```
start outer for loop
I am  healthy
I am  fine
I am  geek
end for loop
start outer for loop
You are  healthy
You are  fine
You are  geek
end for loop
```

## Importing Data

- Notice we have already imported the pandas package as pd, so to import data we can just use the functions available.
- We will start using relative paths with respect to where the Notebook is saved.

```
# Read excel using pandas
df1 = pd.read_excel('../data/twitter.xlsx')
# And we can visualize it
df1.head()
```

Figure: Pandas Data Frame



Out[9]:

| | twitter_handle | text | possibly_sensitive | date | date_consulted | like_count | quote_count | reply_count | retweet_count | |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | BashirAhmaad | RT @francefootball: KARIM BENZEMA IS THE 2022 ... | False | 2022-10-17T20:00:02.000Z | 2022-10-17 | 0 | 0 | 0 | 65106 | https://twitter.com |
| 1 | LandNoli | RT @wikileaks: Julian Assange speaking in 2011... | False | 2022-10-17T04:22:35.000Z | 2022-10-17 | 0 | 0 | 0 | 57220 | https://twitter.cor |
| 2 | SphilileNxumalo | RT @michaelb4jordan: 3/3/23 #creed3 https://t.... | True | 2022-10-17T13:57:54.000Z | 2022-10-17 | 0 | 0 | 0 | 47678 | https://twitter.c |
| 3 | coolfrancess | RT @dipo_smart: The person he's proposing to i... | False | 2022-10-17T06:43:38.000Z | 2022-10-17 | 0 | 0 | 0 | 22621 | https://twitter.cc |
| 4 | MbuyiseniNdlozi | RT @dipo_smart: The person he's proposing to i... | False | 2022-10-17T07:22:18.000Z | 2022-10-17 | 0 | 0 | 0 | 22621 | https://twitter.cc |

## Exporting Data

- To export data we do this directly on a data frame.
- We can generate folders directly from Python using `os`

```
import os # for folder/path manipulation

# Function to create folders using os:
def create_folder(directory):
    """Create folder"""
    # Create target Directory if don't exist
    if not os.path.exists(directory):
        os.mkdir(directory)
        print("Directory ", directory, " created ")
    else:
        print("Directory ", directory, " already exists")

create_folder('../data/out/')
# Saving as parquet
df1.to_parquet('../data/out/twitter.parquet.gzip')
# Saving as Excel
df1.to_excel('../data/out/twitter.xlsx')
```

## Looping through several Folders

- Sometimes the information required is in distinct folders with different names. To import and export this data in an efficient way we could use a loop that goes through the paths:

```
for i in range(1, 3):
    folder_out = f'../data/batch{i}/'
    create_folder(folder_out)
    for e in range(1, 4):
        df = pd.read_excel(f'../data/batch{i}/info{e}.xlsx')
        df.to_parquet(f'{folder_out}info{e}.parquet.gzip')
    print('Done')
```

# Manipulating Data Frames

- Pandas package is the dplyr of Python. With this package we are going to be able to rename, select, drop, create columns, as well as filter our DFs.
- Here are a few important functions:

```python
# Renaming:
df.rename({'twitter_handle': 'username',
    'possibly_sensitive': 'sensitive'},
    axis=1, inplace=True)

# Selecting variables:
df_info = df[['username', 'date', 'treatment']]

# Dropping variables:
df_date = df_info.drop(['username'], axis='columns')

# Creating new variables:
df['int_divided'] = df['total_interactions']/2
```

# Pandas Functions

```
# Cool way to generate a list of variables with the same prefix, suffix
metrics = [col for col in df.columns if '_count' in col]

# Summing the columns (you can do .mean(), etc.)
df['total_ints_2'] = df[metrics].sum(axis=1)
# axis = 1 is axis = 'columns'

# If else in Python using numpy
df['pairs'] = np.where((df.index+1)%2==0, ((df.index+1)/2),
    ((df.index+2)/2)).astype(str)
```

## Pandas Functions: Filtering Data

```
# Duplicates drop:
df_new = df.drop_duplicates('username')

# Filtering:
df_new = df[df['like_count'] > 0].reset_index(drop=True)

# Filtering (more than one condition):
df_new = df[(df['like_count'] > 110) &
    (df['quote_count'] > 10)].reset_index(drop=True)

# Filtering a column by the elements in a list
exclude_usernames = ['AfricaFactsZone', 'renoomokri', 'citizentvkenya']
df_new = df[(~df['username'].isin(exclude_usernames)) &
    (df['like_count'] > 110) &
    (df['quote_count'] > 10)].reset_index(drop = True)
```

## Pandas Functions: Merging Data

```
# Generating a random data frame so we can merge it
import random

df.rename({'treatment': 'post_treatment'}, axis=1, inplace=True)
df_treat = df[['username']].reset_index(drop=True)
df_treat = df_treat.drop_duplicates()
df_treat['user_treatment'] = [random.randint(0, 1)
    for k in df_treat.index]

# Merging with the original data set:
df = df.merge(df_treat, on='username', how='left')
```

## Pandas Functions: Summarizing Data

```
# Getting the mean of a certain variable:
df['retweet_count'].mean()
### Returns: 2848.027932960894

# Or do a table with these locals:
data_summ = {'Type of Interaction': ['RTs', 'Likes', 'Quotes', 'Replies'],
'Mean': [df['retweet_count'].mean(), df['like_count'].mean(),
        df['quote_count'].mean(), df['reply_count'].mean()],
"SD": [df['retweet_count'].std(), df['like_count'].std(),
        df['quote_count'].std(), df['reply_count'].std()],
"Min.": [df['retweet_count'].min(), df['like_count'].min(),
        df['quote_count'].min(), df['reply_count'].min()],
"Max.": [df['retweet_count'].max(), df['like_count'].max(),
        df['quote_count'].max(), df['reply_count'].max()]}

data_summ = pd.DataFrame(data_summ)
print(data_summ.to_latex()) # To LaTeX
```

| | Type of Interaction | Mean | SD | Min. | Max. |
|---|---|---|---|---|---|
| 0 | RTs | 2848.027933 | 7966.844376 | 29 | 65106 |
| 1 | Likes | 391.955307 | 883.994959 | 0 | 7166 |
| 2 | Quotes | 8.240223 | 25.657716 | 0 | 269 |
| 3 | Replies | 52.664804 | 139.827058 | 0 | 1196 |

Table: Summary Statistics

## Pandas Functions: Summarizing Data

```
# We can get grouped summaries:
metrics = ['retweet_count', 'like_count', 'quote_count', 'reply_count']
grouped_summ = df[['post_treatment'] +
        metrics].groupby('post_treatment').mean()

# And export to LaTeX:
print(grouped_summ.to_latex())
```

| post_treatment | retweet_count | like_count | quote_count | reply_count |
|---|---|---|---|---|
| 0 | 2237.584615 | 840.646154 | 11.707692 | 112.569231 |
| 1 | 3196.087719 | 136.122807 | 6.263158 | 18.508772 |

Table: Grouped Summary Statistics

## Example using For Loops and pandas

- Look at the files located in the data folder: notice we have two folders containing three Excel files each.
- We would like to loop through these files and append them one in top of the other, this is what we call aggregating data into one file.
- We would also like to add columns to each file while appending, to do this we can use the pandas package.

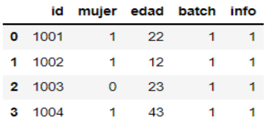
```python
# Example with one file:
example = pd.read_excel('../data/batch1/info1.xlsx')

# Add the batch
example['batch'] = 1

#Add the info:
example['info'] = 1
```

## Example using For Loops and pandas

- We can build a function to use it in a loop:

```
def read_info(batch_num, info_num):
    df = pd.read_excel(f'../data/batch{batch_num}/info{info_num}.xlsx')
    df['batch'] = batch_num
    df['info'] = info_num
return df

# Let's see the output:
example = read_info(1,1)
example
```

Figure: Data Frame Example



| | id | mujer | edad | batch | info |
|---|---|---|---|---|---|
| **0** | 1001 | 1 | 22 | 1 | 1 |
| **1** | 1002 | 1 | 12 | 1 | 1 |
| **2** | 1003 | 0 | 23 | 1 | 1 |
| **3** | 1004 | 1 | 43 | 1 | 1 |

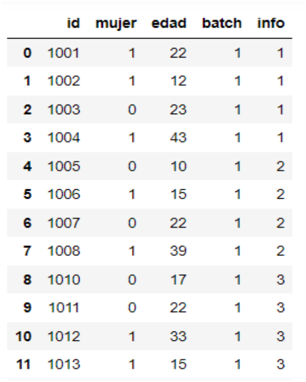
## Example using For Loops and pandas

- Finally we loop through the paths:

```
df_final = pd.DataFrame() # Initialize final df
for i in range(1, 3):
    df_int = pd.DataFrame() # Initialize intermediate df for the inner loop
    for e in range(1, 4):
        df = read_info(i, e)
        df_int = df_int.append(df)
    df_final = df_final.append(df_int)

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

- And we get the following output:

# Example using For Loops and pandas



|    | id   | mujer | edad | batch | info |
|----|------|-------|------|-------|------|
| 0  | 1001 | 1 | 22 | 1 | 1 |
| 1  | 1002 | 1 | 12 | 1 | 1 |
| 2  | 1003 | 0 | 23 | 1 | 1 |
| 3  | 1004 | 1 | 43 | 1 | 1 |
| 4  | 1005 | 0 | 10 | 1 | 2 |
| 5  | 1006 | 1 | 15 | 1 | 2 |
| 6  | 1007 | 0 | 22 | 1 | 2 |
| 7  | 1008 | 1 | 39 | 1 | 2 |
| 8  | 1010 | 0 | 17 | 1 | 3 |
| 9  | 1011 | 0 | 22 | 1 | 3 |
| 10 | 1012 | 1 | 33 | 1 | 3 |
| 11 | 1013 | 1 | 15 | 1 | 3 |

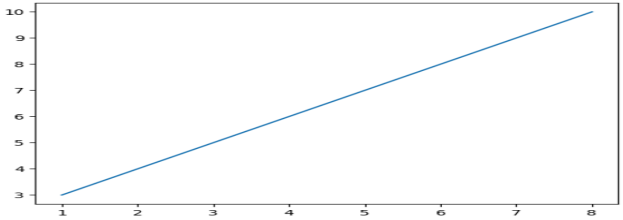
Figure: Data Frame Example

## Matplotlib: Line

- The go to package for graphing in Python is Matplotlib.pyplot

```
xpoints = np.array([1, 8])
ypoints = np.array([3, 10])

plt.plot(xpoints, ypoints)
plt.show()
```

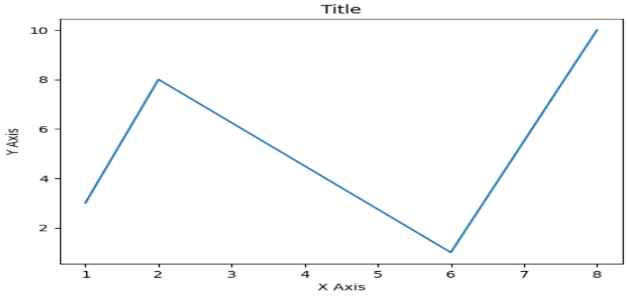
Figure: Line



## Matplotlib: Titles and Axis

```
xpoints = np.array([1, 2, 6, 8])
ypoints = np.array([3, 8, 1, 10])
plt.plot(xpoints, ypoints)
plt.title('Title')
plt.xlabel("X Axis")
plt.ylabel("Y Axis")
plt.show()
```

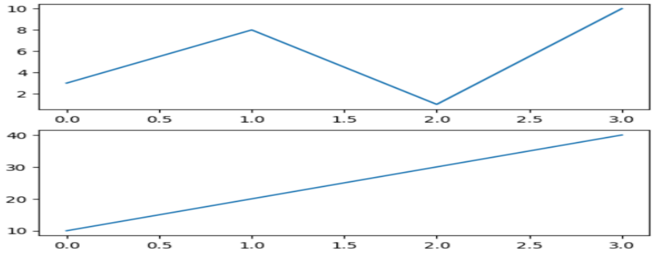
Figure: Titles and axis



# Matplotlib: Subplots

```
#plot 1:
x = np.array([0, 1, 2, 3])
y = np.array([3, 8, 1, 10])
plt.subplot(2, 1, 1)
plt.plot(x,y)

#plot 2:
x = np.array([0, 1, 2, 3])
y = np.array([10, 20, 30, 40])
plt.subplot(2, 1, 2)
plt.plot(x,y)
plt.show()
```

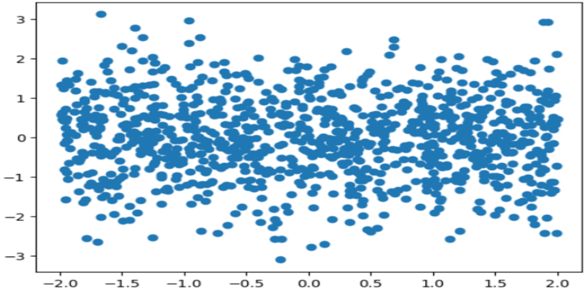
Figure: Subplots



## Matplotlib: Scatter Plots

```
# We generate two vectors of random variables (uniform and normally distributed)
from numpy import random
y = random.normal(loc=0, scale=1, size=(1000))
x = random.uniform(-2, 2, size=(1000))

plt.scatter(x, y)
plt.show()
```

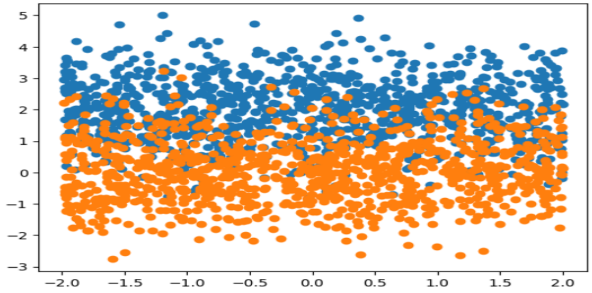
Figure: Scatter Plot



## Matplotlib: Scatter Plots

```
# We can also graph two data points that come from different distributions
y = random.normal(loc=0, scale=1, size=(1000))
x = random.uniform(-2, 2, size=(1000))
y2 = random.normal(2, 1, size = (1000))

plt.scatter(x,y2)
plt.scatter(x, y)
plt.show()
```

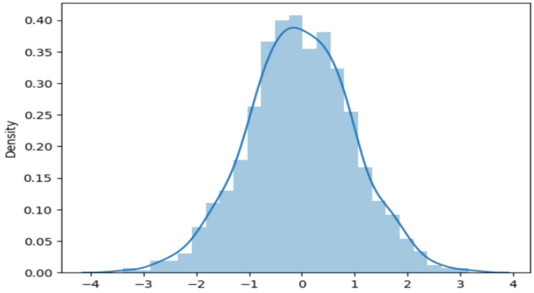
Figure: Mixed Scatter Plot



## Matplotlib: Distributions

```
y = random.normal(loc=0, scale=1, size=(1000))
x = random.uniform(-2, 2, size=(1000))
y2 = random.normal(2, 1, size = (1000))

import seaborn as sns

sns.distplot(y, hist=True)
```

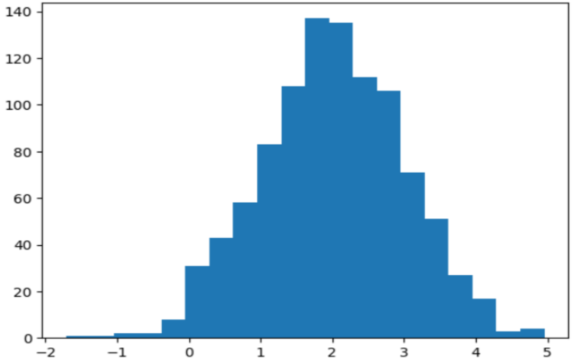
Figure: Normal Distibution



## Matplotlib: Histogram

```
y2 = random.normal(2, 1, size = (1000))
plt.hist(y2, bins = 20)
plt.show()
```

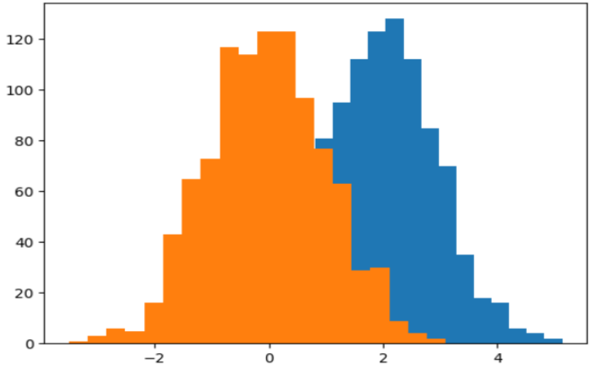
Figure: Histogram



## Matplotlib: Histogram

```
y = random.normal(loc=0, scale=1, size=(1000))
y2 = random.normal(2, 1, size = (1000))

plt.hist(y2, bins = 20)
plt.hist(y, bins = 20)
plt.show()
```

Figure: Histogram



## What is the Twitter API?

- According to Amazon AWS, APIs are mechanisms that enable two software components to communicate with each other using a set of definitions and protocols.
- API architecture is usually explained in terms of client and server. The application sending the request is called the client, and the application sending the response is called the server.
- We will be the clients, doing calls to retrieve info from Twitter, and Twitter will be the server, providing it.

## What is the Twitter API?

- To gain access to the API one must request authentication through the Developer Portal. The most common (for our purposes) access granted is the Academic Research Access. This allows:
  1. Retrieve 10 million Tweets per month
  2. Access to full-archive search and full-archive Tweet counts
  3. Access to advanced search operators
- By endpoint we refer to distinct types of calls (requests) we would like to make to the API to obtain information, or even to perform automated actions such as Tweeting.

## Twitter API in Python: Tweetple

- In this section we are going to take a look at how to access the Twitter API to retrieve relevant social media data such as Tweets, Likes, Followers, Followings, etc. of relevant users.
- To do this we will use the Tweetple package made by ITAM ex-student Daniela Pinto Veizaga.
- To install it, run one of these two commands:

```
# Installing tweetple
!pip install tweetple
# If the above does not work, use:
!pip install tweetple==0.92
```

## Get Profile Information

- When dealing with user IDs, one might want to know their username and the number of interactions they have without having to download all of their data.
- To do this we can use the Get Profile endpoint.

```
bearer_token='####'

# List of handle ids
ids = ['308131814']

# Retrieve users' information
TweetPle.TweepleStreamer(ids, bearer_token).user_lookup()
```

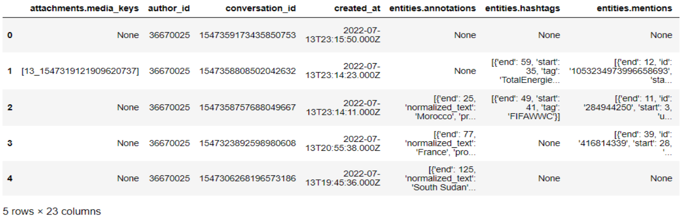
## Get User Timeline

- Users in Twitter are distinguished by two unique identifiers: Twitter handle (@username) and ID.
- If we wanted to obtain the Tweets from a certain list of usernames, from a certain time frame, we would use the following endpoint (User Tweet Timeline)

```
# Load the package:
import tweetple
from tweetple import TweetPle

bearer_token = '#########'
# Date start
start =  '2019-01-01T00:00:00Z'
# Date end
end = '2020-07-01T00:00:00Z'
# IDS:
ids = ['lalozcc', 'j_barrutia']

TweetPle.TweetStreamer(ids, bearer_token, path_save = '../data/your_folder/',
        start_time = start, end_time = end).main()
```

## Output Timeline

Figure: IDs and Date



| | attachments.media_keys | author_id | conversation_id | created_at | entities.annotations | entities.hashtags | entities.mentions |
|---|---|---|---|---|---|---|---|
| **0** | None | 36670025 | 1547359173435850753 | 2022-07-13T23:15:50.000Z | None | None | None |
| **1** | [13_1547319121909620737] | 36670025 | 1547358808502042632 | 2022-07-13T23:14:23.000Z | None | [{'end': 59, 'start': 35, 'tag': 'TotalEnergie... | [{'end': 12, 'id': '1053234973996658693', 'sta... |
| **2** | None | 36670025 | 1547358757688049667 | 2022-07-13T23:14:11.000Z | [{'end': 25, 'normalized_text': 'Morocco', 'pr... | [{'end': 49, 'start': 41, 'tag': 'FIFAWWC'}] | [{'end': 11, 'id': '284944250', 'start': 3, 'u... |
| **3** | None | 36670025 | 1547323892598980608 | 2022-07-13T20:55:38.000Z | [{'end': 77, 'normalized_text': 'France', 'pro... | None | [{'end': 39, 'id': '416814339', 'start': 28, '... |
| **4** | None | 36670025 | 1547306268196573186 | 2022-07-13T19:45:36.000Z | [{'end': 125, 'normalized_text': 'South Sudan'... | None | None |

5 rows × 23 columns

- You also get other variables such as: Tweet ID, number of likes, RTs, replies, and quotes, the text, the username, etc.

## Get Followers Endpoint

- We can also get the followers for a certain user using the Get Followers endpoint:
- This endpoint has a bug, so you must verify that all the ids have been correctly downloaded.

```
bearer_token = '#####'
# IDS:
ids = ['lalozcc', 'j_barrutia']

TweetPle.TweepleStreamer(ids, bearer_token,
        path_save = '../data/your_folder/').followers_lookup()
```

## Getting Tweet Information: Threads, Likes and RTs

- Each Tweet is uniquely identified by an id. Also, each Tweet has a conversation ID, which all Tweets that are replies to the first Tweet have as well.
- While analyzing the Twitter data, one could be interested in seeing who is interacting with who, to do this one must know who is retweeting, replying and liking the Tweets.

```python
# Likes:
TweetPle.TweepleStreamer(ids_tweets, bearer_token,
        path_save='../data/your_folder/').likes_lookup()

# Retweets:
TweetPle.TweepleStreamer(ids_tweets, bearer_token,
        path_save='../data/your_folder/').retweet_lookup()

# Replies (Threads)
TweetPle.get_threads(conversation_ids, bearer_token,
        path_save='../data/your_folder/')
```

## Twitter API in R: academictwitteR

- Unfortunately, Tweetple does not have functions to access all the endpoints available by the Twitter API.
- For the ones not available in Tweetple, we could use an R package called academictwitteR.
- In this package we will find useful functions such as Search Full Archive, Get User Likes, Get User Followings and many more.
- To install the package in R run the following command:

```
install.packages("academictwitteR")
```

## Search Full Archive

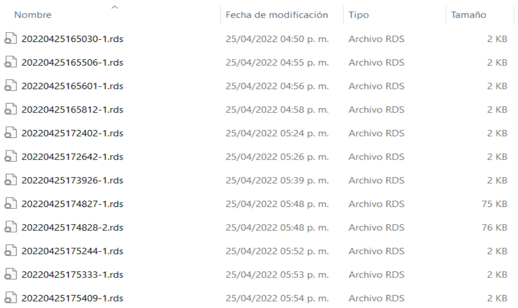
- The most useful function included in this package is the Search Full Archive one.
- To analyze Twitter Trends, one would certainly find it useful scrapping Tweets that have certain word or words in it (query search). To do so we could use `get_all_tweets()` function:

```
tweets <- get_all_tweets(query = "AMLO", # word or words c() to filter for
            start_tweets = "2022-04-01T00:00:00Z",
            end_tweets = "2022-04-02T00:00:00Z",
            data_path = path, # set a path so that the function downloads them gradually
            bind_tweets = FALSE, # Get compiled tweets
            n= 1000 # Number of tweets
            )

# We can later bind them
tweets <- bind_tweets(data_path = path)
```

## Search Full Archive: Output

- When we use this function, we get the following output on the desired folder, we must bind them:

Figure: Output Search Full Archive



| Nombre | Fecha de modificación | Tipo | Tamaño |
|---|---|---|---|
| 20220425165030-1.rds | 25/04/2022 04:50 p. m. | Archivo RDS | 2 KB |
| 20220425165506-1.rds | 25/04/2022 04:55 p. m. | Archivo RDS | 2 KB |
| 20220425165601-1.rds | 25/04/2022 04:56 p. m. | Archivo RDS | 2 KB |
| 20220425165812-1.rds | 25/04/2022 04:58 p. m. | Archivo RDS | 2 KB |
| 20220425172402-1.rds | 25/04/2022 05:24 p. m. | Archivo RDS | 2 KB |
| 20220425172642-1.rds | 25/04/2022 05:26 p. m. | Archivo RDS | 2 KB |
| 20220425173926-1.rds | 25/04/2022 05:39 p. m. | Archivo RDS | 2 KB |
| 20220425174827-1.rds | 25/04/2022 05:48 p. m. | Archivo RDS | 75 KB |
| 20220425174828-2.rds | 25/04/2022 05:48 p. m. | Archivo RDS | 76 KB |
| 20220425175244-1.rds | 25/04/2022 05:52 p. m. | Archivo RDS | 2 KB |
| 20220425175333-1.rds | 25/04/2022 05:53 p. m. | Archivo RDS | 2 KB |
| 20220425175409-1.rds | 25/04/2022 05:54 p. m. | Archivo RDS | 2 KB |

```r
# We can bind them using this function
tweets <- bind_tweets(data_path = path)
```

## Search Full Archive: Complex Queries

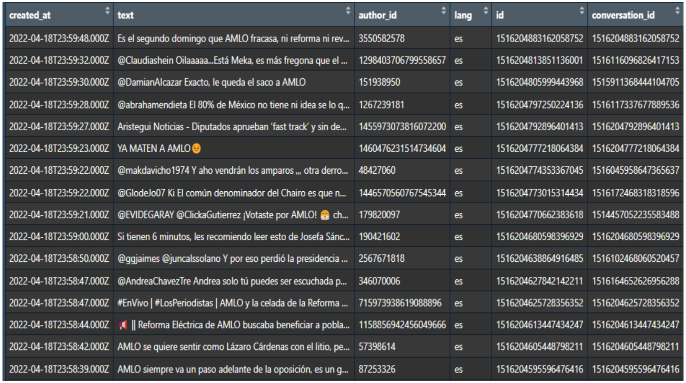
- We can build more complex queries, to do this we can use the `build_query()` function:

```r
query_amlo <- build_query(query = "AMLO", is_retweet = FALSE)

tweets1 <- get_all_tweets(query = query_amlo,
            start_tweets = "2022-03-29T00:00:00Z",
            end_tweets = "2022-04-19T00:00:00Z",
            data_path = path,
            bind_tweets = TRUE,
            n= 100)
```

# Search Full Archive: Output

- We also get variables such as number of likes, RTs, replies, and quotes, the text, the username, etc.

Figure: Output Search Full Archive in Memory



| created_at | text | author_id | lang | id | conversation_id |
|---|---|---|---|---|---|
| 2022-04-18T23:59:48.000Z | Es el segundo domingo que AMLO fracasa, ni reforma ni rev... | 3550582578 | es | 1516204883162058752 | 1516204883162058752 |
| 2022-04-18T23:59:32.000Z | @Claudiashein Oilaaaaa...Está Meka, es más fregona que el ... | 1298403706799558657 | es | 1516204813851136001 | 1516116096826417153 |
| 2022-04-18T23:59:30.000Z | @DamianAlcazar Exacto, le queda el saco a AMLO | 151938950 | es | 1516204805999443968 | 1515911368444104705 |
| 2022-04-18T23:59:28.000Z | @abrahamendieta El 80% de México no tiene ni idea se lo q... | 1267239181 | es | 1516204797250224136 | 1516117337677889536 |
| 2022-04-18T23:59:27.000Z | Aristegui Noticias - Diputados aprueban 'fast track' y sin de... | 1455973073816072200 | es | 1516204792896401413 | 1516204792896401413 |
| 2022-04-18T23:59:23.000Z | YA MATEN A AMLO😡 | 1460476231514734604 | es | 1516204777218064384 | 1516204777218064384 |
| 2022-04-18T23:59:22.000Z | @makdavicho1974 Y aho vendrán los amparos ,,, otra derro... | 48427060 | es | 1516204774353367045 | 1516045958647365637 |
| 2022-04-18T23:59:22.000Z | @GlodeJo07 Ki El común denominador del Chairo es que n... | 1446570560767545344 | es | 1516204773015314434 | 1516172468318318596 |
| 2022-04-18T23:59:21.000Z | @EVIDEGARAY @ClickaGutierrez ¡Votaste por AMLO! 😱 ch... | 179820097 | es | 1516204770662383618 | 1514457052235583488 |
| 2022-04-18T23:59:00.000Z | Si tienen 6 minutos, les recomiendo leer esto de Josefa Sánc... | 190421602 | es | 1516204680598396929 | 1516204680598396929 |
| 2022-04-18T23:58:50.000Z | @ggjaimes @juncalssolano Y por eso perdió la presidencia ... | 2567671818 | es | 1516204638864916485 | 1516102468060520457 |
| 2022-04-18T23:58:47.000Z | @AndreaChavezTre Andrea solo tú puedes ser escuchada p... | 346070006 | es | 1516204627842142211 | 1516164652626956288 |
| 2022-04-18T23:58:47.000Z | #EnVivo \| #LosPeriodistas \| AMLO y la celada de la Reforma ... | 715973938619088896 | es | 1516204625728356352 | 1516204625728356352 |
| 2022-04-18T23:58:44.000Z | 📢 \|\| Reforma Eléctrica de AMLO buscaba beneficiar a pobla... | 1158856942456049666 | es | 1516204613447434247 | 1516204613447434247 |
| 2022-04-18T23:58:42.000Z | AMLO se quiere sentir como Lázaro Cárdenas con el litio, pe... | 57398614 | es | 1516204605448798211 | 1516204605448798211 |
| 2022-04-18T23:58:39.000Z | AMLO siempre va un paso adelante de la oposición, es un g... | 87253326 | es | 1516204595596476416 | 1516204595596476416 |

## Other Functions: Get Followings

- If we wanted to get the users following a certain user, we must use the `get_following()` function.
- This function is extremely easy to use, however, contrary to Tweetple, this package stores in R the info downloaded for the IDs you give it, which is a problem if these are a lot.
- Thus, an easy way to export them as you download them is to loop through these IDs using a function.
- Other functions where this applies are: `get_liked_tweets()` and `get_followers()`

## Other Functions: Get Followings

```r
# Define the function
followings <- function(id_p){
        # Run the academictwitteR function
        df <- get_user_following(id_p, bearer)
        if (length(colnames(df)) <= 1) {
            print("No followings or private")}
        else {
            df <- df |> unnest(public_metrics) |> mutate(url = NA) |>
                select(username, from_id, url, name, description,
                    profile_image_url, created_at, followers_count,
                    following_count, tweet_count, listed_count,
                    verified, protected)

            write_parquet(df, paste0("./your folder/",
            id_p, ".parquet"))
            }
        print(id_p)
    }

# Running the loop
ids |> lapply(function(x){followings(x)})
```

## QUESTIONS: