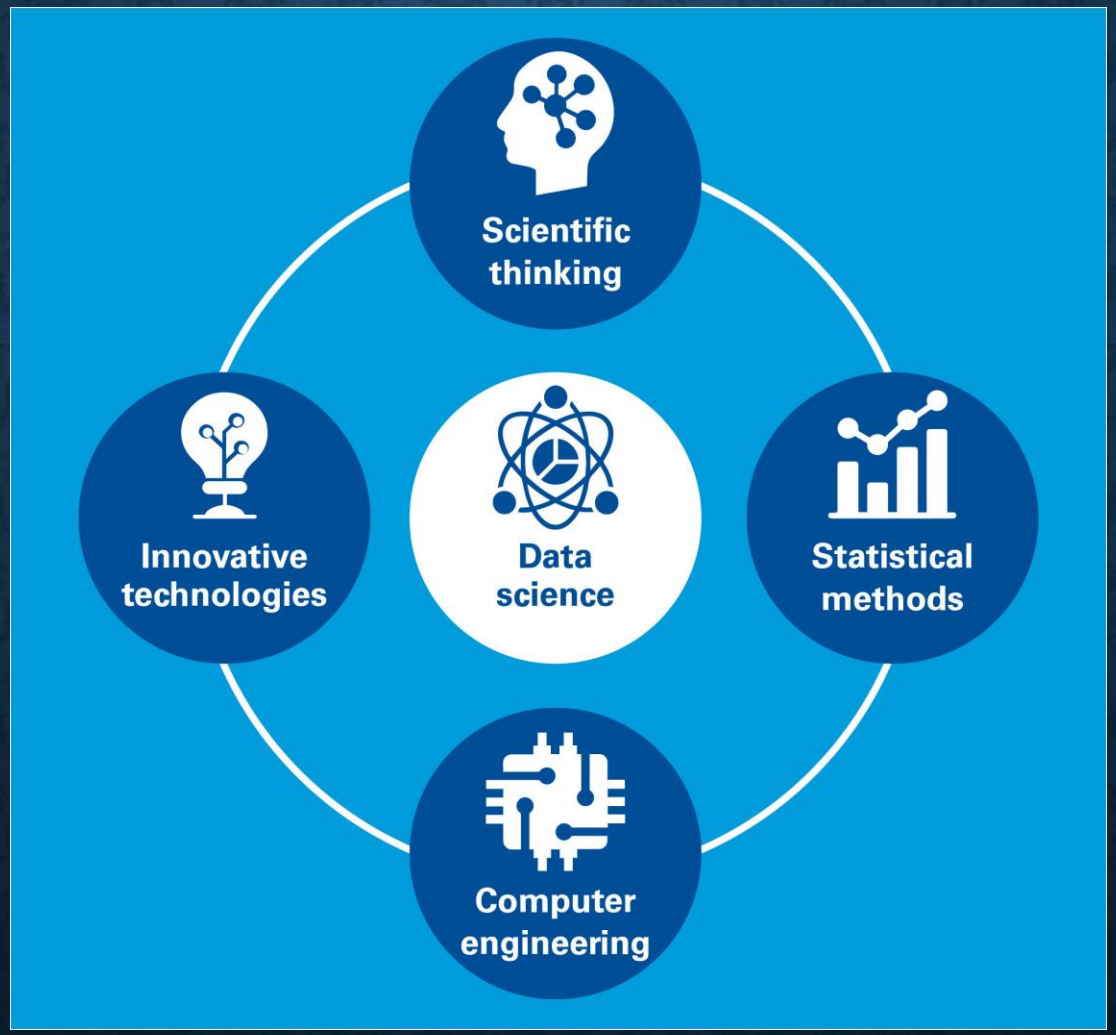
NEED FOR DATA SCIENCE

CORES OF DS

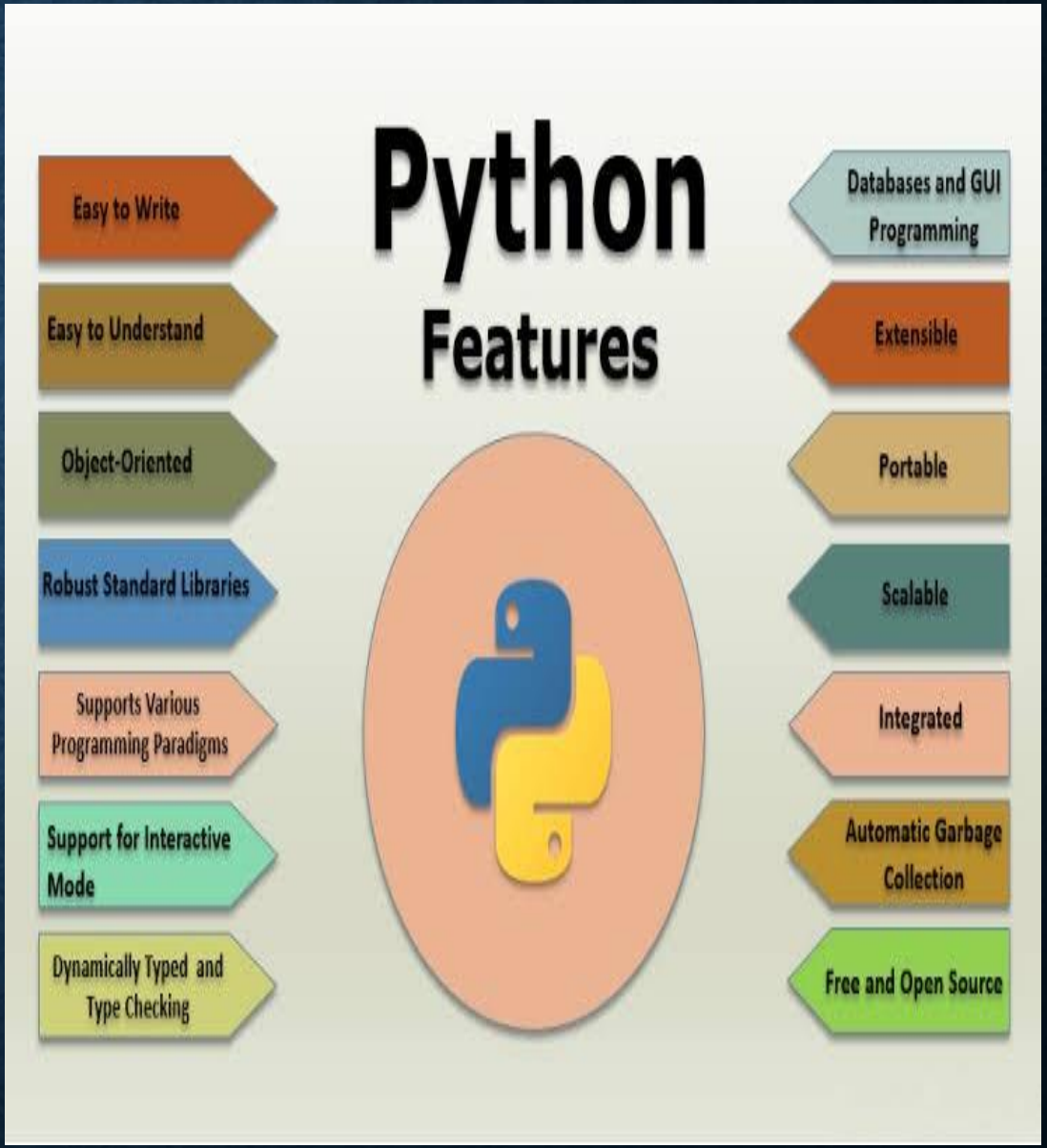

Scientific thinking
Innovative technologies
Data science
Statistical methods
Computer engineering

DATA SCIENCE & ITS IMPORTANCE

- Data is a precious asset of any organization. It helps firms understand and enhance their processes, thereby saving time and money.
- Data Science involves mining large datasets containing structured and unstructured data and identifying hidden patterns to extract actionable insights.
- Eg: using Siri for daily activities or Alexa for recommendations or operating a self-driving car.

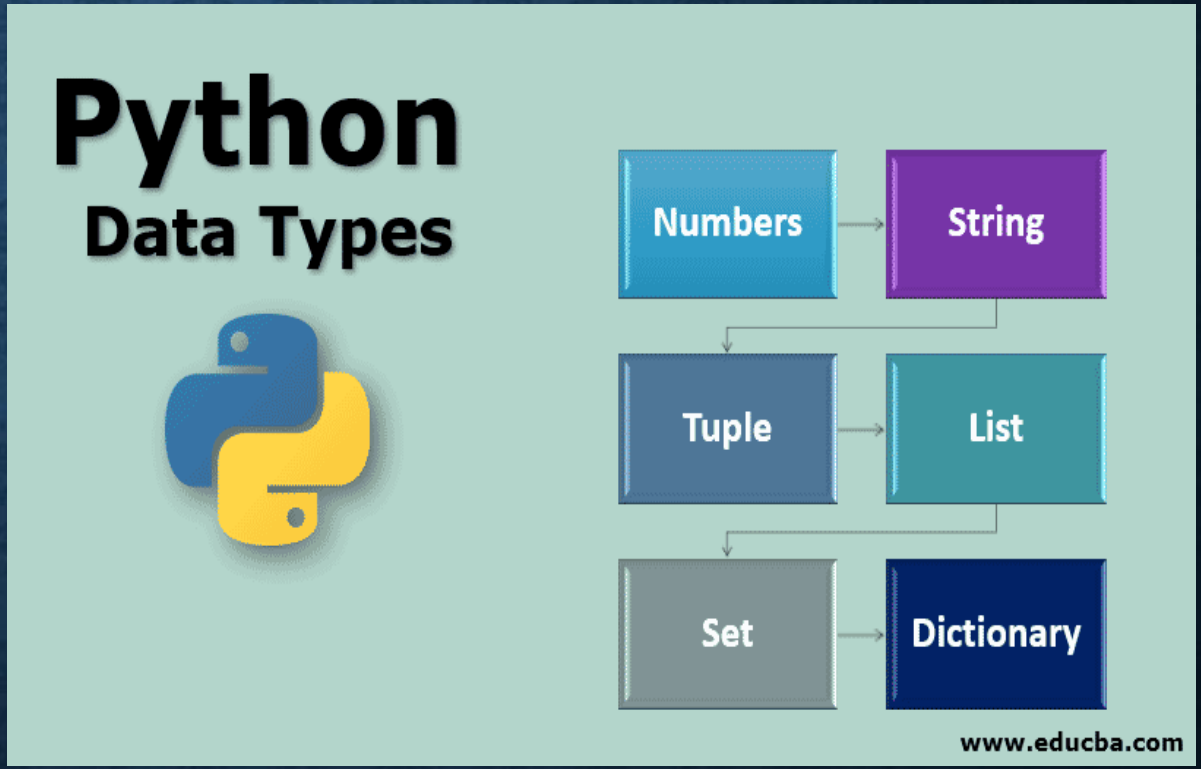
IMPORTANCE OF PYTHON IN DATA SCIENCE

- Data scientists use Python and R for data preparation and statistical analysis.
- Compared to R, Python is used for general purpose, more readable, simpler, and offers more flexibility while learning.
- Moreover, Python is used in several verticals other than Data Science and offers you various applications.



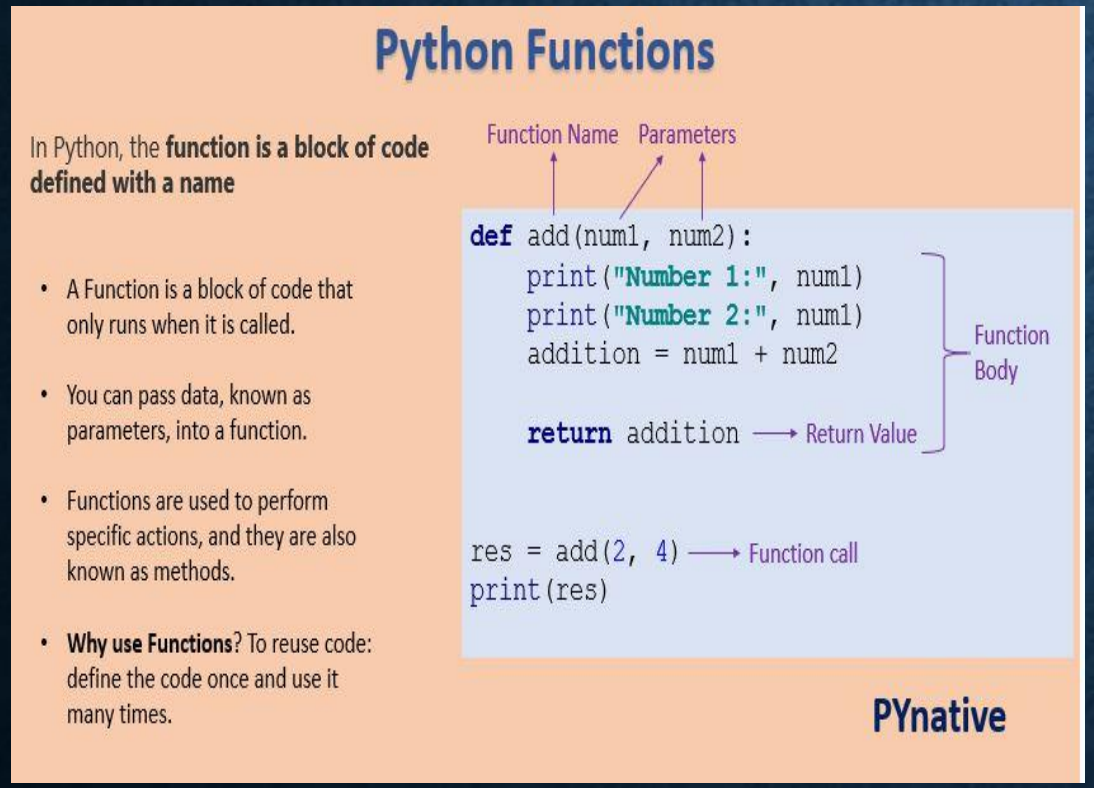
PYTHON HAS SIX STANDARD DATA TYPES:

- Numeric
- String
- List
- Tuple
- Set
- Dictionary



Name	Type	Description
Integers	int	Whole numbers, such as: **3 300 200**
Floating point	float	Numbers with a decimal point: **2.3 4.6 100.0**
Strings	str	Ordered sequence of characters: **"hello" 'Sammy' "2000" "楽しい"**
Lists	list	Ordered sequence of objects: **[10,"hello",200.3]**
Dictionaries	dict	Unordered Key:Value pairs: **{"mykey" : "value" , "name" : "Frankie"}**
Tuples	tup	Ordered immutable sequence of objects: **(10,"hello",200.3)**
Sets	set	Unordered collection of unique objects: **{"a","b"}**
Booleans	bool	Logical value indicating **True** or **False**

PYTHON FUNCTIONS



- Functions: Considerations
- Some important points to consider while defining functions:
- A function should always have a return value.
- If return is not defined, then it returns None.
- Function overloading is not permitted

FUNCTION OVERLOADING

```
def area(l, b):
c = l*b
return c
def area(size):
c = size*size
return c
area(4)
area(5,6)
```

Output:

```
Traceback (most recent call last):
  File "C:/Users/deva/Desktop/python1.py", line 8, in <module>
    area(5,6)
TypeError: area() takes 1 positional argument but 2 were given
>>>
```

PYTHON BUILT-IN FUNCTIONS

Remove items from the list	del statement	x = [1, 2, 3, 4] del x[0]	Removes one or more items with given index
	remove method	x = [1, 2, 3, 4] x.remove('1')	Removes an item with given value
	pop method	x = [1, 2, 3, 4] x.pop(0)	Removes an item with given index
	clear method	x = [1, 2, 3, 4] x.clear()	Removes all items of the list
Copy one list to another	copy method	x = [1, 2, 3, 4] y = x.copy()	
	list function	x = [1, 2, 3, 4] y = list(x)	
Join two lists	concatenation operator	x = [1, 2, 3, 4] y = ['a', 'b', 'c'] z = x + y	
	extend method	x = [1, 2, 3, 4] y = ['a', 'b', 'c'] z = x.extend(y)	

Python Built-in Functions

		Built-in Functions		
abs()	divmod()	input()	open()	staticmethod()
all()	enumerate()	int()	ord()	str()
any()	eval()	isinstance()	pow()	sum()
basestring()	execfile()	issubclass()	print()	super()
bin()	file()	iter()	property()	tuple()
bool()	filter()	len()	range()	type()
bytearray()	float()	list()	raw_input()	unichr()
callable()	format()	locals()	reduce()	unicode()
chr()	frozenset()	long()	reload()	vars()
classmethod()	getattr()	map()	repr()	xrange()
cmp()	globals()	max()	reversed()	zip()
compile()	hasattr()	memoryview()	round()	__import__()
complex()	hash()	min()	set()	
delattr()	help()	next()	setattr()	
dict()	hex()	object()	slice()	
dir()	id()	oct()	sorted()	

CONTROL FLOW STATEMENTS
PYTHON IF…ELSE STATEMENT

Python if Statement Flowchart

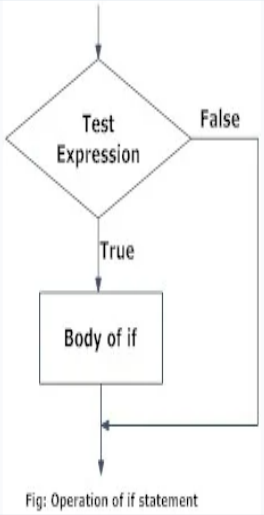


Fig: Operation of if statement

Flowchart of if statement in Python programming

Example: Python if Statement

```
# If the number is positive, we print an appropriate message

num = 3
if num > 0:
    print(num, "is a positive number.")
print("This is always printed.")

num = -1
if num > 0:
    print(num, "is a positive number.")
print("This is also always printed.")
```

When you run the program, the output will be:

```
3 is a positive number
This is always printed
This is also always printed.
```

IF…ELSE STATEMENT

Python if..else Flowchart

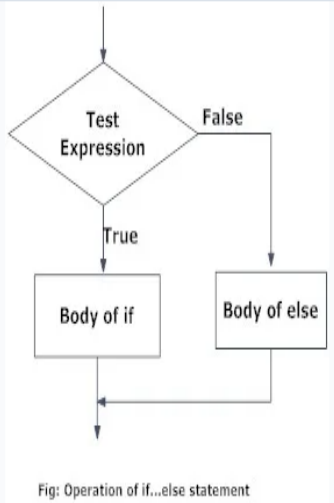


Fig: Operation of if...else statement

Flowchart of if...else statement in Python

Example of if...else

```
# Program checks if the number is positive or negative
# And displays an appropriate message

num = 3

# Try these two variations as well.
# num = -5
# num = 0

if num >= 0:
    print("Positive or Zero")
else:
    print("Negative number")
```

Output

```
Positive or Zero
```

IF…ELIF…ELSE STATEMENT

Flowchart of if...elif...else

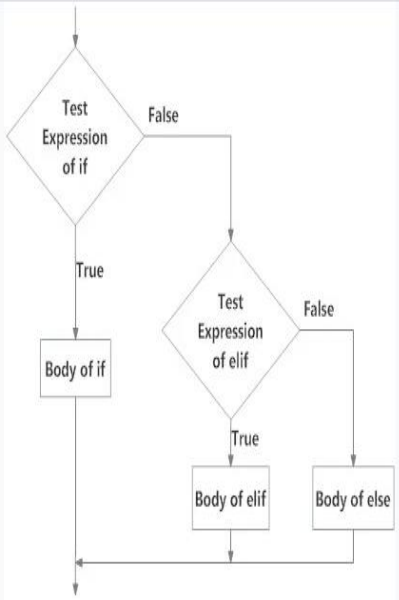


Fig: Operation of if...elif...else statement

Example of if...elif...else

```
'''In this program,
we check if the number is positive or
negative or zero and
display an appropriate message'''

num = 3.4

# Try these two variations as well:
# num = 0
# num = -4.5

if num > 0:
    print("Positive number")
elif num == 0:
    print("Zero")
else:
    print("Negative number")
```

NESTED IF STATEMENTS

```python
'''In this program, we input a number
check if the number is positive or
negative or zero and display
an appropriate message
This time we use nested if statement'''

num = float(input("Enter a number: "))
if num >= 0:
    if num == 0:
        print("Zero")
    else:
        print("Positive number")
else:
    print("Negative number")
```

Output 1

```
Enter a number: 5
Positive number
```

Output 2

```
Enter a number: -1
Negative number
```

Output 3

```
Enter a number: 0
Zero
```

FOR LOOP & RANGE FUNCTION

Syntax of for Loop

```
for val in sequence:
    loop body
```

separated from the rest of the code using indentation.

Flowchart of for Loop

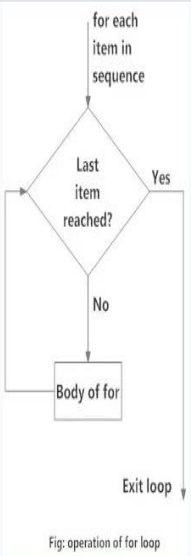


Fig: operation of for loop

Flowchart of for Loop in Python

Example: Python for Loop

```
# Program to find the sum of all numbers stored in a list

# List of numbers
numbers = [6, 5, 3, 8, 4, 2, 5, 4, 11]

# variable to store the sum
sum = 0

# iterate over the list
for val in numbers:
    sum = sum+val

print("The sum is", sum)
```

When you run the program, the output will be:

```
The sum is 48
```

```
print(range(10))
print(list(range(10)))
print(list(range(2, 8)))
print(list(range(2, 20, 3)))
```

utput

```
range(0, 10)
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[2, 3, 4, 5, 6, 7]
[2, 5, 8, 11, 14, 17]
```

```
# Program to iterate through a list using indexing

genre = ['pop', 'rock', 'jazz']

# iterate over the list using index
for i in range(len(genre)):
    print("I like", genre[i])
```

Output

```
I like pop
I like rock
I like jazz
```

FOR LOOP WITH ELSE

```
digits = [0, 1, 5]

for i in digits:
    print(i)
else:
    print("No items left.")
```

When you run the program, the output will be:

```
0
1
5
No items left.
```

```
# program to display student's marks from record
student_name = 'Soyuj'

marks = {'James': 90, 'Jules': 55, 'Arthur': 77}

for student in marks:
    if student == student_name:
        print(marks[student])
        break
else:
    print('No entry with that name found.')
```

Output

```
No entry with that name found.
```

WHILE LOOP

Syntax of while Loop in Python

```
while test_expression:
    Body of while
```

Flowchart of while Loop

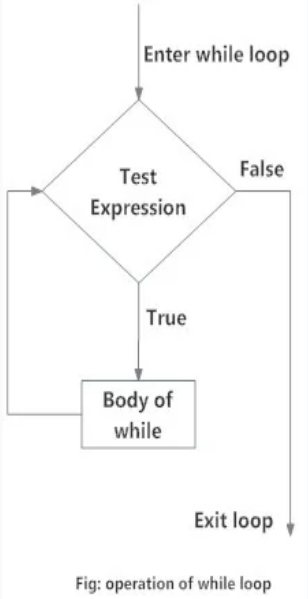


Fig: operation of while loop

Flowchart for while loop in Python

Example: Python while Loop

```
# Program to add natural
# numbers up to
# sum = 1+2+3+...+n

# To take input from the user,
# n = int(input("Enter n: "))

n = 10

# initialize sum and counter
sum = 0
i = 1

while i <= n:
    sum = sum + i
    i = i+1    # update counter

# print the sum
print("The sum is", sum)
```

When you run the program, the output will be:

```
Enter n: 10
The sum is 55
```

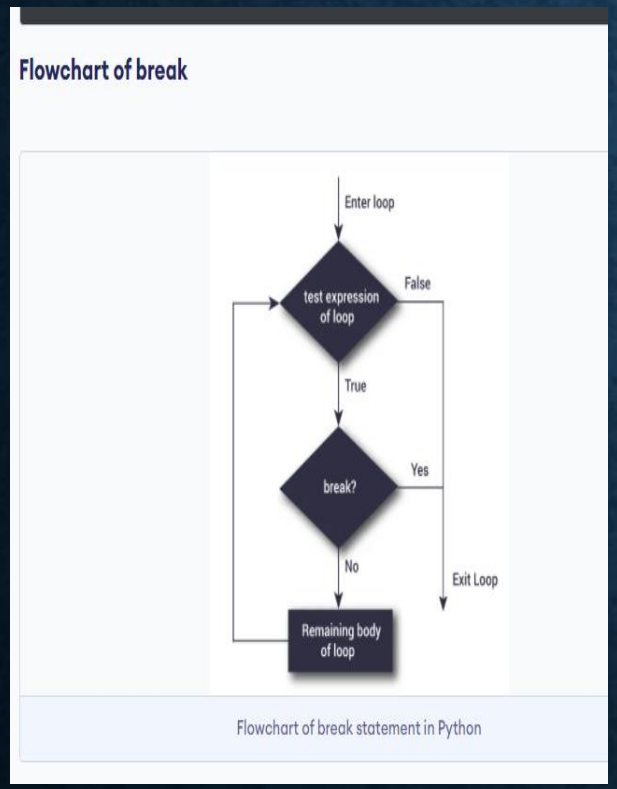
WHILE LOOP WITH ELSE

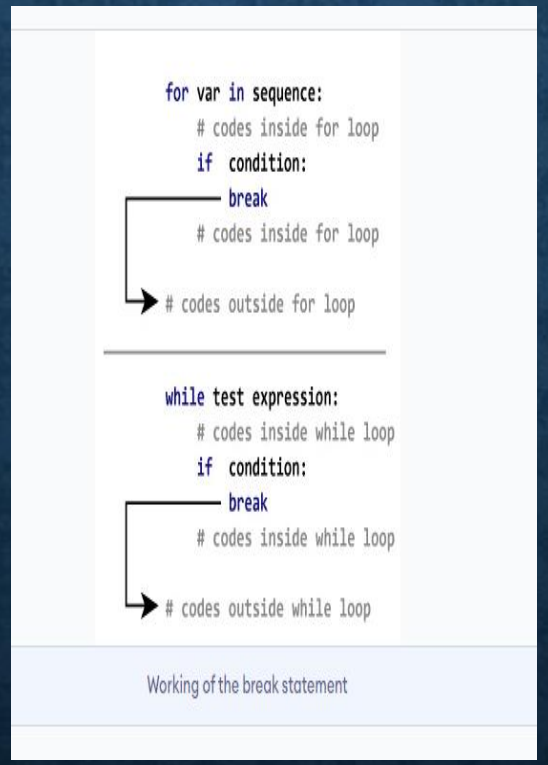
```python
'''Example to illustrate
the use of else statement
with the while loop'''

counter = 0

while counter < 3:
    print("Inside loop")
    counter = counter + 1
else:
    print("Inside else")
```

Output

```
Inside loop
Inside loop
Inside loop
Inside else
```

BREAK AND CONTINUE





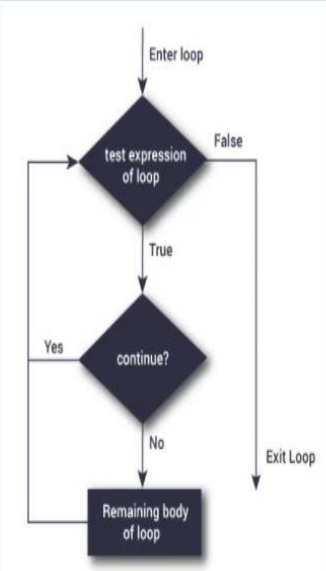

```python
# Use of break statement inside the loop

for val in "string":
    if val == "i":
        break
    print(val)

print("The end")
```

Output

```
s
t
r
The end
```

CONTINUE STATEMENT

Flowchart of continue



Flowchart of continue statement in Python

```
for var in sequence:
    # codes inside for loop
    if  condition:
        continue
    # codes inside for loop

# codes outside for loop
```

```
while test expression:
    # codes inside while loop
    if  condition:
        continue
    # codes  inside while loop

# codes outside while loop
```

How continue statement works in python

```
# Program to show the use of continue statement inside loops

for val in "string":
    if val == "i":
        continue
    print(val)

print("The end")
```

Output

```
s
t
r
n
g
The end
```

PASS STATEMENT

Example: pass Statement

```
'''pass is just a placeholder for
functionality to be added later.'''
sequence = {'p', 'a', 's', 's'}
for val in sequence:
    pass
```

We can do the same thing in an empty f

```
def function(args):
    pass
```

```
class Example:
    pass
```

BASIC CONCEPTS OF OBJECT-ORIENTED PROGRAMMING IN PYTHON

- Object-oriented programming is a programming paradigm that provides a means of structuring programs so that properties and behaviors are bundled into individual objects.
- It could represent an email with properties like a recipient list, subject, and body and behaviors like adding attachments and sending.
- Put another way, object-oriented programming is an approach for modeling concrete, real-world things, like cars, as well as relations between things, like companies and employees, students and teachers, and so on.
- OOP models real-world entities as software objects that have some data associated with them and can perform certain functions.
- The key takeaway is that objects are at the center of object-oriented programming in Python, not only representing the data, but in the overall structure of the program as well.

Python: Classes and objects

- Define a class in python:

```
1. class pet:
```

- Define a property for the class (*indentation*)

```
1. class pet:
2.     number_of_legs = 0
3.
```

- Create an objet of this class

```
1. class pet:
2.     number_of_legs = 0
3.
4. doug = pet()
```

- Access to a property or method of the object

```
1. doug.number_of_legs
```

.__INIT__() METHOD

```
class Dog:
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

Example 1: Creating Class and Object in Python

```python
class Parrot:

    # class attribute
    species = "bird"

    # instance attribute
    def __init__(self, name, age):
        self.name = name
        self.age = age

# instantiate the Parrot class
blu = Parrot("Blu", 10)
woo = Parrot("Woo", 15)

# access the class attributes
print("Blu is a {}".format(blu.__class__.species))
print("Woo is also a {}".format(woo.__class__.species))

# access the instance attributes
print("{} is {} years old".format( blu.name, blu.age))
print("{} is {} years old".format( woo.name, woo.age))
```

Output

```
Blu is a bird
Woo is also a bird
Blu is 10 years old
Woo is 15 years old
```

METHODS
EXAMPLE 2 : CREATING METHODS IN PYTHON

```
class Parrot:

    # instance attributes
    def __init__(self, name, age):
        self.name = name
        self.age = age

    # instance method
    def sing(self, song):
        return "{} sings {}".format(self.name, song)

    def dance(self):
        return "{} is now dancing".format(self.name)

# instantiate the object
blu = Parrot("Blu", 10)

# call our instance methods
print(blu.sing("'Happy'"))
print(blu.dance())
```

Output

```
Blu sings 'Happy'
Blu is now dancing
```

INHERITANCE
EXAMPLE 3: USE OF INHERITANCE IN PYTHON

```python
# parent class
class Bird:

    def __init__(self):
        print("Bird is ready")

    def whoisThis(self):
        print("Bird")

    def swim(self):
        print("Swim faster")

# child class
class Penguin(Bird):

    def __init__(self):
        # call super() function
        super().__init__()
        print("Penguin is ready")

    def whoisThis(self):
        print("Penguin")

    def run(self):
        print("Run faster")

peggy = Penguin()
peggy.whoisThis()
peggy.swim()
peggy.run()
```

Output

```
Bird is ready
Penguin is ready
Penguin
Swim faster
Run faster
```

ENCAPSULATION
EXAMPLE 4: DATA ENCAPSULATION IN PYTHON

```python
class Computer:

    def __init__(self):
        self.__maxprice = 900

    def sell(self):
        print("Selling Price: {}".format(self.__maxprice))

    def setMaxPrice(self, price):
        self.__maxprice = price

c = Computer()
c.sell()

# change the price
c.__maxprice = 1000
c.sell()

# using setter function
c.setMaxPrice(1000)
c.sell()
```

Output

```
Selling Price: 900
Selling Price: 900
Selling Price: 1000
```

POLYMORPHISM
EXAMPLE 5: USING POLYMORPHISM IN PYTHON

```python
class Parrot:

    def fly(self):
        print("Parrot can fly")

    def swim(self):
        print("Parrot can't swim")

class Penguin:

    def fly(self):
        print("Penguin can't fly")

    def swim(self):
        print("Penguin can swim")

# common interface
def flying_test(bird):
    bird.fly()

#instantiate objects
blu = Parrot()
peggy = Penguin()

# passing the object
flying_test(blu)
flying_test(peggy)
```

utput

```
Parrot can fly
Penguin can't fly
```

NUMPY

- 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 stands for Numerical Python.
- The most important object defined in NumPy is an N-dimensional array type called **ndarray**.
- It describes the collection of items of the same type. Items in the collection can be accessed using a zero-based index.
- Every item in a ndarray takes the same size as the block in the memory. Each element in ndarray is an object of the data-type object (called **dtype**).
- An instance of ndarray class can be constructed by different array creation routines described later in the tutorial. The basic ndarray is created using an array function in NumPy as follows
- numpy.array
- It creates a ndarray from any object exposing an array interface, or from any method that returns an array.
- numpy.array(object, dtype = None, copy = True, order = None, subok = False, ndmin = 0)

The above constructor takes the following parameters −

Sr.No.	Parameter & Description
1	**object** Any object exposing the array interface method returns an array, or any (nested) sequence.
2	**dtype** Desired data type of array, optional
3	**copy** Optional. By default (true), the object is copied
4	**order** C (row major) or F (column major) or A (any) (default)
5	**subok** By default, returned array forced to be a base class array. If true, sub-classes passed through
6	**ndmin** Specifies minimum dimensions of resultant array

INSTALLATION OF NUMPY

```
C:\Users\Your Name>pip install numpy
```

Import NumPy

Once NumPy is installed, import it in your applications by adding the `import` keyword:

```
import numpy
```

Checking NumPy Version

The version string is stored under `__version__` attribute.

Example

```
import numpy as np

print(np.__version__)
```

Example 1

```
import numpy as np
a = np.array([1,2,3])
print a
```

The output is as follows –

```
[1, 2, 3]
```

Example 2

```
# more than one dimensions
import numpy as np
a = np.array([[1, 2], [3, 4]])
print a
```

The output is as follows –

```
[[1, 2]
 [3, 4]]
```

Example 3

```
# minimum dimensions
import numpy as np
a = np.array([1, 2, 3,4,5], ndmin = 2)
print a
```

The output is as follows –

```
[[1, 2, 3, 4, 5]]
```

Example 4

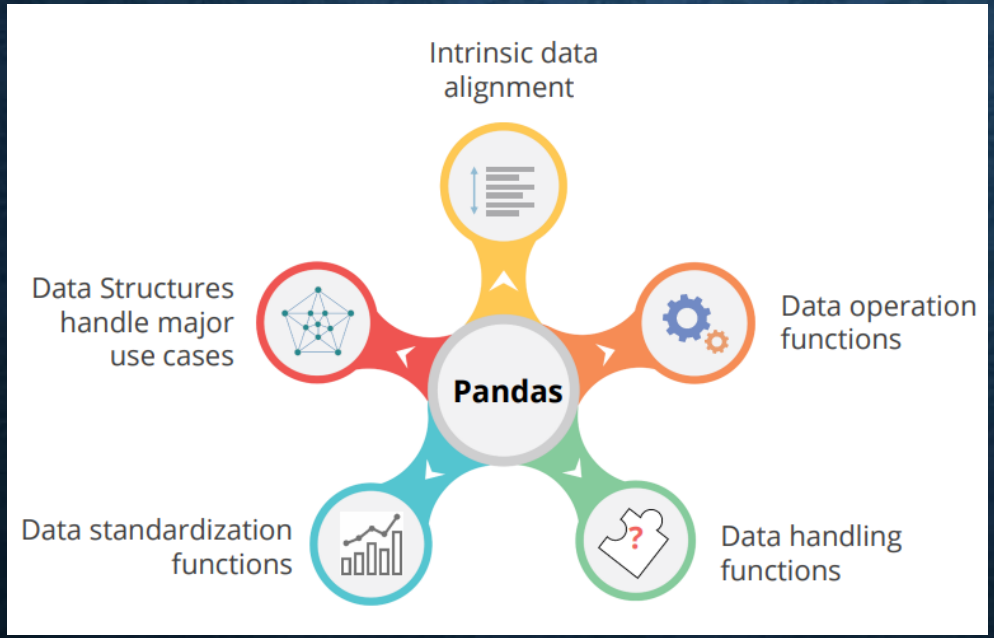
```
# dtype parameter
import numpy as np
a = np.array([1, 2, 3], dtype = complex)
print a
```

The output is as follows –

```
[ 1.+0.j,  2.+0.j,  3.+0.j]
```

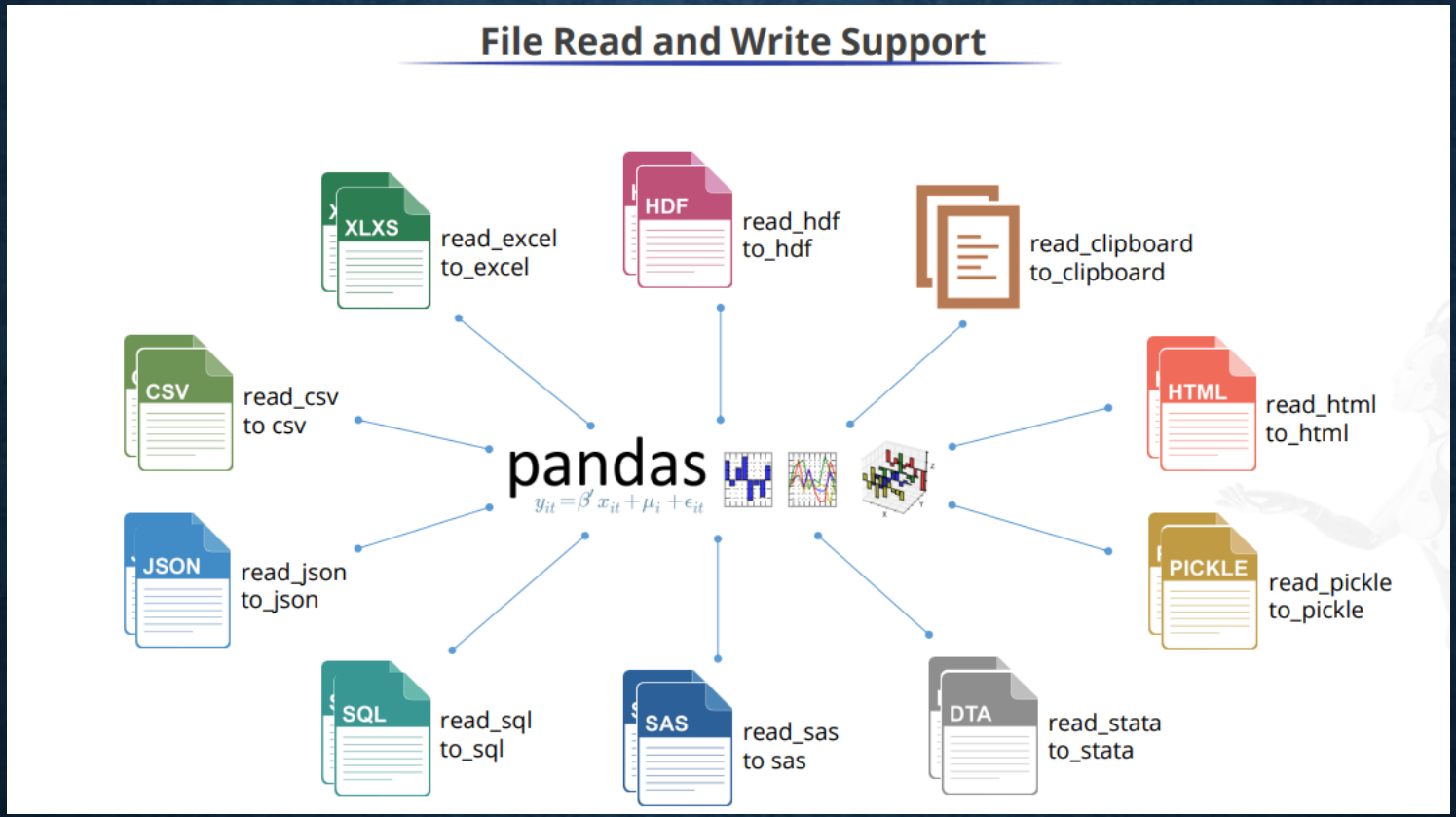
PANDAS

- Pandas is a **Python library** for data analysis.
- It is a fast, powerful, flexible and easy to use open source data analysis and manipulation tool, **built on top of the Python programming language**.



Installing from PyPI

```
pip install pandas
```


File Read and Write Support
XLXS
read_excel
to_excel
HDF
read_hdf
to_hdf
read_clipboard
to_clipboard
CSV
read_csv
to csv
HTML
read_html
to_html
pandas
JSON
read_json
to_json
PICKLE
read_pickle
to_pickle
SQL
read_sql
to_sql
SAS
read_sas
to sas
DTA
read_stata
to_stata

In [3]:
```
from matplotlib import pyplot as plt
import matplotlib.pyplot as plt
import pandas as pd
import numpy as np
import patsy as pt
```

In [4]:
```
# For .read_csv, always use header=0 when you know row 0 is the header row
train_df = pd.read_csv('csv/train.csv', header=0)
test_df = pd.read_csv('csv/test.csv', header=0)
```

In [5]:
```
train_df.head(3)
```

Out[5]:

	PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare	Cabin	Embarked
0	1	0	3	Braund, Mr. Owen Harris	male	22	1	0	A/5 21171	7.2500	NaN	S
1	2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38	1	0	PC 17599	71.2833	C85	C
2	3	1	3	Heikkinen, Miss. Laina	female	26	0	0	STON/O2. 3101282	7.9250	NaN	S

Creating DataFrame from Lists

```
In [1]: import pandas as pd
```

Create DataFrame from dict of equal length lists

```
In [2]: #last five olymnics data: place, year and number of countries participated
        olympic_data_list = {'HostCity':['London','Beijing','Athens','Sydney','Atlanta'],
                             'Year':[2012,2008,2004,2000,1996],
                              'No. of Participating Countries':[205,204,201,200,197]
                             }
```

```
In [3]: df_olympic_data = pd.DataFrame(olympic_data_list)
```

Pass the list to the DataFrame

```
In [4]: df_olympic_data
```

Out[4]:

	HostCity	No. of Participating Countries	Year
0	London	205	2012
1	Beijing	204	2008
2	Athens	201	2004
3	Sydney	200	2000
4	Atlanta	197	1996

WEB SCRAPING USING PYTHON

LibraryUsed:

- AutoScraper
- Requests
- Beautiful Soup 4
- lxml
- Selenium

- Web Scraping is a method or art to get or scrap data from the internet or websites and storing it locally on your system.
- Web Scripting is a programmed strategy to acquire a lot of information from sites.
- Automated web scraping can be a solution to speed up the data collection process.
- You write your code once, and it will get the information you want many times and from many pages.

AUTOSCRAPER

- It makes web scraping smart, automatic fast, and easy.
- You have to provide the URL or HTML content of the web page from where you want to scrap the data.
- The information can be text, URL, or any HTML label worth of that page.

Installation

There are 3 ways to install this library in your system.

- Install from the git repository using pip:

```
pip install git+https://github.com/alirezamika/autoscraper.git
```

- Install using PyPI:

```
pip install autoscraper
```

- Install from source:

```
python setup.py install
```

Importing Library

```
from autoscraper import AutoScraper
```

1. Defining Web Scraping Function

```
url = 'https://www.analyticsvidhya.com/blog/category/machine-learning/'
wanted_list = ['Confusion Matrix: Detailed intuition and trick to learn']
```

wanted list is a list that is sample data that we want to scrape from that page

2. Initiate the AutoScraper

```
scraper = AutoScraper()
```

call the AutoScraper function
We aim to use this function to build the scraper model and perform web scraping on that particular page itself.

3. Building the Object

final step in web scraping using this particular library. Here we create the object and show the result of web scraping.

```
scraper = AutoScraper()
result = scraper.build(url, wanted_list)
print(result)
```

```
['Confusion Matrix: Detailed intuition and trick to learn', 'Top 30 MCQs to Ace Your Data Science Interviews', '20 Must-Know Pa
ndas Function for Exploratory Data Analysis', 'Beginners Guide to Explanatory Data Analysis', 'Beginners Guide to Automation in
Data Science', 'Simple understanding and implementation of KNN algorithm!', 'How to Perform Monte Carlo Simulation?', 'Proximit
y measures in Data Mining and Machine Learning', 'Guide For Feature Extraction Techniques', 'How the Gradient Boosting Algorith
m works?', 'Machine Learning Basics For Data Science Enthusiasts', 'Bring DevOps To Data Science With MLOps', 'Principal Compon
ent Analysis Introduction and Practice Problem']
```

4. Saving the Model

save the model that we have to build so that we can reload it whenever required.

To save the model, use below code

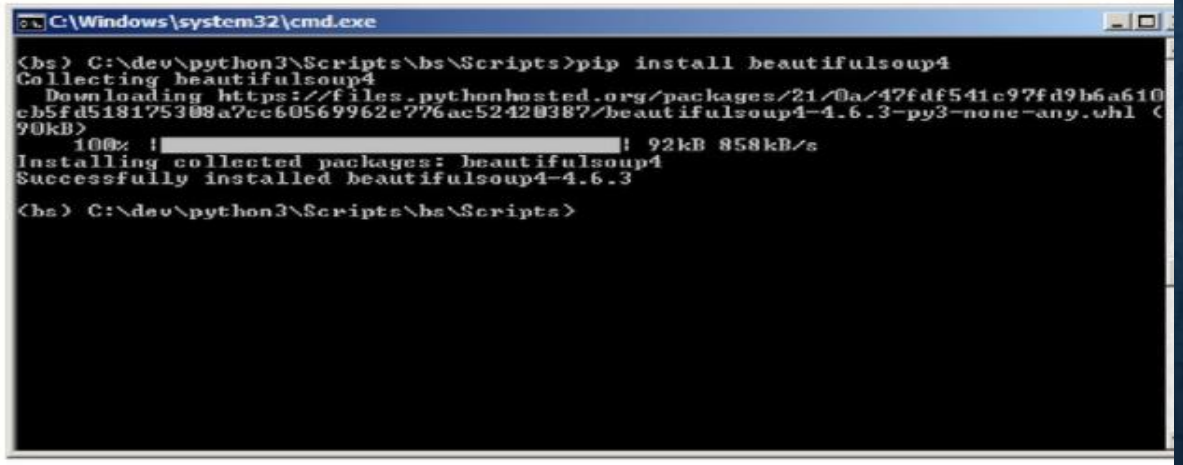
```
scraper.save('blogs')      #Give it a file path
```

To load the model, use the below code:

```
scraper.load('blogs')
```

Beautiful Soup: Build a Web Scraper With Python

pip install beautifulsoup4



BEAUTIFULSOUP 4

Steps involved in web scraping:

- Find the URL of the webpage that you want to scrape
- Select the particular elements by inspecting
- Write the code to get the content of the selected elements
- Store the data in the required format
- BeautifulSoup – Python library for getting data out of HTML, XML, and other markup languages

```python
# !pip install pandas requests BeautifulSoup4
import pandas as pd
import requests
from bs4 import BeautifulSoup as bs
base_url = "https://www.consumeraffairs.com/food/dominos.html"
all_pages_reviews =[]
```

```python
def scraper():
    for i in range(1,6): # fetching reviews from five pages
        pagewise_reviews = []
        query_parameter = "?page="+str(i)
        url = base_url + query_parameter
        response = requests.get(url)
        soup = bs(response.content, 'html.parser')
        rev_div = soup.findAll("div",attrs={"class","rvw-bd"})

    for j in range(len(rev_div)):
    # finding all the p tags to fetch only the review text
        pagewise_reviews.append(rev_div[j].find("p").text)

    for k in range(len(pagewise_reviews)):
        all_pages_reviews.append(pagewise_reviews[k])
        return all_pages_reviews

# Driver code
reviews = scraper()
i = range(1, len(reviews)+1)
reviews_df = pd.DataFrame({'review':reviews}, index=i)
reviews_df.to_csv('reviews.txt', sep='t')
```

```python
import requests

url = 'https://notes.ayushsharma.in/technology'

data = requests.get(url)

print(data.text)
```

```python
import requests
from bs4 import BeautifulSoup
from pprint import pprint

url = 'https://notes.ayushsharma.in/technology'
data = requests.get(url)

my_data = []

html = BeautifulSoup(data.text, 'html.parser')
articles = html.select('a.post-card')

for article in articles:

    title = article.select('.card-title')[0].get_text()
    excerpt = article.select('.card-text')[0].get_text()
    pub_date = article.select('.card-footer small')[0].get_text()

    my_data.append({"title": title, "excerpt": excerpt, "pub_date": pub_date})

pprint(my_data)
```

WEB SCRAPING USING SELENIUM AND PYTHON

- The Selenium API uses the WebDriver protocol to control a web browser, like Chrome, Firefox or Safari.
- The browser can run either locally or remotely.
- It is still used for testing, but it is also used as a general browser automation platform.

Installation

- Install selenium using pip

```
pip install selenium
```

- Install selenium using conda

```
conda install -c conda-forge selenium
```

. Download Chrome Driver:

To download web drivers, you can choose any of below methods-

1. You can either directly download chrome driver from the below link- https://chromedriver.chromium.org/downloads
2. Or, you can download it directly using below line of code-driver = webdriver.Chrome(ChromeDriverManager().install())

Following methods will help us to find elements in a Web-page (these methods will return a list):

- find_elements_by_name
- find_elements_by_xpath
- find_elements_by_link_text
- find_elements_by_partial_link_text
- find_elements_by_tag_name
- find_elements_by_class_name
- find_elements_by_css_selector

Step1:– Import libraries

```
import os
import selenium
from selenium import webdriver
import time
from PIL import Image
import io
import requests
from webdriver_manager.chrome import ChromeDriverManager
from selenium.common.exceptions import ElementClickInterceptedException
```

Step 2:– Install Driver

```
#Install Driver
driver = webdriver.Chrome(ChromeDriverManager().install())
```

Step 3:– Specify search URL

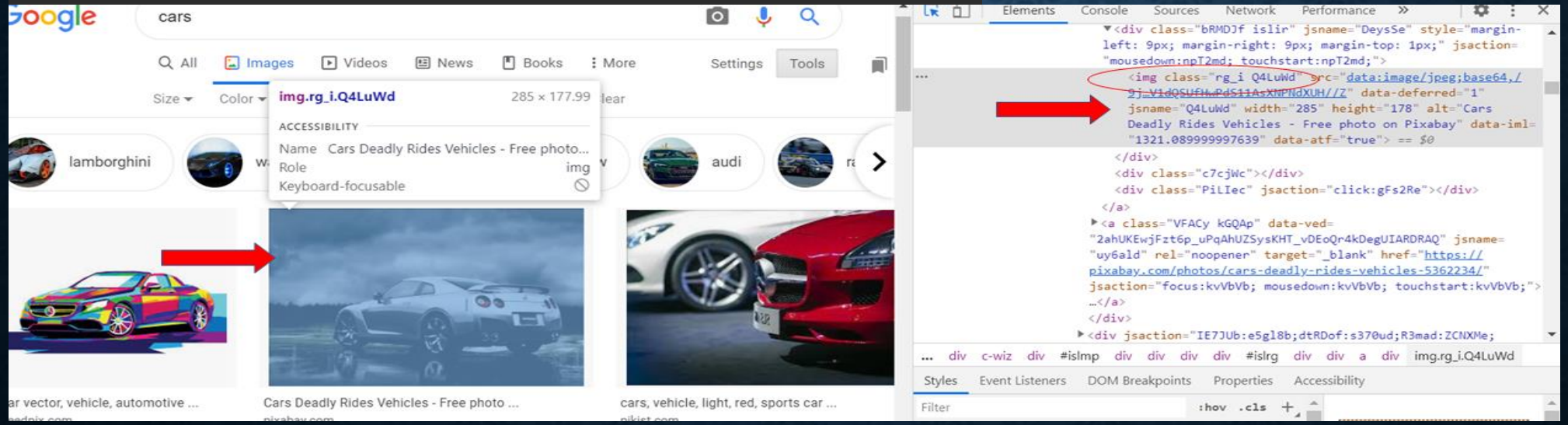
```
#Specify Search URL
search_url="https://www.google.com/search?q={q}&tbm=isch&tbs=sur%3Afc&hl=en&ved=0CAIQpwVqFwoTCKC

driver.get(search_url.format(q='Car'))
```

Step 4:– Scroll to the end of the page

```
def scroll_to_end(driver):
    driver.execute_script("window.scrollTo(0, document.body.scrollHeight);")
    time.sleep(5)#sleep_between_interactions
```

Step 5:– Locate the images to be scraped from the page

```
#Locate the images to be scraped from the current page
imgResults = driver.find_elements_by_xpath("//img[contains(@class,'Q4LuWd')]")
totalResults=len(imgResults)
```



Step 6: – Extract the corresponding link of each Image

```
mg_urls = set()
or i in  range(0,len(imgResults)):
    img=imgResults[i]
    try:
        img.click()
        time.sleep(2)
        actual_images = driver.find_elements_by_css_selector('img.n3VNCb')
        for actual_image in actual_images:
            if actual_image.get_attribute('src') and 'https' in actual_image.get_attribute('src'):
                img_urls.add(actual_image.get_attribute('src'))
    except ElementClickInterceptedException or ElementNotInteractableException as err:
        print(err)
```

Step 7: – Download & save each image in the Destination directory

```
os.chdir('C:/Qurantine/Blog/WebScrapping/Dataset1')
baseDir=os.getcwd()
```

```
for i, url in enumerate(img_urls):
    file_name = f"{i:150}.jpg"
    try:
        image_content = requests.get(url).content

except Exception as e:
        print(f"ERROR - COULD NOT DOWNLOAD {url} - {e}")

try:
        image_file = io.BytesIO(image_content)
        image = Image.open(image_file).convert('RGB')

        file_path = os.path.join(baseDir, file_name)

        with open(file_path, 'wb') as f:
            image.save(f, "JPEG", quality=85)
        print(f"SAVED - {url} - AT: {file_path}")
    except Exception as e:
        print(f"ERROR - COULD NOT SAVE {url} - {e}")
```