Chapter: 1

Overview of python and Data Structure.

(1) Explain Data types and variable.

→ A variable is a way of referring to a memory location used by a computer program. A variable is a symbolic name for this physical location. This memory location contains value, like number, text etc.

→ A variable is a name that refers to a value. The equal (=) operator is used to assign value to a variable.

→ Phython data type includes : Number, Strings, List, Dictionaries, Tuples and Files.

→ Python has no additional command to declare a variable. As soon as value is assigned to it, the variable is declared.

Ex. a = 20

→ Rules for Variables

- Special characters are not allowed.
- Variables are case - sensitive.
- Variables can only contain alpha numeric characters and underscore.
- Variable name always start with character.

→ Python allows us to assign a value to multiple variables in single statment, which is also known as multiple assignment.

Ex. (a) x = y = z = 51
(b) x, y, z = 11, 21, 31

→ Numerical data types are integer, float, complex number and boolean.

→ Integer are used to represent whole number values. Float data type is used to represent decimal point values.

```
i = 13          # data type is integer
i = 13 + 0.7    # data type changed to float
i = "fifty"     # data type is string
```

-> Python automatically takes care of the physical representation for the different data types, an int value store in different memory location than a float or string.

-> Int : Python has no restriction on the length of integer.

-> Float : float is used to store floating point numbers like 3.2, 40.5. It is accurate upto 15 decimal point.

-> Complex : A complex number contains ordered pair like x + iy where x and y denoted the real and imaginary parts.

--> Operator

+ -> addition
* -> Multiplication
- -> Substraction
/ -> division
>> -> Right shift bit.
<< -> Left shift bit.
** -> Exponential
% -> Modulus

(2) Explain object and function.

→ * Function

- A function is a block of code which only runs when it is called.
- You can pass data known as parameters, into function.
  A function can return data as result.
- In python function is defined using [def] keyword.
- To call function use the function name followed by paranthesis.
- Ex.

```
def my-function (fname):
    print (fname + "Porwal")

my-function ("Harsh")
my-function ("Yash")
```

output :

Harsh Porwal
Yash Porwal

In [3]:
```
def my_fun(fname):
    print(fname +" " + "Porwal")

my_fun("Harsh")
my_fun("Yash")
```

```
Harsh Porwal
Yash Porwal
```

(3) Explain String, Array, List, Tuple, Set, Dictionary.

* String

- Strings are collection of characters which are stored together to represent arbiratry text inside a python program.

- Strings in python are used to represent unicode character values. Python does not have character data type, a single character is also called a string.

- We denote or declare the string values inside single or double quotes. To access the values in a string, we use the indexes and square brackets.

- The operator [*] is known as a repetation operator as the operation "Python" * 2 returns 'Python Python'

- Format strings consist of combined character data and markup. Character data is data which is transferred unchanged from the format string to the output string. markup is not transffered from the format string directly to the output.

(i) Concatenation : The addition (+) operator takes on the role of concatenation when used with string. The result of adding two string together is a new string which consists of the original strings.

Ex.

```
first = "Harsh"
second = "Porwal"
print (first + " " + second)
```

Output :- Harsh Porwal

(ii) Repetition : The asterisk (*), when used b/w a string and an integer creates a new string with the old string repeated by the value of the integer.

Ex.

```
a = "abc " * 3
print (a)
```

output : abc abc abc

(iii) Slicing : Python support indexing and slicing operation. To extract a single character from a string, follow the string with the index of the desired character surrounded by square brackets.

Ex.

```
a = "Python"
print(a[2])
```

output : t

In [4]:
```
first = "Harsh"
second = "Porwal"
print(first+" "+second)
```

```
Harsh Porwal
```

In [9]:
```
a ="abc " *3
print(a)
```

```
abc abc abc
```

In [10]:
```
a = "Python"
print(a[2])
```

```
t
```

* Array

- Python does not have built-in support for arrays. Python lists are used to serve the purpose.

- User can treat lists as arrays but cannot constrain the data type of values stored in a list. Only the same type of data can be stored in arrays.

- To create array in python, we need to import the array module to create arrays.

Ex.

```
import array as ary
b = ary.array("i", [2,3,4])
print(b[0])
```

output: 2

* Set

- Set is an unordered collection of simple object in python.
- You can use curly braces to give an expression whose value is a set. Python print sets using curly braces.

Ex.

```
Set = {"abc", "def", "ghi"}
print(Set)
```

output: {"abc", "def", "ghi"}

In [11]:
```
import array as ary
b = ary.array("i",[2,3,4])
print(b[0])
```

```
2
```

In [12]:
```
set = {"abc", "def", "ghi"}
print(set)
```

```
{'def', 'abc', 'ghi'}
```

* List

- List provide a genral mechanism for storing a collection of objects indexed by a number in python.
- A list is a linear data structure, meaning that its elements have a linear ordering.
- Each element can be accessed by an index. Note that python indexes start with 0 instead of 1.
- Lists are a mutable collection of iteams. We use square brackets to surroud the items[].
- The elements of the list are arbitary, they can be number, String, functions, User defined objects or even other lists making complex data structure very simple to express in python.
- After creating list in python, it is possible to search, add, and remove iteams. A list with another list is called nested list.

Ex.

```
my = ['p', 'r', 'g', 'r', 'a', 'm']
print (my [2:5])
print (my [5 :])
print (my [:])
```

output :

```
['g', 'r', 'a']
['m']
['p', 'r', 'g', 'r', 'a', 'm']
```

In [17]:
```
my = ['p' ,'r','g','r','a','m']
print(my[2:5])
print(my[5])
print(my[:])
```

```
['g', 'r', 'a']
m
['p', 'r', 'g', 'r', 'a', 'm']
```

In [18]:
```
myTuple = (0 , "Harsh","22.5")
print(myTuple[2])
```

```
22.5
```

In [19]:
```
myD = {'a':'Harsh', 'b':'Porwal'}
print(myD['b'])
```

```
Porwal
```

* Tuple

- A tuple is a sequence of values. The values can be any type and are indexed by integers, unlike list. Tuples are immutable.

- Tuple is used to contain data that are related but not necessarily of the same type.

- Like python list, tuple supports random collection of object. Tuple is a fixed length object.

- As the tuple are immutable you can't change the size of tuples without making a copy of it.

- A tuple with a single element must have comma inside the parantheses. tuple in python programming language is defined using parantheses with each iteam seprated by commas.

- Tuple iteam contains object of string, Integer, and float types.

Ex.
```
myTuple = (0, "Harsh", "22.5")
print (myTuple[2])
```

Output : 22.5

* dictionary

- Python provides collections, called dictionaries, that are suitable for representing such function. Conceptually a dictionary is a set of key-value pairs.

- The key defined for a dictionary need to be unique. Though value in a dictionary can be mutable or immutable object, only immutable objects are allowed for keys.

- A dictionary in python is defined using key-value pairs, seprated by commas, enclosed within curly braces.

```
d = { "key1" : "value1",
      "key2" : "value2" }
```

- Key and values in dictionaries can be of any type, but the keys should be unique to that particular dictionary.

- It does not support duplicate keys.

- Key must be immutable.

Ex.

```
myD = {'a' : 'Harsh', 'b' : 'Porwal'}
print (myD ['b'])
```

output : Porwal

(4)

List	Tuple	Set	Dictionary
→ It is a non-homogeneous data structure that stores that elements in single or multiple rows and columns.	It is a non-homogeneous data structure that store single and multiple rows and columns.	Set data structure is also non homogeneous data structure but store in single row.	Dictionary is also non-homogeneous data structure which store key and values pairs.
→ Represented by [].	Represented by ().	Represented by {}	Represented by {}
→ allow duplicate elements.	allows duplicate elements	It will not allow duplicate elements.	It does not allow duplicate key values.
→ Created using list().	Created using tuple().	Created using set()	Created using dict().
→ It is mutable	It is immutable.	It is mutable.	It is mutable.
→ It is ordered.	It is ordered.	It is unorder.	It is ordered.
→ Ex. [1, 2, 3, 4]	Ex. (1, 2, 3, 4)	Ex. {1, 2, 3, 4}	Ex. {1, 2, 3, 4}

(6) Explain Feautures and Advantage-Disadvantage of python.

→ * Features

- Python is a high-level interpreted, interactive and object-oriented scripting language.
- It is simple and easy to learn.
- It is portable.
- Python is free and open source programming language.
- Python can easily perform complex task.
- Python can run on all OS such as windows, Linux, Mcintosh, etc
- It provides vast range of libraries for various field such as machine learning.

* Advantage

. Ease of programming.

. Minimizes the time to devlope and maintain code.

. Modular and object-oriented.

. Large community of user.

. A large standard and user-contributed library.

* Disadvantage

. Interpreted and therefore slower than compiled language.

. Decenterlized with packages.

Chapter : 2

Data Science and Python

(1) What is role of python in data science?

-> Data analysts are responsible for interpreting data and analyzing the results utilizing statistical techniques and providing ongoing report.

-> It is one of the best languages used by data scientist for various data science projects. Python provides great functionality to deal with mathamatic, statics and scientific function.

-> One of the main reason python are widely used in scientific and research community because its ease of use and simple syntax which makes it easy to adopt for non engineering background people.

-> Some most popular libraries :

NumPy : It provides support for mathamatical task, multidimensional array and matrics.

Pands : It allows easy manipulation of tabular data for data cleaning and data analysis.

Matplotib : It is used to create static and interactive box-plot, scatter-plot, graphs, bar and charts.

(2) Creating dataScience Pipeline.

→ Data science pipelines are sequence of processing and analysis step applied to data for specific purpose.

→ They are useful in production projects, and they can also be useful in if one expects to encounter the same type of bussiness quention in the future. So as to save design, time and coding.

(i) Preparing the data : Big data enables organization to gather, store, manage and manipulate vast amounts of data at the right speed, and right time, to gain insights.

The raw data not only may vary substantially in format, but you may also need to transform it to make all data perfect to analysis.

(ii) Performing exploratory data analysis:

Exploratory data analysis (EDA) also known as exploration is step in the data analysis process, where number of techniques are used to better understand the dataset being used.
You can turn an almost useable dataset into completley useable datset.

(iii) Learning from data : Discovery is part of being a data scientist.

(iv) Visulizing : It mean seeing the patterns in the data and then being able to react those patterns.
Visulization is a presentation of quantative information in a graphical form.

In other word data visulization turns large and small database into visual that are easier for the human brain to understand the process.

(3) Implementing machine learning using Scikit-learn.

→ Scikit-learn is probably the most useful library for machine learning in python.

→ It is on NumPy, SciPy, and matplotlib this library contains a lot effcient tools for machine learning and statical modeling.

→ In Scikit-learn an estimator for classification is a python object that implements the method fix (x, y) and Predict (T).

→ It provides following functionality

- Classification
- Regression
- clustering
- Dimensionality reduction
- Model selection
- Preprocessing

(a) Describe Numpy and Pandas library in Python.

→ * NumPy

- Numpy has risen to become one of the most popular python science library and just secured a round of grant funding.
- Numpy's multidemensional array can perform very large calculation easy and effciently than using the python standard data types.
- To get started NumPy has many resources on their website including documentation and tutorial.
- Numpy is a perfect tool for scientific computing and perfoming basic to advanced array operation.
- These library offer many hand features perfoming operations on n-arays and matrices in python. It helps to process arrays that store values of the same data type and makes performing math operations on array easier.

* Data analysis using Pandas

- Pandas is one of the most popular python libraries in data science.
- Pandas library provides support for data structure and data analysis tools.
- The library is optimized to perform data science task especially fast and efficiently.
- The basic principle behind pandas is to provide data analysis and modeling support for python that is similar to other languages.
- Pandas is suitable for structured, labelled data, in other words tabular data that has headings associated with each column of data.
- Pandas has two core data structures used to store data: Series and DataFrame

The series is one-Dimensional array like structure designed to hold single array of data and an associated array of data labels, called and index.

The dataFrame represent tabular data, a bit like a spreadsheet. DataFrames are organised into columns and each column can store a single data type such as floating point numbers, string, etc. DataFrame can be indexed by either their row and columns name.

(6) Ploting data Using Matplotib.

→ Matplotib is a comprehensive library for creating static, animated, and interactive visulization in python.

→ It is a plotting library for the python programming language. It allows to make quality charts in few lines of code.

Most of the other python plotting library are build on top of Matplotib.

→ The library is currently limited to 2D output, but it still provides you with the means to express graphically the data patterns.

* Features : It supports large number of graph and chart.
- It can also be used to make 3D plot and animation.
- It provides an OOP API for embedding plots into application using genral purpose GUI toolkits.

Chapter : 3

Getting Your hands dirty with Data.

(1) Explain magic function in python.

→ Jupyter provides a number of so called magic commands that can be used in code cell to simplify common task. Magic commands are interpreted by jupyter and, for Instance, transformed into python code before content is passed on to the kernel for execution.

→ Jupyter has set of predefined 'magic function' that you can call with command line syntax. There are two kinds of magics, line-oriented, cell-oriented.

→ Line magics are prefixed with "%" and work like a command typed in your terminal. Lines magic can be used as expression and their return value can be assign to variable.

→ Cell magic work on a block of code not on a single line. They are prefixed with "%%". To close block of code, when you are inside a cell magic function hit enter twice.

(2) Remove duplicate data.

→ Duplicate data can creates the problem for data science project. If database is large, then processing duplicate data means wasing of time.

→ Finding duplicates is important because it will save time, space false result.

→ We can easily and efficiently remove duplicate data using drop-duplicates().

```
import pandas as pd

rawData = {'firstName': ['Harsh', 'Smit', 'Harsh'],
           'LastName': ['Porwal', 'Patel', 'Porwal'],
           'RNo' : [11, 22, 33]
           'Marks': [50, 70, 80]}

df = pd.rawData (rawData , columns =
    ['FirstName', 'LastName', 'RNO', 'Marks'])

df.drop-duplicates ('firstName')
df
```

Output :

firstName	LastName	RNo	Marks
Harsh	Porwal	11	50
Smit	Patel	22	70

In [6]:
```
#Remove duplicates data using PANDAS

import pandas as pd
rawData = {'FirstName':['Harsh','Yash','Payal','Trisha','Harsh'],
           'LastName':['Porwal','Porwal','Vanzara','Parekh','Porwal'],
           'Rno': [11,22,33,44,55],
           'Marks':[60,70,90,80,60] }

df = pd.DataFrame(rawData,columns = ['FirstName','LastName','Rno','Marks'] )
df
```

Out[6]:

	FirstName	LastName	Rno	Marks
0	Harsh	Porwal	11	60
1	Yash	Porwal	22	70
2	Payal	Vanzara	33	90
3	Trisha	Parekh	44	80
4	Harsh	Porwal	55	60

In [9]:
```
df.drop_duplicates('FirstName')
```

Out[9]:

	FirstName	LastName	Rno	Marks
0	Harsh	Porwal	11	60
1	Yash	Porwal	22	70
2	Payal	Vanzara	33	90
3	Trisha	Parekh	44	80

(3) Explain Categorical Variables.

-> Categorical variable is one of the variable that has specific value from a limited selection of values. The number of value usually fixed.

-> categorical features can only take on a limited, and usually fixed, number of possible values. For example, if dataset is about information related to user, then you will typically find feature like country, gender, age, etc.
Alternatively, if the data you are working with is related to product, you will find feature like Product type, manufacturer, seller, etc.

```
import pandas as pd

c1 = pd.Categorical([1, 2, 3, 4, 5, 6])
print("\n c1 :", c1)

c2 = pd.Categorical(['e', 'm', 'f', 'i',
                     'f', 'e', 'h', 'm'])
print("\n c2 :", c2)
```

Output :-

```
c1 : [1, 2, 3, 4, 5, 6]
Categories (6, int64) : [1, 2, 3, 4, 5, 6]

c2 : ['e', 'm', 'f', 'i', 'f', 'e', 'h', 'm']
Categories (5, object) : ['e', 'm', 'f', 'i', 'h']
```

In [10]:
```
# numpy.pandas.Categorical()

# importing libraries
import numpy as np
import pandas as pd

# Categorical using dtype
c = pd.Series(["a", "b", "d", "a", "d"], dtype ="category")
print ("\nCategorical without pandas.Categorical() : \n", c)


c1 = pd.Categorical([1, 2, 3, 1, 2, 3])
print ("\n\nc1 : ", c1)

c2 = pd.Categorical(['e', 'm', 'f', 'i',
                     'f', 'e', 'h', 'm' ])
print ("\nc2 : ", c2)
```

```
Categorical without pandas.Categorical() :
 0    a
1    b
2    d
3    a
4    d
dtype: category
Categories (3, object): ['a', 'b', 'd']


c1 :  [1, 2, 3, 1, 2, 3]
Categories (3, int64): [1, 2, 3]

c2 :  ['e', 'm', 'f', 'i', 'f', 'e', 'h', 'm']
Categories (5, object): ['e', 'f', 'h', 'i', 'm']
```

(4) Explain Date and times with Pandas.

-> Dates are often provided in different formats and must be converted into single format DateTime object before analysis.

-> Python provides two method of formating date and time.

str() -> It converts dateTime value into string without any formatting.

strftime() function -> It define how user want the determine value to appear after conversion.

```
import datetime

x = datetime.datetime.now()
Current-Time = Str(x)

print("current time is :", current_Time)
```

```
current time is : 2022-08-09 13:55:07.460518
```

```python
# Getting current date and time using now().

# importing datetime module for now()
import datetime

# using now() to get current time
x = datetime.datetime.now()
current_time = str(x)

# Printing value of now.
print("Current Time is :", current_time)
```

```
Current Time is : 2022-09-05 14:55:33.543536
```

(6) Finding Missing data

-> Data can have missing values for a number of reason such as observation that were not recorded and data corruption. Handling missing data is important as many machine learning algorithms do not support data with missing values.

-> In python specifically Pandas, Numpy and scikit-learn, we mark missing value as NaN. Values with NaN value are ignored from arithmetic operations.

-> Pandas dataframe provides a function isnull() it returns a new dataframe of same size as calling dataframe, it contains only True and False. With True at the place NaN in original dataframe and False at other place.

```
import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(3,3),
    index = ['a', 'c', 'e'], columns =
    ['one', 'two', 'three'])
dfr = df.reindex(['a', 'b', 'c'])
print ("NaN replace with 0 :")
print (dfr.fillna(0))
```

	one	two	three
a	0.523	0.869	0.725
b	NaN	NaN	NaN
c	0.869	0.324	0.123

In [49]:
```
#Finding missing value and replace NaN with 0

import pandas as pd
import numpy as np

df = pd.DataFrame(np.random.randn(3, 3), index=['a', 'c', 'e'],columns=['one',
'two', 'three'])

df = df.reindex(['a', 'b', 'c'])


print ("NaN replaced with '0':")
print (df.fillna(0))
```

```
NaN replaced with '0':
        one       two     three
a -1.287786 -0.276673 -1.261441
b  0.000000  0.000000  0.000000
c  0.067903 -1.145021  0.728119
```

(6) Explain Bag of words.

-> Bag of words simply referes to a matrix in which the rows and documents and the columns are words.

-> The value matching a document with word in the matrix could be count of word occurrence within the document or use tf-idf.

-> Classifier are used to train the bag of words and a special kind of algorithm used to break words down into categories.

-> Traditionally, text document represented in NLP as a bag-of-words. This means that each document is represented as a fixed-length vector with length equal to the vocabulary size.

-> Each dimension of this vector corresponds to the count or occurence of a word in a document. Being able to reduce variable length documents to fixed length vectors makes them more amenable for use with a large variety of ML model.

(7) Explain TF - IDF.

-> TF-IDF stands for Term frequency inverse document Frequency of record.

-> It can be defined as the calculation of how relavant a word in a series or corpus is to a text.

-> The meaning increases proportionally to the number of times in the text a word appears but is compensated by the word frequency in the corpus. (dataset).

-> Term frequency : It represent the number of instance of a given word t.
Therefore we can see that it becomes more relavant when a words appear in the text, which is rational.
Since the ordering of term is not significant we can use a vector to describe the text in the bag of term model.

-> Inverse document Frequency : It test how relavant the word is. The key aim to the search is to locate the appropriate records that fit the demand. Since consider all term equally significant, it is not only therefore not only possible to use the term frequencies to measure weight of the term in the paper.

Chapter : 4

Data visulization.

(1) Explain Labels, Annotations, and Legends.

-> **Labels** : Labels help people to understand the significance of each axis of any graph you create.
Without labels, the values portrayed don't have any significance. In addition to a such as rainfall you can also add units of measure, such as inches or centimeters, so that your audience knows how to interpret the data shown.

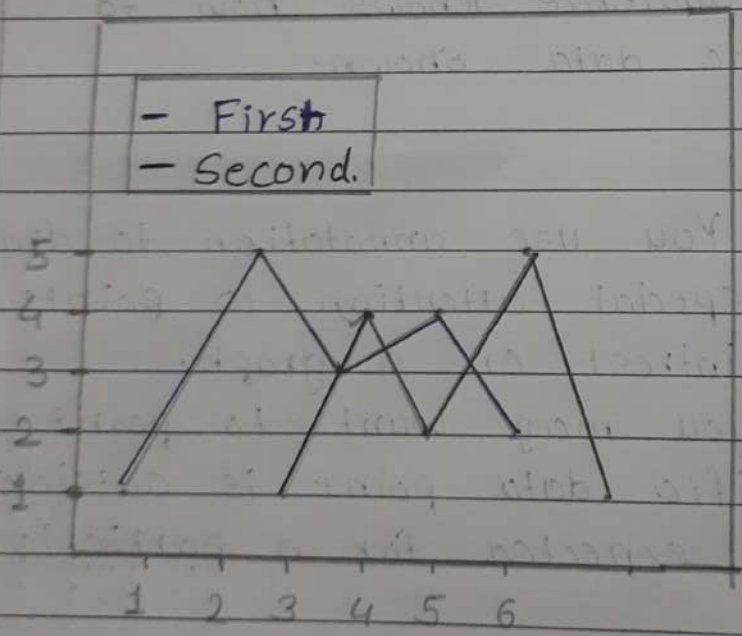
-> **Annotating** : You use annotation to draw special attention to points of intrest on a graph.
For example, you may want to point out that a specific data point is outside the usual range expected for a particular data set.

-> **Legend** : A legend document the individual elements of plot. Each line is presented in a table that contain a lable for it so that people can differentiate between each line.

Ex.

```
impot matplotlib.pylot as plt

values = [1, 5, 3, 4, 2]
values2 = [3, 4, 2, 5, 1]
plt.xlabel('Entries')
plt.ylabel('Values')
plt.plot(range(1,11), values)
plt.plot(range(1,11), values2)
plt.legend([['First', 'Second'], loc=0)
plt.show
```



In [53]:

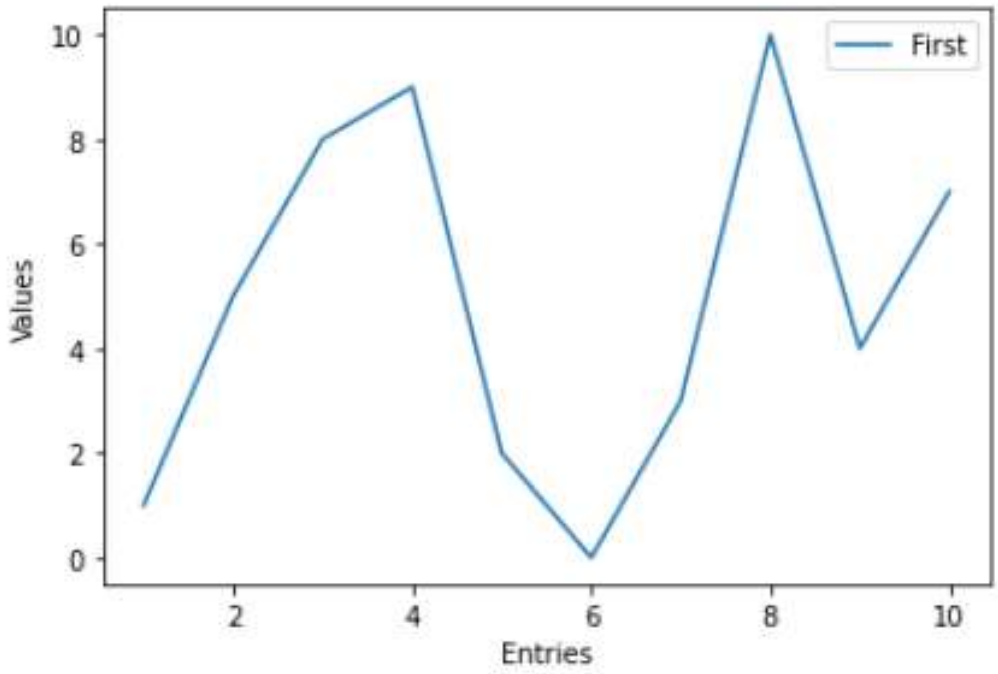
```
#Programm for label ,legend , annotate

import matplotlib.pyplot as plt

values = [1,5,8,9,2,0,3,10,4,7]
values2 =[3,5,6,2,8,4,9,2,6,3]
plt.xlabel('Entries')
plt.ylabel("Values")
plt.plot(range(1,11),values)

plt.legend(['First','Second'] , loc=0)

plt.show()
```



(2) Explain Data visulization.

-> A type of graph is which a circle is divided into sector that each represent a propotion of whole. Each sector shows the relative size of each value.

-> A pie chart display data, information and statics is an easy to read "pie slice" format with varying slice sizes telling how much of one data element exists.

-> Pie chart is also known as circle graph. The bigger the slice, the more of that particular data was gathered. The main use of pie chart is to show comparision.

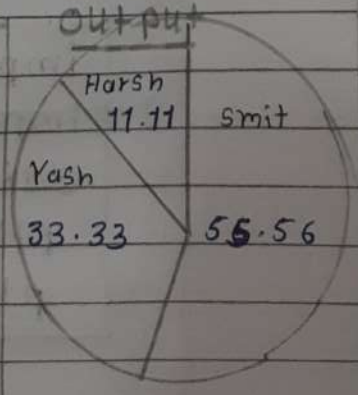
-> Pie chart can be drawn using the function pie() in the pyplot module.

-> By default the pie() function of pyplot arranges the pies or wedges in a pie chart in counter clockwise direction.

Ex.

```
import matplotlib.pyplot as plt
pl = ['Harsh', 'Yash', 'Smit']
marks = [1, 3, 5]
figureObject, axesObject = plt.subplot()
axesObject.pie(marks, labels = pl,
               autopct = '%.1.2f', startangle = 90)
axesObject.axis('equal')
plt.show()
```

Output

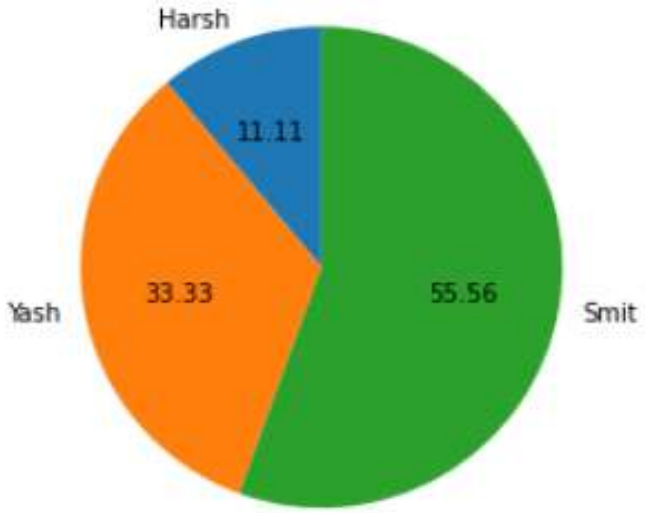


In [69]:
```
# Data Visulization with pie chart

import matplotlib.pyplot as plt

pl= ['Harsh','Yash','Smit']
marks=[1,3,5]
figureObject,axesObject=plt.subplots()
axesObject.pie(marks,labels=pl,autopct='%0.2f',startangle=90)
axesObject.axis('equal') #optional

plt.show()
```



(3) Explain Box-plots.

-> Box plots using boxes and lines to depict the distribution of one or more groups of numeric data.

-> Box plots are used to show distributions of numeric data values, especially when you want to compare them b/w multiple groups.

-> A box plot may also have lines, called whisker, indicating data outside the upper and lower quartiles.

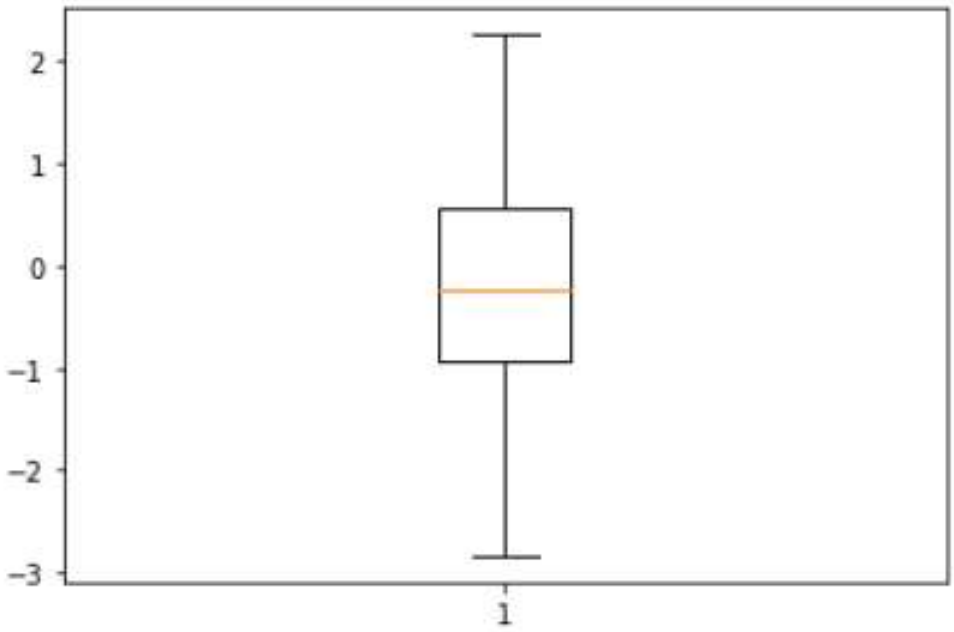
-> In most cases, It is possible to use numpy or python objects, but pandas objects are preferable because the associated name will be used to annotate the axes.
Aditionally, you can use categorical types for the grouping variables to control the order of plot element.

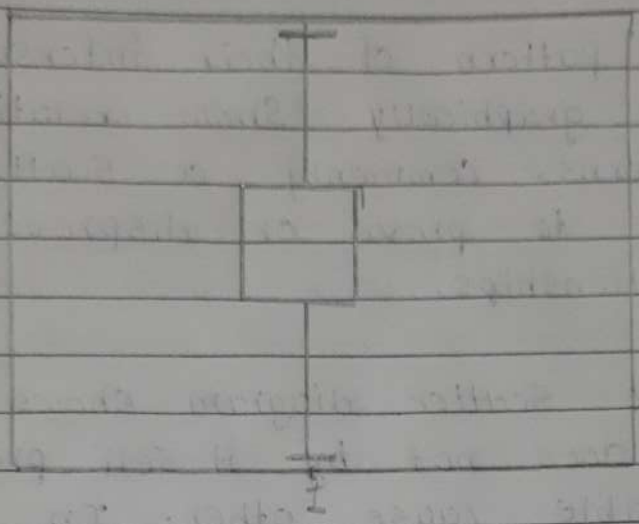
```
import numpy as np
import matplotlib.pylot as plt
data = np.random.randn(100)
plt.boxplot(data)
plt.show()
```

In [70]:
```
# Program for Box-Plots

import numpy as np
import matplotlib.pyplot as plt

data = np.random.randn(100)
plt.boxplot(data)
plt.show()
```





(4) Explain Data pattern using Scatter plots.

-> It displays collection of all the points for the set of data limited only for two values. Also called scatter plot, x-Y graph.

-> While working with statical data it is often observed that there are connection blw sets of data.

-> To find out wheather or not two sets of data are connected scatter diagrams can be used.

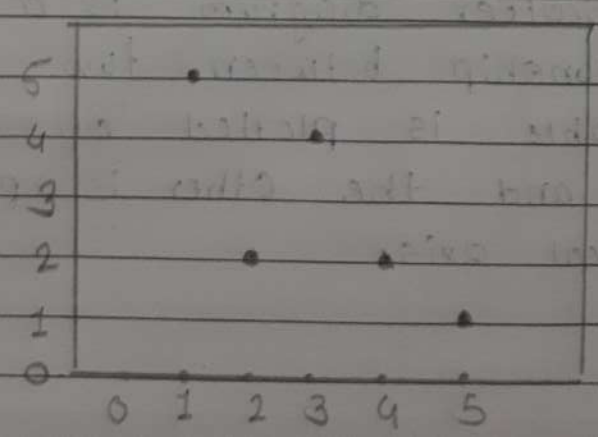
-> A scatter diagram is a tool for analyzing relationship between two variables. One variable is plotted on the horizontal axis and the other is plotted on the vertical axis.

→ The pattern of their intersecting points can graphically show relationship patterns. Commonly a scatter diagram is used to prove or disprove cause-and-effect relationships.

→ While scatter diagram shows relationships it does not by it self prove that one variable cause other. In addition to showing possible cause and effect relationships, a scatter diagram can show that two variables are from a common cause that is unknown or that one variable can be used as a surrogate for the other.

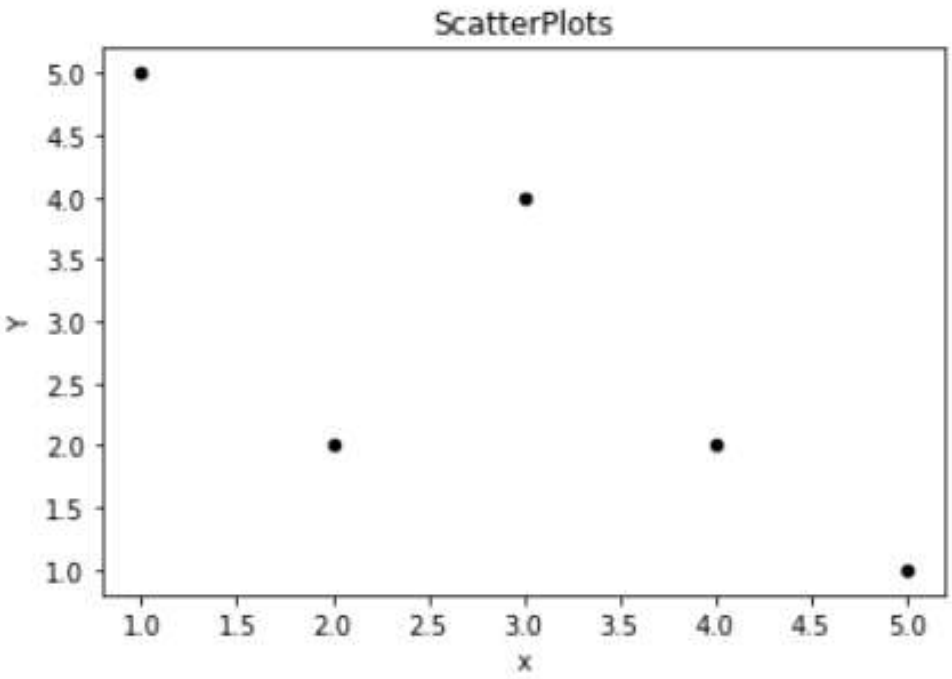
Ex.

```
import matplotlib.pyplot as plt
x = [1, 2, 3, 4, 5]
y = [5, 2, 4, 2, 1]
plt.scatter(x, y, label = "skitcat", color = 'k',
                  s = 20, marker = 'o')
plt.xlabel("x")
plt.ylabel("y")
plt.show()
```



In [10]:
```
# Programm for ScatterPlots

import matplotlib.pyplot as plt
x =  [1,2,3,4,5]
y =  [5,2,4,2,1]
plt.scatter(x,y,label="SkitsCat",color='k',s=20,marker ="
plt.xlabel("x")
plt.ylabel("Y")
plt.title("ScatterPlots")
plt.show()
```



(6) Explain Data visulization using Matplotlib.

-> Data visulization is the process of presenting data in the form of graph or chart.
It helps to understand large and complex ammounts of data very easily.
It allows decision maker to make decision very efficiently and also allows them in identifying new trends and pattern easily.

-> It is also used in high level data analysis for machine learning and exploratory data analysis.

-> Matplotlib is low-level library of python which is used for data visulization.
It is easy to use and emulates MATLAB like graphs and visulization.
This library is buit on the top of NumPy array and consits of serval plots like line chart, bar chart, histogram, etc.

-> The various plots we can utilize using pyplot are line plot, Histogram, scatter plot, Image, contour, and polar.

-> Ans of (3)

-> Ans of (4)

Chapter : 5

Data wrangling

(1) Explain data wrangling.

-> Data wrangling is the process of transforming data from it's original 'raw' form into a more digestable format and organization sets from various sources into a single coherent whole for further processing.

-> It is also called data munging.

-> In other words, It is making raw data usable. It provides substance for further proceedings.

-> Data wrangling cover following processes :

- Getting data from various place into one place.
- Piecing data together according to determined setting.
- cleaning the data and missing elements.

-> Data Exploration : In this process, the data is studied, analyzed and understood by visulizing representation of data.

→ Structuring : This means organizing the data, which is necessary because raw data comes in many different shape and sizes. A single column may turn into serval rows for easier analysis.

→ Cleaning : Most of the datasets having a vast ammount of data contain missing values of NaN, they are needed to be taken care of by replacing them with most frequent value in column or simply dropping field having NaN value.

→ Enriching : In this process data is manipulated according to the requirments, where new data can be added or pre-existing data can be modified.

→ Validating : Validation rules are repetetiv programming sequences that verify data consistency, quality and security.

-> Publishing : Analysts prepare the wrangled data for use down stream, wheather by a particular user or software and document any particular steps taken or logic used to wrangle said data.

(2) List and explain Interface of Scikit-Learn. (classification)

-> Scikit-Learn takes a highly OOP approach to machine learning models.
Every major scikit-learn inherits from sklearn.base.BaseEstimator.

-> All object from scikit-learn share a uniform common basic API consisting of three complemantary Interface : estimator interface for building a fitting models, a predictor for making prediction and transformer for converting data.

(i) Estimator : It is at the core of library. It's define instantiation mechanism of object and expose a fit. method for learning model from training data.

- All supervised and unsupervised learning algorithms are offered as object implementing this interface. ML tasks like feature extraction, feature selection are also provided as estimators.

(ii) Predictors : The predictor interface extends the notion of an estimator by adding a predict method that takes an array X test and produce prediction for X test based on learned parameter of estimator.

- In the case of supervised learning estimators, this method typically returns the predicted labels or values computed by the model.

(iii) Transformers : Since it is common to modify and filter data before feeding it to a learning algorithm, some estimator in library implement a transformer interface which defines a transform method.

- It takes as input some new data X test and yields as output a transformed version of X test.

(3) Explain Exploring Data Analysis and Perfoming t-tests.

→ EDA is a genral approach to exploring datasets by mean of simple summary statics and graphic visulizations in order to gain a deeper understanding of data.

→ EDA is an approach for data analysis that employs a variety of techniques to

- Maximize insight into a dataset.
- Uncover Underlying Structure.
- Extract important variable.
- Detect outliners and anomalies.
- Test underlying assumptions.
- Determine optimal factor settings.

→ With EDA, following function are perfomed:

- Describe of user data.
- Closely explore data distribution.
- Understand the relation blw variables.
- Notice unusual and unexpected situations.
- Place data into group.
- Take note of group difference.

→ Descriptive statics helpful way to understand characterstics of your data and get quickly summary of it. Pandas in python provides an intresting method describe().

-> The t-test is a parametric test for comparing two sets of continuous data. It is used for comparing two sample means.

-> The scipy.stats function ttest_1samp can be used to calculate a T-statics and a P-value for given mean value compared to a sample population.

scipy.stats.ttest_ind_from_stats(mean1, std1, nobs1, mean2, std2, nob2, equal_var=True) [source]

-> mean1 : The means of sample 1.
-> std1 : The standard deviation of sample 1.
-> nob1 : The number of observation of sample 1.
-> mean2 : The mean of sample 2.
-> std2 : The standard deviation of sample 2.
-> nob2 : The number of observation of sample 2.
-> equal_var : If true, perform a standard independent 2 sample test that assume equal population variances

(a) Explain covariance and correlation.

→ Covariance : It is the first measure of the relationship of two variables. It determine wheather both variables have a coincident behaviour with respect to their mean.

- If the single value of two variables are usually above or below their respective averages, the two variable have positive association.

- Covariance is a quantitative measure of the extent to which the deviation of one variable from its mean matches the deviation of the other from its mean.

→ Correlation : correlation b/w to random variables, $\varsigma(x, y)$ is the covariance of the two variables normalized by the variance of each variable. This normalization cancels the units out and normalizes the measure so that it is always in the range $[0, 1]$.

- When people use term correlation, they are actually reffering to specify type of correlation called "Pearson" correlation. It measures the degree to which there is a linear relationship b/w the two variables.

- The alternative measure is "Spearman" correlation, which has a formula almost identical to "Pearson" with the exception that the underlying random variables are first transform into their rank.

→ Difference blw covariance and correlation

- Variance is a measure of variability from the mean.
- Covariance is a measure of relationships blw the variability of 2 variables.
- Covariance is scale dependent because it is not standardized.

- Correlation is a relationship blw the variability of 2 variables.
Correlation is standardized, it's not scale dependent.

(6) Explain Hashing with hash function and hashing trick.

-> The class featureHasher is a high-speed, low-memory, vectorizer that uses a technique known as feature hashing or hashing trick.

-> Instead of building hash table for the feature, apply hash function to the features to determine their column index in sample matrices directly.

-> The result is increased speed and reduced memory usage, at the expense of inspectability.

-> Advantages:

- Handling large data matrices based on text on the fly.
- Fixing unexpected values or variables in textual data.
- Building scalable algorithms for large collection of documents.

* Hash function

-> It can transform any input into an output whose characterstics are predictable. Usually they return a value where the output is bound at specific Interval.

-> In genral, Good hash function is one where a small change in input text will cause a large change in output.

-> In a certain sense, hash function are like a secret code, transforming everything into numbers.
You can't convert the hashed code to it's original value.

* Hashing Trick

-> There is a solution called hashing trick because it is based on the hash function and can deal with box, text and categorical variables in int or string form.

-> It can also work with categorical variables mixed with numeric value form quantative features.

-> Hashing trick can univocally map a value to its position without any prior need to evaluate the feature because it levereges the core charactestics of hashing function.

-> A real problem with the hashing trick is the eventuality of a collision, which happens when two different tokens are associated to the same index.

→ Cosequently, if collision happens probably it will involve two unimportant tokens. When using the hashing trick, probably is on your side because with a large enough output vector, though collision are always possible, it will be highly unlikely that they will involve important elements of the model.

(6) Explain Supervised and Unsupervised ML.

Supervised.	Unsupervised
• Algorithm are trained using labeled data.	Algorithm are trained used against data that is not labeled.
• Computational is simpler.	Computational is complex.
• Highly accurate.	Less accurate.
• No. of classes is known	No. of classes is unknown
• Support vector machine, Neural network, etc	Apriori algorithm, etc.
• Use offline analysis	Use real-time analysis

(1) List and explain different coding style in Python.

→ Different coding style can be chosen for different problems. There are four different coding style.

- Functional
- Imperative
- Object-oriented
- Procedural

→ Functional Coding : In this type of coding, every statment is treated as mathumatical equation and mutable data can be avoided. Most of programmers prefer this type of coding for recursion and lambda calcus.

- The merit of this functional coding is it works well for parallel processing, as there is no state to consider. It is mostly preffered by data scientist.

→ Imperative Coding : When there is change in the program, computation occurs. Imperative style of coding is adopted, if we have to manipulate the data structure. This style establish the program in a simple manner.

- It is mostly used by data scientist. because of its easy demonstration.

→ Object-oriented coding : This type of coding relies on the data fields, which are treated as objects. These can be manipulate only through prescribe method.

- Python does not support this paradigm completely, due to some feature like encapsulation can not be implemented.
- This type of coding support code reuse.

→ Procedural Coding : Usually most of the people begin learning a language through procedural code, where the tasks proceed a step at a time.

It is the simplest form of coding.

- It is mostly used by non-programmers as it is a simple way to accomplish simpler and experimental tasks.
- It is used for iteration, sequencing, selection.

(2) Compare Pandas and NumPy

Pandas	NumPy
• Using this we can work on tabular data.	Using this we can work on numerical data.
• Powerful tools is Data frame and series.	Powerful tool is Arrays.
• Better perfomance when no. of row is 500K +	Better perfomance when no. Of row is less than 50K.
• Indexing is very slow.	Indexing is very fast.
• It is used in lot of organization like. Trivago, Abeja., etc.	It is being used in organization like walmart, Instacart, etc.
• Consume more memory	It is memory efficient.
• It has higher industry application.	It has a lower industry applications.

(3) Explain GroupBy in Pandas.

→ Pandas groupby is used for grouping the data according to the categories and apply a function to the categories

→ Pandas dataframe.groupby() function is used to split data into groups based on some crietria.
Pandas object can be split on any of their axis.

```
import pandas as pd

df = pd.read_csv("nba.csv")
df
```

	Name	Team	Marks
0	Harsh	A	80
1	Yash	A	75
2	Trisha	B	82
3	Payal	B	90

```
gk = df.groupby('Team')
gk.first()
```

	Team	Name	Marks
0	A	Harsh	80
1	A	Yash	75
2	B	Trisha	82
3	B	Payal	90

In [22]:
```
# GroupBy in Pandas

import pandas as pd

df = pd.DataFrame({'Name': ['Harsh', 'Yash','Smit', 'Prayag'],
                   'Speed': [380, 370.8, 24, 26.4],
                   'Surname':['Porwal','Porwal','Patel','Acharya']})
df
```

Out[22]:

	Name	Speed	Surname
0	Harsh	380.0	Porwal
1	Yash	370.8	Porwal
2	Smit	24.0	Patel
3	Prayag	26.4	Acharya

In [21]:
```
gp = df.groupby(['Surname'])
gp.first()
```

Out[21]:

	Name	Speed
Surname		
Acharya	Prayag	26.4
Patel	Smit	24.0
Porwal	Harsh	380.0

(4) Explain web scrapping using beautiful Soup. or
Explain Parsing HTML documents using beautiful soup.

→ Beautiful soup is a python library for pulling data out of HTML and XML files.

→ Beautiful soup helps you pull particular content from a webpage, remove the HTML Markup, and save the information.
It is a tool for web scraping that helps you clean up and parse the documents you have pulled down from the web.

→ Import beautiful Soup constructor

```
from bs4 import BeautifulSoup
```

→ The beautifulSoup constructor function takes in two string arguments, (a) The HTML string to be parsed, (b) optionally, the name of parser.

Ex

```
from bs4 import BeautifulSoup
htmltxt = "<p> Hello World </p>"
Soup = BeautifulSoup (htmltxt, 'lxml')
soup.text
```

Output

```
Hello world.
```

In [20]:
```
from bs4 import BeautifulSoup
htmlTxt = "<P>Hello World</P>"
soup = BeautifulSoup(htmlTxt,'lxml')
soup.text
```

Out[20]:
```
'Hello World'
```

(5) Read data in Python

→ Read CSV

- read_csv() is used to read the comma seprated value file into a pandas data frame.

- Syntax :

pd.read_csv(filepath, sep, header, index_col)

Ex.

```
import pandas as pd

df = pd.read_csv('Marks.csv', index_col=0,
                 header=0)
print(df)
```

	POS	Algo
101	50	75
102	60	80
103	55	85

-> Read Excel

- Read an excel file into a pandas dataframe.
- Supports xls, xlsx, xlsm, xlsb, odf and odt file extension read from local file system. Supports an option to read a single sheet or a list of sheet.
- read-excel() function is used
- Syntax :

```
pd.read-excel(fileName, sheet-name,
              header, index-col)
```

Ex.

```
import pandas as pd

df = pd.read-excel('record.xlsx', sheet-name=
                   'Employees')

print(df)
```

	EmpId	EmpName	Role
0	11	Harsh	CEO
1	22	Smit	Editor
2	33	Yash	MD

In [38]:
```
# Program to show various ways to
# read data from a file.

# Creating a file
file1 = open("myfile.txt", "w")
L = ["This is Delhi \n", "This is Paris \n", "This is London \n"]

# Writing data to a file
file1.write("Hello \n")
file1.writelines(L)
file1.close() # to change file access modes

file1 = open("myfile.txt", "r+")

print("Output of Read function is ")
print(file1.read())
print()

# seek(n) takes the file handle to the nth
# bite from the beginning.
file1.seek(0)

print("Output of Readline function is ")
print(file1.readline())
print()

file1.seek(0)

# To show difference between read and readline
print("Output of Read(9) function is ")
print(file1.read(9))
print()

file1.seek(0)

print("Output of Readline(9) function is ")
print(file1.readline(9))
print()

file1.seek(0)

# readlines function
print("Output of Readlines function is ")
print(file1.readlines())
print()
file1.close()
```

```
Output of Read function is
Hello
This is Delhi
This is Paris
This is London


Output of Readline function is
Hello


Output of Read(9) function is
Hello
Th

Output of Readline(9) function is
Hello


Output of Readlines function is
['Hello \n', 'This is Delhi \n', 'This is Paris \n', 'This is London \n']
```

In [63]:
```
# Write a program using Numpy to count number of “P” element wise in a given array.

import numpy as np

x1 = np.array(['Python', 'PHP', 'JS', 'examples', 'html'])
print("Original Array:", x1)
print()
r = np.char.count(x1, "P")
print("Number of ‘P’:" , r)
```

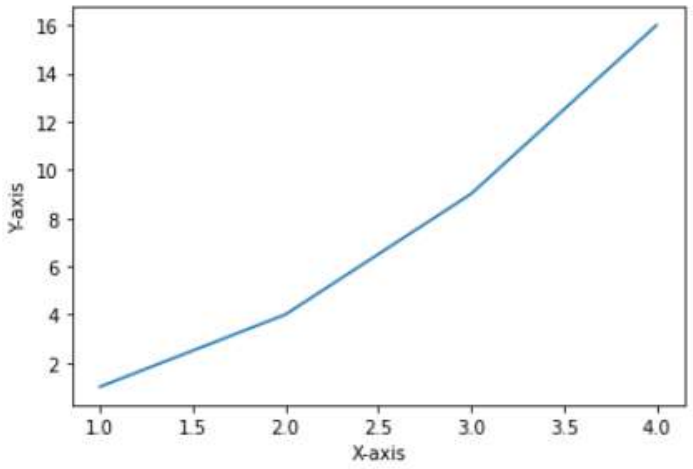
```
Original Array: ['Python' 'PHP' 'JS' 'examples' 'html']

Number of ‘P’: [1 2 0 0 0]
```

In [68]:
```
# Write a simple python program that draws a line graph where x= [1,2,3,4]
#and y = [1,4,9,16] and gives both axis label as “X-axis”and “Y-axis”.

import matplotlib.pyplot as plt
import numpy as np

# define data values
x = np.array([1, 2, 3, 4]) # X-axis points
y = np.array([1,4,9,16]) # Y-axis points
plt.xlabel('X-axis')
plt.ylabel('Y-axis')
plt.plot(x, y) # Plot the chart
plt.show() # display
```

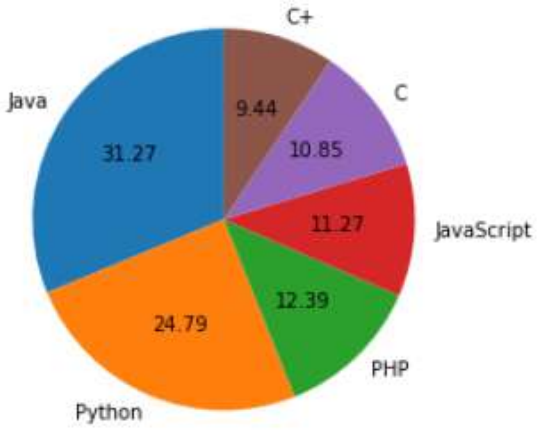


In [76]:
```
#Write a Python programming to create a pie chart with a title of
#the popularity of programming Languages.Sample data:
#Programming languages: Java, Python, PHP, JavaScript, C#,C++
#Popularity: 22.2, 17.6, 8.8, 8, 7.7, 6.7


import matplotlib.pyplot as plt

pl= ['Java','Python','PHP','JavaScript','C','C+']
Popularity=[22.2,17.6,8.8,8,7.7,6.7]
figureObject,axesObject=plt.subplots()
axesObject.pie(Popularity,labels=pl,autopct='%0.2zf',startangle=90)
axesObject.axis('equal') #optional

plt.show()
```



In [10]:
```
# Write a program to check whether the given number is prime or not.

num = int(input("Enter a number :"))
# If given number is greater than 1
if num > 1:
    # Iterate from 2 to n / 2
    for i in range(2, int(num/2)+1):
        # If num is divisible by any number between
        # 2 and n / 2, it is not prime
        if (num % i) == 0:
            print(num, "is not a prime number")
            break
    else:
        print(num, "is a prime number")
else:
    print(num, "is not a prime number")
```

```
Enter a number :5
5 is a prime number
```

In [11]:
```
# Write a program to check whether the given number is prime or not.

num = int(input("Enter a number :"))
for i in range(2, int(num/2)+1):
        if (num % i) == 0:
            print(num, "is not a prime number")
            break
        else:
            print(num, "is a prime number")
            break
```

```
Enter a number :12
12 is not a prime number
```

In [20]:
```
# Write a program to print following patterns.
# 1)
# *
# * *
# * * *
# * * * *

rows = 6
# outer loop
for i in range(rows):
    # nested loop
    for j in range(i):
        # display number
        print("*", end=' ')
    # new line after each row
    print('')
```

```
*
* *
* * *
* * * *
* * * * *
```

In [30]:
```
# Write a program which takes 2 digits, X,Y as input and generates a 2-dimensional array
#of size X * Y. The element value in the i-th rowand j-th column of the array should be i*j.

row_num = int(input("Input number of rows: "))
col_num = int(input("Input number of columns: "))
multi_list = [[0 for col in range(col_num)] for row in range(row_num)]

for row in range(row_num):
    for col in range(col_num):
        multi_list[row][col]= row*col

print(multi_list)
```

```
Input number of rows: 2
Input number of columns: 3
[[0, 0, 0], [0, 1, 2]]
```

In [38]:
```
# Write a program to interchange the List elements on two positions entered by a user

def swapPositions(list, pos1, pos2):

    list[pos1], list[pos2] = list[pos2], list[pos1]
    return list

List = [23, 65, 19, 90]
pos1, pos2  = 1, 3

print("Swapped List :",swapPositions(List, pos1, pos2))
```

```
Swapped List : [23, 90, 19, 65]
```

In [49]:
```
# Factorial of a number using recursion

def factorial(n):
    if n == 1:
        return n
    else:
        return n*factorial(n-1)

num = int(input("Enter a number :"))
print("The factorial of", num, "is", factorial(num))
```

```
Enter a number :7
The factorial of 7 is 5040
```

In [45]:
```
# Example of factorial program in Python using function

def find_factorial(n):
    f = 1
    for i in range(1, n + 1):
        f = f * i
    return f

x = int(input("Enter a number: ")) #for take input number by user
result = find_factorial(x)

print("Factorial is:", result)
```

```
Enter a number: 6
Factorial is: 720
```