THADOMAL SHAHANI TSEC ENGINEERING COLLEGE

CONTENTS

LO.

SR. NO.	EXPERIMENTS	~~PAGE NO.~~	DATE	TEACHERS SIGN.
1.	Python programs to handle decision making and control structures.	1-3 LO1	24/1/22	
2a.	Python programs to elaborate the topic of lists comprehension.	4-6 LO2	26/1/22	
2b.	To understand the use of dictionaries and tuples in Python.	7-9 LO2	10/2/22	
3.	Understanding the use of Numpy in Python programs.	10-12 LO2	26/2/22	
4.	Python programs to implement user-defined and anonymous fxns	13-15 LO2	28/3/22	11/04/22
5.	~~Basic shell scripting~~ Python programs on OOPs	LO3 16-18	9/3/22	
6.	~~Arithmetic~~ Python programs on operator overloading	LO3 19-21	7/3/22	
7.	Python programs on modules and packages.	LO4 22-24	18/3/22	
8.	Python programs on exception handling.	LO4 25-27	18/3/22	
9.	Study and implement file handling operations.	28-30 LO5	26/3/22	
10.	GUI application using Tkinter	LO5 31-33	9/4/22	

THADOMAL SHAHANI
TSEC
ENGINEERING COLLEGE

CONTENTS

SR. NO.	EXPERIMENTS	~~PAGE NO.~~ LO	DATE	TEACHERS SIGN.
11.	CRUD operations using pymsql python package and MySQL database connectivity.	34-36 LO5	26/3/22	
12.	Plots using Numpy and matplotlib.	LO6 37-39	28/3/22	
13.	Basic operations using Pandas	LO6 40-42	26/3/22	
14.	SciPy library and linear algebra using SciPy.	43-45 LO6	28/3/22	
15.	REST API in flask using Python.	46-48 LO6	09/4/22	

CLASSMATE
Date :
Page :

Lab Outcomes :

LO1 - Understand the structure, syntax and semantics of the Python language.

LO2 - Interpret advanced data types and functions in Python.

LO3 - Illustrate the concepts of object-oriented programming as used in Python.

LO4 - Create Python applications using modules, packages, multi-threading and exception handling.

LO5 - Gain proficiency in writing File Handling programs, also create GUI applications and evaluate database operations in Python.

LO6 - Design and develop cost-effective robust applications using the latest Python trends and technology.

Lab Assignment - 1

24/01/22

Ishika Khokhani
S13-39

☆ Aim:

Python programs to handle decision making and control structures. (LO1)

LO1

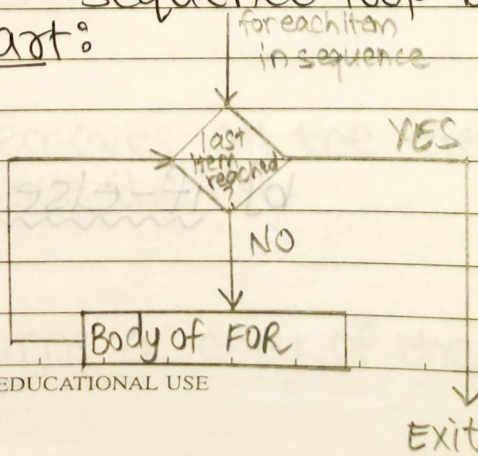
☆ Theory:

① FOR loop -

→ It is used for iterating a sequence (list, tuple, set etc.).
→ A set of statements can be executed with this loop.
→ It does not require an indexing variable to be set beforehand.
→ Iterating over a sequence is called 'traversal'.
→ Syntax:

```
'for' variable takes value of item in sequence loop body.
```

→ Flowchart:



② 'Print' statement -

→ can take multiple values.
→ has a default separator = " "
→ has a default terminator = "\n".
→ Syntax -

```
print(val1, val2 . . )
```

③ 'Range' function -

→ stops at 1
→ step determines increment.
→ returns sequence of numbers,
→ between the given range.
→ syntax -

```
range(stop)
or range(start, stop)
or range(start, stop, sleep)
```

④ selection/decision control statement -

a] simple-'if':
runs the code if the given statement is true.

b] 'if-else':
executes 'if' the condition is true, 'else if' the condition is false.

Lab Assignment - 2(a)

31/1/22

Ishika Khokhani
S13-39

★ Aim:

Python programs to elaborate the topic of lists and list comprehension. (LO2)

★ Theory:

→ A list is a data structure in python that is mutuable or changable, ordered sequence of elements.
→ Each element or value inside of a list is called an 'item'.
→ Just as strings are defined as characters between quotes, lists are defined by having values between square brackets '[]'.

List functions -

1) append():
Adds an element at the end of the list.

2) clear():
Removes all the elements from the list.

3) copy() -
returns a copy of the lists.

4) count()-
returns the number of elements with the specified value.

5) extend()-
Add the elements of a list (or any iterable), to the end of the current list.

6) index()-
returns the index of the first element with the specified value.

7) insert()-
Adds an element at the specified position.

8) pop()-
Removes the element at the specified position.

9) remove()-
removes the first item with the specified value.

10) reverse()-
reverses the order of the list.

11) sort()-
sorts the list

- List comprehension-

→ They are used for creating new lists from other iterables like tuples, strings, arrays, lists etc.
→ It consists of brackets containing the expression, which is executed for each element along with the 'for' loop to iterate over each element.
→ syntax-

newLists=[expression(element) for element in old list 'if' condition]

03/03/24

Lab Assignment - 2(b)

31/10/22

Ishika Khokhani
S13-39

★ AIM:

To understand the use of dictionaries and tuples in Python programs. (L02)

★ THEORY:

- **Dictionaries -**

→ They are made up of key:value pairs.
→ They are organised and accessed using keys and values.
→ They are defined with curly braces.
→ syntax -

```
mydict = {key : value}
```

→ It is the most optimised type of data structure.
→ It is a non-homogenous data struture.
→ Items here are not indexed.
→ eg.:

```
mydict = {'Name': 'Ishika', 'Age' : 20}
print(mydict['Name'])
print(mydict ['Age'])
mydict['City'] = 'Mumbai'
print(mydict ['city'])
```

Adds new item →

o/p :- Ishika
20
Mumbai

- Tuples -

→ They are immutable lists.
→ Their elements cannot be modified, only accessed.
→ They are defined using parenthesis.
→ They are non-homogenous data structure that stores single row and multiple rows and columns.
→ They allow duplicate elements.
→ eg. :

i/p -

empty tuple →
```
mytuple = ()
print(mytuple)
```

```
mytuple = (1,2,3,4) → print(mytuple)
```

mixed datatypes →
```
mytuple = (1,2,"Hi",3,4)
myprint(mytuple)
```

nested type →
```
mytuple = ("world",[5,6,7],(1,2,3,4))
print(mytuple)
```

o/p :-
```
()
(1,2,3,4)
(1,2,'Hi',3,4)
('world',[5,6,7],(1,2,3,4))
```

Conch 9

03/03/22 A+

Lab Assignment - 3

10/2/22

Ishika Khokhani
SI3-39

★ AIM:

Understanding the use of Numpy in Python programs. (LO2)

★ THEORY:

→ Numpy is a python library used for working with arrays.
→ It also has functions for working in the domain of linear algebra, fourier theorem and matrices.
→ They are about 50 times faster than python lists.
→ The array object in numpy is called 'ndarray'.
→ For e.g:

```
import numpy as np
a = np.array([0,90,30,60,45])
print np.sin(a*np.pi/180)
print('\n')
print np.cos(a*np.pi/180)
print('\n')
print np.tan(a*np.pi/180)
```

O/P 8-

[0. 1. 0.5 0.86660254 0.70710678]

[1.000000e+00 6.12323e-17 8.66025e-01
 5.000000e-01 7.07106781e-01]

[0.0000e+00 1.63312394e+16 5.77350269e-01
 1.73205081e+00 1.0000000e+00]

Python Lab Assignment - 4

date 24/2/22

Ishika Khokhani
S13-39

Aim: Write python programs to implement user defined and anonymous functions. (LO2)

LO1

Theory: ★ Lambda Functions:

- Lambda function is a small anonymous function, i.e, it does not have a name.
- It can take any number of arguments, but can only have one expression.
- syntax -

  lambda 'arguments': 'expression'
- They can be used wherever function objects are required.
- Anonymous functions are declared by using the 'lambda' keyword.
- For e.g:-

  i/p:
```
x = lambda a : a+10
print(x)
print("sum= ",x(20))
```

→ Here 'a' is an argument, and 'a+1' is an expression which got evaluated and returned.

→ We first print the function object.

→ Then we print the result of the function after being evaluated.

o/p:

<function<lambda> at 0X0000019E285D16A8>
sum=30

→ For e.g ②:

```
i/p: def x(a):
         return a+10
     print(sum=x(10))
```

The above code yields the same result without the lambda function.

→ For e.g ③:

```
#Program to double each item in a
 list using map().
```

i/p:

```
mylist=[1,5,4,6,8,11,3,12]
newlist = list(map(lambda x:x*2,mylist))
print(newlist)
```

o/p:

[2,10,8,12,16,22,6,24]

→ For e.g ④:

```
i/p: def myfunc(n):
         return lambda a:a*n
     mytripler = myfunc(3)
     print(mytripler(11))
```

o/p: 33

☆Functions:

→ In Python, a function is a group of related statements that performs a specific task.
→ They help break our program into smaller and modular chunks and bits.
→ As our program code increases, they make it more organisad and manageable.
→ It also avoids duplication and make the code reusable.
→ syntax -

```
def myfunc():
    print("Hello There!")
myfunc()
```

→ examples-

①
```
def greeting(name):
    print("Hey, "+name+"!")
greeting("Tom")
```

o/p- Hey, Tom!

②
```
def myfunc():
    x=10
    print("Value inside = ",x)
x=20
print("Value outside = ",x)
```

o/p-

Value inside = 10
Value outside = 20

variable
- Here the 'x' inside the function is local to the function. Hence, it did not affect the value outside the function.

- There are two types of functions in Python:
  1. Built-in functions
  2. User-defined functions

- example for (1):

```
i/p - num = -30
      absnum = abs(num)
      print(absnum)

o/p - 30
```

- example for (2):

```
i/p - def sum_num(x,y):
          sum = x+y
          return sum
      num1 = 10
      num2 = 20
      print("sum = ", sum_num(
            num1, num2))

o/p - sum = 30
```

Concln ?

03/03/20 A-

Lab Assignment 5

7/3/22

Ishika Khokhani
S13-39

Aim: Python programs on OOPs (LO3)

Theory:

- **OOPs in Python-**

→ In python, object-oriented programming (OOPs) is a programming paradigm that uses objects and classes in programming.
→ It aims to implement real-life entities like inheritance, polymorphisms, encapsulation etc. in the programming.
→ Its main concept is to bind the data and the functions that work on that, together as a single unit so that no other part of the code can access this data.
→ It focuses on creating reusable code.
→ The major principles of OOPs system are:

- Class
- Method
- Object
- Inheritance
- Polymorphism
- Data abstraction
- Encapsulation

• Creating classes-

→ A class can be defined as a collection of objects.
→ It is a local entity that has some specific attributes and methods.
→ syntax:

```
class className:
    <statement-1>
    .
    .
    .
    <statement-N>
```

→ For example-

```
class automobile:
    Pass
```

• creating objects-

→ An object is an instantiation of a class.
→ When a class is defined, only the description for the object is defined. Therefore, no memory or storage is allocated.
→ For example-

i/p -

```
class car:
    def __init__(self,model,year):
        self.model = model
        self.year = year
    def disp(self):
        print(self.model,self.year)
c1 = car("BMW",2017)
c1.disp()
```

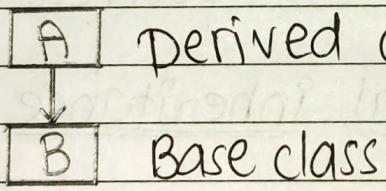
o/p-

BMW 2017

• Inheritance in python -

① single inheritance :

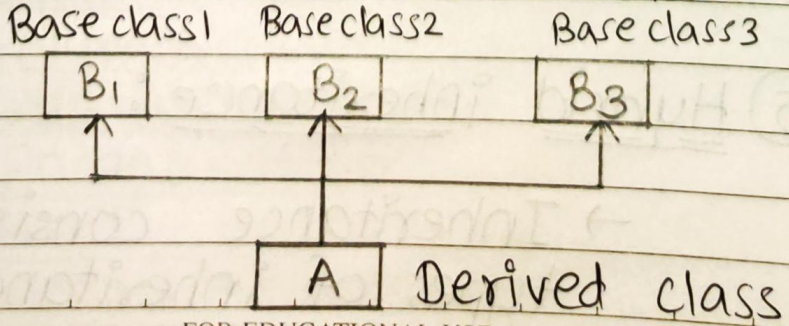
→ It enables a derived class to inherit properties from a single parent class, thus enabling code reusability and addition of new features to existing code.



② Multiple inheritance :

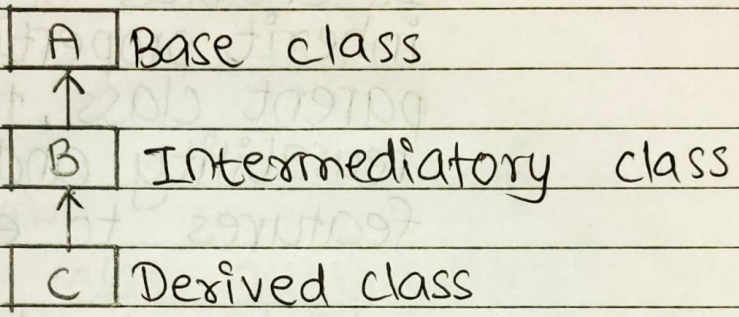
→ When a class can be derived from more than one base class, it is called multiple inheritance.

→ All features ~~are~~ of the base classes are inherited into the derived class.



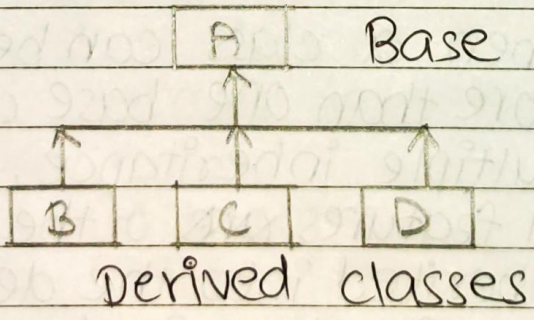
③ Multi-level inheritance:

→ In this, features of the base class and the derived class are further inherited into the new derived class.
→ This is similar to the relationship representing a child and grandfather.



④ Hierarchical inheritance:

→ When more than one derived classes are created from a single base.



Conclusion

30/03/22

⑤ Hybrid inheritance:

→ Inheritance consisting of multiple types of inheritance.

Lab Assignment 6

7/3/22

Ishika Khokhani
S13-39

Aim: Python programs on operator overloading and interface(CLO3)

Theory:

- **Polymorphism in OOPs -**

→ It is the ability of any data to be processed in more than one form.
→ 'poly' means many and 'morphism' means types.
→ It is one of the most important concepts of OOPs.
→ The most common use is when a parent class reference is used to refer to a child class object.
→ In python, Polymorphism lets us define methods in the child class that have the same name as the methods in the parents class.
→ However, it is possible to modify a method in a child class that it has inherited from the parent class.
→ It basically means the same function name being used for different types.

① Polymorphism with class methods:

→ We create a loop that iterates through a tuple of objects.
→ Then call the methods without being concerned about which class type each object is.
→ We assume that these methods actually exist in each class.

② Polymorphism with Inheritance:

→ In python, polymorphism lets us define methods in the class that have the same name as the methods in the parents class.
→ In Inheritance, the child class inherits the methods from the parent class.
→ ~~This particularly~~
→ However, it is possible to modify a method in a class that it has inherited from the parent class doesn't quite fit the child class.
→ In such cases, we re-implement the method in the child class.
→ This process of re-implementing a method in the child class is known as Method overriding.

③ Polymorphism with function and objects:

→ It is also possible to create a function that can take any object, allowing for polymorphism.

- **Python Interface -**

→ It is a collection of method signatures that should be provided by the implementing class.
→ It contains methods that are abstract in nature.
→ The abstract methods will have the only declaration as there is no implementation.

Syntax -

```
class MyInterface(zope.interface.
                  Interface)
```

example -

```
import abc
class myinterface(abc.ABC):
    @abc.abstractclassmethod
    def display():
        pass
print("this is my class")
class Myclass(myinterface):
    def display():
        pass
obj=Myclass()
```

→ Interface acts as a blueprint for designing classes, so interfaces are implemented using "implementer" decorator on class.

→ If a class implements an interface, then the instances of the classes ~~implements~~ provide the interface.

→ Objects can provide interfaces directly, in addition to what their classes implement.

→ Syntax:

```
@zope.interface.implementer(*interfaces)
class Class_name:
    #methods.
```

Conclusions: 9

30/03/22

Lab Assignment 7

18/03/22
Ishika K.
S13-39

Aim: Python programs on modules and packages. (L04)

Theory:

① **Module** -

→ It is a file containing Python definitions and statements.
→ A module can define functions, classes and variables.
→ A module can also include runnable code.
→ Grouping related code into a module makes the code easier to understand and use.
→ It also makes the code logically organised.

② **Package** -

→ A directory with Python files and a file with the name __init__.py.
→ This means that every directory inside of the Python path, which contains a file named __init__.py will be treated as a package by Python.

③ __init__ -

→ The role of __init__.py file is similar to the __init__ function in a Python class.

→ The function file essentially the constructor of the package without it being called such.

Conclusion :

30/03/22

Lab Assignment 8

18/3/22
Ishika K.
S13-39
Page-1

Aim: Python programs on exception handling. (LO4)

Theory:

- **Exception-**

  → It is an event which occurs during the execution of a program that disrupts the normal flow of the program instruction.

  → In Python, exceptions can be handled using a 'try' statements.
  → The critical operation which can raise an exception is placed inside the 'try' clause.
  → The code that handles this is written in the except clause.

- **try-** It tests the excepted error to occur.

- **except-** Here we can handle the error.

- **finally-** Always gets executed either exception is generated or not.

Conclusion ?

30/03/22

```
try:
    #code
except:
    #optional block
    #exception handaling
finally:
    #code
```

Python Assignment 9

26/3/22

Ishika K.
SI3-39

Aim:

To study and implement different file handling operations. (LO5)

Theory:

→ Python, too supports file handling and allows user to handle file i.e to read and write files along with many others file handling options, to operate on files.
→ The concept of file handling has stretched over various other languages, but the implementation in Python is easy and short.
→ Python treats file differently as text or binary and that is important.
→ Each line of code includes a sequence of characters and then form text file.
→ Each line of a file is terminated with a special characters like comma or newline characters.

- **open() function:**

↦ First we have to open the file inorder to perform any read or write operation.
↦ For this we use Python's inbuilt function 'open()'.
↦ When we open the file, we have to specify

the mode in which we open the file.

f= open(filename, mode)

modes -

1. r: read operation
2. w: write operation
3. a: append operation
4. r+: read and write data
5. w+: read and write(overriden)
6. a+: append and read data.

- read() function:

↦ If you need to extract a string that contains all characters in the file then we use read() function.

↦ We can also specify the number of lines to be read in the parenthesis.

↦ syntax:

```
file.read(5)
```

- write() function:

↦ We use this command to write or manipulate the entities in a particular file.

↦ The close() command terminates all the resources in use and frees the system of the particular program.

↦ syntax -

file.write("command")

- append() function :

↦ We use ~~this~~ the lstrip() and rstrip() methods in this mode of file handling.
↦ They strip off the spaces from the left and right side of the line respec-tively.
↦ It is designed to provide much cleaner syntax and exception handling while working with code.

Conclusion -

17/04/24

→ Files are named location on disk to store related information.
→ They are permanently stored in a non-volatile (secondary) memory.
→ To make changes to a file or to write a certain file using certain scripts, we use python.
→ Python provides us file handling capabilities and lets us work and change the content of a certain file.
→ We use the inbuilt open() function to open() function to open the file and other provided methods to make changes to a file.

09/4/22

Python Assignment - 10

Ishika Khokhani
S13-39

Aim: To study GUI application using TKinter. (LO5)

Theory:

→ Python provides the standard library TKinter for creating the graphical user interface for desktop-based applications.

→ Developing desktop-based applications with Python Tkinter is not a complex task.

→ An empty Tkinter top-level window can be created by using the following steps:

- import the Tkinter module
- Create the main application window
- Add the widgets like label, buttons, frames etc. to the window.
- Call the main event-loop so that the action can take place on the user's computer screen.

- syntax:

```
from tkinter import *
#create the application window
top = tk()
#Enter the event main loop
top.mainloop()
```

There are various widgets like button, canvas, entry etc.

1) Button - it is used to add various kinds of buttons to the python app.

2) Canvas - it is used to draw the canvas on the window.

3) Checkbox - checksum is used to display the checkbox on the window

4) Entry - it is used to display the single line text field to the user.

5) Frame - it can be defined as a container to which, another widget can be added and organised.

6) Label - text used to display some message or information about the other widgets.

7) Listbox - it is used to display some message a list of options to the user.

8) Menubutton - displays the menu items to the user.

9) Menu - used to add menu items to the user.

10) Message - display message box to the user.

11) Radiobutton - You can select only one option.

12) Scale - provides slider to the user.

13) Scrollbar - provides user with a scrollbar to go up and down.

14) Top-level - creates a separate window container.

15) Spin-box - entry widget to select from option of values.

16) Paned window - contains horizontal and vertical panes.

17) Label frames - widget acting like a container.

18) Message box - used to display the message box in the desktop-based applications.

Tkinter geometry :

It specifies the method using which, the widgets are represented on display.

1. Pack method
2. Grid method
3. Place method

1. Tkinter pack

Conclusion:

We can use the same components as we have in Java to create a GUI.

Python Assignment - 11

26/3/22

Ishika Khokhani
S13-39

Aim:

To study and implement CRUD operations using pymsql Python package and MySQL Database connectivity. (L05)

Theory:

Steps to install | connect MySQL Database with Python -

1. Install MySQL connector module
   ~use the pip command to install MySQL connector python

2. Import MySQL connector module
   Import using a import mysql.connector statement so you can use this module's methods to communicate with the MySQL database.

3. Use the connector() method
   use the connect() method of the MySQL connector class with the required arguments to connect~~or~~ mysql.

4. Use the cursor() method
   use the cursor() method of a MySQLConnection object to create a cursor object to perform various SQL operations.

5. Use the execute() method
The execute() method runs the SQL Query and returns the result.

6. Execute/Extract result using fetchall().
Use cursor.fetchall() or fetchone() or fetchmany() to read query result.

7. Close cursor and connection objects
use cursor.close() and connection.close() method to close open connections to end the connection.

- ~~get~~
- fetchall() -

It fetches all the rows of a query rebuilt. It returns all the rows as a list of tuples. An empty list is returned if there is no record to fetch.

- execute() -

It helps us to execute the query and return records according to the query.

syntax-

cursor.execute(query.args=None)

- **executemany()** -

It helps us to INSERT or REPLACE multiple records at once.

Conclusion-

Database connectivity with an application is one of the essential steps in order to maintain the relevant data about a particular entity of the application.
Python supports MySQL connectivity and offers a range of functions to execute in order to run a query and store data in the database.
By following simple steps mentioned above, we can connect to a particular DB and execute a query according to our needs.

28/3/22

Python Assignment-12

Ishika Khokhani
S13-39

Aim: To study different types of plots using Numpy and matplotlib. (LO6)

Theory:

→ Matplotlib is an amazing visualisation library in Python for 2D plots of arrays.
→ Its a multi-platform data visualisation library built on Numpy arrays and designed to work with the broader scipy stack.
→ One of the greatest benefits of visualisation is that it allows us visual access to huge amounts of data in easily digestible visuals.
→ Matplotlib consists of several plots like line, bar, scatter, histogram.

- **subplots()** -
  used to display multiple plots in one figure. It takes various arguments such as a number of rows, columns, or sharex, sharey axis.
- **Lineplots()** -
  used to see the relationship between the x and y axis.

→ plot() function in the matplotlibelibrary's Pyplot module is used to create a 2D hexagonal plot of the co-ordinate x and y. plot() will take various arguments like plot(x, y, scalex, scaley, data, **kwargs).
→ x, y are co-ordinates of the horizontal and vertical axis, they are optional args.

Histogram-

→ The most common graph for displaying frequency distribution is a histogram.
→ We create a bin of ranges, then distribute the whole range of value into series of intervals, and count the value which will fall in the given interval.
→ plt.hist() function with arguments like data, bin, color etc.

Bar plot-

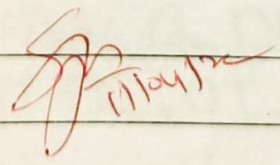
→ Mainly barplot is used to show the relationship between the numeric and categoric values.
→ In a bar chart, we have one axis representing the values or count of the specific category.
→ plt.bar(x, height, width, bottom, align).

Piechart -

→ A pie-chart is used to show the percentage of the whole.
→ Hence it is used when we want to compare the individual categories with the whole.
→ Pie() will take the different parameters like x, labels, autopct.

Conclusion:

→ matplotlib is a comprehensive library for creating static, animated and interactive visualizations in Python.
→ Its numerical mathematics extension Numpy.
→ It provides an object-oriented API for embedding plots into applications.
→ It helps us to represent data in a graphical format for better visualization.
→ It consists of various methods for different types of graphical representations.



26/3/22

Python Assignment - 13

Ishika Khokhani
S13-39

Aim: Basic operations using pandas like series, dataframes, indexing, filtering, combining and merging data frames. (106)

Theory:

→ Pandas is an open-source library that is made mainly for working with relational or labelled data both easily and intuitively.
→ It provides various data structures and operations for manipulating numerical data and time series.
→ This library is built on top of the Numpy library.
→ After installation of Pandas, we import the library into our programs.
→ It generally provides two data structures for manipulating data.

1. **series -**

- Pandas series is one-dimensional labelled array capable of holding data of any type.
- The axis labels are collectively called indexes.
- Pandas series is nothing but a column in an excel sheet.
- Labels need not be unique but must be a hashable type.

- The object supports both integer and label-based indexing and provides a host of methods for performing operations involving the index.
- In the real world, a panda series will be created by loading the datasets from existing storage, can be SQL Database, csv file, an excel file.
- Panda series can be created from the lists, dictionary and from a scalar value.

Dataframe :-

→ Pandas dataframe is a two-dimensional size-mutuable, potentially heterogenous tabular data structure with labelled axes (rows and columns).

→ A dataframe is a two-dimensional data structure i.e data is aligned in a tabular fashion in rows and columns.

→ It consists of three principal components, the data, rows and columns.

Conclusion :

Pandas is an open source Python package that is most widely used for data science/data analysis and machine learning tasks. It is built on top of another package called Numpy, which provides support for multi-dimensional array.

28/3/22

Python Assignment -14

Ishika Khokhani
S13-39

Aim: To study Scipy library and linear algebra using Scipy. (LO6)

Theory:

→ Scipy is a scientific computation library that uses Numpy underneath.
→ It stands for optimisation, stats and signal processing.
→ It is an open-source so it is free to use.
→ It provides many built-in scientific constants.
→ The library supports integration, gradient, optimisation, special functions, ordinary differential equation solvers, parallel programming tools and many more.
→ It has the number of sub-packages for the various scientific computing domains.
→ It provides very fast linear algebra capabilities
→ Linear algebra routine accepts two-dimensional array object and output is also given as a 2-dimensional array.
→ A linear algebra problem can be solved by.
  • linalg.solve()

The linalg.solve is used to solve the linear equation -

$ax + by = z$, for the unknown values x, y.

$x + 3y + 10z = 10$
$2x + 12y + 7z = 18$
$5x + 8y + 8z = 30$

The above equation can be solved as -

```
a = np.array([[1,3,10],[2,12,7],[5,8,8]])
b = np.array([[10],[18],[30]])
x = linalg.solve(a,b)
```

- **linalg.det()** -
  - → The determinant of the square matrix is found by using the linalg.det() fxn.
  - → The determinant A is denoted as |A|.
  - → It accepts a matrix and returns a scalar value.

- **linalg.eig()** -
  - → We use the linalg.eig() fxn to find the eigen value and the corresponding eigenvector of a square matrix (A).

Syntax:

```
l, v = ligalg.eig(a)
```

l → eigen values
v → eigen vectors

- **ligalg.inv()** -
They calculate the inverse and the pseudoinverse of a matrix, they take arguments of the input matrix, condition and rank and check finite are optional arguments.

- **linalg.svd(-)** -
The singular-value decomposition (svd) is a matrix decomposition method for reducing a matrix to its constituent parts to make subsequent matrix calculation simpler.

Conclusion:

11/04/22

→ SciPy is used for computing scientific and technical computations.
→ The linear algebra package from the library is used to perform certain algebraic operations with ease.
→ The unique values of simultaneous equations and matrix operations can be performed using this package.
→ Hence, scientific computations become easier by using the SciPy library.

Python Assignment - 15

Ishika Khokhani
S13-39

(106)

Aim: To study about REST API using Flask in Python.

Theory:

→ Flask is a web framework, it is a python module that lets you develop web applications easily.
→ It has a small and easy-to-extend core: it is a microframework that doesn't include an ORM (object relational manager) etc.
→ Flask is a web app framework written in python.
→ It was developed by Armin Ronacher who leads an international group of python enthusiasts (Pocco).
→ It is based on WSGI toolkits and jinja2 template engine.

- **WSGI:**
  - ↦ stands for web server gateway interface which is a standard for python web application development.
  - ↦ It is considered as the specification for universal interface between the web server and web application.

• Jinja2 -

↦ It is a web template engine that combines a template with a certain data source to render the dynamic web pages.

• Flask Environment Setup -

↦ To install flask, you need to have Python 2.7 or higher installed on our system.

• Installation of flask :

↦ $pip install virtualenv

↦ $ mkdir new
$ cd new
$ virtualenv venv
] creating a new virtual environment

↦ venv/bin/activate
venv\scripts\activate

→ $ pip install flask

→ It then starts a web server which is available only on your computer.
→ In a web brewser open localhost on port 5000 and you will see your output.

→ It is microframework but that does not mean your entire app should be inside one single python file. You can use and should use many files for larger programs, to handle the complexity.

→ 'micro' means the framework is simple but extensible.

→ ~~FLASK~~

Conclusion:

↦ Flask is one of the most popular web frameworks, meaning its up-to-date and modern.
↦ You can easily extend its functionality.
↦ You can scale it up for complex applications.

11/04/22