Room no: C305 **1St Class**

**Introduction To Python:**

\*Python is called as open source language.

\*Python is general purpose object oriented and high level programming language.

\*It can be used in Web applications, Enterprices applications, 3D and CAD.

\*It can't be used in gaming.

\*Current version is 3.10.1

\*Python file formate is .py

\*Officil website https://www.python.org

\*For downloding Libraries and Repositaries https://www.pypi.org

**Features Of Python:**

\*Easy to understand and learn.

\*Python is called as an Interpretend Language.

\*Object oriented language where u will use all the moocs concepts nessesaries.

\*Open source Language.

\*Supports wide range of libraries and frame works.

\*GUI acts as a medium b/w the user and server.

**Various Areas Here Python Is Used:**

\*AI and Data Science, Data Mining, Meachine Learning, Software Development,

Mobile Applications, 3D Apllications, Speach Recognisation, Image Prossecing Apllications

and (\*\*Web Applications\*\*).

**Python Basic Syntax:**

\*Indentations is to defined block of a code and it is nothing but (\*\*White Spaces\*\*).

**Applications Of Python:**

\*You Tube, Netflix, Face\_Book, Instagram, LinkedIn, Quora and Drop\_Box.

**Popular Frame Works In Python:**

\*Web Development(Server Side:-"ADMINS are Server Side"):- Django, Flask, Pyramid Cherrypie.

\*GUI based applications:- TK, PYGTR, PYQL, PYJS.

\*Machine Learning:- TensorFlow, pytorch, Matplotlib, Mathematics(NumPy or pandas).

**Limitations In Python:**

\*Not Good for Gaming.

\*More Libraries.

\*No dedicated supports for the libraries and the libraries are in huge.

**Important Library Functions In Python:**

\*datetime, OS, CSV, \*\*sqllite3\*\*(SQL:-Structured Query Language), JSON.

**Requests For Library:**

\*Django and Flask.

**Data Types In Python:**

\*int, float, String , Complex, Boolean, list, tuple, set, dictionary, class.

**Important Basic Functions:**

\*len():- returns length of the collection.

\*type():- To know the data type of a variable.

\*dir:- to know the functionalities of individual data type.

\*print():- used to display

\*help:- to know the description and to resolve the error.

\*input():- to read input from keyboard.

**Comment Lines In Python:**

1.# :- used for single line command.

2."text" :- Multi line comment or documents frame.

**Python Data Structures:**

1.Primitive:

\*integer, float, String and Boolean.

2.Non Primitive:

\*It is categorised into built in and user defined.

1.Built In:

List, Tuple, Dictionary, Set.

2.User Defined:

Stack, Queue, Linked List and Tree.

**List In Python:**

\*Python list holds the ordered Collection of Items that indicates the Sequence.

\*List is mutable(modifications can be done).

\*The Items od the list are enclosed with in the square brackets([]) separated by coma(,).

\*Example:

a=[1,"Likith",12.5,true]

print(len(a))

print(a)

**Tuple In Python:**

\*It is used to store sequence of immutable python objects.

\*The Items od the list are enclosed with in the square brackets(()) separated by coma(,).

\*The Tuples are immutable(Modifications can't be done).

\*Example:

a=(1,"Likith",12.5,true)

print(len(a))

print(a)

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**2n Class**

**List:**

list=["abcd",786,2.23,"john",70.3]  
tinylist=[123,"tiny"]  
print(list)  
print(list[0])  
print(list[1:3])  
print(list[2:])  
print(tinylist\*2)  
print(list+tinylist)

**OUTPUT:**

C:\Users\rocky\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/rocky/PycharmProjects/pythonProject/list.py

['abcd', 786, 2.23, 'john', 70.3]

abcd

[786, 2.23]

[2.23, 'john', 70.3]

[123, 'tiny', 123, 'tiny']

['abcd', 786, 2.23, 'john', 70.3, 123, 'tiny']

Process finished with exit code 0

**Tuple:**

tuple=("abcd",786,2.23,"john",70.3)  
tinytuple=(123,"tiny")  
print(tuple)  
print(tuple[0])  
print(tuple[1:3])  
print(tuple[2:])  
print(tinytuple\*2)  
print(tuple+tinytuple)

**Output:**

C:\Users\rocky\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/rocky/PycharmProjects/pythonProject/tuple.py

('abcd', 786, 2.23, 'john', 70.3)

abcd

(786, 2.23)

(2.23, 'john', 70.3)

(123, 'tiny', 123, 'tiny')

('abcd', 786, 2.23, 'john', 70.3, 123, 'tiny')

Process finished with exit code 0

**Dictionaries:**

\*It Stores the data in the (key, value) pair format.

\*In dictionary each key is separated from its value by a colon.

\*Keys must be unique and value can be Integer, Float or Character.

\*Dictionaries are mutable.

\*The empty curly brasses {} are used to create empty dictionary.

\*Example:

dept={“CSE”:02,”ID”:2100031023,”Name”:”Likith”}

Update[“CSE”]=”Sample”

print(dept)

dict={"cse":"Likith","ID":2100031023}  
dict["name"]="Likith"  
dict["cse"]="branch"  
print(dict)  
print(dict["name"])  
print(dict.values())  
print(dict.keys())  
print(dict.items())

**OUTPUT:**

C:\Users\rocky\PycharmProjects\pythonProject\venv\Scripts\python.exe C:/Users/rocky/PycharmProjects/pythonProject/dict.py

{'cse': 'branch', 'ID': 2100031023, 'name': 'Likith'}

Likith

dict\_values(['branch', 2100031023, 'Likith'])

dict\_keys(['cse', 'ID', 'name'])

dict\_items([('cse', 'branch'), ('ID', 2100031023), ('name', 'Likith')])

Process finished with exit code 0

**Sets:**

\*A Python sets is a collection of an ordered elements.

\*The Elements in the set cannot be duplicates.

\*The Elements are immutable.

**Strings:**

\*Is a collection or sequence of characters.

\*It is a Collection of the character surrounded by single, double or triple quotes.

\*A single character written as (‘A’) is treated as a String of length 1.

\*Strings are immutable.

**Name Spaces, Built In Class Attributes and Base Overloading methods**

**Name Spaces:**

**1.**Built in name space.

2.Global

3.Local

**def \_\_init\_\_():** (It is called as a constructor and is a reserved method in python class. This method is called when a object is created from a class and it allows the class to initialise the attributes of class)

**\_\_doc\_\_:** Class Documentation String

**\_\_dict\_\_:** Dictionary containing the class name spaces

**Self Keyword:**

\*Used to represent the instance of a class

\*Self keyword can access the attributes and the methods of class.

**Example:**

Class klu:

#statements

cse=klu() (\*\*\*Object Creation\*\*\*)

**Inheritance:**

**Types Of Inheritance:**

1.Single

2.Multiple

3.Hybrid

4.Multilevel

5.Hyrarichal

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**3rd CLASS**

**Single Inheritance:**

\*The Communication will be b/w parent and child.

**Multiple Inheritance:**

\*The Communication will be b/w One Parent and more than One Children.

**Multilevel Inheritance:**

\*The Communication will be b/w more than One Parent and more than One Children.

**Hybrid Inheritance:**

\*The combination of single and multiple Inheritance.

**Hierarchal Inheritance:**

\*It is a tree like structure which consists of more than one level of parent and child.

**Super Function:**

\*super is a built in function.

\*super() implements code reusability.

\*The methods declared in the parent will be automatically inherited(Properties and methods) to

the child class.

\*We can access the methods from the superclass to the subclass and it is a part of inheritance.

**Example:**

class Parent:  
 def \_\_init\_\_(self, text):  
 self.msg = text;  
  
 def printmsg(self):  
 print(self.msg)  
  
  
class Child(Parent):  
 def \_\_init\_\_(self, txt):  
 super().\_\_init\_\_(txt)  
  
  
x = Child("Hello, and Welcome!")  
x.printmsg()

**OUTPUT:**

Hello, and Welcome!

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**4th Class**

**Random Function In Python:**

\*Random is built in function.

\*It is used to perform actions randomly on the list and tuples.

**Methods In Random Function:**

\*random.choice(list) :- random sequence will be displayed and that can be of integer

Or characters

\*random.choices(list,k=3) :- \*here k is range and also called as randrange.

\*Choose multiple random list.

\*random.choice(range(10,101)) :- \*displays a single random number from 10 to 101.

\*random.getrandbits(1) :- \*It displays the Boolean value.

\*random.choice(list,dict) :-It displays a random key from the dictionary.\

**Regular Expressions In Python(short form re):**

\*re is a built in function.

\*re is a module in python.

\*it is used to process the input text and can be imported by using the syntax

Import re

**Methods In re:**

\*re.match() :- Checks for a match only at beginning of the string.

\*re.search() :- Search for the regular expression pattern and displays the 1st occurrence.

\*re.span() :- Print all the beginning and last character in a tuple.

\*re.string() :- Prints all the occurrence’s of the strings.

\*re.group() :- The part of the string is returned when a match is found.

\*re.findall() :- Search for all occurrence’s that matches a given pattern.

\*re.split() :- Used for creating and displaying the space in the string.

\*re.sub() :- Replaces 1 or more Strings that matches the given pattern.

\*re.compile() :- It looks for occurrence’s of the same pattern inside the strings without we

We writing it.

**Matching Expressions :-**

1.”^”

\*This expression matches the start of a string.

2.”w+”

\*This expression is for matching the alphanumeric characters in the string.

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**5th Class**

**File Handling:**

**Modes:**

\*r:- which indicates viving of the file

\*w:- which is making

\*x:- executable files

\*a:- append(adding content to the existing data)

\*r+,w+,a+ :- here we can perform both read and write operations.

**Git Hub:**

\*It is an open source tool where we can store the data and can be accessible by every person.

**Commands In The Git**:

\*commit (it means changes that are made to already existing applications)

\*push (it is where we will send the data to local machine(our desktop) to remote

machine(git))

\*push (it is where we will send the data from remote machine(git) to local

Machine(our desktop))

\*fork (it is nothing but it creates a duplicate copy and save in the github account)

\*clone (creating a duplicate copy of our data)

**Steps to follow for integrating python to git:**

\*Installed a git

\*click on the vcs and select the enable version control

\*right click on program.pi select git select add (at this point the program color is changed

from red to green Stating that program is version enabled

\*go to git select commit

\*we should select unversioned file type a msg click commit enter mail id and click send

commit

\*click on git github and select share project via github and add the description and add the

account

\*login via github and my data and projects will be displayed in the repository

**6th Class**

**PYTEST:**

\*It is a testing frame work that allows users to write test cases using python programming language

\*We can write scalable and simple test cases for the data bases GUI(graphical user interface) and

UI(user interface).

**Advantages Of PyTest:**

\*Easy to implement because of it’s simple syntax.

\*We can run test in parallel.

\*Automatically detect test.

\*It is an Open Source.

\*The programe in pytest must be save as test\_py

+++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**7th Class**

**Integrating Pytest With Python IDE:**

\*Open the program which is saved in the folder and click on open with select

Pycharm ide

\*Click on file go to settings select python project name select python interpreter

\*Click on the + symbol search for pytest and install (This imports the pytest package

Into our python program

\*Go to file and select setting and select tools and select python integrated tool

Select pytest as the defaoult runer and give the path in syntax working sirectory

\*Click on run select edit configurations click on the + symbol select pythontest

Select pytest

\*Give the path in the target and in the working directory

\*Right click in the program module and run

**6th Class**

**Pytest Methods:**

\*assertion:

\*pytest assertions or the checks that returns either true or false.

\*If assertion is added with the code and the method execution gets failed it will

Continue to test next method.

\*Example:

If(x==y)=T/F

assert %x=y

\*parameterised:

\*Running is single or test suit or test case against multiple test cases

\*Fixture or Marker:

\*These are used when we want to run the code before every test method

\*File Handiling CSV(Coma Saperated Value):

\*Methods:

\*open method.

\*Syntax: file=open(“fn.CSV”)

\*for extracting the field names use the next method

\*syntax: file=next(“filename.csv”)

\*for closing the file

\*syntax: file.close

**7th Class**

**MongoDB:**

\*It is a database for storing your frontend data in the mangodb database

**MangoDB Crud Operations:**

\*ccreate

\*read

\*update (update to one and update to many)

\*delete (find one and find all)

Integrating MongoDB With Pycharm:

\*Go to official website [www.mongodb.com](http://www.mongodb.com) in that select products select compass download

\*compass acts as mongodb clint

\*On the compass page click on onprimises---enterprise server---download

\*Open mongodb program in pycharm install pymongo package and run the program

\*Open the compass create a new data base(data base name and collection name should be

same)

\*back to the pycharm replace database name in the 1st 2 lines of program with the name

u created in the compass

\*Back to compass refresh the page and data will be reflected in the database we created

Integrating MySQL with MongoDB:

\*Open Compass create the database(in the collections select caped collection and give the

Size of the file like 8000 kb) then create

\*Open the data base u created click on add data import file chose the csv extension

Select the file and import

Note:

\*While creating the new.csv file open an excel sheet insert the data click on save as chose

.CSV with MS-DOS extension

++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++++

**Redirect KeyWord:**

\*redirecting the url to a new location

\***Syntax:**

@app.route(‘/user/<name>’)

def hello\_user(name):

if(name==’admin’):

return redirect(url\_for(‘hello\_admin))

else:

return redirect(url\_for(‘hello\_guest’,guest=name))

5/9/22

\*sqlite3 is an inbuild database available in python.

\*You can use by using import sqlite3

\*sqlite3 is used only for the small-scale applications.

Syntax for creating a database:

\*conn = sqlite3.connect(“pfsd.db”)

Syntax for creating a Table:

\*CREATE TABLE table\_name(Attr1 Type1,Attr2 Type2, …… Attrn Typen);

Syntax for inserting data into the Table:

\*INSERT INTO table\_name VALUES(value1, value2, ….., valuen)

Syntax for updating data in the Table:

\*UPDATE table\_name SET column1=value1, column2=value2,…… WHERE condition;

Syntax for creating the Cursor:

\*The sqlite3.Cursor class is an instance used for fetching the statements and executing them.

\*Cursor = conn.cursor()

13/9/2022

Flask web Forms:

\*The data will be sent from clint to server (this process is called as request)

Attributes for the request:

\*forms: it is a dictionary object consists of key value pairs in the form of parameters and

Values

\*args: Contents of a query string in the URL after question mark

\*cookies: dictionary objects holding the Cooke names and values

\*files: files are mainly for Storing our data

\*methods: methods are mainly for request and response parameters

\*

21/9/22

Bootstrap:

\*It is a free frame work for web development which includes HTML, CSS, Design Templets for

Forms(it is a dictionary attribute key and value pair), buttons, tables, navigations, models, images

and Optional java script plug in

Advantages Of Bootstrap:

\*Easy to use

\*Responsivnes

\*Browser Compartability

\*Bootstarp current version is 5

\*

28/9/22

Sql alchemi:

\*It is an python tool kit and object relational mapper used for storing relational database structures and algorithms

\*The database will be considered as an argument by create\_engine() function

Key words:

1, columns :- it is used to storing the data in the columns

2,integer:- for storing the integer data in the field

3,primary key:- unique values to be stored

4,string:- storing the character data

5,unique:- it returns true when the data in the field is unique

6,nullable:- it returns false such that the data in the field is not null

7,\_\_repr\_\_:- it is mainly for declaring the statements and used to represent the objects in the table

Django:

Workflow of the project:

Manage.py

Settings.py

Urls.py(write project)

Urls.py(application)

Views.py

11/10/22

DJANGO

\*Django is a high end web frame work used for developing Pragmatic content

\*Django follows MVT architecture(Model, View, Templets)

\*Model:

\*Consists of fields and methods connected to a database

\*Useful for handling databases

\*View:

\*Executes business logic and interacts with model to carry out the operations on the data

\*Templet:

\*It is a html file mixed with DTL(Django templet language and djngo design controller)

Work Flow Of Django Project:

\*The execution will start from

\*manage.py(of project)

\*settings.py(of project)

\*urls.py(of application)

\*urls.py(of project)

\*views.py(of application)

Features Of Django:

1.Scalability:

\*Adding more web pages to the project

\*Scaling will be done horizontally

2.Portability:

\*The Django projects must run on Widows, Linux and Mac OS

3.Versatality:

\*The Django code must be integrated to run using HTML, CSS, XML and CLINT SERVER ARCHITECHTURES

4.Packages:

\*Django must support 100s of many packages

5.Easy To Use:

\*Django supports admin and user modifications and there will be an inbuilt admin role supported

Django Structure:

1.Project Structure:

\*init.py(it is the first file): Django treats init.py as package

\*settings.py(contains all the project settings)

\*urls.py(consists of all the links of the project and a function to call them into the project)

\*wsgi.py(web server gatway interface): used to deploy your projects over wsgi that means into a network or into a server

\*manage.py: it is a project localdjango machine for interacting with your project through terminal or command line

2.Application Structure:

\* init.py(it is the first file): Django treats init.py as package

\*admin.py: Helps us to make the application admin modifiable

\*models.py: stores all the application models

\*tests.py: used for unit testing

\*views.py: contains all the application views