DATE 3/Oct.
PAGE NO.:

Python SESSION - 12

#jupyter notebook

jup nb

```
db = ['vimal@lw.com', 'popa@gmail.com', "tom@", "krish@"]

db.append("jack@lw.com")

x = [1,2,3,4,5]
for i in x:
    print(i)
```

~~for i in x:~~
~~j = i~~
~~print(j)~~

```
y = []
for i in x:
    j = i * i
    y.append(j)
```

List comprehension

```
m = [i * i for i in x]
```

- # : comments the line in py.
- If we do not get the desired output then it is called a bug.

DATE : / /
PAGE NO. :

```
m = [i*i for i in a if i>0]
```

~~m = [i*i for i in a if i>0 else 0]~~

```
for i in x:
    if i>0:
        j = i*i
        y.append(j)
    else:
        j=0
        y.append(j)
```

```
m = [i*i if i>0 else 0 for i in x]
```

FUNCTION

```
def iiec():
    print("Hi")
    print("Hello")
    print("Bye")
```

```
def iiec(m,n):
    a = m+n
    print(a)

iiec(6,7)
```

DATE : / /
PAGE NO. :

creates dynamic parameter

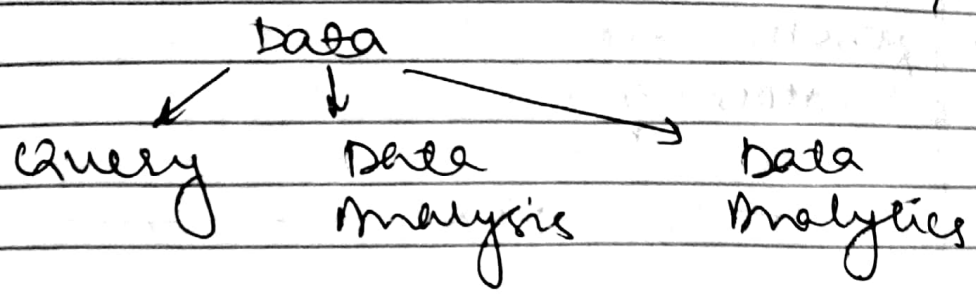
```
def iiec(*i):
    m = 0
    for j in i:
        m = j+m
    print(m)
```

If we function want to use m. Instead of print we use return keyword.

DATE: 11/Oct
PAGE NO.:

SESSION - 13 Py

1:28:44



- Dimensions: How many data you have per entry.

pip list
↳ lists all the library installed

dir(numpy)
↳ all the f^n in module numpy.

- Query - row wise
  Analysis - column wise

numpy.array(db)

DATE : 17 Oct
PAGE NO. :

SESSION - 14 Py

pip install <library_name>

NUMPY

a = numpy.array([1,2,3])
a.shape
o/p (3,) ⟶ 1D Array / Vector
↳ 1 column, 3 rows.

a = numpy.array([["abi", 111], ["jack", 222], [ ],])
a.shape
o/p (3,2) ⟶ column
↳ row
dtype = <U4

- 1D Array → Vector
2D, 3D, ... nD Array → Matrix

s = score.astype(int)
↳ changes into int from '<U4'
dtype = int32

s.mean()
↳ Now it will work!!!

s.dtype = {'names': ['sname', 'phone', 'marks']}

changes/ tags title columns name

DATE : / /
PAGE NO. :

key ↙ value ↙

{ 'formats' : [np.str , np.int32 , np:int32 } ,

↗
Dictionary

creates the format for each column.

PANDAS

```
# import pandas
dir(pandas)

d=
pandas.read_csv('Book2.csv')
d['score']                    # print score
d[['score', 'phone']]         # " score & phone
d.columns                     must be in a list

d.iloc[1]
```

DATE: 5/Dec
PAGE NO.:

SESSION: 15 py

TASK:

gedit student.py

```
def st1():
    print("a")
```

↳ Module

gedit case.py

```
from student import st1
student.st1()
```

or

```
import student
student.st1()
```

- Python always create an extra namespace (NS) whenever you save a file or launch an interpreter.

dir(student)

student.__name__

↳ namespace
or workspace

- __main__ is the namespace created by python everytime we create or launch a interpreter.

DATE: 6/Dec.
PAGE NO.:

SESSION: 16 py

ps -aux
↳ shows all the processes running in an OS.

- Behind the scene all the processes are stored in a directory. The PID is the name of the dir

- when we terminate/kill a process they remove the directory.

- The dir are stored in /proc ⇒ folder.

- when a program is executed, instructions are sent one by one to the CPU.

- The instructions are known as Thread. Till now all the processes are single threaded.

- cat /proc/3982/status
  ↳ Threads : 1
  ∴ single Threaded process

import threading
~~threading~~
def a():
  while True:
    print ("aaa")

```
def b():
    while True:
        print(" bbb ")

x1 = threading.Thread(target = a)
x2 = threading.Thread(target = b)

x1.start()
x2.start()
```

DATE : 12 Dec
PAGE NO. :

SESSION : 16 Py

Exception Handeling

```
# try :
    print ("ffff")
    print ("hi")
  except :
    print (" send mail to dev....")


# print ("welcome")
try:
    ch = input ("Enter")
    if int (ch) == 1:
        print ("One")
    elseif int(ch) == 2:
        print ("two")
    else:
        print ("no")
except :
    print ("mail to admin")
print ("Bye")
```

- *. For any kind of error

except SyntaxError :
  ↳ It will print only when there is syntax Error

DATE: / /
PAGE NO.:

try:
_____
_____
_____

except SyntaxError:
    print ("Syntax Error")
except ValueError:
    print ("Enter number")
except:
    print ("Mail Admin")

- If you want a block of code to always run then write it after the except keyword.

Or

we can use the keyword "finally".

finally:
    print ("End of code, Bye")

DATE: 13 Dec.
PAGE NO.:

SESSION: 17 py

- File only contains the inside / first initial characters linked with the address.

- storage == file
  file Name == variable

open("iiec.txt", mode='r')
  ↳ It won't show the data.
  file only contains the name & address of the content/file.

# ~~f=open~~
f = open("iiec.txt", mode='r')
f.tell()
  ↳ They print 0 if they know the file; they are at 0th position
~~#~~ f.read()
  ↳ prints the entire data.
f.read()
  ↳ Now it won't print the data.

- read() is like a for loop. After each character the pointer moves and ~~at~~ at the end there is a character, End of file which states the end and after that there is no pointer for read() to move.

DATE : / /
PAGE NO. :

# f.close()
f.read()
↳ we will get a error because we can read a closed file.

# f = open("iiec.txt", mode = 'w')
f.write("jjjj")
↳ It overwrites the file previous data is erased.

↳ file pointer / file handler

PYTHON

# x = recvfrom("Enter :") } Error
print(x)
↳ we can enter through n/w.

- Packet = Header + Payload
  ↳ {IP : Port} or Socket

SOCKET PROGRAMMING.

# ip = 192.168.0.129 → import socket
port = 1234
bind (ip, port)

- TCP : ~~Transmission~~ Control Protocol → Ack
  UDP : User Datagram Protocol → No Ack
  ↳ slow | ↳ fast
  ↳ Eg: Live streaming

PRACTICAL 07

DATE: / /
PAGE NO.:

TCP: smtp, http, ftp
UDP: DNS, DHCP

import socket

ip = 192.168.0.129
port = 1234
bind (ip, port)

SENDER / LINUX

# import socket as s
# udp protocol
myp = s.SOCK_DGRAM
# network address: IPv4.
afn = s.AF_INET
s1 = s.socket(afn, myp)

ip = "192.168.0.129"
port = 1234
s.bind((ip, port)) # Tuple

~~x = s.recvform (1024)~~
~~print (x)~~

x = s.recvform (1024)
print (x) ↳ buffer size

ps -aux | grep s.py
netstat -tnlp
↳ show tcp program

DATE : / /
PAGE NO. :

* netstat -unlp
  ↳ for UDP

* import socket ass
  s = s.socket (s.AF_INET, s.SOCK_DGRAM)
  s.sendto("hello".encode(), ("192.168.0.129", 1234))
  ↳ TO convert the string to bytes.

Python

gedit server1.py

server:

```
import socket
s=socket.socket (socket.AF_INET,
                 socket.SOCK_DGRAM)
s.bind (("192.168.0.129", 1234))
x=s.recvfrom(10)   # buffer size
print(x)
```

{ b"hello"
"hello".encode() }

receive

```
import socket
s=socket.socket(socket.AF_INET,
                socket.SOCK_DGRAM)
s.bind (("192.168.0.129", 1234))
# server ip   # port
data = "hello"
s.sendto(data.encode(), ("192.168.0.129",1234))
```

PRACTICAL 07

DATE : / /
PAGE NO. :

sender: (inside while)

```
clientip = x[1][0]
print(clientip)
data = x[0].decode()
print(data)
    or
print(clientip + " : " + data)
'import time
#import os
#print(os.system(data))
time.sleep(no)
```

PYTHON

- IP : Port
  - ↳ socket.
- UDP is connectionless protocol. ⌐ Not Ack.
  TCP is connection oriented protocol.
  - ↳ Ack
  - ↳ Session is created
  - ↳ 3 way handshake

TCP socket Programming

```
# receiver.py

import socket
s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
ip = "192.168.0.129"
port = 1234
s.bind((ip, port))
#now we might be able to see the
#port running bec. it has not
#created a session
s.listen()
    #now it it will show.
x = s.accept()
print(x)
```

O/P will show

PAGE NO. :
DATE : / /

x=s.recv(1024)
#s.send(b"hi from server")
s.recv()
print(x)

#sender.py
import socket
s=socket.socket(socket.AF_INET, socket.SOCK_STREAM)
s.connect(('ip', port))
↳ #("192.168.0.129", 1234)
#s.close() #TO terminate the session
s.send(b"hi i'm client")
s.recv(1024)

netstat -nat
↳ TO see all the live connections

PAGE NO.:
DATE: 12/Jan/21

SESSION: 7+ Pj

- Sequential search or linear search

x = [10, 1, 5, 93, 51, 7, 16]
0, 1, 2, 3, 4, 5, 6
find: 16

loop → while pos == 6
if 10 == 16
found
else
checknext element

item (~~const~~)	worst case(O)	Best case (Ω)	Avg (θ)
item ~~present~~ present	n	1	n/2
item not present	n	n	n

jupyter notebook

```
x = [10, 1, 5, 93, 51, 7, 16]
len(x)
position = 0
found = False

while position < len(x) and not found:
    ~~position+=1~~
    ~~found = True~~
    if x[position] == 16:
        found = True
    else:
        position += 1
```

PAGE NO. :
DATE : / /

Create a fn.

```
def sequential(arr, item):
    position = 0
    found = False
    while position < len(arr) and
                        not found:
        if x[position] == item:
            found = True
        else:
            position += 1
    return found
```

PAGE NO.:
DATE: 13 Jan 21

SESSION: 72 Py

- sometimes we write a algo with higher time complexity for the sake of simplicity.
- RAM = Buffer + Cache + Dirty(stack)

Py ~~Py~~ # ——— py

```
import dis
dis.dis(ww)   # ww is a fn.
```

↳ shows how they are working on RAM.

```
def ww(x):
    y = x*x
    return y
```
high comp

```
def ww(x):
    return x*x
```
low comp.

*. The value of x is temporary. So it is stored in stack Memory

```
# list(range(5))
↳ [0, 1, 2, 3, 4]
```

Program for factorial
↳ Iteration
↳ ~~Recursion~~ Recursive

PAGE NO.:
DATE: 16/Jan/21

SESSION: 73 Py

- TCP is more reliable than UDP.

```
#mkdir lws25
#gedit server.py
    import socket
    s=socket.socket()      # By default they use TCP & IPv4.
    port = 2222
    ip = "192.168.0.129" or "" i program will automatic bind ip
    s.bind((ip, port))
    s.listen()
c,addr=s.accept()
    # it can accept only
    # one client
    print(addr)
```

```
a = [1,2]
i,j = a
{a[0]=i
 a[1]=j}
# tuple unpacking
```

```
#netstat -tnlp | grep 2222
    ↳ 0.0.0.0 : 2222
        Any ip can connect to 2222
```

client

```
serverip = "192.168.0.129"
serverport = 2222
s.connect((serverip, serverport))
```

For more than one connection we need to accept two times.

PAGE NO.:
DATE: / /

server
client A
c.send (b" i am server")
data = c.recv (100)
print (data)

client B
[same as CA]

client recv A
print(S.recv(100))
S.send(b" i am client A")

client recv B
print(S.recv(100))
S.send (b" i am client B")

- when we again try to use it will show error that port in use
  # netstat -nct

server
s.setsockopt(SOL... SOL_SOCKET,
SO_REUSEADDR, 1)

while True:
—— accept
print(data)

PAGE NO. :
DATE : / /

```
def iiecs():
    # c, adde = S.accept()      (comment)
    print(adde)
    c.send(b"i m server")
    data = c.recv(100)
    print(data)

while True:
    iiecs()
```

under

```
import multithreading as mt.

while True: c, adde = s.accept()
    t1 = mt.Thread(target = iiecs)
    t1.start()
```

PAGE NO :
DATE : 17/Jan/21

SESSION: 74 Py

```
# def iiec():
    a=5
    y=2
    return**y

# def email(*i):
    return i+'@gmail.com'
↳ email("vimal","jack","pep")

        or

# def email(*i):
    for j in i:
        # print(j+"@g.c")# print the values
        return(j+"@gmail.com")

        or

# def ____
    k=[]
    for ____:
        k.append(j+"@g.c")
    return k
↳ h=email("__","__","__")
    print(h)
        ↳["__","__","__"]

# def db(i,j,h):
    return i+j+h
↳ db("jack","mumbai","India")
    ↳'jackmumbaiIndia'
```

PAGE NO.:
DATE: / /

- Arguments are order sensitive.

# db(city = "mumbai", name = "krish",
country = "India")

def db (name, city, country):

- (Parameter = "value")
  - Order insensitive
- Docstring is the doc of a func or comment. To call it:
  - <<func_name>>.__doc__

or

print (<<func_name>>.__doc__)

- For default arguments:

def db(name, city, country = "India"):

- Variable length keyword arg:

** kwargs : dictionary

k = db

Alias or now k is a fn.

(db)

- Functional Programming is based on readability.
- Lambda == Nameless fn == Anonymous fn.

Syntax

lambda i, j : i+j

m(3, 4)

PAGE NO. :
DATE : / /

- Higher order fn: when a func takes another fn as an argument.

```
# def fun(fun2(), j):
```

Or we can use lambda.

```
# db=[1,3,6,4,90,78,56,4,45,79]
# find even no.
list(filter(lambda x: x%2==0, db))
```

PAGE NO.:
DATE: 28/Jan/21

SESSION: 82 DSA-Py

- Pre-requisit of the Binary search is that the data should be sorted.
- Bubble sort:

  space Complexity = O(1)

	O(n)	O(n-1)	O(n-2)	
X	I	II	III	------ n times
2	2	2	2	
6	6	4	4	
9	4	6	1	
4	8	1	5	
8	1	5	6	
1	5	7	7	
5	7	8		
7	9			

- Bubble sort compares two adjacent elements and swap if they are in wrong order.

  Time Complexity = $O(n^2)$

URL: visualgo.net (visual for sorting algo.)

- Bubble is best in terms of space complexity.

PAGE NO.:
DATE 29/Jan/24

SESSION: 83 DSA-py

```
# .py
if 5==5:
    pass
else:
    print("no")
```

""" # if we want if block to be empty then use pass keyword. """

```
# def wbs(data):
    for i in range(len(data)-1):
        for s in range(len(data)-1-i):
            if data[s] > data[s+1]:
                data[s], data[s+1] = data[s+1], data[s]
    return data
```

- Bubble sort: $O(n^2)$

PAGE NO.:
DATE: 7 Feb 21

SESSION: 22 Python

jupyter notebook
↳ .py

# class iiec_visitor:
    title = "IIEC Visitor Form"

} class initialization

jack = iiec_visitor() # instantiation
jack: object ; instance

jack.title
↳ prints the title.

Similarly,
krish = iiec_visitor()
krish.title

# class iiec_visitor:
    title = " — " # variable == attribute
    name = " "
    phone = " "
    def get_name(): # function == method
        print ("name is: ", name)
    def send_sms():
        ≡ Print ("phone is: ", phone)

↳ jack.name = "jack xyz"

eric = iiec_visitor()
eric.send_sms()
↳ error bec. eric do not have phone.
↳ whenever we call any method from a class we need to pass the address as a parameter.

correct method
def send_sms(self):
print("send sms:", self.phone)
# print(x)
↳ eric.send_sms()

def get_name(self):
print("name is:", self.name)
* #compulsory in py to use self as an argument in a method.

• constructors are are functions that are by default called when we create a new object.
↳ def __init__(self):

# def __init__(self):
print("constructor test")
self.name = x
self.phone = y

PAGE NO. :
DATE : / /

```
# def __init__(self, x, y):
    self.name = x
    self.phone = y

tom = iiec_visitor("TOM", 123)
```

PAGE NO.:
DATE: 20Feb21

SESSION: 97 Py

- pass : If we wanna to create an empty block of code.

```
# class visitor:
      pass
↳ jack = visitor()
↳ dir(jack)
    ↳ __dict__
  jack.__dict__
    ↳ If we want to see the data inside the block of code.

↳ jack.phone = 12345
  jack.address = "____"

  jack.__dict__
    ↳ shows phone & adde.


# class visitor: → country = "IN" ← class var
  def __init__(self, n, l, p):
      self.name = n ← Instance var
      self.lname = l
      self.phone = p

↳ jack = visitor("jack", "xyz", 12345)

      def ret_phone(self):
          return self.phone)
↳ jack.ret_phone()
```

PAGE NO. :
DATE : / /

```
# class visitor:  <-- Parent class
    country = "IN"
    num_of_visitor = 0
    def __init__ (self):
        self.role = "visitor"
        visitor.num_of_visitor += 1
    def ret_phone(self):
        return self.phone
    def ret_full_name(self):
        return self.name + " " +
                self.lname
    def ret_role:
        return self.role
class emp(visitor):  <-- child class
    pass

vinal = emp("vinal", "daga", 1111)
vinal.ret_full_name

# class emp(visitor):
    def ret_role(self):
        self.role = "emp"
        return self.role.
```

(a) This is FUNCTION OVERRIDING

PAGE NO. :
DATE : 21/Feb/21

SESSION: 96 Py

#.

```
class visitor:                                  (region)
    def __init__(self, first, last, mob):
        self.first = first
        self.last = last
        self.mob = mob
    def get_fullname(self):
        return self.first + " " + self.last
    def set_mobile(self, newmob):
        self.mob = newmob
```

instance method

```
class emp(visitor):
    pass
# jack = emp("jack", "xyz", 1111)
jack.__dict__
```

```
class emp(visitor):
    def __init__(self, title):
        self.title = title

jack = emp(—, title = "manager")
```

FAILED!!

- If the same fn name is avail in child class as parent class then that fn is overridden.

PAGE NO. :
DATE : / /

```
class emp(visitor):
    super().__init__()
    def __init__(self, first, last, mob, title):
        super().__init__(first, last, mob)
        self.title = title
```

↳ now it will retrive all ~~user~~ its four properties

region

class method

```
region = 'IN'
def get_region(cls):
    return cls.region
```

address of class

↳ failed

```
class visitor:
    region = 'IN'        # class method
    # annotation / decoration
    @classmethod
    def get_region(cls):
        return cls.region
    @classmethod
    def set_region(cls, country):
        cls.region = country
```

PAGE NO.:
DATE: 6/Mar/21

SESSION: 108 py

flask ⟶ Full Stack Dev
↳ WebApp
↳ RestAPI
→ Framework to create a webapp on the top of restapi.

win
pip install flask
~~jupyter notebook~~ mkdir flask
notepad app.py

• flask have its own inbuild web server

app.py module class

from flask import Flask, render_template

~~app~~ ~~from flask~~ ~~Flask()~~ ← Instantiated the class in app

app = Flask("myjcc") ← app name

def myhello():
    print("hello")
def search():
    print("search ...")

@app.route("/search")

• Every func. creates its URL that's how client is able to use that func.

@app.route("/email")
def myemail():
    print("email ....")

PAGE NO. :
DATE : / /

flask run
↳ bydefault picks the file (app.py)

- flask runs on port 5000
  ↳ They'll print the URL.
    ↳ Error!

- Print always prints in the terminal not in the web page.

def mysearch():
  return ("search")
replace print with return

- re-run the app.

def mysearch():
  return (" search... ")

Paid
```
import os
data = os.system("data /t")
return (data)
```

```
import subprocess as sp
data = sp.getoutput('data /t')
return ('data')
```

PAGE NO. :
DATE : / /

~~#Create pad .html~~

* mkdir/cd templates
* notepad "e.html"

* app.py

```
@app.route("/email")
def myemail():
    return render_template("e.html")
```

PAGE NO.
DATE: 7/Mar/24

SESSION 109 py

Rhel

~~python~~
pip3 install flask
mkdir menuapp
cd menuapp
vim app.py

```
from flask import Flask, render_template
app = Flask("myapp")
@app.route("/")
def home():
    return "i am in home!"
```

flask run

export
FLASK_ENV=development
(if win ; set FLASK_ENV= ____)

flask run

```
@app.route("/menu")
def mymenu():
    name = "Jack"
    ~~return "html code <b> hello </b>"~~
    htmlcode = render_template("mymenu.html")
    return htmlcode
```

mkdir templates
cd
vim mymenu.html
~~this is my menu html file~~.
Hi, Vimal

```
htmlcode = render_template("myname.html",
                myname = name)
return htmlcode
```

menu.html

```
Hi, {{myname}}
```

vim myform.html ← By default

```
<form method='GET'>
  Name <input name='x' />
  <br />
  Company Name <input name='y' />
  <br />
  Click <input type='submit' />
</form>
```

app.py

```
@app.route("/form")
def myform():
    return render_template("myform.html")
```

```
<form method='GET' action='/menu'>
```

↳ sends the data to /menu.

PAGE NO.
DATE / /

```
from flask import request

@app.route("/menu", methods=['GET'])
def mymenu():
    name = request.args.get("n")
    return render_template("mymenu.html",
                           myname=name,
                           cname=c)

c = request.args.get("y")
```

PAGE NO.:
DATE 20/Mar21

SESSION: 118 Py(Flask)

- App ⟶ DB
  Flask ⟶ sqlite
  or mysql / Ms / Oracle / ...

- object Relational Mapper (ORM)

pip install Flask-SQLAlchemy

notepad db1.py

```
from flask import Flask
from flask_sqlalchemy import SQLAlchemy

app = Flask("liecapp")
db = SQLAlchemy(app)

app.config["SQLALCHEMY_DATABASE_URI"] =
        'sqlite:///mydb1/data.sqlite'

db = SQLAlchemy(app)
```

mkdir mydb
notepad db1.py

```
class IEC(db.Model):
    name = db.Column(db.Text)
    age = db.Column(db.Integer)
    remarks = db.Column(db.Text)
    id = db.Column(db.Integer,
            primary_key = True)
```

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

#Create

```
db.create_all()

jack = IIEC("jack", 20, "ok")
db.session.add(jack)
db.session.commit()
```

#Read

```
""
r2 = IIEC.query.get(2)
print(r2)
print(r2.name, r2.age)
""
```

#Query

```
""
rall = IIEC.query.all()
print(rall)

rage = IIEC.query.filter_by(age=10)
print(rage)
```

PAGE NO. :
DATE : 10 Apr 21

SESSION: 139 Py

REST API
↳ Application programming interface

- Microservices are based on rest api.

#hosted app.py
from flask import Flask

app = Flask("myiiec")

@app.route('/form')
def myform()

@app.route('/data')
def mydata()

- If we want to run flask code with some other python libraries
  ∴ now instead of flask run we can run the code with python app.py cmd

#Set
- Visual Studio code
  ↳ IDE

PAGE NO. :
DATE : / /

* /templates/basic.html

```
<form action =
Name < input name = 'x' />
<br />
<input type = 'submit' />
</form>
```

```
@app.route('/data', methods = ['POST'])
def mydata():
    if request.method = 'POST':
        data = request.form.get("x")
    return data.upper()
```

(from get)

POST

header

URL

GET (args.get)

- Challenge in GET method is, it is not suitable for sensitive info. And we also can't pass big files like img, vids.

So, we change the method to POST.

- Action (verbs)

C R U D

- C → Create (POST)
- R → Read (GET)
- U → Update (PUT)
- D → Delete (DELETE)

PAGE NO. :
DATE : / /

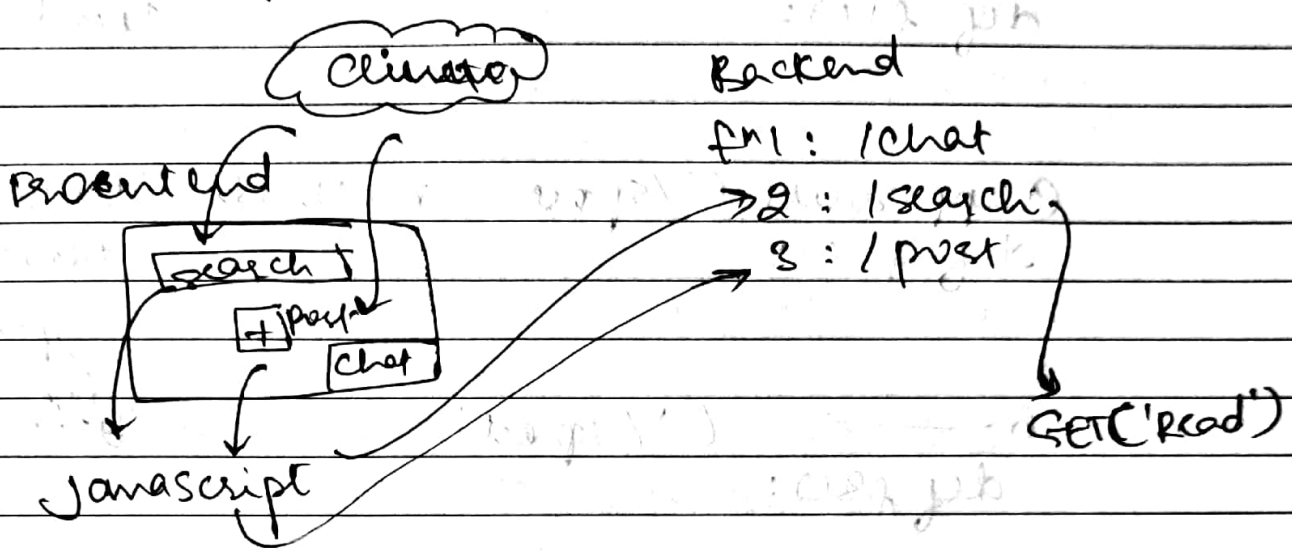
↖ Dynamic Route

```
@app.route('/name/<y>')
def myname(y):
    return y
```

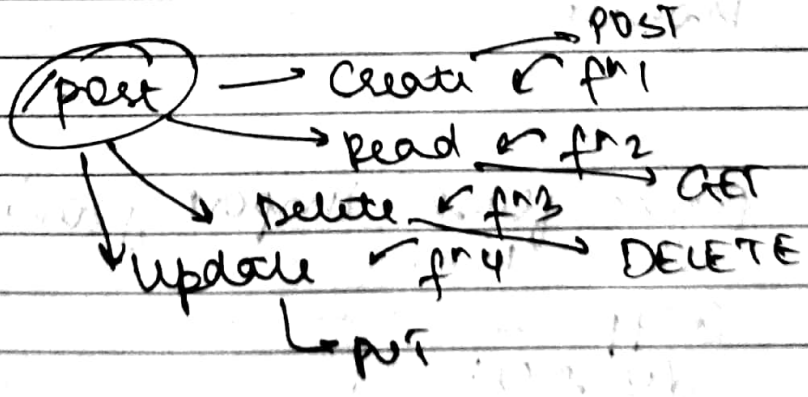
PAGE NO.:
DATE: 11 Apr 21

Session: 141 By

Rest API



- Here, the client does not directly goes to the backend. It only interacts with the frontend and the program (Javascript) goes to the backend. The advantage is that the client not have to change the URL for dif. functionalities running in dif servers.



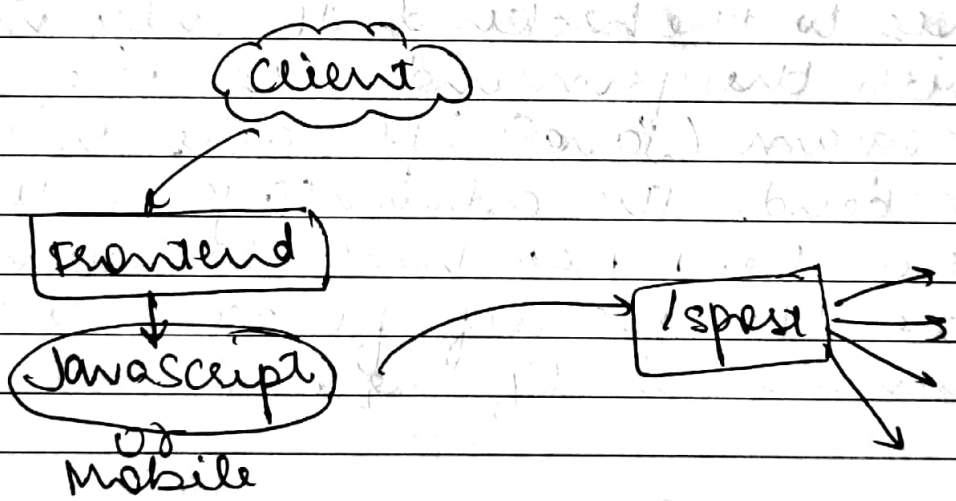
PAGE NO. :
DATE : / /

#.

```
@app.route('/spest', methods=["GET"])
def f1():
```
Read

```
@app.route('/spest', methods=["DELETE"])
def f2():
```
delete

```
@_____('/spest', _____=['PUT'])
def f3():
```
update



#. from flask import jsonify
db = list.
@app.____
def f1():
return jsonify(db)