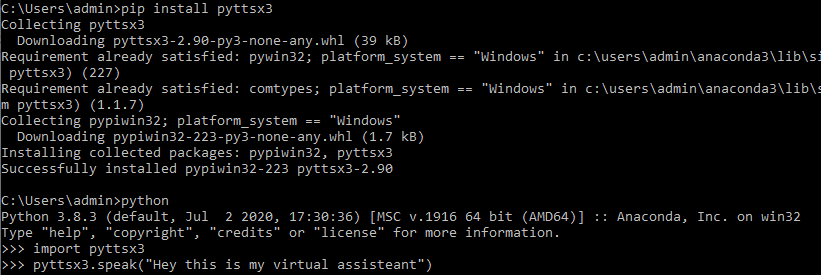
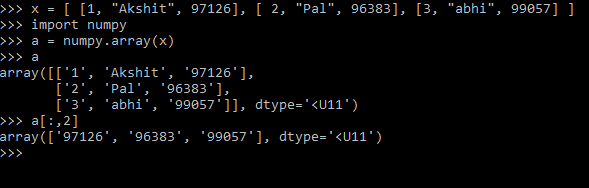
**Session 01 – basic languages**

* We need programming language, so we can communicate with our OS, because OS only knows programming language.
  + But if OS don’t know python language then we cannot do anything, for this we have to install python interpreter in our OS.
  + Here we are using Python 3.8 version of python.
  + We are using anaconda distribution for installing python interpreter.
    - * While installing click on add path.
    - It will install python interpreter.
    - Jupyter notebook
    - And lots of library.
* For storing value inside ram, we have to create one box, for this we use variable so we can get value using variable whenever we require.
* We have 2 way to communicate to OS.
* For using base windows, linux commands we have to use system function.
  + There is some function known as **builtin function**, which comes with python, we don’t have to import them.
    - Eg, print()
    - Dir(\_\_builtins\_\_)
  + System() is not a builtin function, so we have to tell our language where is function located.
    - It is inside OS module (program file – contains lots of function).
    - We have to import this module for using this.
  + SO, for this we have to use os.system() for running.
    - Os.system(“notepad”)
    - Os.system(“chrome yahoo.com”)
* If you want to tell your OS to speak instead of print we have to use speak function.
  + It will come from pyttsx3 module.
    - For this we have to install library.
    - We have a pip command to install library in python.
  + Pip install pyttsx3
    - Pyttsx3.speak(“How are you”)



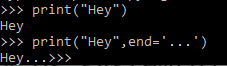
**Session 02**

* We have three way to run commands – programs.
  + Offline
    - Using notepad or vim.
  + Online
    - Your command prompt – shell – CMD.
  + IDE
* When you have any file stored in your H.D. which have some code it is known as program file.
  + When you run your program, it will known as process.
    - Here your entire code load inside your ram.
* When you create variable in other language you have to tell datatype of variable.
  + In python we don’t have to define datatype.
  + Python dynamically find datatype of variable; it is known as Type Inference.
* Anything inside quotes (“”) are known as string.
  + In python we don’t have difference between “ “ or ‘ ‘ .
    - Between “ “ or ‘ ‘ if you write escape sequence it will process this in python.
      * In different language ‘ ‘ will not process escape sequence.
* In online interpreter, python shell come with REPL.
  + Read, evaluate, print, loop.
* List datatype
  + X= [ “Akshit” , “Pal” , “Abhi”]
    - X[0]
      * Behind the seen x.\_\_getitems\_\_(0)
    - X[-1]
    - Len(x)
    - X[0:2] == x[:2]
      * [ “ Akshit ”, “ Pal ”]
      * It will exclude last number.
      * It is known as slicing operator.
  + Y= [ [ 1, ”Akshit” , 97126] , [ 2, “pal”, 96383] , [ 3, “Abhi”, 99067] ]
    - Y[2][2]
      * 99067
* List only works on row wise; it will not work column wise.
* We have array datatype for getting column wise data.
* It will come from numpy.
  + A = numpy.array(x)



**Session 03 – if-else**

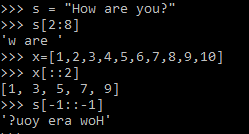
* Jupyter notebook
  + To launch notebook.
  + In jupyter file of code is known as notebook.
* Escape sequence
  + Between “ “ or ‘ ‘ if you write escape sequence it will process this in python.
    - In different language ‘ ‘ will not process escape sequence.
  + You write print(r” Welcome here”)
    - It will print as it is as raw string.
  + Help(print)
    - It will always end with new line.
    - Print(“mdfbd”, end=’\n’)



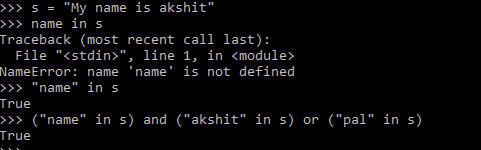
* If – else
  + If x > 2:  
     print(“HI”)  
    else:  
     print(“not”)
    - Here first x>2 evaluated.
      * If condition match it will return True else False…
      * It will always return Boolean.
    - So, if True  
       do this.

**Session 04 – while**

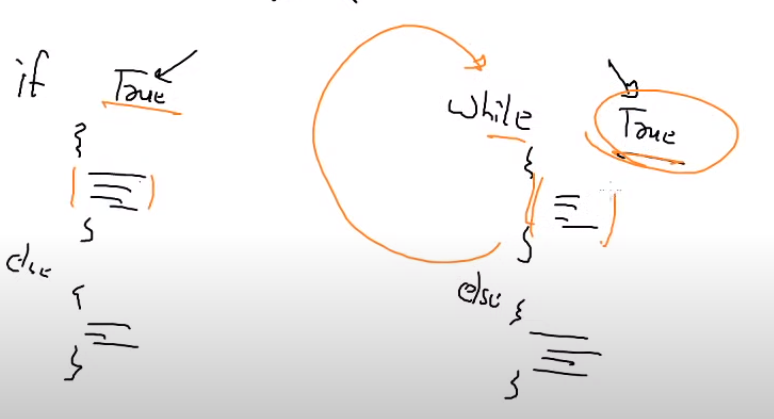
* String is also like a list.
* String = ‘’How are you?’’
  + String[2:8]  
    “w are “
  + By default it will jump with 1.
    - String[2:8] == string[2:8:1]



* + String [-1: :-1]
    - To print reverse string.
* In, and, or



* While loop
  + When you want to run same thing, some finite number of times then we use while loop.



* You can compare while loop with if – else.
* There is only one difference between them.
  + While run multiple times, but if run only one time.
* We can use break for exiting from of loop.

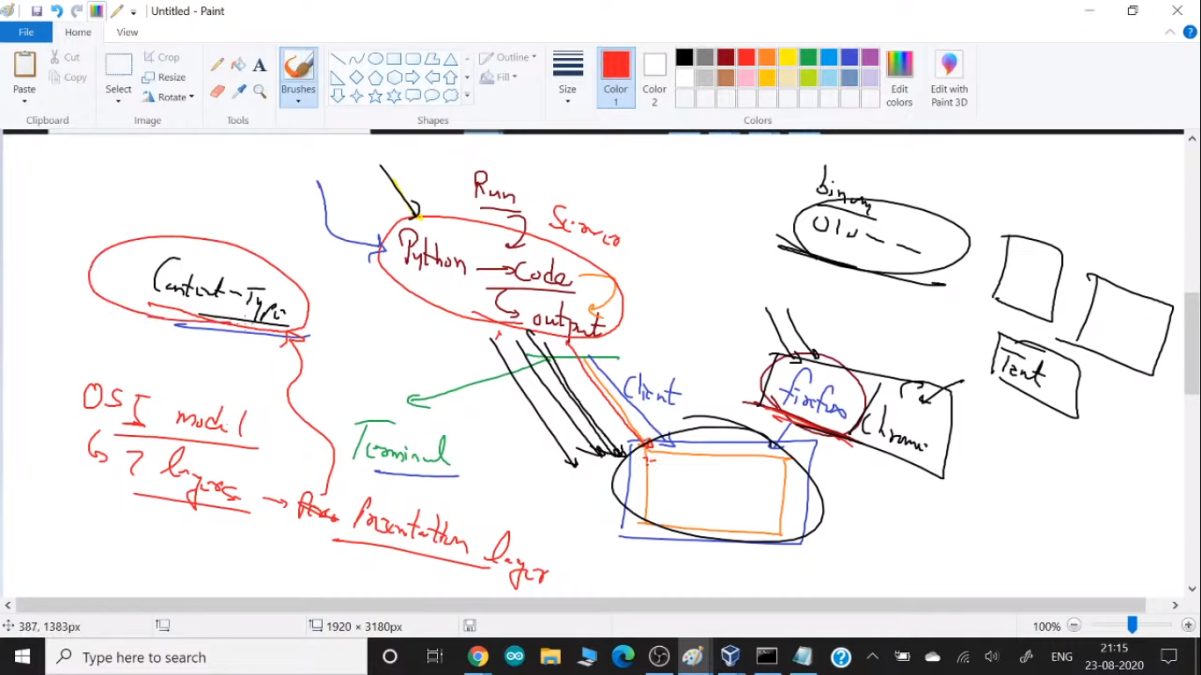
**Session 05 –CGI/API**

**CGI**

* If you want run code from different laptop without local or remote login (You don’t know login – password), run a code and see output in your local system.
* You have to send packet for running a code… so you have to send a request.
  + Here we need some program in that system, which will take your request.
  + So, on your behalf this program will go and run a program and will send output to you.
    - Without remote login
    - Without going there
    - Without knowing login—pass
    - On your behalf
  + So, you can think it as anInterface.
  + That system provides you one gate – Interface.
    - This is a common gateway for any programming language, for running any code.
  + This program is known as **Common Gateway Interface (CGI).**
* This program is giving you service so it is known as server and here you are known as client.
* Web program or web services are giving you this type of CGI services.
  + Concept name – CGI
  + For Implement concept – enable webserver
  + That system – webserver
* Installing – configuring webserver
  + CGI facility is already enabled in Apache software.
    - You have to write your code inside /var/www/cgi-bin.
  + Yum install httpd
  + Systemctl start httpd
* Connectivity
  + You have configured webserver.
  + Client need web client program.
    - chrome
  + client have to use https protocol.
* If you want to run your code as a webApp as a CGI, then you have to deploy your code in /var/www/cgi-bin.
* For accessing this you have to use <http://192.168.43.128/cgi-bin/mycode.py>
  + If instead of you someone else (webserver) want to run your program you have to make file executable.
    - Chmod +x mycode.py
  + Now you have to tell which interpreter or compiler to run.
    - Extension is only for you, for your reference.
    - For this you have to use #! this symbol.
      * This is known as shebang / hashbang.
      * This symbol will tell which interpreter to use.



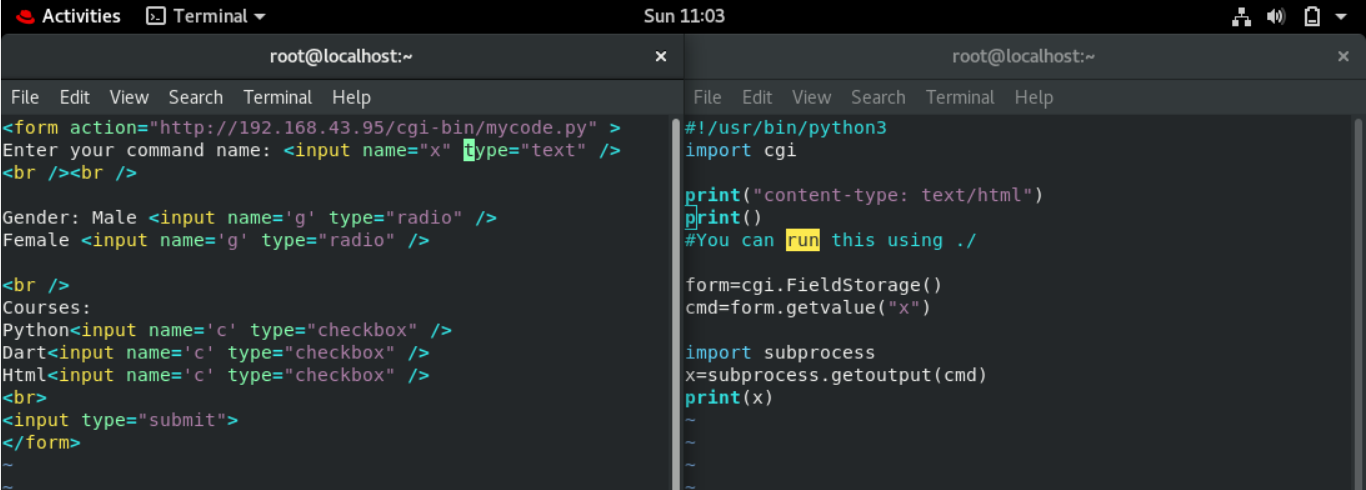
* + - * + Now you can also run this code using ./mycode.py
    - Now you have to do one more thing.
    - Here your output is going to the client browser and printed there.

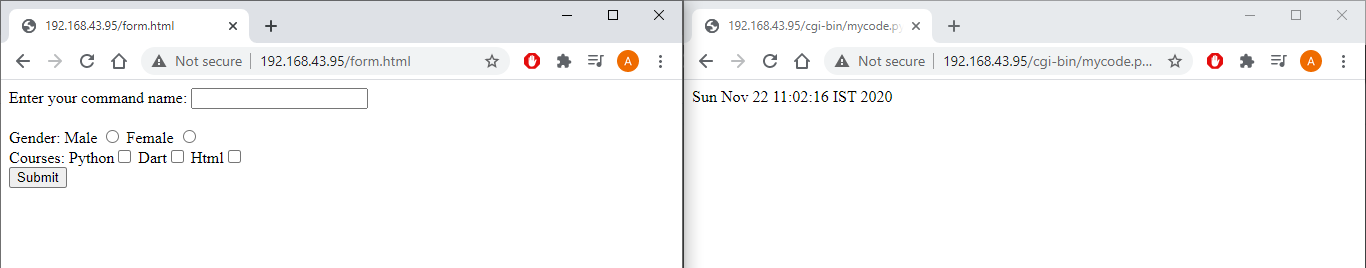


* + Here output is printing on the terminal or on top of client browser.
    - Here you have to tell which format you want print output on top of browser.
    - Here python output goes to the client using network, it will convert into the binary.
    - So, your browser doesn’t know convert this binary to the image, video or normal text.
  + This is the meaning of presentation layer.
  + Before printing output browser look into first line and search for content-type.
    - Content-type is a header, to tell your browser is header is till here and after that it is body part. For telling this we have to use one line before body.
* Sometime OS module not work. So instead of that you can use subprocess instead of OS module.
  + Import subprocess
  + X = Subprocess.getoutput(“date”)
  + Print(x)
* <http://192.168.43.128/cgi-bin/mycode.py>
  + This URL is going to OS and run something for you.
  + This is interface between client and your program.
  + Here you have to manage, what application you want to run.
  + This is known as Application Program Interface (API).
* In today’s world always all the URL server giving you API.

**Session 06 – html**

* Here our website / page is static, they are only running date command, but we need dynamic webApp.
  + For this they have to ask / prompt you in the webApp.
  + For this we have to use fieldstorage() function.
* In html some tags don’t have closing tags.
  + Eg, <br>
  + So, instead of this we can write like this.
    - <br />
    - This is the best way to write this type of tags.
* For taking input we have to use <input> tag.
  + By default, input type is text, for password we can write type=”password”
  + Input type=”submit”
  + You have to use name =”x” variable for holding data.
    - But still it will not send data via URL.
    - If you want to send data, all the box would part of big box – form tag.
  + You can also send data from one page to another page, for this you have to use action=” URL/path”





**Session 07 – redirection**

* Redirection
  + When you type <http://www.gmail.com> then it will redirect you to the different URL of Gmail.
  + In header of the webserver it is mentioned that when someone come to this URL then redirect you to this different URL.
  + You can check this thing using curl command.
    - Curl -I <URL>
    - Print(“location: <http://www.google.com>”)

**Session 08 – speech recognition**

* Import webbrowser
* Webbrowser.open([http://192.168.43.95/cgi-bin/mycode.py?x=”date](http://192.168.43.95/cgi-bin/mycode.py?x=)”)
  + It will open this URL in your default browser and run date command for you.
  + So, using this we can call API from python code.
* Library
  + One of the uses of library is, mike is one device, so library give you one driver so you can contact to the device.
  + For this we have to install pyaudio library.
    - This library can connect to your mic.
    - Pip install pyaudio
      * Here it will not work.
      * Instead of this we can use conda which comes from anaconda distribution.
      * This is more stable then pip.
    - Conda install pyaudio
  + For speech recognition we have to use one more library.
    - Here, your sound (Frequency) comes inside your system, but OS don’t know convert into which… English language or any other language or some other things.
    - Here we are using API from google for this.
      * Pip install speechrecognition
    - Import speech\_recognition as sr
    - R = Sr.recognizer()
    - With Sr.microphone() as source:  
       print(“Start speaking”)  
       audio = r.listen(source)  
       print(“Stop speaking”)
      * Here this line connects to pyaudio.
        + Sr.microphone() open your mike and frequency captured in source variable.

You can use any variable.

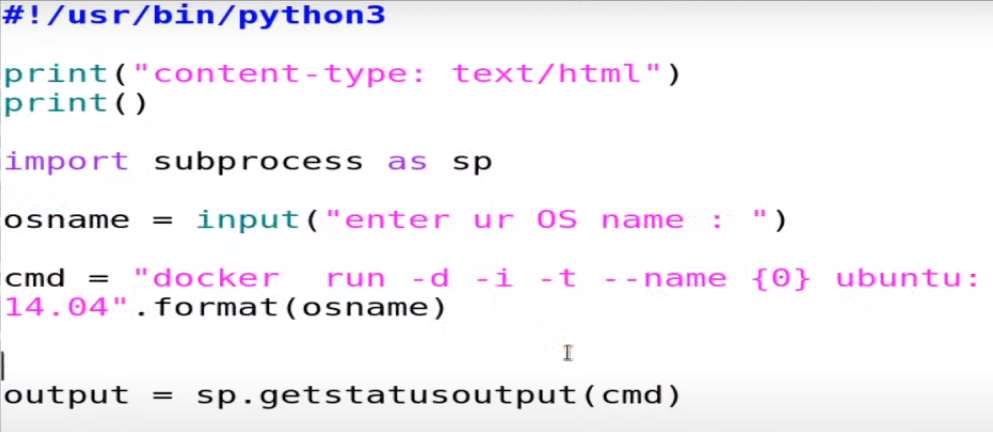
* + - * + Now mike will listen using r.listen to source variable.
      * When you stop talking with understand and disconnect microphone.
        + Here with is doing both open and close mike.
    - R.recognize\_google(audio)
      * It will give you text output of what you say.
      * Here we are using API and going to google so he can convert it into text.

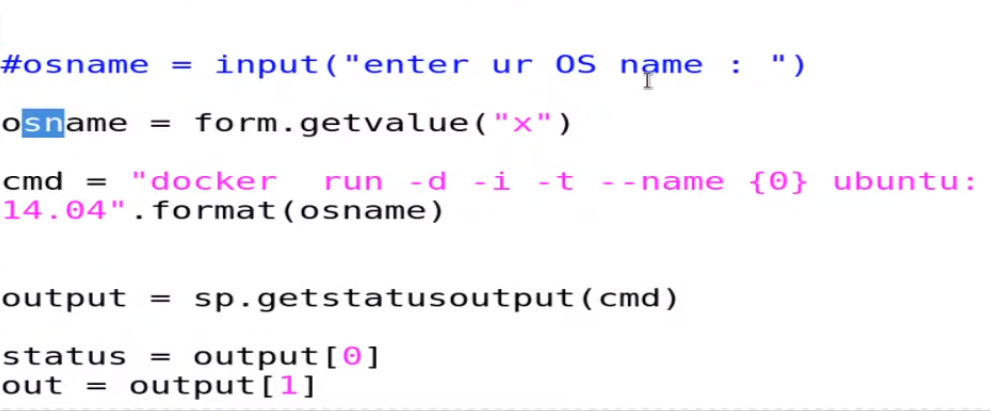
**Session 09 – getpass**

* When you type something, it will visible to your screen and when you pass password, sometimes it will show you \*\*\*\*\*\* or sometimes it will doesn’t show you anything.
  + Here when you typing something it will by default echo back to your screen but in case of password it is giving you something else. So, it will secure your credentials.
    - Keyboard is not deciding what to print.
    - It is decided by some program.
  + In python for this we have to use getpass() function.
  + This function comes from module called getpass.
    - Import getpass
    - Password = Getpass.getpass(“Enter password”)
* System vs subprocess
  + Os.system() will return you output.
    - Here in bg they have used print function so it will always print.
  + Subprocess.getoutput() will return you output.
    - It will not print the output if you don’t want.
    - Instead of this you can store output inside a variable.

**Session 10 – docker**

* Docker description – launching OS – basics.





**Session 11**

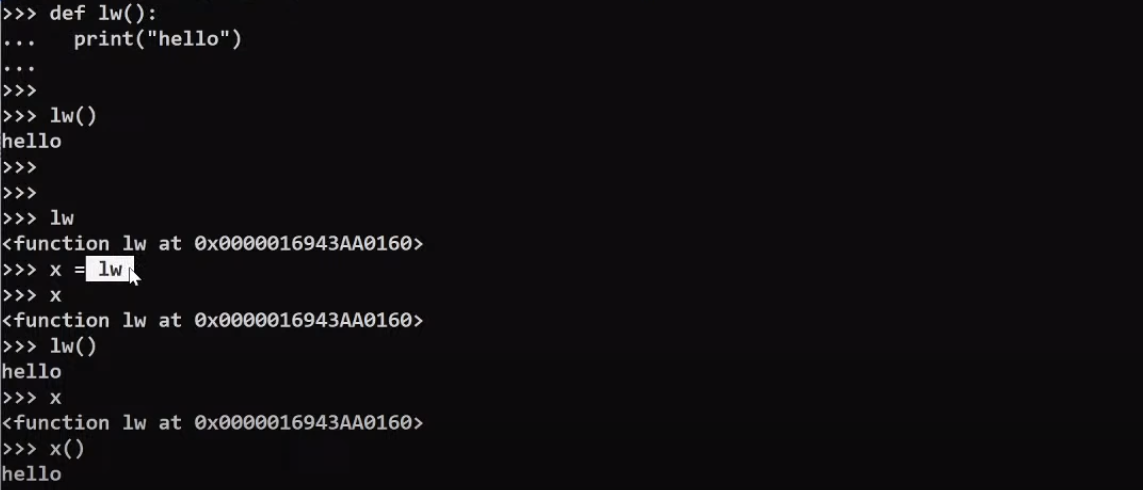
* Here when you use webUI or API – cgi then you are using apache user for doing anything.
  + But apache user doesn’t have more permissions so it will not run all the commands in all conditions.
  + For this you have to give power to the apache user.
  + For this you have to use /etc/sudoers file.
* Integrating docker with webApp.

**Session 12 – list comprehension/function**

* When we want to use something repeatedly finite number of times then we have to use while loop, but here we want to iterate same thing but we don’t know for how much time we want to execute this, here we have to use for loop.
  + If you using the database so it will repeat loop for till last record comes in the database and after last item we will come out from the for loop.
* **List comprehension** 
  + Without list comprehension
    - Y = [ ]  
      for I in x:  
       if I > 0:  
       j=i\*i  
       y.append[ j ]  
       else:  
       j=0  
       y.append[j]
  + With list comprehension
    - Y = [ i\*iif i>0 else 0 for I in x ]
  + Concept of readability
    - If you writing some business logic, try to complete the code in one single line.
      * May be in future you will also not able to read.
    - For this you have to use inline, it will improve code readability.
  + In newer approach we use functional language.
* **Functions**
  + When you code something you don’t have control on it. When this file run your code will automatically run.
    - But if we want to control it. We don’t want to run some codes automatically, then we have to use functions.
    - Here whenever you call function then only that block of code run.
    - Eg,
      * def ak():  
         print(“ Hey”)  
         print(“ This is a function”)
      * when you call this only then it will run.
    - Here we have to write this block of codes and whenever you need just call them.
  + **Benefits**

1. Control code when to use
2. Run one block of code multiple times.
3. You can make It dynamic using arguments.
   * There are two types of functions.
4. Static
5. Dynamic
   * To make function dynamic we have to use argument, we can pass argument while calling a function.
     + Def ak(\*I ):  
        print( i )  
       ak(1,2,3,4,5)  
       ak(1,2,7,8,5,6,7,8)
     + \*i will make you argument as tuple.
     + So, in one single variable you can store multiple values.

* Here **function is exactly equal to the variable**.
  + Variable will be one particular location in ram.
  + You can say same thing in function in case of python language.
    - You can assign function to variable.
  + Eg.



**Session 13 – dataset**

* Data
  + If you are collecting one information – 1 column of data, it is known as 1D data.
    - 2 columns == 2D data
  + Raw == record / column / object
  + Column == field / feature / dimension
  + You must know why you collecting data & in which format to store data.
* **After collecting data what can you do on top of dataset?**

1. Query
   1. It is also known as searching.
      1. Mobile number 97xxx belongs to which person.
      2. Yas has purchased which course.
   2. You must have data.
      1. If value is not there it will give you error.
      2. Historical data
   3. Query we do in raw wise.
2. Data analysis
   1. Who do this job is also known as data analyst?
      1. Total number of python course sold.
      2. Average mark in maths subject.
      3. Top three students in science.
   2. Here you can only analyze data.
      1. Data must be present.
      2. Historical data
   3. Analysis we do in column wise.
3. Data analytics
   1. prediction
      1. How much mark will anil get in this subject?
      2. They will give you data which is not present.
         1. But in the basis of past data.
   2. Artificial intelligence – machine learning

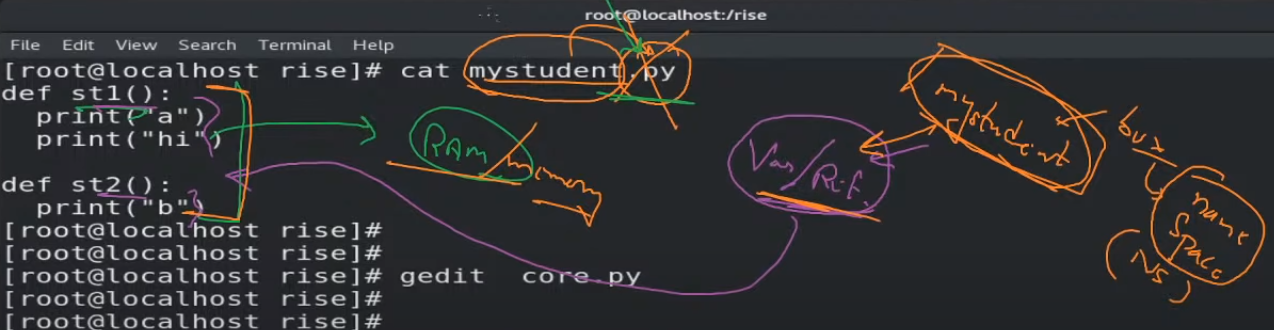
* Numpy
  + Db = [ ,fdsfdsf ,dfsd , sddsg]
  + Import numpy
  + A = numpy.array(db)
    - A.max()

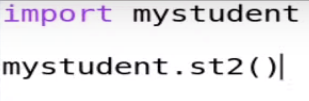
**Session 14 – numpy**

* A=np.array(db)
  + A.shape()
* In mathematics world 1D array == vector
  + Multidimension == matrix
* Here numpy have flexible datatype (<U4)by default.
  + They have many datatypes.
    - Numpy.ndarray is a structure type of np.
  + So, you cannot perform mathematics operation on top of this. If you try you will get error like can not perform operation on flexible datatypes.
  + Marks.astype(int)
    - It will convert flexible datatype into integer.
    - For this variable contains must be integer numbers.
* **Can we give column name in the numpy datatype?**
  + Yes
  + a=np.array(db)
  + a.dtype = { ‘names’: [ ‘StudentName’ , ‘Phone’, ‘Score’ ], ‘formats’: [ numpy.str, numpy.int32, numpy.int32] }
    - names, format are by default property of numpy.
    - But this thing is harder when you have thousands of columns.
  + So, on top of numpy developers have created pandas datatype in top of numpy.
  + It will give you lots of things.
    - One of the things it provides you is read\_csv.
    - Csv == comma saperated values.
  + Import pandas
  + Db = pandas.read\_csv(“file”)
    - Db[‘column\_name’]
    - Db[ [ ‘column\_1’, ‘column\_2’] ]
    - D.columns
  + D.iloc[1,1]

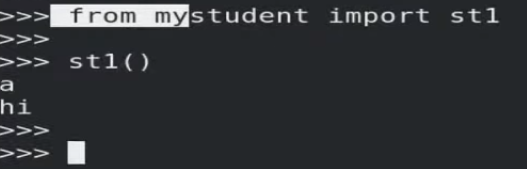
**Session 15 – Modules**

* For managing our codes, we are using functions, but sometimes we have to create 100’s of functions.
  + So, now it is harder to manage functions.
  + For this we can use modules.
  + For this we can create separate file for functions.
  + This program files are known as modules.
* When you import this module inside other file, then at that time (You can copy complete data here), python save this file all code in top of ram with variable same as module name.



* When you write import, they create name space and put entire data in this namespace.
  + - Mystudent variable created in ram. Here known as name space.
    - Name space contains 2 things.
      * Function created by us.
      * Some system related info.
  + In normal file
    - Function
    - Variable
  + In module
    - Method
    - Property
* Here when you import, they will consume space on ram.
  + So, it will waste our memory.
  + We need only one method and import complete module.
  + So, instead of this we can import only one function.
  + For programming world main namespace is entry gate.
    - In python we don’t create main function but behind the seen python create this for you.
* When you use import, it will create a completelynew namespace and give name same as the module name.
  + Then it will copy the complete code there.
* When you use **from module import method**, then it will **not create** a new namespace but copy code inside same namespace.

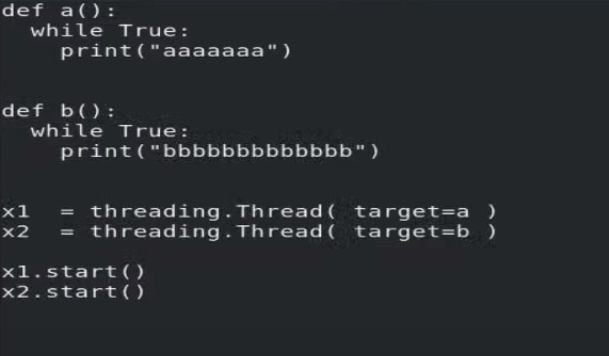
1. Save memory
   1. Don’t create new namespace
2. Here they copy code inside main namespace, so we don’t need to use module.method(), we can directly use method()



* When you import any module, first it will check to the current folder, then it will go to python module path(/usr/lib64/python3.6).

**Session 16 – multi** **thread**

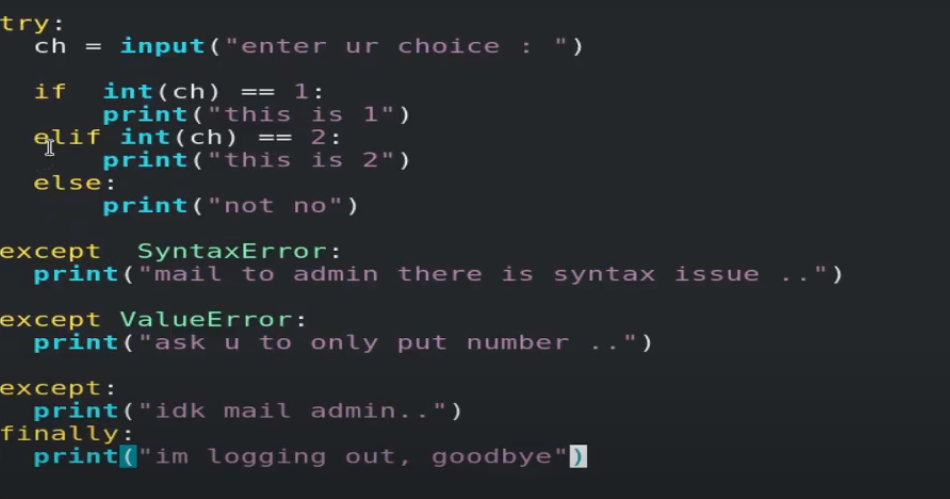
* Kernel will give every process a unique id known as process id(PID).
* **What is process id, any file folder or just a number, or a table?**
  + It is actually a folder or directory.
  + When you execute any program.
  + All code and related data they maintain in this folder and this folder is created on top of ram.
  + They give name to this folder known as PID.
  + **This folder is linked/mounted to one folder, /proc.**
    - Link == mount
  + When you kill the process or close the firefox or anything, then they remove this folder from ram.
    - Cd /proc
    - Ls
      * Cd <pid>
      * ls
* There is one limitation of process.
  + May be, we have 100’s of statement, process only send one instruction at a time.
  + CPU run this instruction and give you output.
    - Until the CPU gives output to this instruction, this process will not send any new instruction to CPU.
    - So, you can say this is a single threaded process.
  + **This is not a limitation of CPU; this is a limitation of process.**
  + If you want to see process is a single threaded or not.
    - For this inside process folder, go to status.
  + For solving this we have to send multiple statement at same time.
    - So, this is known as multi-threading.
    - In one process (Chrome), they are doing multiple things in parallel, then this is known as multi-threading.
    - OS doing lots of things parallel (Notepad, chrome), then this is known as multi-tasking.
  + Import threading



* + - So, we have 3 thread here, main and 2 different threads.
  + **Till the time program run, main function is kept on running.**
  + So **main thread**is continues running.
  + If you want to start the thread then you have to use x1.start, x2.start.

**Session 17 – Exceptional Handling**

* Sometimes functions have some limitation and we don’t know about every limitation. So, if by chance we pass some different value, then our complete App or Website can be crashed.
* For this we have to handle this type of exception.
* When you (System)don’t able to understand something then exception come up.
  + When exception come up then error comes up.
* We have to use try… except block for this.

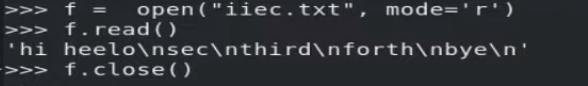


**Session 18 – File handling**

* We assign variable to particular location, so we can get value of that location using the variable, but in HD perspective it is not known as variable, it is known as File.
  + Ram – Variable/reference
  + HD – File/Filename
* In location of file (in INODE table), the location of first character available.
* If you use cat command you can do only one thing at a time.
  + Read
  + Write
  + Append
  + Read-write 🡪r+
    - We cannot create new file.
  + Read-write 🡪w+
    - Can create a new file.
    - If file already exists. Remove and create new file.
    - When you append you cannot read.
* Open(“Ak.py”, mode=’r’)
  + It will give you location of first character.
  + So, lets store it in the variable.
  + F = open(“Ak.py” , mode=’r’)
    - F.tell()
      * It will give you first letter position.
        + 0 (first character location is 0)
    - F.read()
      * In python read is equal to the for loop.
      * It will read first character, then print and move pointer to second position, do these things until EOF comes.
      * F.tell() will give you location, now it is on EOF.



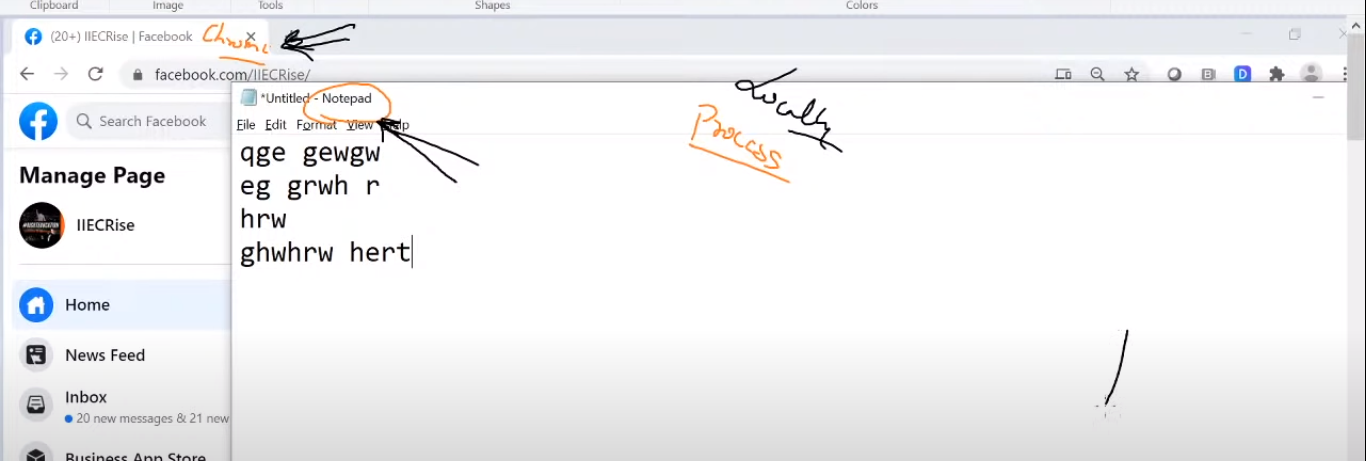




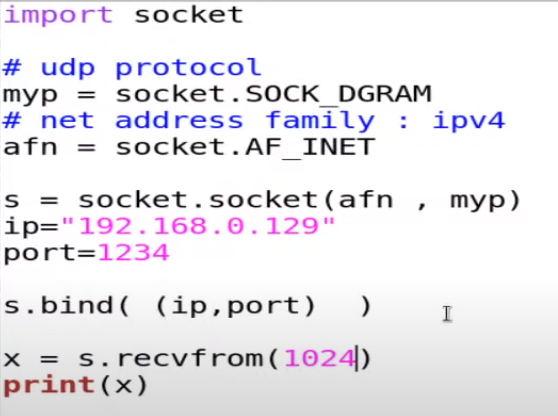
* + F.read(3), f.read(3)
    - Read first three character, next 3 character.
  + For reading again we have to close file and start again, so pointer can be come to first place.
    - Rewind the pointer is known as seek operation.
    - F.seek(3,0)
      * At third place.
* F = open(“Ak.py”, mode = ‘w’)
  + F.write(“Akshit here /n”)
    - it will remove data and new data stored in ram, until you close file.
  + F.close()

**Session 19 – socket programming**

* To take input from a network we have to use function called recvfrom().
  + Recvfrom(“Enter name:”)
* Here maybe we have multiple OS, so where to send data using network, may be multiple program in a single OS reading data from a network, so who receive data?



* + If you are using locally you can use mouse pointer to go to the different process.
    - Eg, you are typing and click on chrome behind to go to chrome process.
* But when you connect to remote system.
  + Here we have to use network.
  + Using IP, we can go to system, but we do not know which process we have to connect.
    - Here we cannot see program name or process name.
    - So that system (remote system) has given some unique no to that process.
  + This type of unique no is known as PORT number.
  + Using IP & port number data reach to the program.
  + Now its that program choice what to do with data.
* Some programdoesn’t want to get data from outside program, so this program is not bind with any port number.
  + Eg, notepad
* Combination of IP & port is also known as socket.
  + If you want to anybody can connect to your program you need socket.
  + For connecting IP & port number together we have to use bind function.
    - For using bind function we have to use bind module.
* Here we have to use anyone of the protocol, UDP or TCP.
* Here we are using UDP protocol.
  + Here we have lots of ways to give address one of famous version is IPv4.
    - For checking other address family name use dir(socket).



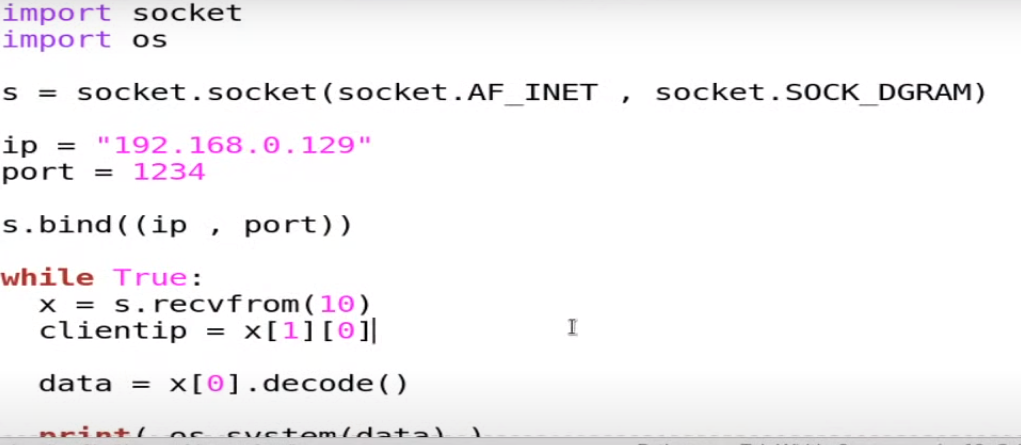
* + Now we can get data from network.
  + S = socket.socket(afn, myp)
    - Here we are passing which type of address family and which protocol you want to use.
  + Port 1234 is also started, for this you can use netstat –unlp.
    - U for UDP protocol.
  + You can send maximum 1024 size data,
    - It is known as maximum buffer size.
* Client side program.



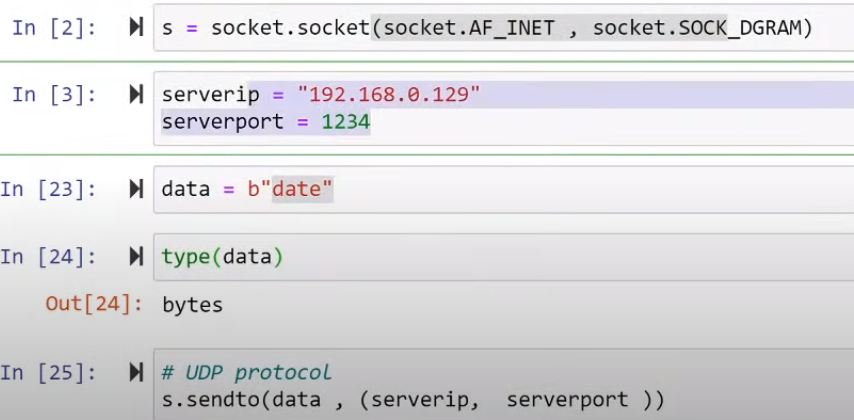
* + Python don’t support direct string over network.
  + While transferring data into network, python use bytes datatype.
    - For converting string into bytes we have to use string.encode().
    - This is the case of python, may be other language don’t transfer with this datatype.
    - String.decode() to decode string.

**Session 20 – socket programming//UDP**

* When server/receiver program will receive program, it will come up with socket of sender/client.
* If you want to print IP first you have to convert it into the normal string, so you can do some operation on top of this.
  + For this you can use string.decode().
  + Now we can also use loop so program is continuously running.
* **Server side**



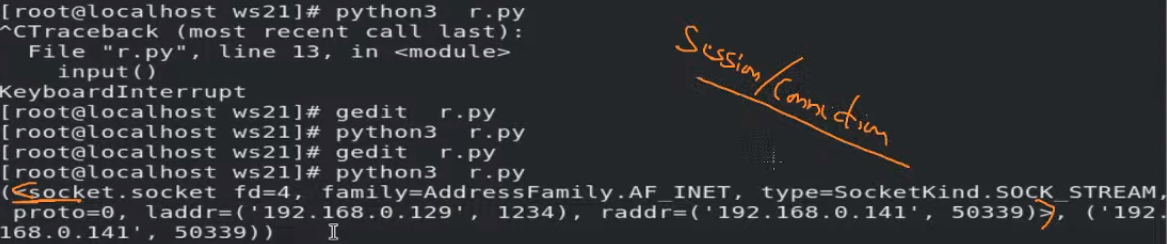
* **Client side**



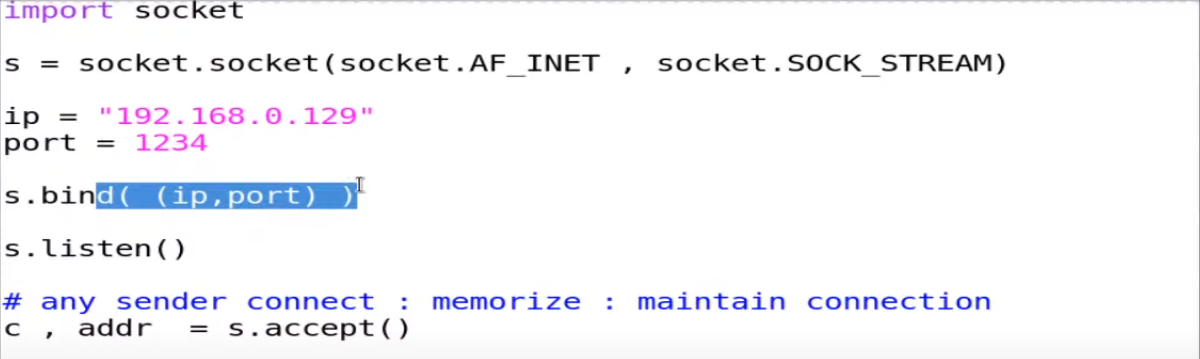
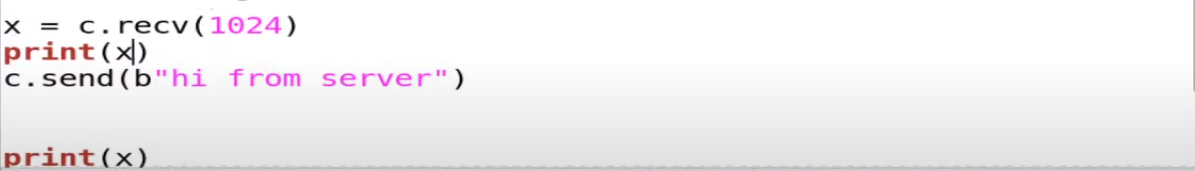
* Here if we use multiple client and give data at same time, it will not support so we have to use Multi-threading concept here.

**Session 21 – socket programming//TCP**

* In TCP first we establish connection using 3 way handshaking.
* In UDP there is no connection establish.
* Suppose if A is chatting to B.
  + Suddenly C come and msg to B, and B receive it.
* In TCP they will create separate connection for everyone.
* When sender connect to receiver, using TCP they will maintain some information like IP and port number.
* And they reply back to same information, IP & port number, so when you use firefox to connect to google.
  + One port starts in your system.
  + It is a sender port and your req go to google.
  + It will create one connection.
  + This thing is known as connection oriented.
  + It is known as transmission control protocol.
* TCP will create a session for you.
  + Sometime it is also known as stream.
  + If you want TCP will maintain the state then you have to use listen function, so they can listen.
* For connecting to receiver sender has to use s.connect( (IP,port) ).
  + It will create a connection for you.
  + For close the connection sender has to use s.close()
* Netstat -nct
  + To see on going connection.
  + Monitor live connection.
* Here B need capability so he can remember the session of A.
  + So, they can use this session for send// receive something.
  + He(B) has to accept the connection request from A.



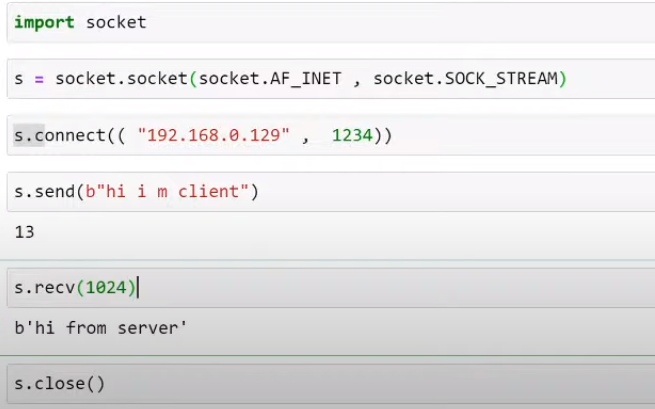
* + This entire box is an information about a sender and second thing is a sender address.
    - Here raddr == remote address
  + Here s.accept will accept the req from first user.
  + For multiple client you have to use multi threading.
* **Server**

* Server will wait for receiving the data and after that reply back to the client.
* This program will only run for one single client.
  + It is accepting only one request.
* **client**



* If client is busy or not receiving output, then the data(received) is in the buffer and when client free it will be available in the screen.
* Server program is stopped before receiving data still we can receive data.
  + Because it is available in our buffer.

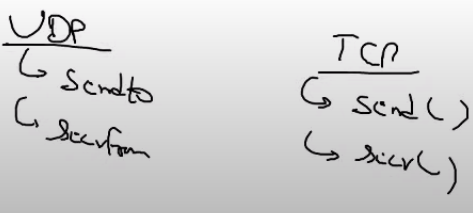


**UDP**

* No connect establish
  + Non reliable
* Receiver down
  + no error
  + package drop

**TCP**

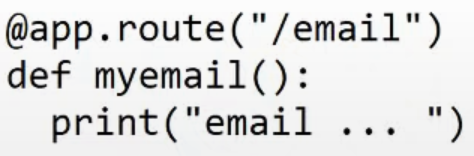
* Connection oriented
  + Reliable
* Receiver down
  + Try for some time
  + Gives you error if sender does not connect.
* Receiver reply back to sender using sender Info (IP,port).
* **Accept the session**.

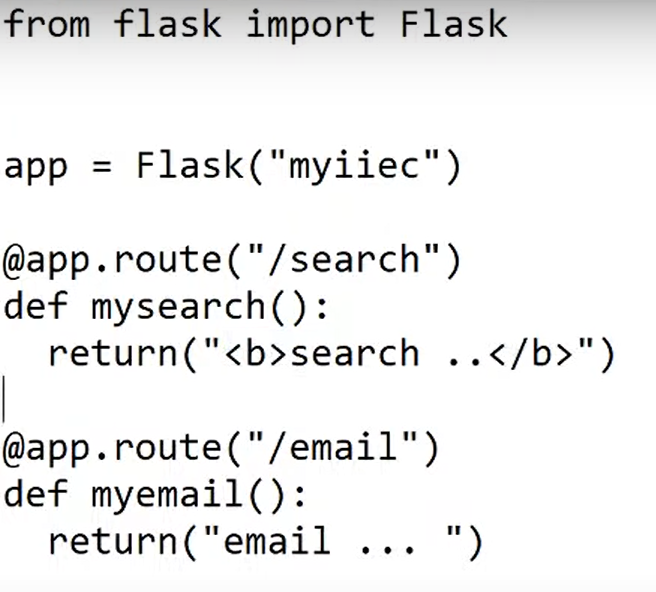


**Session 22 –**

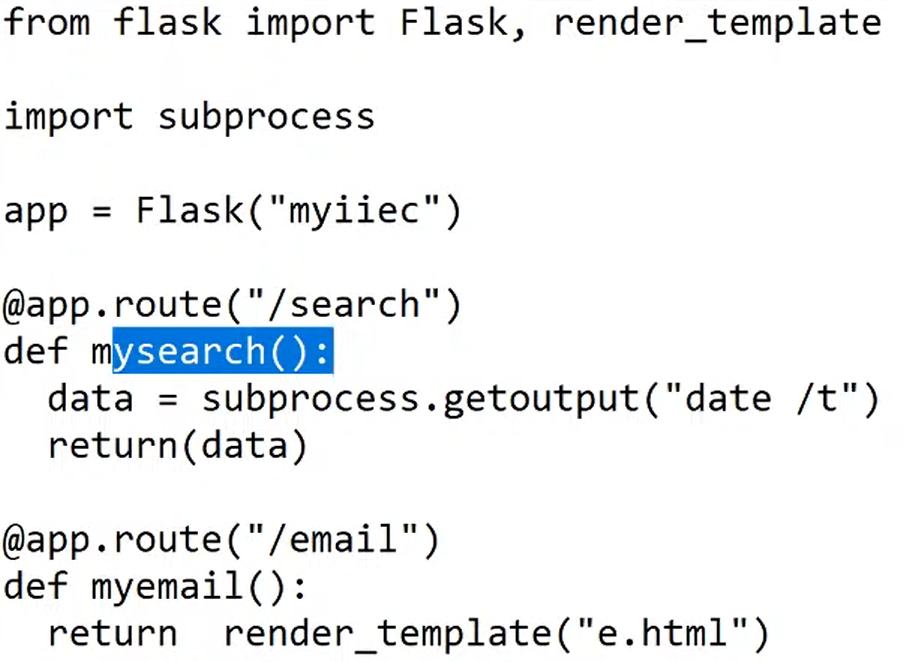
* In real world we require multiple clients connect to server parallelly.
* By default, socket program work for IPv4 & TCP protocol.
* If you don’t give IP name in server file, they will replace that with current IP address of system.
  + Here they first replace with 0.0.0.0 & when client connect they will allow current IP address and port combination.

**Session 27 – full stack development//Flask**

* We can create **webapp & restAPI** both using flask.
* Flask is a framework to create a RestAPI.
  + Here you don’t have to write basic internal codes.
  + Instead of this we can use pre created library.
  + It is known as **framework**.
* Pip install flask.
* Create a file with **app.py**
  + If we use print it will print in your terminal.
  + But we want to send it to the client.
    - This type of thing we have done in CGI classes.
* We are using flask for doing this.
  + For webserver we are using apache product.
  + But flask comes up with inbuild webserver.
  + Flask() will create a complete app for you.
    - You have to pass some name (App name) inside small brackets.
    - So here most of time we use object name app.
    - You can use route function, and it will automatically map to the function you have defined below that line.
    - But for this we have to use decorate here. 
    - Flask by default go to your current folder & search for app.py file and run for us.
    - Print will print in the black screen, we have to send it to client.
    - You can also add html tags here.

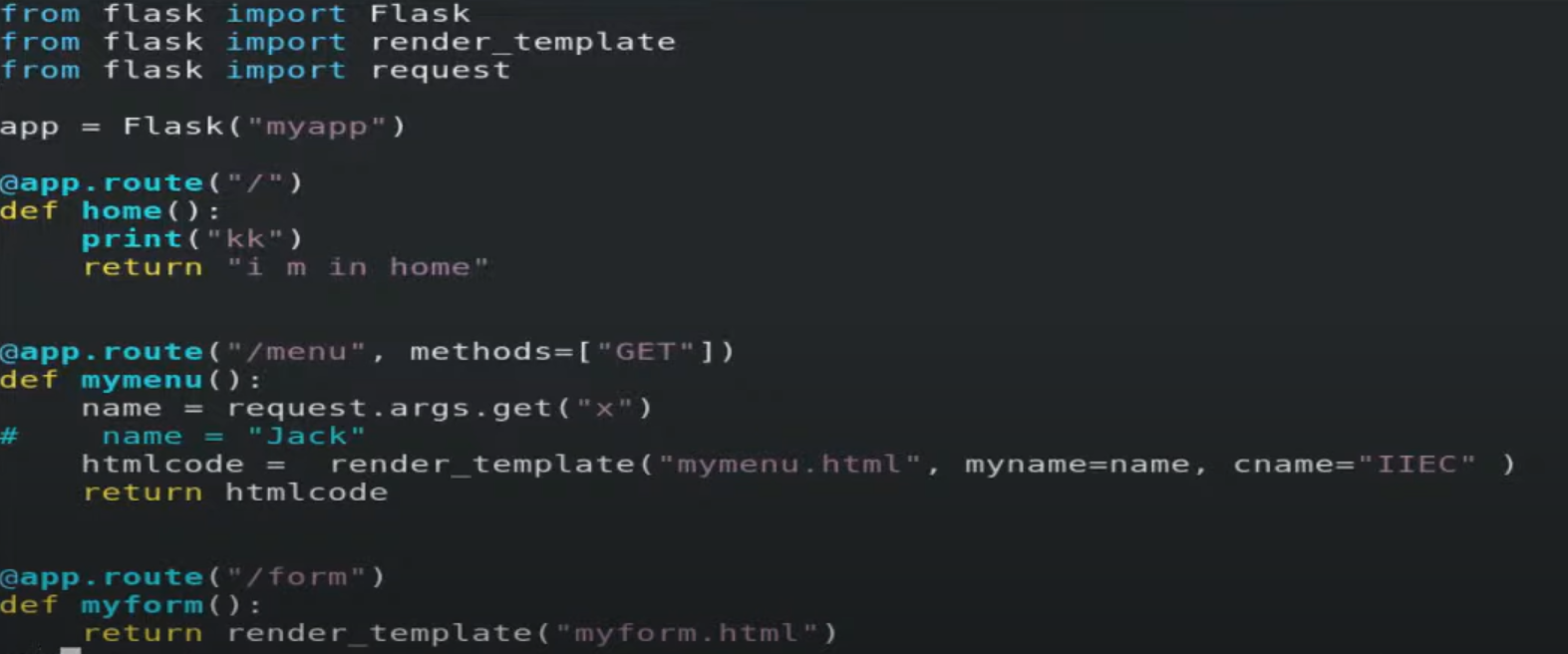


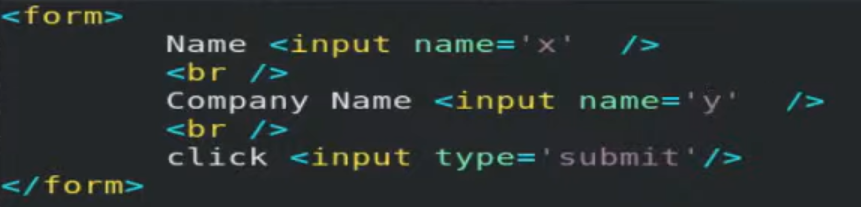
* + **Flash run** 
    - Flask is also show you logs when someone comes to your web page.
* In real world we have lots of code of html css javascript.
  + We cannot put all codes in single file.
  + So we can create a new file for it.
  + For html codes we have to create a folder templates and inside this folder create your html file.
    - For using that particular **template** you have to use **render\_template(“file\_name.html”)**



**Session 28 – hot reload//template//form**

* Sometimes it is harder to see error in black **terminal** as a **developer**.
* Client cannot able to see error.
* When we run flask, in terminal you can see we are running with production environment.
  + **Export FLASK\_ENV=development**
    - If you want flask can use this variable then you have to export it.
    - For windows you have to use set keyword.
  + Now if you run flask.
  + It will run with **developer environment**.
    - Here it comes with debug mode on.
    - Now if you check in browser it will show you errors also.
    - You can check errors here also.
    - At end you can see symbol of terminal.
      * Here you require to give pin.
      * It will come when you run flask run.
    - And from this you can access console here.
  + In developer mode, program will run as a **hot reload**.
    - It will auto detect change in live code and change restart flask automatically.
  + We have already created some static pages using flask.
    - Now suppose we want to change something on the fly.
    - We can also pass this in render\_template function.
    - For tell out html file this is variable we have to put variable inside {{ }}.
  + Any document who has capability to change data dynamically known as **template**.
    - We have created a form and give route /form.
  + Here we want to get input from client.
    - Client add input and submit it.
    - But we want to get this data in different route.
      * Eg want to print name of company in menu page and we want to enter it from client.
      * So we are getting input from form and want this data in menu page.
    - By default form send input to the same location where file exists.
    - So for this we can use action attribute.
  + For taking input from form in menu route we have to use **methods** keyword in same line where we declare route.
    - For getting data we have to use **requests.args.get()** function.





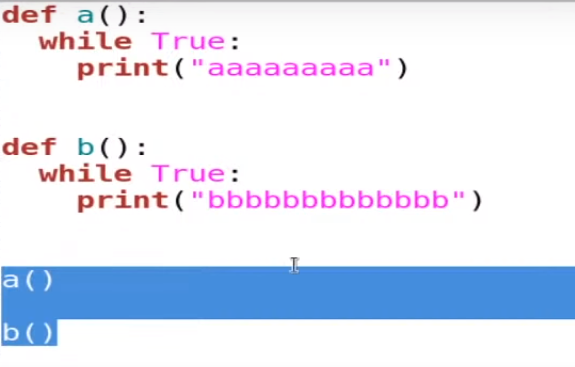
**Questions**

* How to read data from ram?
* How to WhatsApp anybody?
* How to reverse a string?
  + String = ‘’How are you?’’
  + String[-1: : -1]
* What is the difference between checkbox and radio box?
  + Radio box is just a variable, while checkbox is a list.
    - So, you can store only one value in one variable, while in checkbox you can give multiple choices.
    - We use this concept in radio box, so when you select first option it will overwrite and last data removed.
  + So, you can say you can select only one button, it’s a functionality of variable. It’s not because of radio box.
  + So, there is mainly 2 difference.

1. Radio box stores data in variable, checkbox store in list.
2. You can unselect in checkbox, but in radio box you cannot unselect selected choice.



* What is a difference between list and tuple?
* Can we give column name in the numpy datatype?
  + Yes
  + a=np.array(db)
  + a.dtype = { ‘names’: [ ‘StudentName’ , ‘Phone’, ‘Score’ ], ‘formats’: [ numpy.str, numpy.int32, numpy.int32] }
    - names, format are by default property of numpy.
* How to run both infinite function parallel?
* PID is just a number or any file or folder? Is it any table?
  + It is actually a folder or directory.
  + When you execute any program.



**Content**

* **Session 01 – basic languages**
  + Programming language
  + Installing python
  + Anaconda distribution
  + Variables
  + Builtin function
    - Dir(\_\_builtins\_\_)
  + Modules – program file
  + Os.system()
  + Pip install pyttsx3
    - Pyttsx3.speak(“Hey how are you”)
* **Session 02**
  + Program file
  + Program – process
  + Type Inference
  + List datatype
  + Slicing operator
  + Numpy
* **Session 03 – if-else**
  + Jupyter notebook
  + Escape sequence
  + If – else
* **Session 04 – while** 
  + String
  + In, And, Or
  + while
  + break
* **session 05 – CGI/API**
  + Common Gateway Interface (CGI)
  + Installing webserver – httpd
    - /var/www/cgi-bin
    - Execute file
  + Stop firewall
  + Shebang – hashbang (#!)
  + Header – body
  + Presentation layer
  + Subprocess module
  + Application program interface (API)
* **Session 06 – html** 
  + Static-dynamic webapp
  + Html
    - Form – input – radio button-checkbox
  + Subprocess
  + Fieldstorage()
* **Session 07 – redirection** 
  + Redirection
  + Curl -I <url>
* **Session 08 – speech recognition** 
  + Webbrowser.open()
  + Conda install pyaudio
  + Pip install speechrecognition
  + Started using mic and take input using this.
* **Session 09 – getpass** 
  + Echo back to screen
  + Getpass
  + Os.system() vs subprocess.getoutput()
  + Server-side language
* **Session 10 – docker** 
  + Docker basics
  + Starting Integrating with cgi
* **Session 11**
  + Apache user’s permissions
    - /etc/sudoers
  + Integrating docker with webApp.
* **Session 12 – list comprehension**
  + For loop
  + List comprehension
    - Inline
    - Readability
  + Functions
    - Call – argument
    - Types
    - Benefits
    - Return
  + Functions is equal to variable.
    - You can assign function to variable.
* **Session 13 – dataset** 
  + Data
    - Query
    - Data analysis
    - Data analytics
  + Numpy
    - np.array()
    - Slicing operator
      * A.max()
      * A.sum()
      * A.mean()
* **Session 14 – numpy** 
  + 1D array == vector
  + Multi dimension array == matrix
  + Flexible datatypes
  + unicode
  + Marks.astype(int)
  + Giving column name of np
    - A.dtype
  + Numpy
  + Csv == comma saperated values
    - Read\_csv()
  + D.ilock[1]
* **Session 15 – modules** 
  + Modules – methods
  + Namespace
  + Import
    - Create new namespace copy code there.
  + From
    - Copy code inside main namespace.
  + Python module path
* **Session 16 – multi thread**
  + kernel
  + Process id (PID)
  + Instruction
  + Limitation of process
  + Cat /proc/<pid>/status
  + Import threading
    - X1=threading.Thread(target = a)
    - X1.start()
* **Session 17 – Exceptional handling**
  + Limitations/bugs
  + Website crash
  + Error and exception
  + Try – cath – finally
* **Session 18 – File handling** 
  + Ram
    - Variable/reference
  + HD
    - Filename
  + Mode
    - R
    - W
    - A
    - R+
      * Read write – old file
    - W+
      * Read write – create a new file
      * Remove old file
    - A+
  + F = open(“File”, ‘mode’)
    - F.read()
    - F.tell()
    - F.seek()
    - F.write()
    - F.close()
* **Session 19 – socket programming**
  + Recvfrom()
  + Port
  + Socket
  + Bind function
  + Network address family
  + Protocol
  + Maximum buffer size
  + S.encode()
* **Session 20 – socket programming//UDP**
  + S.decode()
  + Client-server using udp socket programming
* **Session 21 – socket programming//TCP**
  + 3 way handshaking
  + Connection oriented
  + Stream
  + Netstat -nct
* **Session 27 – full stack development//flask** 
  + Framework
  + Pip install flask
  + Templates
  + App.py
  + Flask run
  + render\_template(“file\_name.html”)
* **session 28 – hot reload//templates//form**
  + developer environment
    - debug
  + Export FLASK\_ENV=development
  + Hot reload
  + Templates
  + requests.args.get()