PYTHON

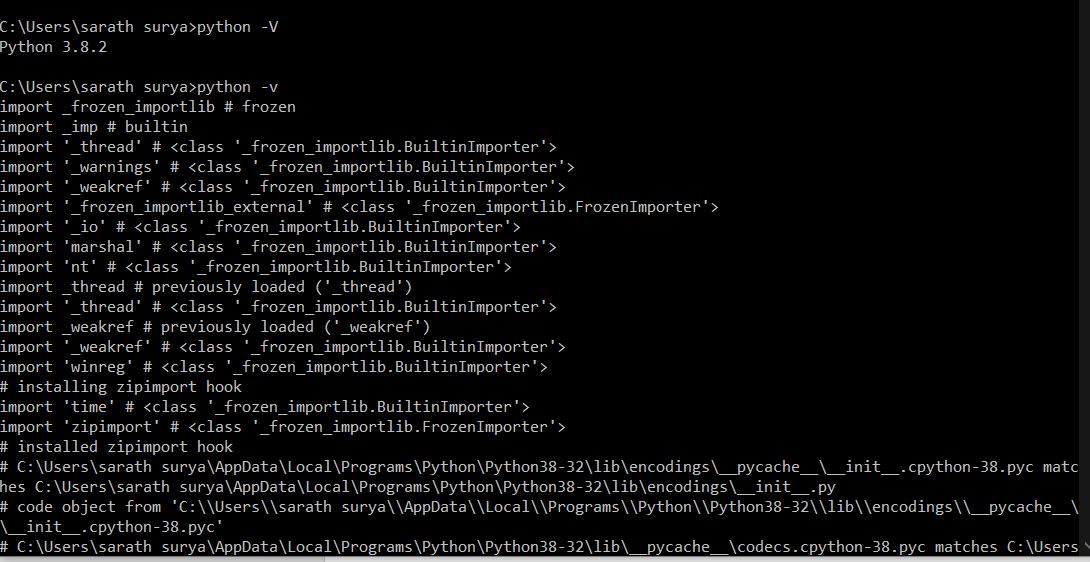
Basics:

1. To Check the python is install or not check

Open CMD type python -V. it shows the current version

* 1. By using the python -v. It shows all the Supported data

By using a and b



1. Use visual studio code in that users click on extensions download the python and code runner (code runner is used easy execute the program)

Sample Program:

* >> is the symbol of a command-line interpreter
* Helloworld.py
* In the Editor

Print(“HI”);

* For output type python Helloworld.py

In terminal

* Output: HI
* The comment for the single line is #
* The Comment for Multiple lines is “”” and the end is “””
* In python take as a (‘ ‘) single quote and (“ ”) double as the same way

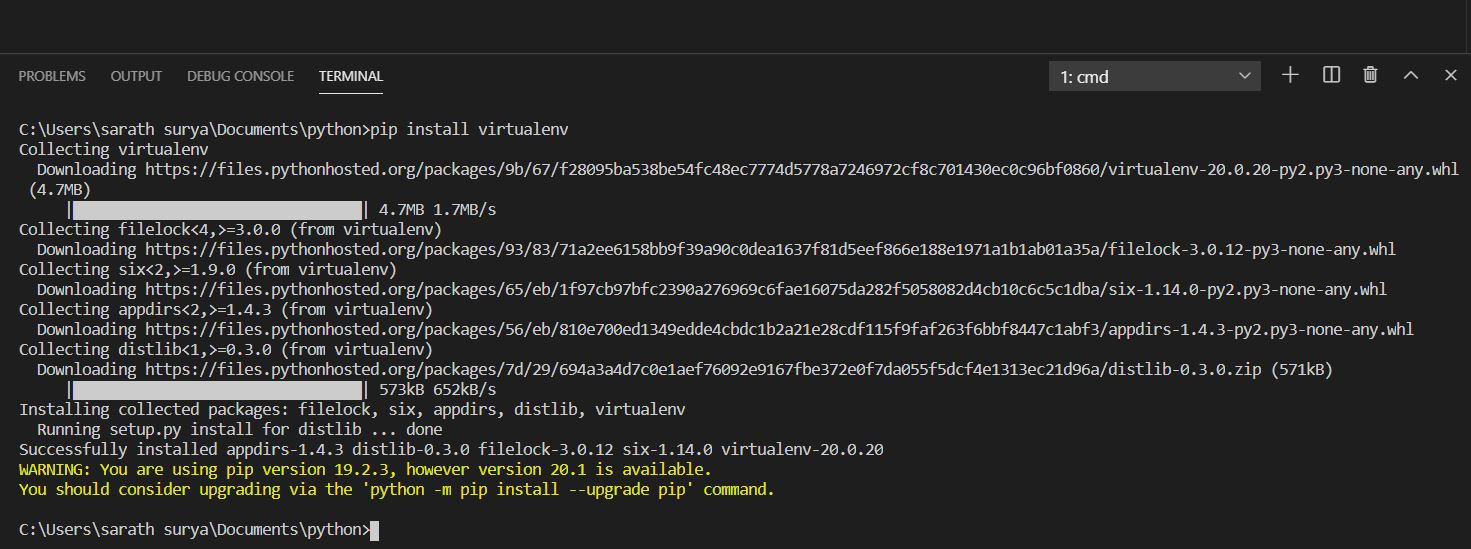
Create a Virtual Environment:

* Used to create a secure environment
* Capable of support any new version available

(Generally complex programs are not capable of new versions. it does not support)

1. INSTALL:

* Click on the terminal in visual studio code
* Type pip install virtualenv
* Pip is package manager, to install any package we use pip



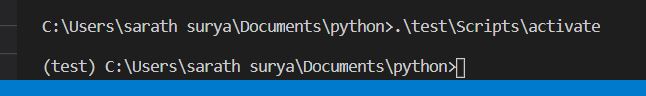
1. Create:

* Type virtualenv test to create the virtual folder

1. Activate:

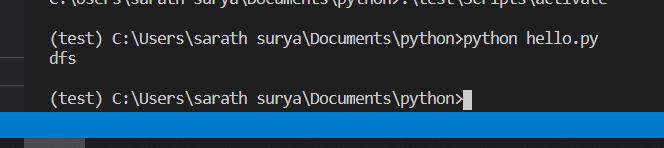
* To activate the folder type

.\test\Scripts\activate



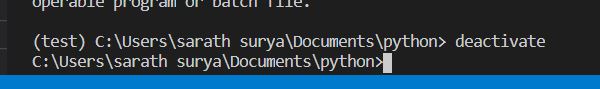
1. Run:

* To run the program type. Python hello.py



1. To Exit

* Type deactivate



Unix Basic Commands used in Terminal:

* mkdir is used to create the directory

EX- mkdir folder

* cd is changing the directory

EX- cd folder

* sdir [ls] is the list in windows
* rmddir is used to delete the directory

EX- rmdir folder

* echo %cd% is pwd present working directory

echo %cd%

C:\Users\sarath Surya

* cls [clear] is the clear the windows terminal
* cd .. to back the directory

gTTS:

* **gTTS** (Google Text-to-Speech)
* a Python library and CLI tool to interface with Google Translates text-to-speech API. Write spoken mp3 data to a file
* To Install type pip install gTTS

Text to audio:

# # Creating an audio file and play

# import google text to speech library

from gtts import gTTS

# import os package

import os

# process text to audio

text = gTTS("welcome to Xplore")

# write the file

text.save('welcome\_to\_xplore.mp3')

#play the file

#os.system("mpg321 welcome\_to\_xplore.mp3") # mac command to play mpg321

os.system("start welcome\_to\_xplore.mp3")

#windows command

Assignment day 1:

* To get the sentence from the user and convert into the British accent and American accent

from gtts import gTTS

import os

a=input("write the sentence\n")

num=int(input("press 1. British accent 2. American accent\n"))

if num==1:

    b="    in british accent"

    c=a+b

    x=gTTS(c , lang='en-uk')

else:

    b="    in American accent"

    c=a+b

    x=gTTS(c , lang='en-us')

x.save("audio.mp3")

Variable:

**Variables** are the names you give to computer memory locations which are used to store values in a computer program.

* In python a=10
* We can use a semicolon or not python compiler doesn’t care.
* In python it is an automatic data type we do not need to mention the datatype. (float, double, char, int, long)
* a = 10
* print(a)
* a = "hello"
* print(a)
* first output 10 and second output hello.

Changing a Variable:

A=110

B=A

Converting to a string:

print("sd" +str(123))

* output: sd123
* a="hell"
* print("sd "+a)
* outpu: sd hell

Condition Statements:

**If Statements:**

An **if** statement is a programming conditional statement that, **if** proved true, performs a function. Otherwise it checks the **elif**(else) statement. If it is true it performs otherwise it will go to the **else** statement.

In python, a colon (:) is added after the condition. the statements should be written in the **tab space or indentation**.

: it is a scope operator.

a = 10

b = 20

if(a>b):

    print("a is big")

elif(a<b):

    print("b is big")

else:

    print("same")

output: is b is big

**In a single line:**

If the condition satisfies it can take prefix otherwise postfix. ( interview)

a = 10

b = 20

msh="a is big" if(a>b) else "b is big"

print(msh)

Function:

Def say that defining the function. Def basic():

Function to work we should invoke the function. Basic().

def basic():

    print("this basic")

basic()

output: this basic

**function with arguments:**

Sending the parameter. Defining in a variable.

def sample(a,b):

    print(a,b)

sample(10,20)

output: this 10,20

**function with Return:**

the Returning value is sent in sum and it is collected by the s.

def add(a,b):

    sum=a+b

    return sum

s=add(20,30)

print(s)

output: this 50

We can send the value without an variable and print

def mul(a,b,c):

return a\*b\*c

print(mul(10,20,30))

def mul(a,b,c):

    s=a\*b\*c

    return s

s=mul(10,20,3)

print(s)

**Function with default value:**

If we do not declare a variable it took as default. While c=1 is a default value in the function.

def mul(a,b,c=1):

    m=a\*b\*c

    return m

m=mul(10,20)

print(m)

def pow(n,e=1):

    result=1

    i in range(e):

        result=result\*n

    return result

print(pow(5,2))

**Function with default value:**

Assignment day 2:

**Speech Recognition:**

First we need to install the

pip install SpeechRecognition

**saved audio to text:**

1. import speech\_recognition as sr         # import speech library
2. r = sr.Recognizer()                     # create a speech object
3. file = sr.AudioFile('sample.wav')       # take file path
4. with file as source:                    # load the file
5. audio = r.record(source)            # record from source
6. text = r.recognize\_google(audio)        # recognize the speech
7. print(text)                             # print the speech

**2.audio to open a browser and search**

## Microproject to open browser-based on the audio clip

# import speech library & web browser

import speech\_recognition as sr

import webbrowser

# create recognizer object, load the file and open the audio

r = sr.Recognizer()

file = sr.AudioFile('open.wav')

with file as source:

    audio = r.record(source)

# use google speech recognition to convert audio to text

text = r.recognize\_google(audio)

# open the webbrowser based on the text

if(text=='open Facebook'):

    webbrowser.open('https://facebook.com')

else:

    print(text+' - unable to recognize')                             # print the speech

**Function with variable Arguments:**

It takes \* arg as 54,45,5,66 all the elements and arg as no variables

def adf(\*arg):

    sum=0;

#if we print args it shows(53,54,5,6,6)

    for i in arg:

        sum=sum+i;

    return sum;

print(adf(54,45,5,6,6))

**Celsius to Farenheit:**

def cal(a):

    ans=(a \* 9/5) + 32

    return ans

f=int(input("c to fh\n"))

if (f<100):

    print(cal(f))

else:

    print("invalid")

**Global variable:**

a=1

def g():

    print(a)

g()

**Output:** 1

When we are declaring a local variable the variable is consist inside only

a=1

def g():

    a=10

    print(a)

g() #10

print(a) #1

**Output:** 10 and 1

When we declare it global keyword it remains constant. We have to declare global a not global a=10

a=1

def g():

    global a

    a=10

    print(a)

g()

print(a)

**Output:** 10 and 10

global a

a=1

def g():

    a=10

    print(a)

g()

print(a)

**Output:** 1 and 10

**Looping:**

**While:**

It will repeat the process unless the condition is false

i=0

while(i<5):

    print(i)

    i=i+1

**output:** 1234

**INCREMENT OR DECREMENT:**

Increment and decrement does not work in python we can use i=i+1 or i+=1 (Interview)

**For:**

It will repeat the process with in the range

The interviewer asks the is the upper bound in for loop will print or not answer no

for i in range(2,5):

    print(i)

**Output:** 2,3,4

for i in range(5):

    print(i)

**Output:** 01234

the third one is the step function is used to skip that numbers

for i in range(2,10,3):

    print(i)

**Output: 258**

**For loop in string:**

s="hello"

k=len(s)

for i in range(0,k):

    print(s[i])

**Output:**

h

e

l

l

o

**Break:**

It used to break the statements

for i in range(10):

    print(i)

    if(i==5):

        break

**Output:** 012345

**Continue:**

for i in range(10):

    if(i%2==0):

        continue;

    print(i)

**Output:** 13579

Assignment day 3:

First, we need to install the pillow

In cmd: pip install pillow

from PIL import Image,ImageFilter

#loading img

im = Image.open('h.jpg')

#turing img

n=1

m=int(input("image manipulation\n1.B/W\t2.Blur\t3.Emboss\t4.two pics horizontal\n"))

if(m==1):

    #b\w

    im = im.convert("L") #grey scale

elif(m==2):

    #blur

    im=im.filter(ImageFilter.BLUR)

elif(m==3):

    im=im.filter(ImageFilter.EMBOSS)

elif(m==4):

    #2 pics

    im1= Image.new('RGB', (im.width + im.width, im.height))

    #creating a empty img

    im1.paste(im, (0, 0))

    im1.paste(im, (im.width,0))

    im1.save('h1.jpg')

    im1.show()

    n=0

if(n==1):

    #rotating

    r=int(input("Do you want to rotate the pic or not\n1.yes 2.no\n"))

    if(r==1):

        r1=int(input("Enter the angle\n"))

        im=im.rotate(r1)

    #entering the quality

    q=int(input("enter the quality\n"))

    #size

    x=int(input("enter the x axis size\n"))

    y=int(input("enter the y axis size\n"))

    im=im.resize((x,y))

    #saving

    im.save('h1.jpg')

    im.show()

**For loop else:**

It is an additional feature in the python. If the loop is success Completed then it will execute the else statement. If the loop is terminated then else block will not Execute.

For loop success

for i in range(10):

    print(i)

else:

    print("exe")

**Output**: 123456789 exe

For loop Terminated

for i in range(10):

    print(i)

    if(i==5):

        break

else:

    print("exe")

**Output**: 12345

**List:**

This is an array type in the python. Used to store the values.

We use [] Square brackets.

l1=[5,6,8,6,9,7]

1.

l1=[5,6,8,6,9,7]

for i in l1:

    print(l1)

**Output:**

[5, 6, 8, 6, 9, 7]

[5, 6, 8, 6, 9, 7]

[5, 6, 8, 6, 9, 7]

[5, 6, 8, 6, 9, 7]

[5, 6, 8, 6, 9, 7]

[5, 6, 8, 6, 9, 7]

2.

l1=[5,6,8,6,9,7]

for i in l1:

    print(i)

**Output:**

5

6

8

6

9

7

3.

l1=[5,6,8,6,9,7]

for i in l1:

    print(l1[2])

**Output:**

8

8

8

8

8

8

**4.Enumerate:**

Enumerate is a function which returns key and value

Here ‘i’ is the key value and ‘j’ is the value

l1=[5,6,8,6,9,7]

for i,j in enumerate(l1):

    print(i,j)

**Output:**

0 5

1 6

2 8

3 6

4 9

1. 7

**5.Printing full list:**

l=[3,4,48,48,56,56,12,5,4]

print(l)

**Output:**

[3, 4, 48, 48, 56, 56, 12, 5, 4]

**6. Printing based on the index:**

l=[3,4,48,48,56,56,12,5,4]

print(l)

print(l[2])

print(l[2],l[5])

**Output:**

48

48 56

**7.Printing Start and End:**

print(l[0:5])

**Output:**

[3, 4, 48, 48, 56]

**8.Printing Step:**

print(l[0:8:2])

**Output:**

[3, 48, 56, 12]

**9. Manpulate:**

l=[3,4,48,48,56,56,12,5,4]

l[2]=34

print(l)

l.append(54) #adding at last 0,1,2

print(l)

l.insert(2,89) #insert at two

print(l)

l.remove(56) #remove 56

print(l)

i=l.index(12) #position of 12

print(i)

**Output:**

[3, 4, 48, 48, 56, 56, 12, 5, 4]

48

48 56

[3, 4, 48, 48, 56]

[3, 48, 56, 12]

6

**10.Generate the List with Range Function:**

We can use step function or nor.

r=list(range(2,10,3))

print(r)

**Output:**

[2, 5, 8]

**11.Tuples: -immutable**

We cannot change the values.By using a circlar Brackets. We cannot add,remove the values

t=(1,2,3)

print(t)

print(t[0])

Output:

(1, 2, 3)

1

**12.String Operations:**

kk="sarath surya".capitalize() #only Fist letter Capital

print(kk)

a = "Hello, World!" #sting lenght

print(len(a))

a = " Hello, World! "

print(a.strip()) # returns "Hello, World!"

a = "Hello, World!"

print(a.lower()) #lower case

a = "Hello, World!"

print(a.upper())

a = "Hello, World!"

print(a.replace("H", "J"))

a = "Hello, World!"

print(a.split(",")) # returns ['Hello', ' World!']

txt = "50800"

x = txt.isdigit() #is nummber

print(x)

txt = "CompanyX"

x = txt.isalpha() #is alphabet

print(x)

txt = "Company12"

x = txt.isalnum()   #is number and alphabet

print(x)

txt = "Hello, welcome to my world."

x = txt.index("welcome") #tells index

print(x) #7

txt = "I love apples, apple are my favorite fruit"

x = txt.count("apple") #how many repeated

print(x) #2

txt = "banana"

x = txt.center(20) #creates Space

print(x)

**Output:**

Sarath surya

13

Hello, World!

hello, world!

HELLO, WORLD!

Jello, World!

['Hello', ' World!']

True

True

True

7

2

       banana

**Multiline String:**

It can take a string as multiple lines.

line= """trgf

fdg

gf

dg"""

print(line)

**Output:**

trgf

fdg

gf

dg

**Format String:**

Used to push the values

1.

s="my name is {},and my age is {}".format("sarath",22)

print(s)

**Output:**

my name is Sarath, and my age is 22

2.

s="my name is {1},and my age is {0}".format("sarath",22)

print(s)

**Output:**

my name is 22, and my age is Sarath

3.

If we prefix the f. the compiler takes as a Format String

name="sarath"

age=22

s=f"my name is {name},and my age is {age}"

print(s)

**Output:**

my name is Sarath, and my age is 22

Assignment day :

# Write a program to print names of months using lists and functions

# in different formats

# format 1 - Januaray, Febuary...December

# format 2 - Jan, Feb... Dec

# format 3 - 1 Jan,2 Feb,....12 Dec

months = ['January','February','March','April','May','June','July','August','September','October','November','December']

def format1():

    str = ''

    for m in months:

            str = str + m + ' '

        print(str)

format1()

def format2():

    str = ''

    for m in months:

        m = m[0:3]

        str = str + m  + ' '

    print(str)

format2()

def format3():

    str = ''

    for num,month in enumerate(months):

        print(num+1,month)

format3()

**Output:**

January February March April May June July August September October November December

Jan Feb Mar Apr May Jun Jul Aug Sep Oct Nov Dec

1 January

2 February

3 March

4 April

5 May

6 June

7 July

8 August

9 September

10 October

11 November

12 December

**Python Dictionaries:**

A dictionary is a collection that is unordered, changeable, and indexed. In Python, dictionaries are written

with curly brackets, and they have keys and values.

1.

#dic

d={"name":"sarath","age":20,"ad":"hyd","male":True}

#all

print(d)

#values

print(d["name"])

**Output:**

{'name': 'sarath', 'age': 20, 'ad': 'hyd', 'male': True}

sarath

2.

for i in d:

    print(i,d[i])

for i in d.keys():

    print(i)

for i in d.values():

    print(i)

for i,j in d.items():

    print(i,j)

**Output:**

name sarath

age 20

ad hyd

male True

name

age

ad

male

sarath

20

hyd

True

name sarath

age 20

ad hyd

male True

3.

d["name"]="surya"

print(d["name"])

print("roll" in d)

print(20 in d.values())

**Output:**

surya

False

True

**Set:**

It does not have index.

a=set("12340")

b=set("45656")

print(a) #duplicates not allowed the numbers change each time execution it change the order

print(a-b) #only A

print(a &b) #aᴖb

print(a|b) #aUb

**Output:**

{'1', '3', '0', '2', '4'}

{'2', '1', '3', '0'}

{'4'}

{'1', '0', '5', '2', '4', '3', '6'}

**Exception:**

**1.**

try:

    a=int("hello")

    print(a)

except:

    print('error')

**Output:**

error

2.Named Error

#value error

try:

    a=int("hello")

    print(a)

except ValueError:

    print('error in value')

except ZeroDivisionError:

    print("10/0 how re")

except:

    print("errror")

(or)

#or

try:

    a=int("hello")

    print(a)

except ValueError:print('error in value')

except ZeroDivisionError:print("10/0 how re")

except:print("errror")

**Output:**

error in value

3.If error no found then it will execute and go to else block

#else in exception

try:

    a=int("54")

    print(a)

except ValueError:

    print('error in value')

except ZeroDivisionError:

    print("10/0 how re")

except:

    print("errror")

else:

    print("error notfound")

**Output:**

54

error notfound

4.Error reporting

It used to customize error

def func(\*args):

    l = len(args)

    if(l<1):

        raise TypeError("Arguments cannot be less than one")

    for i in args:

        print(i)

try:

    func()

except TypeError as e:

    print(f"TypeError: {e}")

**Output:**

TypeError: Arguments cannot be less than one

**Files:**

r- read only format

w- write only format

rstrip() – used to delete the extra line in the end of file.

## operating with files

# reading the file and printing

reader = open('files/sample.txt','r')

#read and print lines

for line in reader:

    print(line.rstrip())

reader.close()

**Output:**

writing the file

reader = open('files/sample.txt','r')

# write files

reader = open('files/sample.txt','r')

writer = open('files/file.txt', 'w')

for line in reader:

    print(line)

    writer.writelines(line)

# close files

reader.close()

writer.close()

Assignment day 5:

**Face Detection:**

First, we need to install the library

1. pillow - Image library (pip install pillow)

2. opencv - Image and the video processing Library (pip install opencv-python)

3. imutils - Image Library (pip install imutils)

4. numpy - image dimensions (pip install numpy)

5. PIL - Image processing (pip install pillow)

Function Module:

It imports from the rec.py

It is sen.py

import rec

#with filename.function name

rec.hel()

rec.py

#hello recieving

def hel():

    print("hello")

Output:

Hello

It also takes the input

Sen.py

name=input()

rec.cn(name)

Rec.py

#calling name

def cn(name):

    print("name call "+name+ " is red")

Input: sa

Output:

Name call sa is red

We can import from a name

import rec as r

r.cn("flash")

here rec is considered as r

Import from a specified function name of the second file

Sen.py

from rec import hel,cn

hel()

cn("sarath")

#the shortcut of the single folder

from rec import hel as h

h()

rec.py

#hello recieving

def hel():

    print("hello")

#calling name

def cn(name):

    print("name call "+name+ " is red")

Output:

hello

name call sarath is red

hello

To import all the functions we use \*

from rec import \*

hel()

cn("j")

Output:

hello

name call j is red

Calling the Subfolder

Unlike the calling the same folder, we use dot to go to depth folder

sen.py

import name.sub

name.sub.s()

name folder -> sub.py

sub.py

def s():

    print("calling sub problems")

Output:

Calling sub problems

\_\_name\_\_:

It tells the main function or not

import rec

print(rec.\_\_name\_\_)

output:

rec

because it invoking

print(\_\_name\_\_)

output:

\_\_main\_\_

Assignment day 6:

Funmath.py

import name.math as ms

ms.add(34,54,67,985,65)

print(ms.pow(2,2))

math.py \*in subfolder

def add(\*args):

    k=0

    for i in args:

        k+=i

    print(k)

def pow(a,b):

    return a\*\*b

Output:

1205

4

Database:

* If we want to save some data in System we use Data Base.
* If we use .txt it will be slow because it uses line by line.
* SQL means Structure Query Language
* Ex DB – mongoDB, mySQL, Oragle, sqlite
* Download sqlite 3 suitable version
* To use GUI rather than commands then use DB browser for SQLite

In sqlite3

Creating a Database –

>>sqlite3 test.db

Creating basic DB

>>.tables

It shows present tables or create

>>.exit

Exits the SQL

Frequently SQL Commands :

* Creating a table

CREATE TABLE student(id INT primary,name VARCHAR(20),age INT);

Here id is Attribute

INT is Data type

Primary it does not create duplicates

* Inserting the values

INSERT INTO student(id,name,age)VALUES(1,”sarath”,20);

* Select the data

SELECT name FROM student;

It will show all names in the database

SELECT \* FROM student;

It selects all the records

SELECT name FROM student WHERE id=1;

To select a particular Attribute

* Update

Changing the old values

UPDATE student SET age=40 WHERE id=2;

* Delete

DELETE FROM student WHERE id=2;

Python in DB:

import db

# create conncetion object

conn = db.connect('test.db')

db.version(conn)

# insert statement

query="INSERT INTO student (name,rollnumber,age) VALUES ('Suresh',1004,29)"

db.insert(conn,query)

# select multiple records

query = "SELECT \* from student"

result = db.fetchall(conn,query)

for r in result:

    print(r)

# select one record

query = "SELECT \* from student where id = 1"

result = db.fetchone(conn,query)

print(result)

# update a record

query = "UPDATE student SET age=30 WHERE id=2"

db.update(conn,query)

# delete a record

query = "DELETE from student WHERE id = 3"

db.delete(conn,query)

# close connection

db.close(conn)

OR

import sqlite3 as s #importing

con =s.connect("prac.db") #connecting

cur =con.cursor() #creating cursor

cur.execute("SELECT name FROM student") #details

r= cur.fetchall() #fetching all the to r

print(r)

TRY- EXCEPT:

import sqlite3 as s

try:

    con = s.connect("prac.db")

    cur =con.cursor()

    cur.execute("SELECT fd FROM student")

    r=cur.fetchall()

    print(r)

except s.Error as e:

    print("error is ",e)

More try-Catch:

import sqlite3 as lite

# connect to the database

def connect(database\_name):

    try:

        connection = lite.connect(database\_name)

        return connection

    except lite.Error as error:

        print("Error while connecting to sqlite", error)

# print sqlite version

def version(connection):

    try:

        cursor = connection.cursor()

        query = "select sqlite\_version();"

        cursor.execute(query)

        record = cursor.fetchall()

        print("SQLite Database Version is: ", record)

        cursor.close()

    except lite.Error as error:

        print("Failed to print database version", error)

# insert one record into table

def insert(connection,query):

    try:

        cursor = connection.cursor()

        cursor.execute(query)

        connection.commit()

        print("Successfully inserted one record")

        cursor.close()

    except lite.Error as error:

        print("Failed to insert data into  table", error)

# fetch all records from table

def fetchall(connection,query):

    try:

        cursor = connection.cursor()

        cursor.execute(query)

        records = cursor.fetchall()

        cursor.close()

        return records

    except lite.Error as error:

        print("Failed to fetch records the data from table", error)

# fetch one record from table

def fetchone(connection,query):

    try:

        cursor = connection.cursor()

        cursor.execute(query)

        record = cursor.fetchone()

        cursor.close()

        return record

    except lite.Error as error:

        print("Failed to fetch one record from table", error)

# update one record in the table

def update(connection,query):

    try:

        cursor = connection.cursor()

        cursor.execute(query)

        connection.commit()

        print("Successfully updated one record")

    except lite.Error as error:

        print("Failed to update data into table", error)

# delete one record from table

def delete(connection,query):

    try:

        cursor = connection.cursor()

        cursor.execute(query)

        connection.commit()

        print("Successfully deleted one record")

    except lite.Error as error:

        print("Failed to delete one record ", error)

# close the connection

def close(connection):

    connection.close()

Classes and objects:

Classes are the blueprint or templates of objects

An object is the real-time Entities

class student:

    name="sarath"

    age=20

    def fullname(self): #self is like a pointer

        print("surya sarath")

s=student()

s.fullname()

print(s.age)

output:

surya sarath

20

Empty attribute changing:

class student:

    name=""

    age=""

    def setname(self,n):

        self.name=n

s=student()

s.setname("surya")

print(s.name)

output:

surya

multiple Empty attribute changing:

class student:

    name=""

    age=""

    def setname(self,n):

        self.name=n

s=student()

s.setname("surya")

print(s.name)

s.setname("sarath")

print(s.name)

output:

surya

sarath

Directly changing attribute

class student:

    name=""

    age=""

    def setname(self,n):

        self.name=n

s=student()

s.name="sarath"

print(s.name)

**Constructor method:**

class student:

    name=""

    age=""

    def \_\_init\_\_(self,n,a): #it must be double (\_\_) main

        self.name =n

        self.age =a

s=student("sarath",20)

print(s.name)

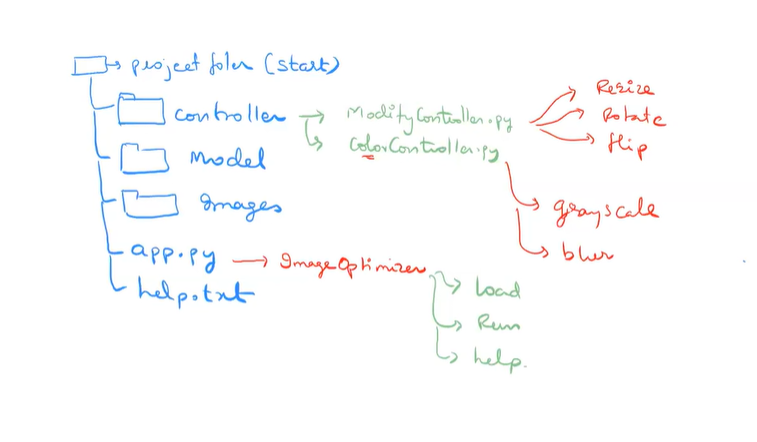
print(s.age)

**MVC view:**

M is model (data-based related files)

V is a view (Design based files)

C is Controller(logic Files)



Python provides various options for developing graphical user interfaces (GUIs). Most important are listed below.

* **Tkinter** − Tkinter is the Python interface to the Tk GUI toolkit shipped with Python. We would look this option in this chapter.
* **wxPython** − This is an open-source Python interface for wxWindows [http://wxpython.org](http://wxpython.org/).
* **JPython** − JPython is a Python port for Java which gives Python scripts seamless access to Java class libraries on the local machine [http://www.jython.org](http://www.jython.org/)

**Tkinter:**

The tkinter package (“Tk interface”) is the standard Python interface to the Tk GUI toolkit. Both Tk and tkinter are available on most Unix platforms, as well as on Windows systems. (Tk itself is not part of Python; it is maintained at ActiveState.)

## **Tkinter Modules**

Most of the time, [tkinter](https://docs.python.org/3/library/tkinter.html" \l "module-tkinter" \o "tkinter: Interface to Tcl/Tk for graphical user interfaces) is all you really need, but a number of additional modules are available as well. The Tk interface is located in a binary module named \_tkinter. This module contains the low-level interface to Tk, and should never be used directly by application programmers. It is usually a shared library (or DLL), but might in some cases be statically linked with the Python interpreter.

In addition to the Tk interface module, [tkinter](https://docs.python.org/3/library/tkinter.html" \l "module-tkinter" \o "tkinter: Interface to Tcl/Tk for graphical user interfaces) includes a number of Python modules, tkinter.constants being one of the most important. Importing [tkinter](https://docs.python.org/3/library/tkinter.html" \l "module-tkinter" \o "tkinter: Interface to Tcl/Tk for graphical user interfaces) will automatically import tkinter.constants, so, usually, to use Tkinter all you need is a simple import statement:

**import** **tkinter**

Or, more often:

**from** **tkinter** **import** \*

## **Example:**

#!/usr/bin/python

import Tkinter

top = Tkinter.Tk()

# Code to add widgets will go here...

top.mainloop()

This would create a following window −



1. **Button**:

To add a button in your application, this widget is used.  
The general syntax is:

w=Button(master, option=value)

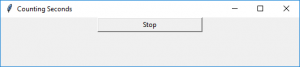
master is the parameter used to represent the parent window.  
There are number of options which are used to change the format of the Buttons. Number of options can be passed as parameters separated by commas. Some of them are listed below.

* + **activebackground**: to set the background color when button is under the cursor.
  + **activeforeground**: to set the foreground color when button is under the cursor.
  + **bg**: to set he normal background color.
  + **command**: to call a function.
  + **font**: to set the font on the button label.
  + **image**: to set the image on the button.
  + **width**: to set the width of the button.
  + **height**: to set the height of the button.

Program:

|  |
| --- |
| import tkinter as tk  r = tk.Tk()  r.title('Counting Seconds')  button = tk.Button(r, text='Stop', width=25, command=r.destroy)  button.pack()  r.mainloop() |

Output:



1. **Canvas:**

It is used to draw pictures and other complex layout like graphics, text and widgets.  
The general syntax is:

w = Canvas(master, option=value)

master is the parameter used to represent the parent window.

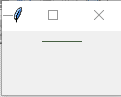
There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

* + **bd**: to set the border width in pixels.
  + **bg**: to set the normal background color.
  + **cursor**: to set the cursor used in the canvas.
  + **highlightcolor**: to set the color shown in the focus highlight.
  + **width**: to set the width of the widget.
  + **height**: to set the height of the widget.

Program:

|  |
| --- |
| from tkinter import \*  master = Tk()  w = Canvas(master, width=40, height=60)  w.pack()  canvas\_height=20  canvas\_width=200  y = int(canvas\_height / 2)  w.create\_line(0, y, canvas\_width, y )  mainloop() |

Output:



1. **CheckButton:**

To select any number of options by displaying a number of options to a user as toggle buttons. The general syntax is:

w = CheckButton(master, option=value)

There are number of options which are used to change the format of this widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

* + **Title**: To set the title of the widget.
  + **activebackground**: to set the background color when widget is under the cursor.
  + **activeforeground**: to set the foreground color when widget is under the cursor.
  + **bg**: to set he normal backgrouSteganography

Break

Secret Code:

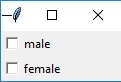
Attach a File:nd color.

* + **command**: to call a function.
  + **font**: to set the font on the button label.
  + **image**: to set the image on the widget.

**Program**:

|  |
| --- |
| from tkinter import \*  master = Tk()  var1 = IntVar()  Checkbutton(master, text='male', variable=var1).grid(row=0, sticky=W)  var2 = IntVar()  Checkbutton(master, text='female', variable=var2).grid(row=1, sticky=W)  mainloop() |

Output:



1. **Entry:**

It is used to input the single line text entry from the user.. For multi-line text input, Text widget is used.  
The general syntax is:

w=Entry(master, option=value)

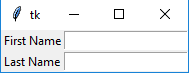
master is the parameter used to represent the parent window.  
There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

* + **bd**: to set the border width in pixels.
  + **bg**: to set the normal background color.
  + **cursor**: to set the cursor used.
  + **command**: to call a function.
  + **highlightcolor**: to set the color shown in the focus highlight.
  + **width**: to set the width of the button.
  + **height**: to set the height of the button.

Program:

|  |
| --- |
| from tkinter import \*  master = Tk()  Label(master, text='First Name').grid(row=0)  Label(master, text='Last Name').grid(row=1)  e1 = Entry(master)  e2 = Entry(master)  e1.grid(row=0, column=1)  e2.grid(row=1, column=1)  mainloop() |

Output:



1. **Frame:**

 It acts as a container to hold the widgets. It is used for grouping and organizing the widgets. The general syntax is:

w = Frame(master, option=value)

master is the parameter used to represent the parent window.

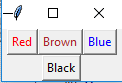
There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

* + **highlightcolor**: To set the color of the focus highlight when widget has to be focused.
  + **bd**: to set the border width in pixels.
  + **bg**: to set the normal background color.
  + **cursor**: to set the cursor used.
  + **width**: to set the width of the widget.
  + **height**: to set the height of the widget.

Program:

|  |
| --- |
| from tkinter import \*    root = Tk()  frame = Frame(root)  frame.pack()  bottomframe = Frame(root)  bottomframe.pack( side = BOTTOM )  redbutton = Button(frame, text = 'Red', fg ='red')  redbutton.pack( side = LEFT)  greenbutton = Button(frame, text = 'Brown', fg='brown')  greenbutton.pack( side = LEFT )  bluebutton = Button(frame, text ='Blue', fg ='blue')  bluebutton.pack( side = LEFT )  blackbutton = Button(bottomframe, text ='Black', fg ='black')  blackbutton.pack( side = BOTTOM)  root.mainloop() |

Output:



1. **Label**:

It refers to the display box where you can put any text or image which can be updated any time as per the code.  
The general syntax is:

w=Label(master, option=value)

master is the parameter used to represent the parent window.

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

* + **bg**: to set he normal background color.
  + **bg** to set he normal background color.
  + **command**: to call a function.
  + **font**: to set the font on the button label.
  + **image**: to set the image on the button.
  + **width**: to set the width of the button.
  + **height**” to set the height of the button.

Program:

|  |
| --- |
| from tkinter import \*  root = Tk()  w = Label(root, text='GeeksForGeeks.org!')  w.pack()  root.mainloop() |

Output:



1. **Listbox**:

It offers a list to the user from which the user can accept any number of options.  
The general syntax is:

w = Listbox(master, option=value)

master is the parameter used to represent the parent window.

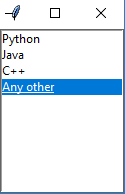
There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

* + **highlightcolor**: To set the color of the focus highlight when widget has to be focused.
  + **bg**: to set he normal background color.
  + **bd**: to set the border width in pixels.
  + **font**: to set the font on the button label.
  + **image**: to set the image on the widget.
  + **width**: to set the width of the widget.
  + **height**: to set the height of the widget.

Program:

|  |
| --- |
| from tkinter import \*    top = Tk()  Lb = Listbox(top)  Lb.insert(1, 'Python')  Lb.insert(2, 'Java')  Lb.insert(3, 'C++')  Lb.insert(4, 'Any other')  Lb.pack()  top.mainloop() |

Output:



1. **MenuButton**:

It is a part of top-down menu which stays on the window all the time. Every menubutton has its own functionality. The general syntax is:

w = MenuButton(master, option=value)

master is the parameter used to represent the parent window.

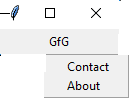
There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

* + **activebackground**: To set the background when mouse is over the widget.
  + **activeforeground**: To set the foreground when mouse is over the widget.
  + **bg**: to set he normal background color.
  + **bd**: to set the size of border around the indicator.
  + **cursor**: To appear the cursor when the mouse over the menubutton.
  + **image**: to set the image on the widget.
  + **width**: to set the width of the widget.
  + **height**: to set the height of the widget.
  + **highlightcolor**: To set the color of the focus highlight when widget has to be focused.

Program:

|  |
| --- |
| from tkinter import \*    top = Tk()  mb =  Menubutton ( top, text = &quot;GfG&quot;)  mb.grid()  mb.menu  =  Menu ( mb, tearoff = 0 )  mb[&quot;menu&quot;]  =  mb.menu  cVar  = IntVar()  aVar = IntVar()  mb.menu.add\_checkbutton ( label ='Contact', variable = cVar )  mb.menu.add\_checkbutton ( label = 'About', variable = aVar )  mb.pack()  top.mainloop() |

Output:



1. **Menu**:

It is used to create all kinds of menus used by the application.  
The general syntax is:

w = Menu(master, option=value)

master is the parameter used to represent the parent window.

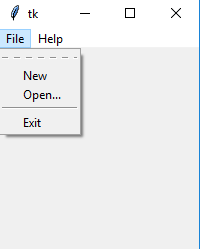
There are number of options which are used to change the format of this widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

* + **title**: To set the title of the widget.
  + **activebackground**: to set the background color when widget is under the cursor.
  + **activeforeground**: to set the foreground color when widget is under the cursor.
  + **bg**: to set he normal background color.
  + **command**: to call a function.
  + **font**: to set the font on the button label.
  + **image**: to set the image on the widget.

Program:

|  |
| --- |
| from tkinter import \*    root = Tk()  menu = Menu(root)  root.config(menu=menu)  filemenu = Menu(menu)  menu.add\_cascade(label='File', menu=filemenu)  filemenu.add\_command(label='New')  filemenu.add\_command(label='Open...')  filemenu.add\_separator()  filemenu.add\_command(label='Exit', command=root.quit)  helpmenu = Menu(menu)  menu.add\_cascade(label='Help', menu=helpmenu)  helpmenu.add\_command(label='About')  mainloop() |

Output:



1. **Message**:

It refers to the multi-line and non-editable text. It works same as that of Label.  
The general syntax is:

w = Message(master, option=value)

master is the parameter used to represent the parent window.

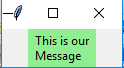
There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

* + **bd**: to set the border around the indicator.
  + **bg**: to set he normal background color.
  + **font**: to set the font on the button label.
  + **image**: to set the image on the widget.
  + **width**: to set the width of the widget.
  + **height**: to set the height of the widget.

Program:

|  |
| --- |
| from tkinter import \*  main = Tk()  ourMessage ='This is our Message'  messageVar = Message(main, text = ourMessage)  messageVar.config(bg='lightgreen')  messageVar.pack( )  main.mainloop( ) |

Output:



1. **RadioButton:**

It is used to offer multi-choice option to the user. It offers several options to the user and the user has to choose one option.  
The general syntax is:

w = RadioButton(master, option=value)

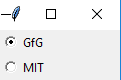
There are number of options which are used to change the format of this widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

* + **activebackground**: to set the background color when widget is under the cursor.
  + **activeforeground**: to set the foreground color when widget is under the cursor.
  + **bg**: to set he normal background color.
  + **command**: to call a function.
  + **font**: to set the font on the button label.
  + **image**: to set the image on the widget.
  + **width**: to set the width of the label in characters.
  + **height**: to set the height of the label in characters.

Program:

|  |
| --- |
| from tkinter import \*  root = Tk()  v = IntVar()  Radiobutton(root, text='GfG', variable=v, value=1).pack(anchor=W)  Radiobutton(root, text='MIT', variable=v, value=2).pack(anchor=W)  mainloop() |

Output:



1. **Scale:**It is used to provide a graphical slider that allows to select any value from that scale. The general syntax is:

w = Scale(master, option=value)

master is the parameter used to represent the parent window.

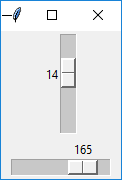
There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

* + **cursor**: To change the cursor pattern when the mouse is over the widget.
  + **activebackground**: To set the background of the widget when mouse is over the widget.
  + **bg**: to set he normal background color.
  + **orient**: Set it to HORIZONTAL or VERTICAL according to the requirement.
  + **from\_**: To set the value of one end of the scale range.
  + **to**: To set the value of the other end of the scale range.
  + **image**: to set the image on the widget.
  + **width**: to set the width of the widget.

Program:

|  |
| --- |
| from tkinter import \*  master = Tk()  w = Scale(master, from\_=0, to=42)  w.pack()  w = Scale(master, from\_=0, to=200, orient=HORIZONTAL)  w.pack()  mainloop() |

Output:



1. **Scrollbar**: It refers to the slide controller which will be used to implement listed widgets.  
   The general syntax is:

w = Scrollbar(master, option=value)

master is the parameter used to represent the parent window.

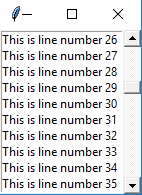
There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

* + **width**: to set the width of the widget.
  + **activebackground**: To set the background when mouse is over the widget.
  + **bg**: to set he normal background color.
  + **bd**: to set the size of border around the indicator.
  + **cursor**: To appear the cursor when the mouse over the menubutton.

Program:

|  |
| --- |
| from tkinter import \*  root = Tk()  scrollbar = Scrollbar(root)  scrollbar.pack( side = RIGHT, fill = Y )  mylist = Listbox(root, yscrollcommand = scrollbar.set )  for line in range(100):     mylist.insert(END, 'This is line number' + str(line))  mylist.pack( side = LEFT, fill = BOTH )  scrollbar.config( command = mylist.yview )  mainloop() |

Output:



1. **Text:**To edit a multi-line text and format the way it has to be displayed.  
   The general syntax is:

w =Text(master, option=value)

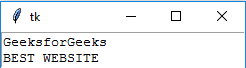
There are number of options which are used to change the format of the text. Number of options can be passed as parameters separated by commas. Some of them are listed below.

* + **highlightcolor**: To set the color of the focus highlight when widget has to be focused.
  + **insertbackground**: To set the background of the widget.
  + **bg**: to set he normal background color.
  + **font**: to set the font on the button label.
  + **image**: to set the image on the widget.
  + **width**: to set the width of the widget.
  + **height**: to set the height of the widget.

Program:

|  |
| --- |
| from tkinter import \*  root = Tk()  T = Text(root, height=2, width=30)  T.pack()  T.insert(END, 'GeeksforGeeks\nBEST WEBSITE\n')  mainloop() |

Output:



1. **TopLevel:**This widget is directly controlled by the window manager. It don’t need any parent window to work on.The general syntax is:

w = TopLevel(master, option=value)

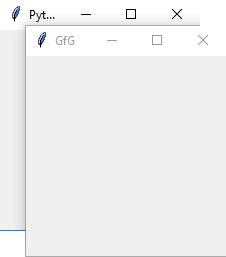
There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

* + **bg**: to set he normal background color.
  + **bd**: to set the size of border around the indicator.
  + **cursor**: To appear the cursor when the mouse over the menubutton.
  + **width**: to set the width of the widget.
  + **height**: to set the height of the widget.

Program:

|  |
| --- |
| from tkinter import \*  root = Tk()  root.title('GfG')  top = Toplevel()  top.title('Python')  top.mainloop() |

Output:



1. **SpinBox:**It is an entry of ‘Entry’ widget. Here, value can be input by selecting a fixed value of numbers.The general syntax is:

w = SpinBox(master, option=value)

There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

* + **bg**: to set he normal background color.
  + **bd**: to set the size of border around the indicator.
  + **cursor**: To appear the cursor when the mouse over the menubutton.
  + **command**: To call a function.
  + **width**: to set the width of the widget.
  + **activebackground**: To set the background when mouse is over the widget.
  + **disabledbackground**: To disable the background when mouse is over the widget.
  + **from\_**: To set the value of one end of the range.
  + **to**: To set the value of the other end of the range.

Program:

|  |
| --- |
| from tkinter import \*  master = Tk()  w = Spinbox(master, from\_ = 0, to = 10)  w.pack()  mainloop() |

Output:



1. **PannedWindow**It is a container widget which is used to handle number of panes arranged in it. The general syntax is:

w = PannedWindow(master, option=value)

master is the parameter used to represent the parent window.  
There are number of options which are used to change the format of the widget. Number of options can be passed as parameters separated by commas. Some of them are listed below.

* + **bg**: to set he normal background color.
  + **bd**: to set the size of border around the indicator.
  + **cursor**: To appear the cursor when the mouse over the menubutton.
  + **width**: to set the width of the widget.
  + **height**: to set the height of the widget.

Program:

|  |
| --- |
| from tkinter import \*  m1 = PanedWindow()  m1.pack(fill = BOTH, expand = 1)  left = Entry(m1, bd = 5)  m1.add(left)  m2 = PanedWindow(m1, orient = VERTICAL)  m1.add(m2)  top = Scale( m2, orient = HORIZONTAL)  m2.add(top)  mainloop() |

Output:



**COLLEGE**

"""

print("Bus")

bno,st,dest,time=123,"hyd","ibp",6

print("Busno",bno)

print("start",st)

print("destination",dest)

print("time",time)

print(type(bno) , type(time))

print(st[1])

"""

"""

sT = "d djdkjflsdfjkls"

print(sT)

print("3", sT[3])

#write 60 is end of string or more than of last line

print(sT [5:60])

 """

'''

sT = "d djdkjflsdfjkls"

print(sT[-3])

'''

"""

print(chr(97))

print(ord("a"))

sT = "d djdkjflsdfjkls"

print(len(sT))

"""

"""

#format

age=20

name="sarath"

print("{0} is a good boy age {1}" .format(name,age))

print(name+' is '+str(age)+' years old' )

print("{} is a good boy age {}" .format(name,age))

print("{n1} was {a}" .format(n1=name,a=age))

print(f"{name} is a good boy age {age}")

"""

'''

#format contolling float values

value=0.1564644664

print("{0:.2f}" .format(value))

print("%0.2f" %value)

'''

'''

#implicit

ivar=10

svar=12

svar=int(svar)

tvar=ivar+svar

print(tvar)

'''

"""

#square

x,y=2,3

z=x\*\*y

print(z)

"""

#repeat

'''

x,y="software",3

z=x\*y

print(z)

'''

'''

#add

x,y="software","lol"

z=x+y

print(z)

'''

'''

#bitwiseAND

x=2

y=4

z=x&y

print(z)

#bitwiseOR

z=x|y

print(z)

#bitwiseXOR

z=x^y

print(z)

#bitwiseCOMPLEMENT

z="x

print(z)

#bitwiseLEFTSHIFT by 2

z=x<<2

print(z)

#bitwiseRIGHTSHIFT by 2

z=x>>2.

print(z)

'''

'''

#binary

x=5

print(bin(x))

'''

"""

#AND

x=2

y=4

z=x is y

print(z)

#or

z=x is not y

print(z)

"""

'''

x="hello"

y={1:'a'}

print('a' in y)

print('h' in x)

print('h' not in x)

print(1 in y)

'''

#conditional

"""

n=0

if n<0:

    print("negitive")

elif n==0:

    print("zero")

else:

    print("positive")

"""

#for

'''

k=[1,2]

for val in k:

    print(val)

'''

'''

for k in range(10):

    print(k)

'''

'''

for i in range(2,10):

    print(i)

'''

'''

for i in range(2,10,5):

    print(i)

'''

'''

for i in [1,2,3,4,5]:

    if i==6:

        print(i)

        break

    else:

       print("sd")

else:

print("h")

'''

#vote

'''

k=int(input())

if k<18:

    print("NO vote")

else:

    print("Vote")

'''

"""

import time

k=['delhi','haryana','indore','goa','mumbai','hyderabad']

de=input("Enter the destination ")

print("source to destination")

for val in k:

    time.sleep(2)

    print(val)

    if de==val:

        break

time.sleep(5)

print("destination arrived" ,val)

"""

#create a final marks list of an integer

#ICEcream near

#list

'''

a=[3342,23,234343]

print(a[1])

a=["dfs3342","dfdf23",234343]

print(a[1])

print(a)

print(type(a))

'''

#ice

'''

icecream=["orange","mango","choco"]

price=["25","30","50"]

for fl in range(3):

    print(icecream[fl],price[fl])

print(max(price))

print(min(price))

pri=[34,23]

print(sum(pri))

print(sorted(price))

print(sorted(icecream))

'''

#sorted arraylist

'''

col= list("suryasarathbharadwaj")

print(type(col))

print(sorted(col))

'''

#fruits = ['apple', 'banana', 'cherry']

#inserted

'''

fruits.insert(1, "orange")

print(fruits)

'''

#lastinserted

'''

fruits.append("sb")

print(fruits)

'''

#removeat2 only cherry

'''

fruits.pop(2)

print(fruits)

'''

#removeSB

'''

fruits.remove("sb")

print(fruits)

'''

#whereisit

'''

x = fruits.index("cherry")

print(x)

'''

#rev

'''

fruits.reverse()

print(fruits)

'''

#howmanyit

'''

fr = [1, 4, 2, 9, 7, 8, 9, 3, 1]

x = fr.count(9)

print(x)

'''

#stack

'''

st=[23,54,34,56,54]

print(st)

#insert

st.append(12)

print(st)

#pop

st.pop()

print(st)

'''

#queue

'''

st=[23,54,34,56,54]

print(st)

#insert

st.insert(0,34)

print(st)

#pop

st.pop()

print(st)

'''

'''

line=["sai","manoj","asish","surya","sarath"]

print(line)

line.remove("asish")

print(line)

'''

#from collections import deque

#q = deque(['dsaasd','sdsd','sdg','ashish','asdsd','sai'])

#add

'''

q.append('a')

print(q)

'''

#remove

'''

print(q.popleft())

print(q)

'''

#asish remove

'''

p=q.index("ashish")

del q[p]

print(q)

'''

#collections

#removes duplicate

'''

A={34,5444,34,23,34}

print(A)

'''

#dicto

'''

d  = {1: 'value','age':43}

print(d)

print(d['age'])

d['age']=45

print(d['age'])

'''

'''

k = {

    1:"kalyan",

    2:"sarat"

}

print(k[2])

'''

'''

dic={

    "male":{

        "name":"sarath",

        "age":"20"

    },

    "female":{

        "name":"rahulam",

        "age":"45"

    }

}

print(dic["male"])

print(dic["male"]["name"])

'''

#male to female print

'''

task={

    1:{

        "name":"sarath",

        "sex":"male",

        "age":"20"

        },

    2:{

        "name":"rahul",

        "sex":"male",

        "age":"1000"

        },

    3:{

        "name":"kajal",

        "sex":"female",

        "age":"20"

        },

    4:{

        "name":"samantha",

        "sex":"female",

        "age":"30"

        },

}

inp=input("gender ")

'''

'''

if inp=="male":

    for i in task:

        if task[i]["sex"]=="female":

            print(task[i]["name"])

if inp=="female":

    for i in task:

        if task[i]["sex"]=="male":

            print(task[i]["name"])

'''

'''

for i in task.keys():

    if task[i]["sex"]!=inp:

        print(task[i]["name"])

'''

#function

"""

def gt():

    print("fggg")

gt()

"""

#pass

"""

def gt(ID):

    print("fggg",ID)

gt("df")

"""

'''

def gt(ID="dfs"):

    print("fggg",ID)

gt()

'''

#point func

"""

def gt(\*ID):

    print("hello ",ID)

    for i in ID:

        print("hell ",i)

gt("df","er")

gte = ["df","gh"]

gt(gte)

"""

"""

def gt(ID):

    print("hello ",ID)

    for i in ID:

        print("hell ",i)

gt("df")

gte = ["df","gh"]

gt(gte)

"""

#lambada

'''

va = lambda a,b:(a\*b)+2

print(va(2,4))

'''

#rocket launch

'''

import time

de=input("Do you want to launch the rocket (y/n)? ")

if de=="n":

    print("The Rocket launch is Stoped")

else:

    print("Count down begins")

    print(" ")

    print("Count:5")

    time.sleep(1.5)

    print("Count:4")

    time.sleep(1.5)

    print("Count:3")

    time.sleep(1.5)

    print("Count:2")

    time.sleep(1.5)

    print("Count:1")

    print(" ")

    print("SUCCESSFULLY LAUNCHED")

'''

#calculator

#shuffle

'''

import random as rd

a = ["kalyan","rahul","vamshi","vikranth"]

rd.shuffle(a)

print(a)

print("female")

print(a[0])

'''

"""

import random as rd

print(rd.random())

print(rd.randint(1,20))

print(rd.randrange(1,5,1))

"""

#rd vs

"""

import random as rd

a=["t1","t2","t3","t4","t5","t6","t7","t8"]

rd.shuffle(a)

for i in range(4):

    print(a[i]+ "vs"+a[i+4])

"""

#comment print

'''

def my():

    """print(a)"""

    print("fg")

my()

print(my.\_\_doc\_\_)

'''

#print in f

#print(f"{'ddfsdf'}")

#function update

'''

x=10

def my():

    x=29

    print(x)

my()

print(x)

'''

'''

def my():

    global x

    x=29

    print(x)

my()

print(x)

'''

#new line

#print("dsfsfdsdf\nasdasd")

#error Exception

'''

import sys

ra =['a',0,2]

for entry in ra:

    try:

        print("The entry is ",entry)

        r=1/int(entry)

        break

    except:

        print("oops ",sys.exc\_info()[0]," occured")

        print("Next Entry")

print("the recipocal of",entry,"is",r)

'''

#errorhandling only first error next terminate

'''

a,b=1,0

try:

    print(a/b)

    print("This wont be printed")

    print('10'+10)

except TypeError:

    print("you added values of incompatible types")

except ZeroDivisionError:

    print("you divided by 0")

'''

'''

try:

    print('10'+10)

    print(a/b)

except TypeError:

    print("you added values of incompatible types")

except ZeroDivisionError:

    print("you divided by 0")

'''

#normal except

'''

try:

    print('10'+10)

    print(a/b)

#except TypeError:

 #   print("you added values of incompatible types")

except ZeroDivisionError:

    print("you divided by 0")

except:

    print("caught error yo!")

'''

#raise

'''

try:

    print('1'+1)

except:

    raise TypeError("raised")

'''

#assert true

#assert(1==1)

#import

from paymentnotreceived import \*

rec = False

am = 500

try:

    if rec==False:

        raise paymentnotreceivedException

except:

    print("not rec")

finally:

    print("will this print?")

**Assignment: Object-Tracking:**

**Folder View:**

5.1. App.py

**5.2. Models –**

5.2.1 Authmodel.py

**5.3. Controllers –**

5.3.1 AuthController.py

**5.4.Views –**

5.4.1 Authview.py

5.4.2 Details\_view.py

5.4.3 home.py

**5.5. Lib –**

5.5.1 Colordetect.py

5.5.2 db.py

Read without input

''' Read input from STDIN. Print your output to STDOUT '''

#Use input() to read input from STDIN and use print to write your output to STDOUT

import sys

def main():

a =sys.stdin.readline()

b = sys.stdin.readline()

s=int(b) - int(a)

s = str(s)

sys.stdout.write(s)

# Write code here

main()

**For in negative :**

for i in range(10, -11, -1):

print(i)

**O/P**:

10

9

8

7

6

5

.

.

-10

**Space for in negative number :**

for i in range(10, -11, -1):

print(i, end = ' ')

**O/P:**

10 9 8 7 6 5 4 3 2 1 0 -1 -2 -3 -4 -5 -6 -7 -8 -9 -10

**Space in List:**

num = [1, 2, 3, 4, 5]

print(\*num)

print(num)

**O/P:**

1 2 3 4 5

[1, 2, 3, 4, 5]