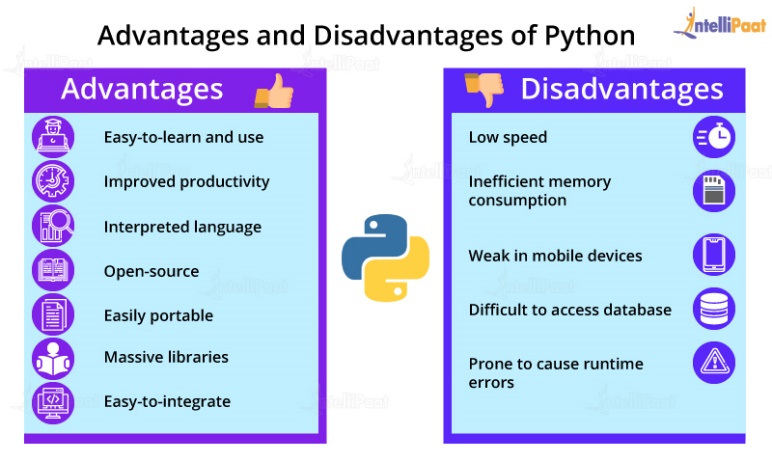
# BRIEF ABOUT PYTHON and benefits of python

Python is a high level language , interpreted , open source ,dynamic-typed ,platform independent , procedural, opps support language.

Current version of 3.11.3

It was created by Guido van Rossum, and released in 1991.

It is used for:

* web development (server-side),
* software development,
* mathematics,
* to connect database system.
* handle big data and perform complex mathematics

# application of python?

* Web Development
* Machine Learning and Artificial Intelligence
* Data Science
* Game Development
* Audio and Visual Applications
* Software Development
* Business Applications
* Desktop GUI
* Web Scraping Application

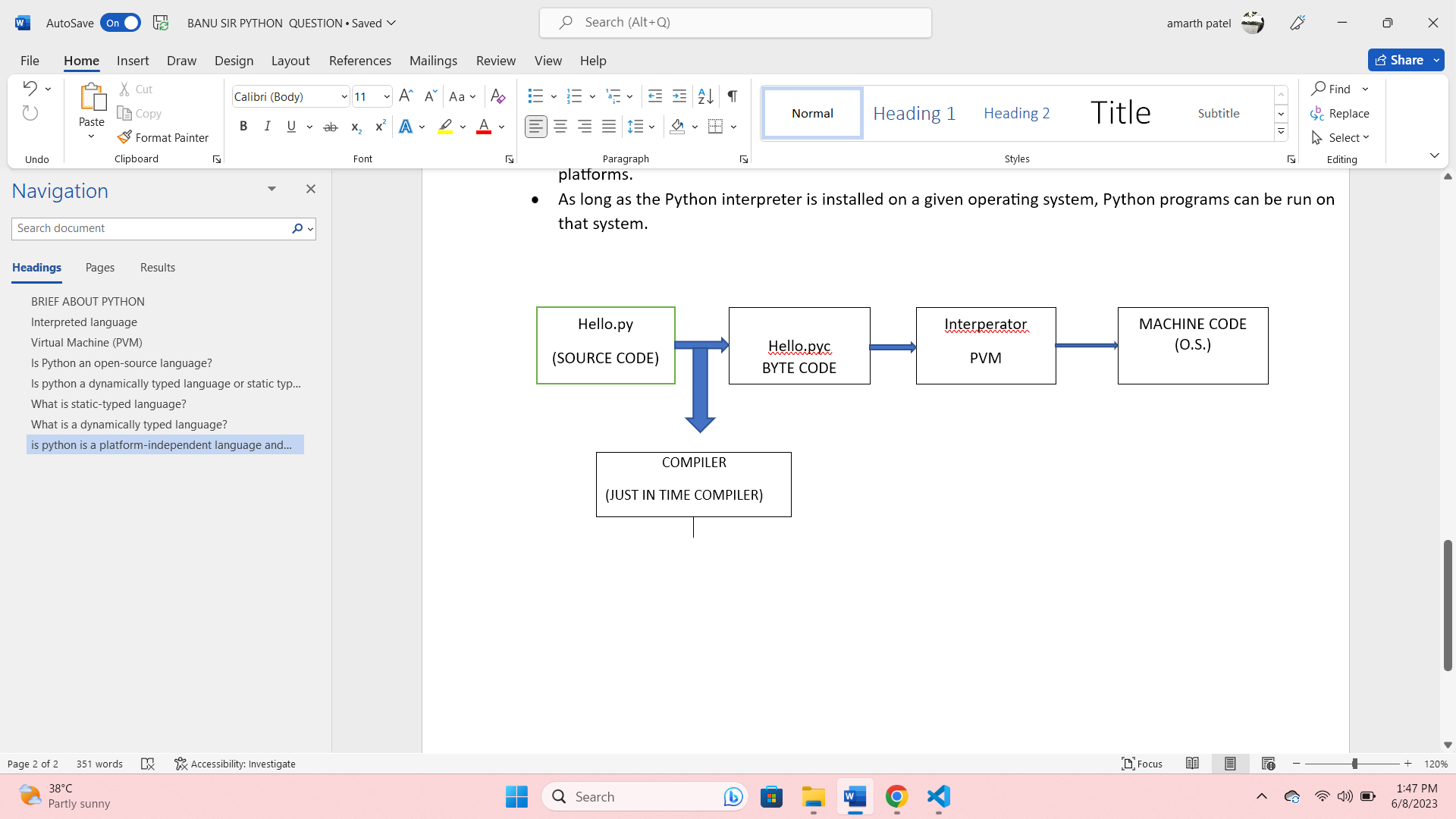
# Interpreted language

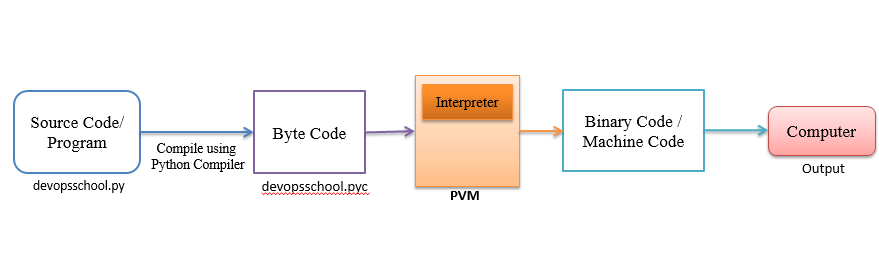
An interpreted language is a language in which the implementations execute instructions directly without earlier compiling a program into machine language.

Interpreted = execution of code line by line .

Python is an interpreted language, which means the source code of a Python program is converted into bytecode that is then executed by the Python virtual machine.

# Virtual Machine (PVM)

Python Virtual Machine (PVM) is to convert the byte code instructions into machine code so that the computer can execute those machine code instructions and display the final output.



# Is Python an open-source language?

Yes, it is an open source language, It is a language that is free to use and modify.

# Is python a dynamically typed language or static typed language?

Python is dynamically typed language

# What is static-typed language( strong type )?

* A statically-typed language is a language (such as Java, C, or C++) where variable types are known at compile time.
* This means that before source code is compiled, the type associated with each and every single variable must be known

# What is a dynamically typed language (weak type)?

* Yes, in dynamically typed languages, type checking takes place at [runtime](https://www.baeldung.com/cs/runtime-vs-compile-time#runtime) or execution time.
* This means that variables are checked against types only when the program is executing.
* Some examples of programming languages that belong to this category are Python, JavaScript, Lisp, PHP, Ruby, Perl, Lua, and Tcl.

# is python is a platform-independent language and what is ?

* Yes, Python is considered a platform-independent language.
* It can run on various operating systems, including Windows, macOS, Linux, and others, without requiring any modifications to the code.
* This is because Python code is executed by the Python interpreter, which is available for different platforms.
* As long as the Python interpreter is installed on a given operating system, Python programs can be run on that system.

# What is server-side lang and client-side lang?

|  |  |
| --- | --- |
| Client-side language | Server-side language |
| It is used to develop the front-end or user side of product  which can be seen and interacted with by users  example : html ,css ,js | Is used to develop the core functionality of the  Product and is usually hidden from the end-users  Example – python ,ruby , c++ |

# What is PIP?

PIP is a package manager for Python packages, or modules

Pip command is used to install the packages and modules

Ex. Pip install flask

# What is IDE ? Suggest five IDE for Python?

An integrated development environment (IDE) is a software application that helps programmers develop software code efficiently.

It increases developer productivity by combining capabilities such as software editing, building, testing, and packaging in an easy-to-use application.

Some IDEs

* **Online Compiler from Programiz** ·
* IDLE ·
* Sublime Text 3 ·
* Atom ·
* Thonny ·
* PyCharm ·
* Visual Studio Code ·

# Membership operators

* Membership operators are used to test if a sequence is presented in an object:

|  |  |  |  |
| --- | --- | --- | --- |
| **Operator** | **Description** | **Example** | **Try it** |
| in | Returns True if a sequence with the specified value is present in the object | x in y | [Try it »](https://www.w3schools.com/python/trypython.asp?filename=demo_oper_membership1) |
| not in | Returns True if a sequence with the specified value is not present in the object | x not in y |  |

# What is Garbage Collection(GC)?

Garbage collection is a memory management technique used in programming languages to automatically reclaim memory that is no longer accessible or in use by the application.

It helps prevent memory leaks, optimize memory usage, and ensure efficient memory allocation for the program.

# What is type conversion in Python (type casting)?

There are two types of type conversion in Python.

Implicit Conversion - automatic type conversion

Explicit Conversion - manual type conversion

Python Implicit Type Conversion

In certain situations, Python automatically converts one data type to another. This is known as implicit type conversion.

Example 1: Converting integer to float

Let's see an example where Python promotes the conversion of the lower data type (integer) to the higher data type (float) to avoid data loss.

integer\_number = 123

float\_number = 1.23

new\_number = integer\_number + float\_number

# display new value and resulting data type

print("Value:",new\_number) output : value: 124.23

print("Data Type:",type(new\_number))

**Output**

Value: 124.23

Data Type: <class 'float'>

In the above example, we have created two variables: integer\_number and float\_number of int and float type respectively.

Then we added these two variables and stored the result in new\_number.

As we can see new\_number has value **124.23** and is of the float data type.

It is because Python always converts smaller data types to larger data types to avoid the loss of data.

Explicit Type Conversion

In Explicit Type Conversion, users convert the data type of an object to required data type.

We use the built-in functions like int(), float(), str(), etc to perform explicit type conversion.

This type of conversion is also called typecasting because the user casts (changes) the data type of the objects.

a= ’2;

b=12

c=int(a)

# Is python case sensitive?

Yes, Python is a case−sensitive programming language. This means that it considers uppercase and lowercase letters differently.

# Is Python purely interpreted language?

Python is both compiled as well as an interpreted language, which means when we run a python code, it is first compiled and then interpreted line by line.

# what is inplace function in python. give example

# Python is truly a interpreted language. Explain

# What is the difference between lists and tuples?

# What is Dictionary?

# What are the Key features of Python?

# What are decorators in Python?

# How does break, continue, and pass work?

# what is inplace function in python.give example

# explain any five list function

# explain any five tuple function

# explain any five dictionary function

# What are the two types of functions in Python?

# What are the functions map lambda?

# What is a map function in Python?

# What is recursion?

# What is scope in python ?

Th scope of variable is a region of program where a variable is visible or accessible .

Lifetime of a variable is duration of which aa variable exist in the memory.

The existence and accessibility depend on the declaration of a variable in the program

Three type of scope in python , global , local

**Local Scope**

A variable created inside a function belongs to the local scope of that function, and can only be used inside that function.

def myfunc():  
  x = 300  
  print(x)  
  
myfunc()

As explained in the example above, the variable x is not available outside the function, but it is available for any function inside the function:

def myfunc():  
  x = 300  
  def myinnerfunc():  
    print(x)  
  myinnerfunc()  
  
myfunc()

**Global Scope**

A variable created in the main body of the Python code is a global variable and belongs to the global scope

**Naming Variables**

If you operate with the same variable name inside and outside of a function, Python will treat them as two separate variables, one available in the global scope (outside the function) and one available in the local scope (inside the function):

. x = 300  
def myfunc():  
  x = 200  
  print(x)  
myfunc() #300  
print(x) #300

**Global Keyword**

If you need to create a global variable, but are stuck in the local scope, you can use the global keyword.

The global keyword makes the variable global.

def myfunc():

global x

x=300

print(x)

myfunc() #300

print(x) #300

pyhton is a strong type language

strong type

only perform operation with same data type

a="1"+1  #error

# libaray 🡺 Package 🡺 modules?

Python modules may contain several classes, functions, variables, etc. whereas Python packages contain several modules. In simpler terms, Package in Python is a folder that contains various modules as files.

libaray --> packages ==> modules ==> function , class,object

# Python indentation

Python indentation is a way of telling a Python interpreter that the group of statements belongs to a particular block of code

# Exception handling

**note: try--> either with except or with finally**

try:

    print(1+'1')                                     #this is type error

except:

    print("error is occured")

finally:

    print("finally block is executed")

output:

error is occured

finally block is executed

try:

    for i in range(3.4):

        pass

except TypeError:

    print("type error occured")

output:

type error occured

def fun():

    try:

        print(1/0)

    except ZeroDivisionError:

        print("zero division error occured")

fun()

output:

zero division error occured

try:

    n1=int(input("enter first number :"))

    n2=int(input("enter second number :"))

except:

    print("error occur ")                           # here next code works

try:

    for i in range(10):

        print("hello")

except:

    pass

try:

    n1=int(input("enter first number :"))

    n2=int(input("enter second number :"))

    print("add =",n1+n2)

    # print("div =",n/n2)           # Name error

    print("div =",n1/n2)

except NameError:

    print("Name error occur ")

#Output -:

# enter first number :23

# enter second number :43

# add = 66

# Name error occur

except ZeroDivisionError:

    print("Zero Division Error occur ")

#Output -:

# enter first number :23

# enter second number :0

# add = 23

# Zero Division Error error occur

    # print("div =",n1/n2)

    # print("add =",n1/n2)

    # then add code not get executed because div error handle

try:

    n1=int(input("enter first number :"))

    n2=int(input("enter second number :"))

    l=[1,2]

    print(l[3])

    d={1:'mon'}

    print(d[5])

    print("add =",n1+n2)

    print("div =",n1/n2)

except NameError : print(NameError)

except ZeroDivisionError : print(ZeroDivisionError)

except ValueError : print(ValueError)

except TypeError : print(TypeError)

except SyntaxError : print(SyntaxError)

except AttributeError : print(AttributeError)

except KeyError : print(KeyError)

except FileNotFoundError : print(FileNotFoundError)

**attribute error**

try:

    a=90

    a.append(90000)

except AttributeError:

    print("ATTRIBUTE ERROR IS OCCURED")

OUTPUT:

ATTRIBUTE ERROR IS OCCURED

def F1():

    try:

        d={1:"mon" , 2:"tue" }

        a=int(input("enter key : "))

        try:

            print("value of key :" , d.get(a))

            result=d.get(a)

            print(result+1)

        except:

            print("exception occured in inner try")

    except:

        print("exception occured in outer try")

F1()

#OUTPUT:

# WHEN YOU ENTER 3 THE INNER Exception OCCURED

# enter key : 3

# value of key : None

# exception occured in inner try

# WHEN YOU ENTER W THE INNER Exception OCCURED

# enter key : W

# exception occured in outer try

**# else**

try:

    a=int(input("enter : "))

except:

    print("some error is occured")

else:

    print("no error is occured")         #when no exception occured else execute

**# output**

# enter : 41

# no error is occured

# enter : d

# some error is occured

**# finally**

#  finally chalna hi hai

try:

    a=int(input("enter : "))   #24

    b=int(input("enter : "))   #0

    print(a/b)

except ValueError:

    print("some error is occured")

finally:

    print("finally block")

    for i in range(2):

        print("welcome")

# output:

# enter : 24

# enter : 0

# finally block

# welcome

# welcome

# output:

# enter : 24

# enter : 24

# finally block

# welcome

# welcome

**# assert statement**

assert statement -- is used for  debugging purpose

a=90

assert a > 10

print("yess")      #yess

a=90

assert a > 100

print("yess")      #output:AssertionError

a=90

a > 10

print("yess")      #yess

2

# exception raise (user defined exception)

#\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_

# raise exception ki help se hum  user exception create karte hai.

#wap to create a exception if user entered the wrong value (other value) contain in a list else.

'''l=["delhi","mumbai","agra","bhopal","banglore","hydrabad"]

print(l)

ch=input("enter your city:")

if ch.lower() in l:

    print("ok")

else:

    raise Exception("invalid input")'''

num=int(input("enter any number:"))

if num>0:

    print("fine")

else:

    raise Exception("wrong no")

#wap to create a list with only even number(n, limit)

# 1) if user enter odd number u have to raise a exception

'''n=int(input("how many element do you want to insert in your list:"))

l=[]

for i in range(n):

    ele=int(input("enter your list's elements:"))

    if ele%2==0:

        l.append(ele)

    else:

        raise Exception("number must be even")

print(l)'''

# 2) handle exception

n=int(input("how many element do you want to insert in your list:"))

l=[]

while n>0:

    try:

        ele=int(input("enter your list's elements:"))

        if ele%2==0:

            l.append(ele)

            n=n-1

        else:

            raise Exception("")

    except:

        print("u have to enter a even number")

print(l)

**Module**

module: it is normal .py  file ,  here we defined some functions or classes to take access

of this function and classes in different .py file .

**how to make a package in python**

**package** - collection of related module .

1. Make a package name folder
2. In package folder make a folder name = \_\_pycache\_\_
3. In package folder make a file name = \_\_init\_\_.py
4. In package folder make a module name = abc.py

**\_\_pycache\_\_** =

       COMPILATION                INTERPRETER

S.C ----------------- BYTE CODE --------------- OC

FIRST.PY            \_\_PYCACHE\_\_

we have to made \_\_init\_\_.py file in package folder .

file name = modul.py

def add(a,b):

    print(a+b)

import modul

modul.add(1,2)

**wap to define a function which are going to create a list of n number in a module abc.py and also take access of that module in a separate xyz.py file**

this file name is abc.py this is a module

def create (lis,n):

    for i in range(n):

        num=int(input("enter a number :"))

        lis.append(num)

    print(lis)

this file name is xyz.py this is normal file

# import xxx

import xxx as a

# from xxx import create , ........ , ........ ,

l=[]

a.create(l,3)

output:

enter a number :1

enter a number :2

enter a number :3

[1, 2, 3]

# special variable =  \_\_name\_\_

\_\_name\_\_ = it is a special variable

1. \_\_name\_\_ = name of the .py file (when run by other module)

2.\_\_name\_\_ = "\_\_main\_\_" when run by same module

alias = as

|  |  |
| --- | --- |
| This file is aaa.py this is module  if \_\_name\_\_=="\_\_main\_\_":    for i in range(2):      print("hello")  print(\_\_name\_\_)  #output  hello  hello  \_\_main\_\_ | This file is xxx.py this is module  # import xxx  import xxx as a  print(\_\_name\_\_)  output:  xxx |

 ================================MODULE==========================================

module can be classified as

1.user defined Module

2.bulid-in module

bilud-in module.

random module.

datetime module.

os module.

3.third party package. =install though - pip install django , pip install pandas .

=================================================================================

pip is python package manager

=================================================================================

RANDOM MODULE

random() ---> 0,1

to generate number between 0 to 1 result in decimal form

import random as a

for i in range(3):

    print(a.random())

ouput:

0.46546478277124836

0.6288117016857782

0.618669873806503

generate number between 4 and 8

from random import uniform

print(uniform(4,8))  #ouput = 4.849273674513222

from random import \*

print(uniform(4,8))  #ouput = 6.282665355873903

for i in range(2):

    import random as a

    print(a.random())

ouput:

0.40778735574676706

0.8864799409807297

generate random number in interger use randint

from random import \*

print(randint(1,10),end=" ") #2 6 1 5 6

generate otp

from random import \*

print(randint(10000,99999),end=" ") #53967

from random import \*

# choice(lis)

l=["one","two","three","four"]

print(choice(l))

output:

give random number each time

choice()

from random import \*

choice(lis)

l=[1,2,3,4,6]

print(str(choice(l))+choice("hello"))

output:

give random number each time

#shuffle() = only work on  list

from random  import \*

l=["apple","banana","hello","by"]

shuffle(l)

print(l)  #['hello', 'by', 'banana', 'apple']

# file handling

f=open("file.txt","w")

f.close()

f = file object(we need a file object to work with file ,its acts as interface (buffer) between PM (PRIMARY MEMORY) and SM(SECONDARY MEMORY))

F= READ ,WRITE , APPEND,CLOSE (ALL OPERATION DONE WITH HELP OF THIS )

open(f\_name\_with\_full\_path , f\_mode)

**file mode**

**read** = 'r' (default mode) --> filee is not there,its throws error , position of pointer at beginning

**write**='w' -->file is not there ,its create a file , position of pointer at begginning , if previous written content is there , its got overwritten by new content.

**append**='a' --> file is not there ,its create a file , position of pointer at ending , if previous written content is there , new content is add after old one.

**read+write="r+" =**

**write+read="w+"**

**append+read="a+"**

**method related to read mode**

read(n) = n(number of bytes)

readline() return only one line

readlines() return list

**method related to write mode / append mode**

write()

writeline([iterable pss])

**file.txt**

today we learn file handling

a=open("file.txt","r")

r=a.read()

print(r) # today we learn file handling

a.close()

a=open("file.txt","r")

r=a.read(5)

print(r) #today

a.close()

a=open("file.txt","r")

r=a.readline() #today we learn file handling

print(r)

a.close()

a=open("file.txt","r")

r=a.read(5)

print(r) #today

rr=a.read(8)

print(rr) # we lear

a.close()

a=open("file.txt","w")

data="hello wolrd"

a.write(data)

print("data inserted")

a.close()

# old data overwrite , hello wolrd inserted in file.txt

a=open("file.txt","w")

data="hello wolrd\n"

data2="ok kok oko\n"

a.writelines([data,data2])

print("data inserted")

a.close()

**#wap to creata a text  in which user can add content until we wants  with infinite loop?**

f=open("aaa.txt","w")

while True:

    data=input("enter your content : ")

    f.write(data + "\n" )

    ch=input("do tou want to continue y/n")

    if ch=='n' or ch=='N':

        break

print("OK AD YOUR CONTENT BELOW")

# tell() and seek(n)

# tell()  --  current position of file pointer

# seek(n)  -- to move the pointer

# f=open("abc","w")

# f.close()

# with open ("abc.txt" , "w") as f :

#     f.write("my data")

# hello my name is amarth   ( abc.text file)

with open ("abc.txt" , "r") as f :

    print(f.tell())

    print(f.read(5))

    print(f.tell())

    print(f.read(6))

    print(f.tell())

    f.seek(0)

    print(f.read())

# output:

# 0

# hello

# 5

#  my na

# 11

# hello my name is amarth

# recursion:

whenever a function call itself

In every Recursion function ,we have to mention a exit point , with the help of this exit point we can easily come out of the function

intermediate result holds in a stack memory.

def add(a,b):

    a+b

    add(a,b)

factorial find out using Recursion

5!=5\*4\*3\*2\*1=120

1!=1

0!=1

-1(negative)=not possible

def factorial(a):

    if a==1 or a==0:

        return 1

    elif a<0:

        return "not possible"

    else:

        return a\*factorial(a-1)

print(factorial(5))     #120

print(factorial(0)) #1

print(factorial(-5))#not possible

print(factorial(1)) #1

a=4

4\*factorial(3) = 12

12factorial (2)=24

24factorial(1)=24

# print a table

def table(i,n):

    if i>10:

        return

    else:

        print(n,"\*",i,"=",n\*i)

        return table(i+1,n)

table(1,12)

output:

12 \* 1 = 12

12 \* 2 = 24

12 \* 3 = 36

12 \* 4 = 48

12 \* 5 = 60

12 \* 6 = 72

12 \* 7 = 84

12 \* 8 = 96

12 \* 9 = 108

12 \* 10 = 120

# wap to print febonazi series using  recursion

# Fn = Fn-1 + Fn-2, where n > 1.

def febonazi(n):

    if n<=1:

        return n

    else:

         return (febonazi(n-1)+ febonazi(n-2))

print(febonazi(6))

# Decorater

|  |
| --- |
| def deco(f):      def sub(a,b):          print(a-b)          f(a,b)      return sub  @deco  def add(a,b):      print(a+b)  add(1,2)  ot:  3  -1 |

it is a  wrapper function, used to enhance the functionality of already existing function.

decorator '@'

example

@wrapper

def wrapper(f):    #here wrapper --->decorater function(we pass a functioin)

    def sub(a,b):

       print("sub of {} and {} is {}".format(a,b,a-b))

       f(a,b)                    #f is reference of add(a,b) function

    return sub

@wrapper

def add(a,b):

    print("add of {} and {} is {}".format(a,b,a+b))

add(4,5)

# output:

# sub of 4 and 5 is -1

# add of 4 and 5 is 9

**# creation of list --->create(emptyy list()) , find highest and lowest**

def wrapper(f):    #here wrapper --->decorater function(we pass a functioin)

    def lowest(li):

      print("lowest of is {}".format(min(li)))

      f(li)                    #f is reference of add(a,b)

    return lowest

@wrapper

def greates(li):

    print("greates of is {}".format(max(li)))

li=[4,5,6]

greates(li)

pass function to decorator as argument

def wrapper(f):

    def sub(a,b):

        print(a-b)

        f(a,b)

    return sub

def add(a,b):

    print(a+b)

add=wrapper(add)

add(1,1)

op : 0 = 2

# map function

def square(n):

    return n\*n

l=[1,2,3,4]

print(list(map(square,l)))

def cube(n):

    return n\*n\*n

print(tuple(map(cube,l)))

# map function with lambda

print(list(map(lambda n:n\*n,l)))

a=["5" , " 4" ,"4"]

print(list(map(int,a))) #[5, 4, 4]

# filter function

def even(n):

    if n % 2 == 0:

        return n

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

print(list(filter(even,l)))

# print(list(filter(lambda n:n%2==0,l)))

# reduce(function,iterable)

from functools import reduce

def great(a,b):

    if a>b:

        return a

    return b

list=[1,5,5,4,10,-12]

print(reduce(great,list)) #10

from functools import reduce

a=[1,1,1,1]

def aa(a,b):

    return a+b

print(reduce(aa,a)) #4

#wap to create a dict  d which has 1 to 15 as a key and the respective square became the value

of that particular key

# d={1:1 , 2:4 ,3:9 , 4:16 ,............... 15:225}

d={}

for i in range(1,16):

    d[i]=i\*i

print(d)

# output:

# {1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100, 11: 121, 12: 144, 13: 169, 14: 196, 15: 225}

# setdefault(key,value)  function

#return -----> value

#new key-value--> updation

#old key value--> no updation

# d={1:"mon" , 2 : " tue"}

# a=d.setdefault(3,"wed")

# print(d)  #{1: 'mon', 2: ' tue', 3: 'wed'}

# print(a)  #wed

d={1:"mon" , 2 : " tue"}

a=d.setdefault(4)

print(d)  #{1: 'mon', 2: ' tue', 4: None}

print(a)  #None

**#wap to create a dict using two list**

list1=["ajay","akash","rajesh" , "rahul"]

list2=["bhopal","vidisha" , " gwalior" , " rewa"]

d={}

for i in range(len(list1)):

      d[list1[i]]=list2[i]

print(d)

output:{'ajay': 'bhopal', 'akash': 'vidisha', 'rajesh': ' gwalior', 'rahul': ' rewa'}

# list1=["ajay","akash","rajesh" , "rahul"]

# list2=["bhopal","vidisha" , "gwalior" , "rewa"]

# d=dict(zip(list1,list2))

# print(d)

# output:{'ajay': 'bhopal', 'akash': 'vidisha', 'rajesh': ' gwalior', 'rahul': ' rewa'}

**# wap to find the frequency of list**

l = [24, 24, 5, 24, 3, 3, 9]

d = {}

for i in l:

    d[i] = l.count(i)

print(d)

output: {24: 3, 5: 1, 3: 2, 9: 1}