WRITTEN BY: AI ENGr. SAAD

What is an IDE:

Ide is integrated development environment.it provide facilities to programmers. an ide normally consist of source code editor(which run that specific code), built automation tools, and a debugger. Here we will use pycharm ide.

What is pip:

Pip is the package manager of the python.

What is interpreter:

An interpreter is a virtual machine or you can say a software which read the code and execute it.

What is compiler:

A compiler read a programming language and run it according to syntax and lines of code.

What is a module in python:

A module is a pre written code or extension which can import from inside or outside. We can say a module a library that contain set of functions which we want to include in our application.

🡪python is an interpreter based language.

Difference between interpreter and compiler:

An interpreter run the code line by line and if error occur at any line it stop there but a compiler scan the whole file and indicate the error at specific line if any.

Difference between ide and interpreter:

Integrated development environment provide a complete set of environment for a specific language. Pycharm is an ide. Interpreter is a computer software which execute the code while writing and if any error occurs, send it to user. Rather than compiler which compile the whole code first and then execute it.

Comments in python:

For single line comment use #

For multiline comments use three “”” at the start and end of the lines.

These are lines you simply want an extra information irrelevant to syntax.

Escape sequence characters:

Escape sequence characters are those which are used in string for special behavior in the string like /n give us the new line /t give us a tab after that word etc. it does not represent itself while using inside the string.

Type casting:

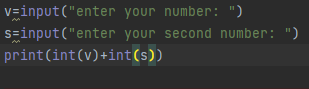
Type casting is changing the type of variable from one form to another.

Printing and taking input from user:

If you wanna print anything just write print(“anything”) in python and it will print it.

If you wanna take input from user just type input() and it will ask for a value in output section.

Add two numbers of your choice:



If you will not type int with the number it will be a string by default.

And if you will type type inside the print outside variable, it will tell what is type of that variable.

String slicing:

If you want the changes in the string than you can use the method of string slicing.

Let take some examples:

Let assume you have string in a variable name as v,

If you want to print string until a specific word or alphabet, just write print(v[0:4])

If you want to skip alphabets in an order, just write print(v[0:10:2]), it will skip one word after printing other one.

And if you want that the string print opposite then write print(v[::-1])

If a string start from zero in forward side than it start from -1 in opposite.

Sometimes you print anything in an ordered way the system gives the output in form of true or false. Example: if you tells system that your string ends with that name then it will gave output as true or false.

If you want to know how many alphabets are there in that string just write print(len(v)).

If you want to want to know that how many times any specific alphabet repeating just write print(v.count(“b”)). It will count that alphabet.

If you want to capitalize the first letter of string then just write print(v.capitalize())

If you wanna find a specific alphabet or word in string just write print(v.find(“that number”)). It will tell you the index at which it is.

If you wanna change a string in lower or upper case just write print(v.upper()).

If you wanna change any word with other just write print(v.replace(“is”,”are”)).

Lists:

Lists are the sequence of different integers or strings in a variable.

I.e. v=[9,4,3,5]

Or v=[“bricks”,”gold”,”love”]

Slicing and different functons in a list:

If you wanna print a list till any string or number just write print(v[0:4]).

If you wanna length print(len(v)).

If you wanna maximum number print(max(v)).

If minimum print(min(v)).

If you wanna add one more number in list just write

Suppose list is v=[4,3,5,6]

You wanna add 9

Just write

v.append(9)

print(v).

if you wanna insert number or string at any index

v=[4,3,5,6]

v.insert(index,number)

print(v)

for removing any number

v.remove(6)

print(v)

if you wanna remove the last number

v.pop()

print(v)

if you wanna replace a number with other

v[1]=45

print(v)

* A list is editable. It represented by square brackets.

TUPLE:

🡪 a tuple is not editable. Its represented by parathesis ().

V=(3,2,4,5) – a tuple.

Tuple is immutable.

🡪replacing two numbers in traditional way:

A=5

B=6

C=a

A=b

B=c

Print(a,b)

But here

A,b = b,a

Print(a,b)

If you wanna reverse the list

v.reverse()

print(v).

Dictionary:

A dictionary is a mutable python container which store the values with a unique key. Then you can access that value easily. It is surrounded by curly brackets. The key values pairs are surrounded by commas.

Lets see some examples:

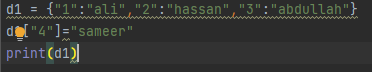


Here in print() function there is d1[“3”] which means it is accessing the Abdullah with key name 3.

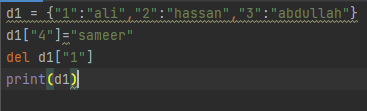
You can add a dictionary into a dictionary:



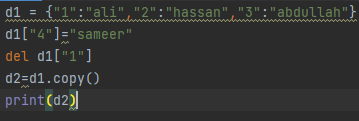
You can add anything later in your dictionary



You can delete any key



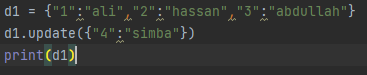
You can copy a dictionary into another variable



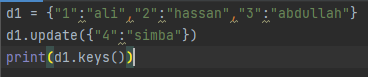
You can get any key value from the function:



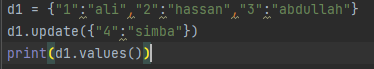
You can update anything in dictionary



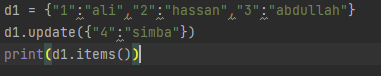
You can get only keys:



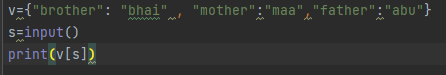
If you want the values



If you wanna print items of all key\_value pairs:



If you want that a user enter a value and if it is in the dictionary it will be printed

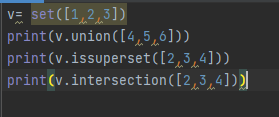


Set:

A set is collection of data which is unordered and immutable. Set stores unique values. Set can store lists, tuples or dictionaries.

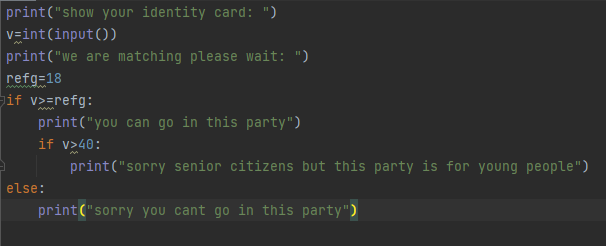


There are many funtions of set

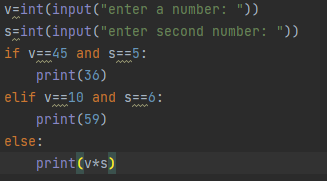


If else:

This statement in python used to match any condition with any reference. If that condition will true it will print that desire thing placed for output if not then it will go for else. You can use if statement inside an if statement too. You can use an elif for second if statement as it represent that if previous condition was not true then you can use this one.



Let have a special exercise

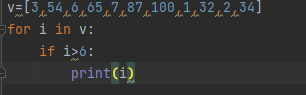


In and not in key words:

It execute result as true or false. That if what you are saying is true then it will print true otherwise false. Like a list v=[1,2,3,4] print(4 in list) it will print true. Or print(4 not in list) it will print false.

For loop in python:

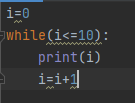
If you want to run a code in an iteration for a specific range or execute until desire condition will not meet then you use the for loop.



This code uses number of indexes as range and the number for if statement.

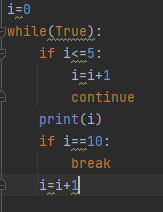
While loop:

While loop will give output until the condition will remain true. For loop print value for a range but while loop work on condition.

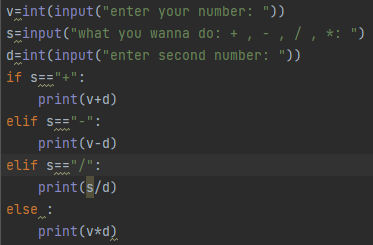


Continue and break statement:

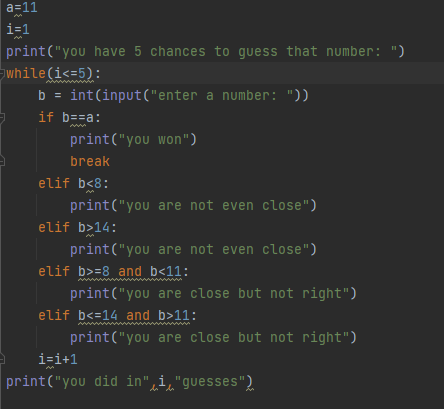
Continue and break statements are used to meet a condition. If that condition will meet than it will break the loop or continue the loop.



Task: simple calculator in python:



Task: Guess a number:



Operators in python:

Arithmetic operator:

These operators are simply +,-,/,\*

Assignment operator:

These operators are used if you want to assign a value to any variable. Its =.

Comparison operator:

This operator compare the two variables or values. These are <=, >=, ==, <,>,!=.

Logical operators:

These operators work on condition. These are and, or, not.

Identity operators:

These operators are is , is not.

Membership operators:

These are in , not in.

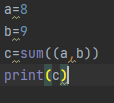
Bitwise operators:

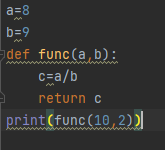
These perform bitwise operations.

These are in form of pairs of 0’s and 1’s.

Functions:

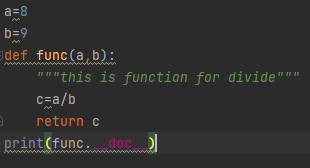
Functions are parts of code or you can say chunk of code which can use to perform an operation again and again instead of writing whole code. These are of two type which are built in functions and user defined functions.

 this is built in function.

this is a user defined function.

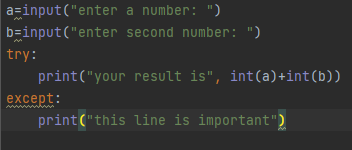
Docstring in function:

Docstring is the first string of comment which tell the specific information about that function.



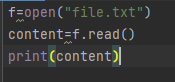
Try except:

If you know or guess that your program can gave error while compiling, and error line will be above any important line which you really want to print than you write the try except command. Means it takes an alternate action in case of error occur.



**File reading in python:**

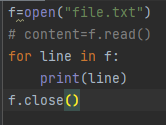
In python we can read a python file by first open it then read and then print.

  
if you wanna read first 5 or limited letters then just write the number read(5) in read mode.

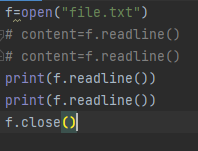
If you wanna read in binary mode just write open(“file.txt”,”rb”)

If you want to read and print it again, the all characters will be dumb in first time and second time it will ignore it.

If you wanna read the file line by line



Readline() function print only one line of the code. The more you will write it, the more lines of code will print.



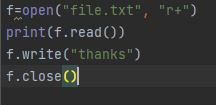
If you want to print in a form of list then just write readlines instead of readline and it will print in form of list.

Writing, appending and read and write both in a file:

If you want to write over a file over its existing data you use f.write(“anything you wanna write”) command. But you have to indicate mode in open() with “w” that it is write mode.

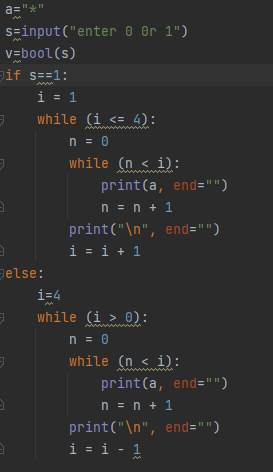
If you want that you wanna something additional but do not want to over write then use append mode indicate “a” in open. And write the same other code like write.

If you want to read and write both, then use “r+” mode in open().



Exercise:

Half-pyramid in python:

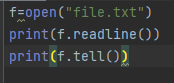


For printing mirror of half pyramid

a=5  
i=1  
while(i<=a):  
 j=1  
 while(j<=a-i):  
 print(" ",end=" ")  
 j=j+1  
 k=1  
 while(k<=i):  
 print("\*",end=" ")  
 k=k+1  
 print()  
 i=i+1

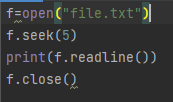
Tell():

If you want to know where is our file pointer we write this command. It tell us how many characters we have read.



Seek():

If you want to start reading your file from a specific location of characters. You can use this function.



Open file using with block:

By using with block you will not need to close the file after you opened it. with block will do it by itself.



Exercise:

Health management system:

def ali\_ex():  
 with open("ali exercise","r+") as f:  
 c=int(input("you want to read or write? \n press 1 for read and 2 for write: "))  
 if c==1:  
 print(f.read())  
 return  
 else:  
 f.write(input("what you want to write? "))  
 return  
  
def ali\_diet():  
 with open("ali diet","r+") as f:  
 c=inputint(("you want to read or write? \n press 1 for read and 2 for write: "))  
 if c==1:  
 print(f.read())  
 return  
 else:  
 f.write(input("what you want to write? "))  
 return  
  
def hamza\_ex():  
 with open("hamza exercise","r+") as f:  
 c=int(input("you want to read or write? \n press 1 for read and 2 for write: "))  
 if c==1:  
 print(f.read())  
 return  
 else:  
 f.write(input("what you want to write? "))  
 return  
  
def hamza\_diet():  
 with open("hamza diet","r+") as f:  
 c=int(input("you want to read or write? \n press 1 for read and 2 for write: "))  
 if c==1:  
 print(f.read())  
 return  
 else:  
 f.write(input("what you want to write? "))  
 return  
  
def asad\_ex():  
 with open("asad exercise","r+") as f:  
 c=int(input("you want to read or write? \n press 1 for read and 2 for write: "))  
 if c==1:  
 print(f.read())  
 return  
 else:  
 f.write(input("what you want to write? "))  
 return  
def asad\_diet():  
 with open("asad diet","r+") as f:  
 c=int(input("you want to read or write? \n press 1 for read and 2 for write: "))  
 if c==1:  
 print(f.read())  
 return  
 else:  
 f.write(input("what you want to write? "))  
 return  
i=1  
while(i<2):  
 a = int(input("whose file you want to open? \n 1-ali \n 2-hamza \n 3-asad \n"))  
 if a == 1:  
 b = int(input("what you want to open for ali? \n 1-diet \n 2-exercise"))  
 if b == 1:  
 print(ali\_diet())  
 else:  
 print(ali\_ex())  
 if a == 2:  
 d = int(input("what you want to open for hamza? \n 1-diet \n 2-exercise"))  
 if d == 1:  
 hamza\_diet()  
 else:  
 hamza\_ex()  
 if a == 3:  
 e = int(input("what you want to open for asad? \n 1-diet \n 2-exercise"))  
 if e == 1:  
 print(asad\_diet())  
 else:  
 print(asad\_ex())  
 i=i+1

Global variables and local variables:

While making a function, if we gave any variable value inside the function then it is local variable. But if you gave outside the function it is global variable.

a=3  
def func1():  
 c=3+3  
 print(c)  
func1()

in above code a is global variable and c is local variable.

If you will not declare a variable locally and call it inside the function, then that variable will print the value of global variable.

Global keyword:

You cant change a global variable value inside a function. But if you wanna change, then you use global keyword inside the function.

a=3  
def func1():  
 global a  
 a=4  
 c=3+3  
 print(c)  
func1()  
print(a)

now you can see that value of global variable is changed.

Recursion:

A recursion means that a function call itself again and again until it reach a desire output.

Iterative approach:

Achieve desire value using iterations.

def func(n):  
 a=1  
 for i in range(n):  
 a=a\*(i+1)  
 return a  
print(func(5))

Recursive approach:

Now calling function again in a function.

def func(n):  
 if n==1:  
 return 1  
 else:  
 return n\*func(n-1)  
print(func(5))

Here function calling itself again and again until it will meet the desire value.

Lambda function:

These are type of anonymous function which are not actually functions but perform like a function. They are one liner function.

#lambda function:  
c= lambda x,y:x-y  
print(c(4,3))

Sort function:

This is function write as sort() which print the values of list in ascending order.

Additional info 🡪 pycharm is an ide means integrated development environment. While visual studio code is an editor where you can write code and visual studio is an ide.

Modules in python:

Modules are the set of functions we want to use or include in our code. Modules are simply files with .py extension which include python code we import into another program. Modules contain sub modules or functions which we use in our code.

import random  
c=random.randint(0,3)  
print(c)

here in this module we are generating a random number using random module.

String formatting:

String formatting is technique of inserting a new variable or string in a predefined string.

# 1  
#percentage method  
b=3  
c="this is a %s"%b  
print(c)  
  
# 2  
#tuple method  
b=7  
c=3  
a="this is a %s %s"%(b,c)  
print(a)  
  
#4  
#dot method  
b=3  
c=6  
a="this is a {} {} "  
b=a.format(b,c)  
print(b)  
#3  
# f string method:  
s=2  
d=3  
a=f"this is a {s} {d}"  
print(a)

These all are string formatting method. F string(fast string) is covered in it.

Args and kwargs:

Args parameter will collect all the passed parameter into a tuple then it will processed by function. It is denoted by single asterisk \* with the variable.

Kwargs parameter will collect all the passed parameter into a dictionary and then it will be processed by the function. It is denoted by double asterisk \*\* with the variable.

def func(\*args,\*\*kwargs):  
 print(args)  
 for key,value in kwargs.items():  
 print(key,value)  
has=["saad","ali","hamza"]  
sd={"saad":"ML engineer","ali":"employee","saim":"employee"}  
func(\*has,\*\*sd)

time module:

time module contain the functions which do tasks related to time in the program.

Like if you want to find the execution time of a while loop, remove the time taken to that specific time to the initial time of for loop.

import time  
ini=time.time()  
# print(ini)  
k=0  
while(k<200):  
 print("hello")  
 k +=1  
print(time.time()-ini)

if you want to print local time.

import time  
gh=time.asctime(time.localtime(time.time()))  
print(gh)

* Want to open powershell at any where in windows, just press shift and right click.

requirements.txt:

if you want to know which packages with their versions are installed in the interpreter just write pip freeze > requirements.txt.

* If you want to include all packages that are installed in your system to your virtual environment then just write pip virtualenv –system-site-packages filename

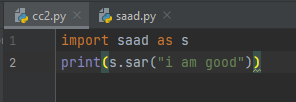
Enumerate function:

Enumerate function that iterate over a sequence and track the index of item. It gives the both index and the item number.

l=["banana","aloo","apple"]  
for a,b in enumerate(l):  
 print(a,b)

IMPORT:

Import brings the other modules and packages or functions into your current directory of python and you can you them there.



* If \_\_name\_\_==’\_\_main\_\_’:

This is the main function and used to separate the code that is only execute when that script only run but not to other script when it is being used as module.

def sp(ss):  
 print("hii",ss)  
def sr(se):  
 print("hii",se)  
if \_\_name\_\_=='\_\_main\_\_':  
 sr(43)

join function:

join function is used if you want to join the list of strings and adding a specific character or word in it without using the for loop.

l=["bro","brother","bhai"]  
a=" ".join(l)  
print(a)

map() function:

the map function is use to perform an action on an iterable like list, tuple or set instead of using for loop.

l=[1,2,3,4]  
def func(x):  
 return x\*x  
print(list(map(func,l)))

filter() function:

if you want to filter out specific element from the list or a condition, you can use this function. It return the print the true conditions.

l=[2,3,4,5,6,7,8,9]  
def func(n):  
 return n>5  
c=list(filter(func,l))  
print(c)

reduce() function:

reduce function is a function in functools module which is use to do specific tasks over iterables like list or tuple. It do task and merge them and return a single value.

l=[2,3,4,5,6,7,8,9]  
from functools import reduce  
print(reduce(lambda x,y:x+y,l))

“is” and “==” operator:

The is operator compare the exact value if they are immutable and on exact memory location. But the == operator compare the identical values.

a=5  
b=5  
print(a is b)  
print(a==b)

OOP:

Oop is object oriented programming. It is a programming model that revolves around objects. Objects are fundamental building blocks of oop. Oop was design to deal with real world entities and to organize and maintain code.

Oops further have different concepts. The concept of class, object and method are basic.

Class:

A class is a blueprint or template which is user defined that defines the structure and behavior of objects. It is use to organize the code and maintain it and reuse it. It a non-live entity. A class has **data**(state) and **method**(functions). Method work on that data. The state means the attributes of object like which color it is, which condition it is now etc. the method mean that in which state it is going. Ike change of state. It is the behavior of that object that how it behave.

e.g: fruit is a class. And apple is its object.

Object:

An object is an instance in the class which has its own attributes and behavior. It is a real-word or live entity. An object is create on the basis of blue print created by its class. Object is that actual instance that created on the basis of that particular structure and behavior defined by that class.

* Entity is something that exist and has identity.

Example:

class mobile():  
 name="s22 ultra" #attributes of class  
 model="23"  
 price="13k"  
samsung=mobile() #samsung is instance of class called object  
print(samsung.name,samsung.model,samsung.price)

* The attributes in the class are different than the attributes of object.
* We can access class variables through instance.
* Attributes are the characteristics or properties which lie within class or object. While parameters are the values or information which lie within the function.
* Parameters are variables that pass to functions while declaration while arguments are the actual values that pass to function while calling the function.

Method:

A method is the behavior of the object within class. It is the function that defines the actions that an object in class can perform. Each object can call the method.

Self:

Self is the conventional first parameter of method in class. Self refers the values of object to the method which is called on. It refers the current instance of the class. You can write it whatever you want instead of self.

class man:  
 # man\_colour="brown"  
 def manspecs(self): 🡪 this is method of class  
 return f"man height is {self.height}. weight is {self.weight}"  
ali=man()  
hamza=man()  
hamza.height=5  
hamza.weight=65  
print(hamza.manspecs())

constructor:

a constructor is a special function in the class which has attributes for object and takes the parameters while creating the object and set the values.

class man:  
 def \_\_init\_\_(self,a,b):  
 self.height=a  
 self.weight=b  
 def manspecs(self):  
 return f"man height is {self.height}. weight is {self.weight}"  
ali=man(4,5)  
print(ali.manspecs())

@classmethod:

A class method is a decorator whose first argument is class itself. It one of use it that it can access and change the class instance variable.

class man:  
 colour=5  
 def \_\_init\_\_(self,a,b):  
 self.height=a  
 self.weight=b  
 def manspecs(self):  
 return f"man height is {self.height}. weight is {self.weight}"  
 @classmethod  
 def rr(cls,new):  
 cls.colour=new  
ali=man(4,5)  
ali.rr(4)  
print(ali.colour)

class method as alternative constructor:

class man:  
 colour=5  
 def \_\_init\_\_(self,a,b):  
 self.height=a  
 self.weight=b  
 def manspecs(self):  
 return f"man height is {self.height}. weight is {self.weight}"  
 @classmethod  
 def rr(cls,new):  
 sp=new.split("-")  
 return cls(sp[0],sp[1])  
ali=man(4,5)  
ali=man.rr("44-55")  
print(ali.height)

Static method:

A static method in python is a method which belong to class only rather than instances of class. You can access it directly with class name. It do not have access to instance or its attributes.

Encapsulation and abstraction:

Encapsulation and abstraction are two different concepts in object oriented programing but similar to each other. Abstraction is not possible to achieve without encapsulation. Encapsulation means that storing the data and all of the code related to that data at one place which should be restricted from outer word. But abstraction means the hiding the all internal functionalities and show only essential information.

Encapsulation:

class bike():  
 def \_\_init\_\_(self,model,cc):  
 self.model=model  
 self.cc=cc  
 def start(self):  
 print("bike is starting")  
honda=bike(23,150)  
print(honda.model)  
honda.start()

abstraction:

class bike():  
 def \_\_init\_\_(self):  
 self.name="honda"  
 def specs(self):  
 self.name="honda"  
 self.cc="170"  
 self.color="black"  
 print(self.name,self.cc,self.color)  
 def owndel(self):  
 self.name="saad"  
 self.age="22"  
 self.work="ML engineer"  
 print(self.name,self.age,self.work)  
hi=bike()  
print(hi.name)  
hi.specs()  
hi.owndel()

* Hiding the details of implementation.

Inheritance:

Inheritance in python means that deriving a new class from the existing class. The existing class is called parent class and the derived class is called child class. By using this concept you can inherit the properties of existing class to new class.

* Reduce the programmers efforts.
* Increase the code reusability.

Single inheritance:

Single inheritance means that deriving one new class from the existing class.

class product:  
 prod1="cake"  
 prod2="roti"  
class property(product):  
 prop1="sweet"  
 prop2="healthy"  
gg=property()  
print(gg.prod1 + " is " + gg.prop1)

multiple inheritance:

if a class is derived or inherited from more than one existing class than it is multiple inheritance.

class name:  
 first="ali"  
 second="usman"  
 third="umer"  
class grade:  
 position1="first"  
 position2="second"  
 position3="third"  
class result(name,grade):  
 prize1="10000"  
 prize2="5000"  
 prize3 = "3000"  
res=result()  
print(res.first+"\n"+res.position1+"\n"+res.prize1)

multilevel inheritance:

multilevel inheritance means that deriving a new class from existing class which will be called as child class. But here we will derive a grandchild class from the child class.

class name:  
 first="ali"  
 second="usman"  
 third="umer"  
class grade(name):  
 position1="first"  
 position2="second"  
 position3="third"  
class result(grade):  
 prize1="10000"  
 prize2="5000"  
 prize3 = "3000"  
res=result()  
print(res.first+"\n"+res.position1+"\n"+res.prize1)

hierarchical inheritance:

if we derive more than one child class from the same parent class then it is called hierarchical inheritance.

class name:  
 first="ali"  
 second="usman"  
 third="umer"  
class grade(name):  
 position1="first"  
 position2="second"  
 position3="third"  
class result(name):  
 prize1="10000"  
 prize2="5000"  
 prize3 = "3000"  
res=result()  
print(res.first+"\n"+res.prize1)

hybrid inheritance:

hybrid inheritance contains more than one form of inheritance. There is no sequence that which class will be derived from whom. You can use it according to your requirements.

#Syntax\_Hybrid\_inheritance

class PC:

pass

class Laptop(PC):

pass

class Mouse(Laptop):

pass

class Student3(Mouse, Laptop):

pass

# Driver's code

obj = Student3(

Access specifiers:

Access specifier in python are used to limit the access of class variables and class method outside the class while implementing the concept of inheritance.

There are three types of access specifiers. They are public, protected and private.

Public access specifiers:

All the methods and variables in class are public by default. You can access variables and methods by any child class.

Protected access specifiers:

By using protected access specifiers, you can access the variables and methods inside the class by the class itself and child class but not any other class. It can achieve by putting an underscore after self. Keyword.

Privates access specifiers:

The variables and methods inside the class can only be achieve by class itself but by any child class or any other class. It can be achieved by putting two underscores after self keyword.

class var:  
 def \_\_init\_\_(self):  
 self.age="13"  
 self.\_name="ali"  
 self.\_\_height=15  
class var2(var):  
 pass  
obj=var2()  
print(obj.age)

super() function:

if you want to access the parent class variables, methods or constructor from child class having same name, we use super function.

class var:  
 x="hello"  
 # def \_\_init\_\_(self):  
 # print("parent class constructor")  
class var2(var):  
 x="good"  
 def \_\_init\_\_(self):  
 print(super().x)  
obj=var2()

polymorphism:

polymorphism is a Greek word which means many forms. Python support polymorphism through inheritance.

In python there are following types of polymorphism:

Operator overloading.

Method overriding:

Method overriding is a technique in which we can create a method in derived class that is already existed in superclass. The new method will override the old method.

class Abc:  
 def func(self):  
 print("Parent class")  
class xyz(Abc):  
 def func(self):  
 print("child class")  
obj = xyz()  
obj.func()

Magic methods or dunders:

Magic methods or dunders are methods which contains two underscores (\_\_) before and after the method name. they have many types. The \_\_init\_\_(self) constructor is also a dunder.

When we print our \_\_init\_\_() constructor, it prints the type and address but if we want to print the instance variables in it we use \_\_repr\_\_() for that purpose.

class Test:  
 def \_\_init\_\_(self, greet):  
 self.greet = greet  
 def \_\_repr\_\_(self):  
 return f"object is: {self.greet}"  
obj = Test("Hey")  
print(obj)

Smilarly there are \_\_add\_\_(), \_\_mul\_\_() dunders also for multiply and add.

Abstract class:

Abstract class in python is a class which contain one or more abstract method. These abstract methods have an empty body. If you want to run a method forcefully in any child class inherited from the abstract class, you can use this class. You have to import ABC module from abc library.

from abc import ABC, abstractmethod  
class caus(ABC):  
 @abstractmethod  
 def mul(self):  
 pass  
class RR(caus):  
 def \_\_init\_\_(self):  
 self.a=4  
 self.b=5  
 def mul(self):  
 return self.a \* self.b  
obj=RR()  
print(obj.mul())

Yield:

Yield is a keyword just like return. But return gives the last value but yield do not return the last value. Instead, it return a generator object. When there will be a yield keyword while the execution of program, the program will immediately stop.

Generator:

Generator is a special type of function which do not contain a value at the end but a iterateabe generator object. You can fetch the values from a generator by using list(), \_\_next\_\_(), for loop.

def func():  
 for i in range(1,5):  
 yield i  
z=func()  
print(z.\_\_next\_\_())  
  
for i in z:  
 print(i)

decorators:

decorators are the special type of functions use to enhance the functionality of a function. You can add extra functionality in the function without changing the original function code.

def dec\_fun(decr):  
 def ecr():  
 x=6  
 print(x)  
 decr()  
 return ecr  
@dec\_fun  
def function():  
 print("this is six")  
function()

There are two types of data structure: built-in(list, tuple, dictionaries) and user defined.

Built-in:

List:

List is the ordered collection of data just like arrays in other programming languages.

List=[1,2,3,4]

In python, the starting index of list is 0 while the ending is n-1. In reverse, the starting index of list is -1.

Multi-dimentional list is list which contains more than one list inside a list.

Tuple:

Tuple is similar to list but immutable in nature.

Tuple=(1,2,3,4)

You can pass a list in tuple.

Set:

Python set is a mutable collection of data which do not allow any duplication.

A={1,2,3,”saad”,4}

Frozen set:

Frozen sets are the sets which contain distinct values like other set but it do not allow any mutability. It is immutable. This is to ensure that the set values remain same while the execution of program.

A=({1,2,3,4})

String;

Python do not have any char data type. So a single character is actually a string with the length of 1. A string is immutable, modifying a string result in a new copy.

String=”this is a string”

Dictionaries:

Dictionary is the collection of unordered data which stores values in the format of key values.

A={“name”:”ali”}

Matrix:

In python, a matrix is a 2D array which consist of rows and columns. Each element in the matrix is identified by its column and row indices allowing you to manipulate and access these elements using these indices. Each element in matrix should be same size. You can represent a matrix with a nested lists. The outer list represent the rows and inner list represent the element within the rows.

matrix = [  
 [1, 2, 3],[4, 5, 6],[7, 8, 9]  
]  
print(matrix)

Byte and Bytearray:

Byte is immutable sequence of integers ranging from 0-255.

Bytearray is mutable sequence of integers ranging from 0-255.

a=[1,100,200,255]  
b=bytes(a)  
for i in b:  
 print(i)

if you print directly, it will print address of that element.

**Data structure and algorithm:**

There are two types of user defined data structure.

Linear:

Linear also have two types:

* Static = array
* Dynamic = linked list, queue, stack

Non-linear data structure:

* Graph
* Tree

Array:

Array is collection of items at contiguous memory location. The concept of array was to store the elements of different data types together. Arrays and list have same structure.

A=[1,2,3,4]

Linked list:

A linked list is a linear data structure (elements are stored in a sequential order) where elements are not stored at contiguous memory location. A linked list is the list of nodes connected to each other. Each node consist of a data and a pointer.

class node:  
 def \_\_init\_\_(self,data):  
 self.data=data  
 self.next=None  
class linkedlist:  
 def \_\_int\_\_(self):  
 self.head=None  
if \_\_name\_\_=='\_\_main\_\_':  
 my\_list=linkedlist()  
 my\_list.head=node(1)  
 second=node(2)  
 third=node(3)  
 my\_list.head.next=second  
 second.next=third

linked list traversal:

traversal means that moving through the list and printing the data stored in it.

class node:  
 def \_\_init\_\_(self,data):  
 self.data=data  
 self.next=None  
class linkedlist:  
 def \_\_int\_\_(self):  
 self.head=None  
  
 def printlink(self):  
 temp=self.head  
 while(temp):  
 print(temp.data)  
 temp=temp.next  
if \_\_name\_\_=='\_\_main\_\_':  
 my\_list=linkedlist()  
 my\_list.head=node(1)  
 second=node(2)  
 third=node(3)  
 my\_list.head.next=second  
 second.next=third  
my\_list.printlink()

Stack:

The stack is the linear data structure which stores elements in first in last out or last in first out manner. The new element add at one end and can be removed from that end only. The insertion and deletion can be done using append and pop for push pop.

my\_list=[]  
my\_list.append(2)  
my\_list.append(3)  
my\_list.append(5)  
print(my\_list)  
my\_list.pop()  
my\_list.pop()  
print(my\_list)

Queue:

Queue is a linear data structure that stores elements in first in first out or last in last out manner.

my\_list=[]  
my\_list.append(2)  
my\_list.append(3)  
my\_list.append(5)  
print(my\_list)  
my\_list.pop(0)  
my\_list.pop(0)  
print(my\_list)

Binary tree:

A binary tree is a hierarchical data structure which consist of more than more child. The two children are the left child and right child. The top most node is called root and the bottom most nodes are the leaves of tree. The nodes above something are its parents and the nodes below something are its children. The binary tree consist of:

Data

Pointer to left child

Pointer to right child

* Time complexity of binary tree is O(n).