

Serialization (using constructor)

Constructor: A constructor is a special method defically name `--init--()` that is automatically called when object of the class is created. The main purpose of the constructor to initialize the instance variable or attribute of object with specific value. In another word we can say that constructor in python is special method of class for creating and initializing of and object of the instance.

Example:

Class PERSON:

```
def __init__(self, name, age, city, mobile):
```

```
    self.name = name
    self.age = age
    self.city = city
    self.mobile = mobile
```

```
def introduction(self):
```

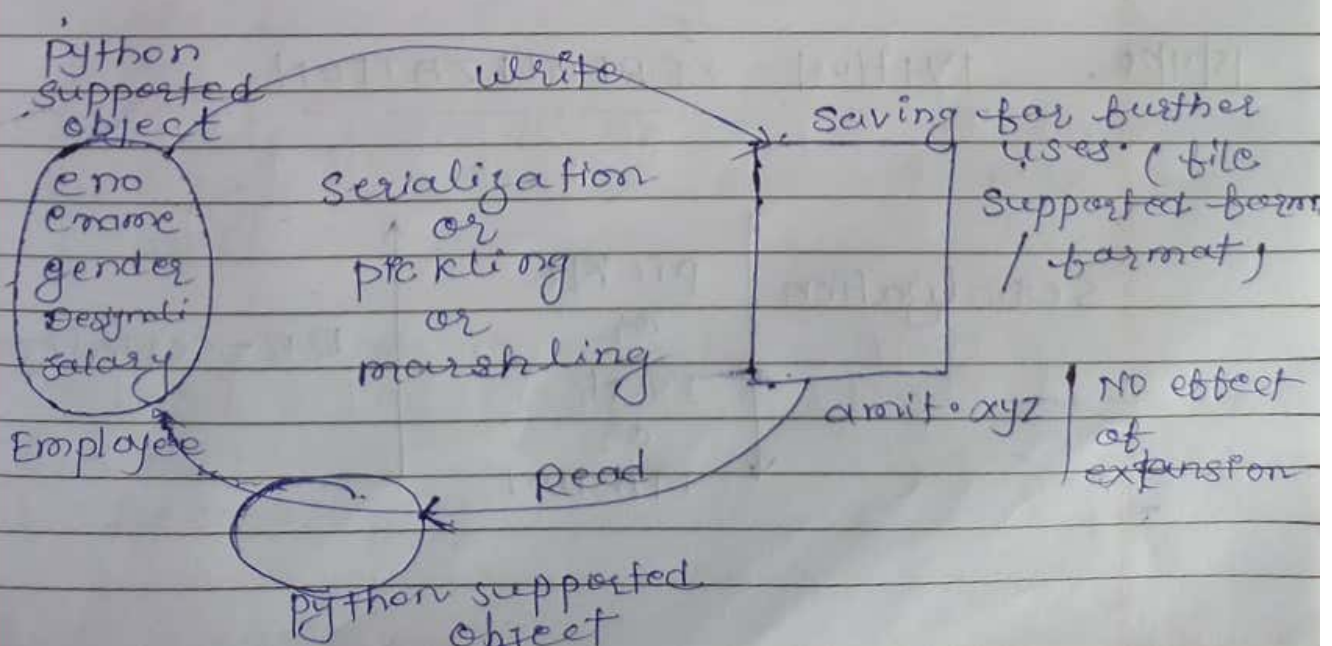
```
    print(f"my Name is {self.name}
           age is {self.age}
           city is {self.city}
           and mobile is {self.mobile}")
```

```
my-person = PERSON ("Ramesh", 25, "Delhi" 9758329)
my-person2 = PERSON ("Priya", 22, "Ranchi" 4325678)

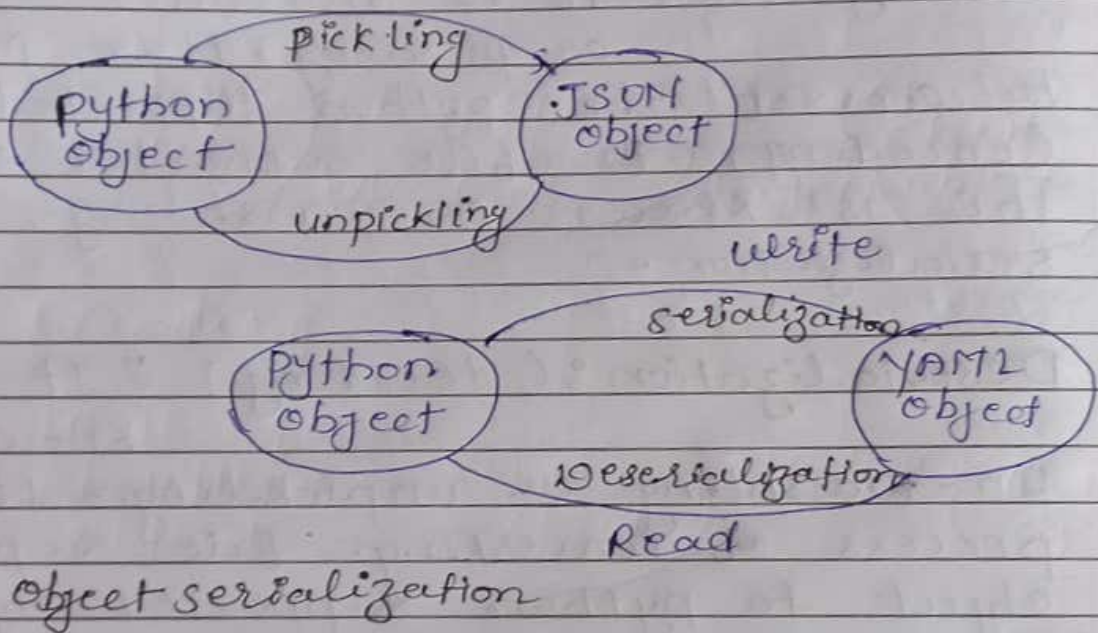
my-person.introduction()
my-person2.introduction()
```

Serialization: It is also called marshaling or pickling. It is process to convert the python supported object (encoding) to file supported object. This is known as marshaling or serialization.

Deserialization: (decoding) It is also known as un-marshaling or unpickling. It is a process of converting file supported object to python supported object is known as unmarshaling.

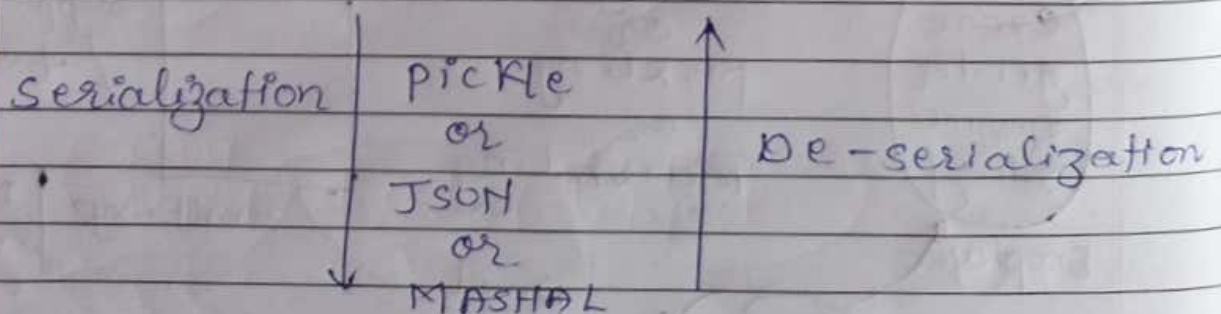


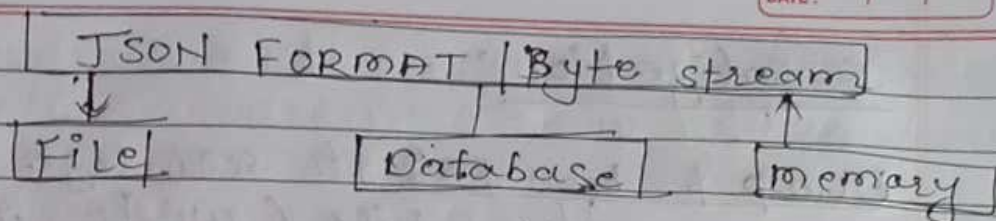
In another word we can say that the process of converting an object from python to file supported form or network supported form either be a JSON or YAML (yet a another markup language)



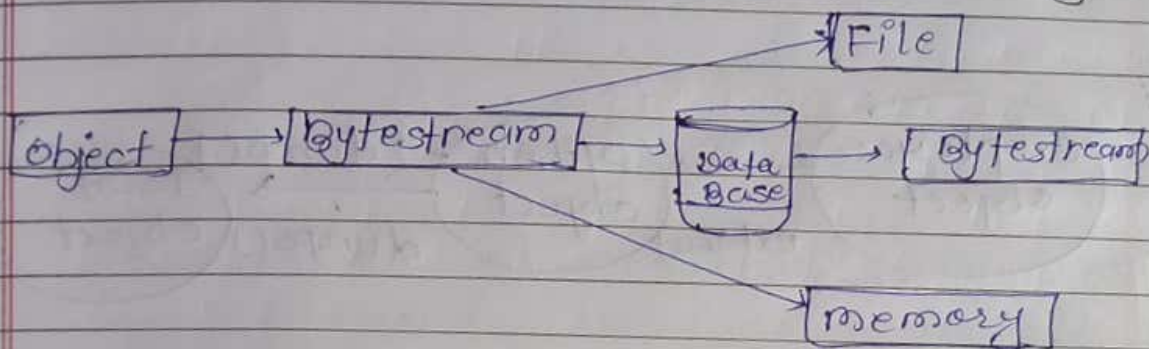
Object serialization

15/4/25 PYTHON SERIALIZATION





process of pickling and unpickling



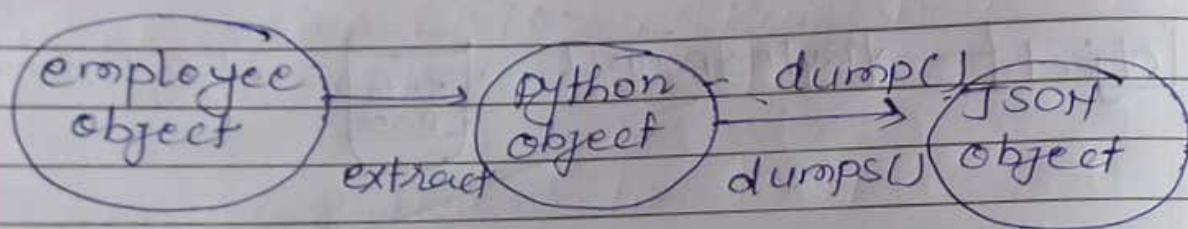
pickling is a process where a python object into bytestream to written into file and memory this phenomena is known as pickling (pickle) whereas unpickling is a process where a object stored a bytestream and converted into a python object is known as unpickling.

we can use 4 method for conversion of python object into another object (JSON) or (YAML).

- ① dump() or dumps()
- ② load() or loads()
- ③ Encoding()
- ④ Decoding()

Serialization

Dump & Dumps: It serializes to open the file (write into file supported format).

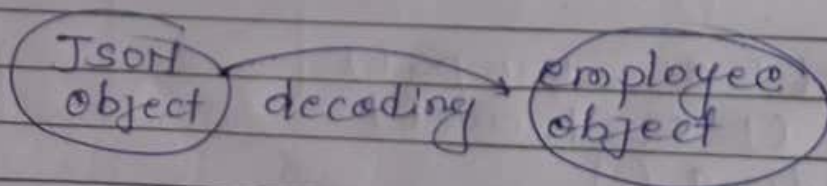
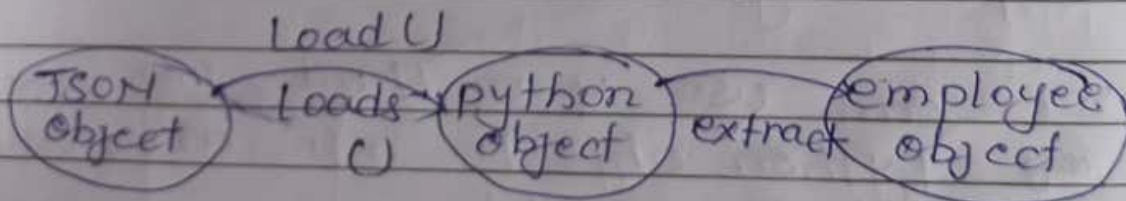


Encoding

In the form of python object



Deserialization



By using JSON pickle module we can serialize our custom class object directly to JSON by input method and this serialize JSON to our custom class object directly by the help of decode method.

Note:

JSON module not directly available by default we can install JSON from PIP.

Example:

program (serialization / de-serialization) by JSON module.

```
class EMP:
```

```
    def __init__(self, eno, ename, ecity,  
                  emobile, esal):
```

```
        self.eno = eno
```

```
        self.ename = ename
```

```
        self.ecity = ecity
```

```
        self.emobile = emobile
```

```
        self.esal = esal
```




```
def display (self):  
    print (f "Employee No = {self.eno}  
            Employee name = {self.ename}  
            Employee city = {self.city}  
            Employee mobile = {self.emobile}  
            Employee salary = {self.esal}")
```

```
emp1 = emp (5768 , "Ramesh" , "Delhi",  
            567898 , 12000)
```

Serialization to JSON

```
Json-string = jsonpickle.encode(emp1):
```

```
print ("After conversion from emp object  
to JSON object = " , Json-string)
```

Thread programming

Thread: A single flow of execution is known as thread.

Multi-thread: A multiple flow of execution is known as multithread. A multithreading is much-much similar to multitasking or multiprocessing.

Multitasking:

Task: A running state of program / job process is known as task.

Task are of two type

- ① single tasking
- ② multitasking

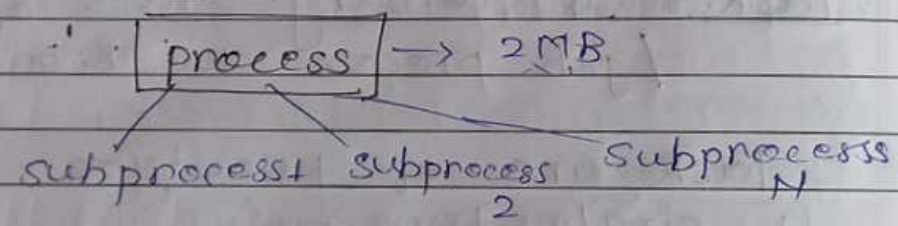
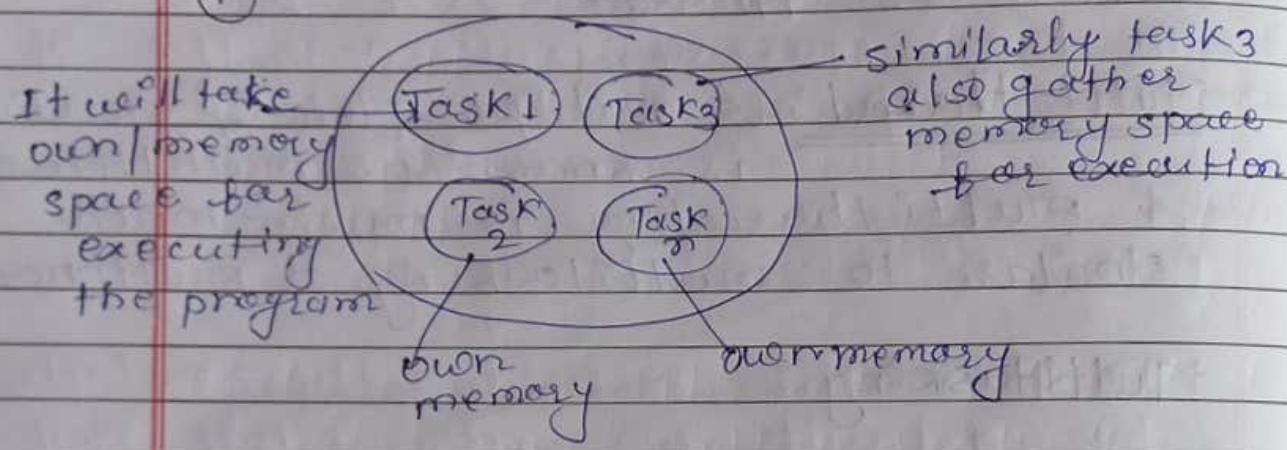
① Single tasking: At a time user can perform single process or job.

② multitasking: At a time more than one job or process. multitasking means executing more than one task at the same time.

Multitasking can be classified into two categories.

- ① process based multitasking
- ② Thread based multitasking

① process based



② Thread based multitasking: It will be ~~ex~~ increased the program efficiency as compared process based

↳ It is based on scheduling algorithm

Packet: small amount of information.

↳ Each task has independent program or process. and every process contain own memory or individual memory

or several execution of the program the process based multitasking work on the principle of scheduling specially Round Robin as first come first serve.

- ↳ Each task is independent thread and execute at separate part of the program.

EX: Suppose we have hundred student in coaching center each batch has 10 student and 4th batch going on by teacher that is based on time slot if company hire 3 more teacher and batch distribute among them then all batch are going parallelly that means every teacher is a strength and this phenomena is known as thread based multitasking or parallelism.

Thread: It is a operating system object that execute the instruction of the program. It follow the scheduling algo also. (RR scheduling)

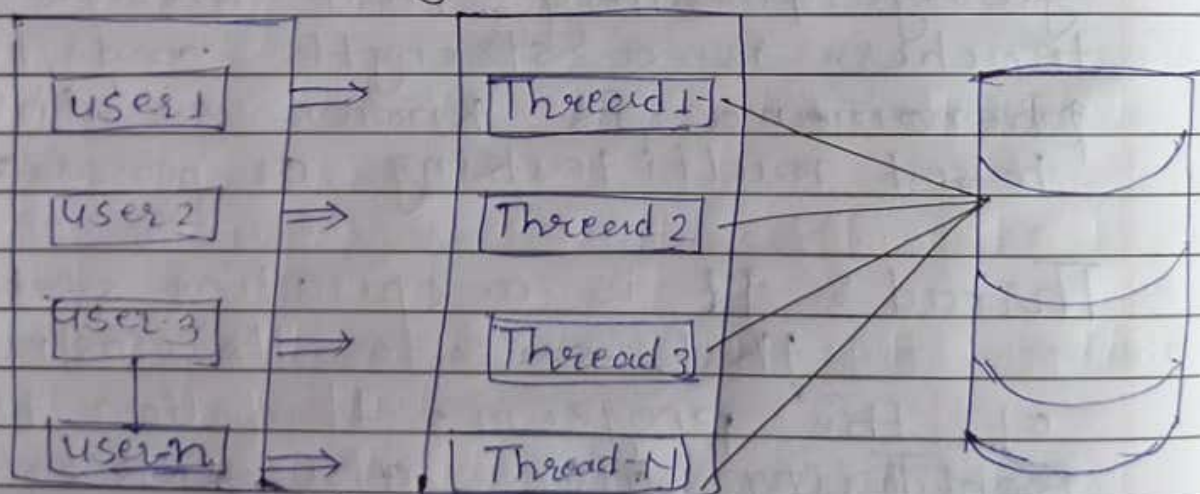
main thread: when we start to create a python program and one thread running automatically immediately is known as main thread of the program.

Multithreading: It is a conceptual programming where a program is divided into more than one or two or n sub programs and all programs will be executed concurrently or simultaneously. This phenomenon is known as multithreading programming.

Note: main thread is created by pvm (python virtual machine). pvm is some time known as python interpreter.

Multithreading server

Database



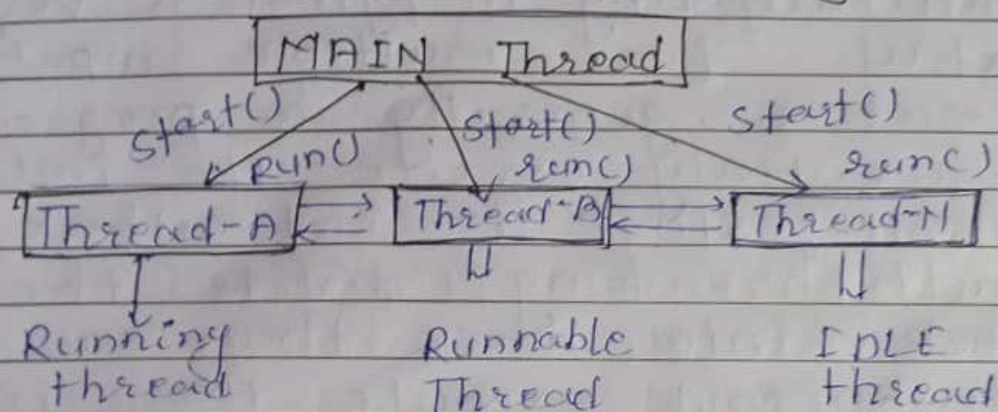
Example of Multithreading:

① pubgi (In pubgi user can perform

Several task at the same time concurrently

- (I) Image displaying
- (II) multimedia graphics.

Thread programming



Running thread: A program / processing is running at current time slice / slot.

Runnable thread: A thread is ready for the execution but waiting for availability of the processor. Processor is busy in execution of another thread.

Idle thread: It is not dead or not alive. It may be either suspended or waiting for processor.

Note: A multithreading allow you to run multiple thread concurrently within a single process. This is known as thread based parallelism.

Multithreading in python is very useful for multiple input/output operation, generally a program sequentially execute the instruction from start to end where as multithreading divide the main task into more than one sub task and execute them in overlapping manner.

Creating thread:

How to create thread in python)

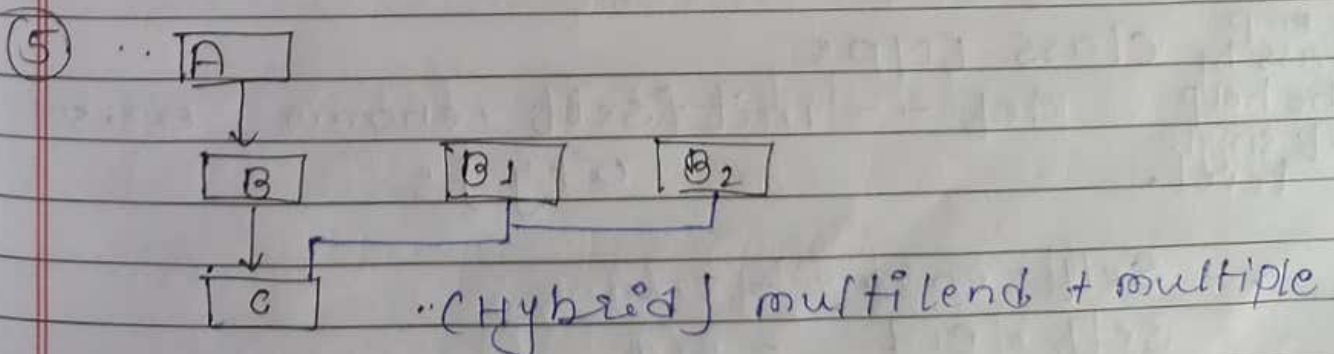
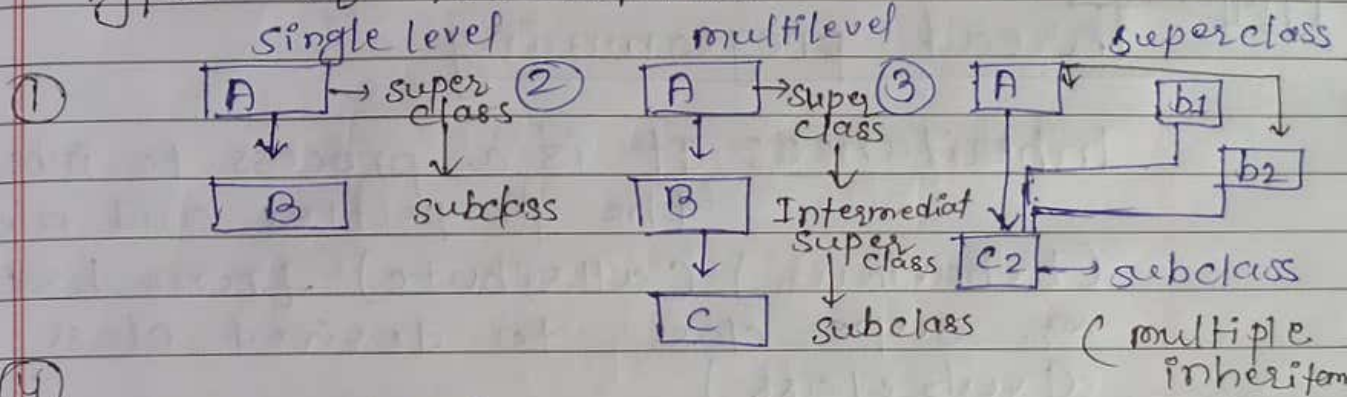
we can create a thread into two way

- ① By thread class present in thread module.
- ② By Extending thread class.
(Inheritance)

✱ Inheritance : Inheritance is very important concept of oops that follows the python.

that means we can say that it is mechanism to allow the subclass or child class or derived acquire the attribute / property / method / behaviour of parent class or super class.

Types of inheritance



Note: The inheritance allow the user for code reuse that means and reducing data redundancy as well as ~~from promoting~~ the hierarchical structure of oops.

17/4/25

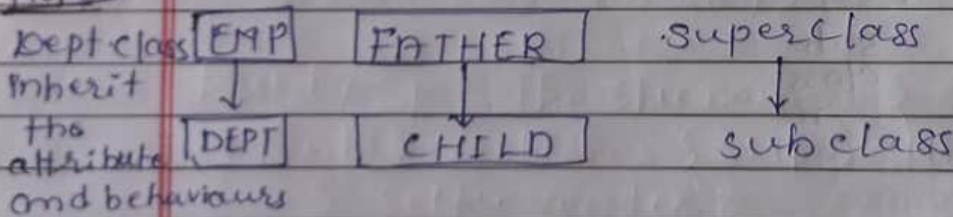
Thread programming

Inheritance: It is a process to inherit the properties and method (behaviours) (attribute) from base class or super class to derived class (sub class)

Advantages:
 ① code Re-useability
 ② data redundancy

① Example of inheritance (single level)

Note:



class EMP:

def __init__(self, ename, eid, salary, city):

self.ename = ename
 self.eid = eid
 self.salary = salary
 self.city = city

```
class DEPT(EMP):
```

```
    def __init__(self, ename, eid, salary, city, deptname, sub):
```

```
        def emp-info(self):
```

```
            print(f"The EmpName = {self.ename} \n Identification = {self.eid} \n salary {self.salary} \n city {self.city} \n")
```

```
super().__init__(ename, eid, salary, city)
```

```
self.deptname = deptname;
```

```
self.sub = sub
```

```
def emp-info1(self):
```

```
    super().emp-info()
```

```
print(f"The DeptName = {self.deptname} \n and sub = {self.sub}")
```

```
dept1 = DEPT("AMIT", 1234, 10000, "Ranchi", "MCA", "python")
```

```
dept1.emp-info()
```

```
dept1.emp-info1()
```

- Q. Write a program in python using oops inheritance. Input name, roll, gender, 5 submarks. Find total and avg and display result using single level inheritance.

19/4/25

PAGE NO.:

DATE: / /

Multi threading

→ How to create a thread class

step 1: using API package
[from threading import *]
or

import threading

step 2 - How we can create new thread with the help of thread object

object name = threading.Thread(target
; args = ())
~~optional~~

or

object name = Thread(target = ())

step 3 - using two method to execute and stop the thread.

(a) start thread

object-name.start()

(b) stop thread

object-name.stop()

or

object name.join()

WAP to print first 10 natural using thread class

```
from threading import *
class Natural:
    def compute(self):
```

```
        num = 0
        while (num < 10):
            print (num)
            num = num + 1
```

```
obj1 = Natural()
thread1 = threading.Thread(target=obj1.compute())
thread1.start()
thread1.join()
```

WAP to implement thread class and find even no. within given range.

```
import threading
class EvenNumbers:
    def display(self):
        i = 10
        j = 20
```

for loop syntax
for counter-variable in range(start, stop, increment/decrement)

```
for num in range(i, j+2):
    if (num % 2 == 0)
```



```
print(num)
```

(Or)

```
print(f"The even number i = {self.num}")
```

```
en = EvenNumber()
```

```
thread1 = Thread(target = en.display())
```

```
thread1.start()
```

```
thread1.join()
```

WAP in python to create two thread and find even and odd number from 30 to 50 without using thread class by function.

```
import threading
```

(Or)

```
from threading import *
```

```
def EvenNumber():
```

```
    print("List of even number")
```

```
    for num in range(30, 51 + 2):
```

```
        print(num)
```

(Or)

```
print(num, end=" ")
```

```
def odd.Number():
```

```
print("List of odd Number")  
for num1 in range(31, 50+2):
```

```
    print(num1)
```

or

```
print(num1, end = " ");
```

```
thread1 = threading.Thread(target = Even number)  
thread2 = threading.Thread(target = Odd number)
```

```
thread1.start()
```

```
thread1.join()
```

```
thread2.start()
```

```
thread2.join()
```

Note: using print (xyz, end = " ")

takes the cursor to
the next line

"double space within
inverted comma"

Thread programming

How to create thread by extending thread class

We know that we can create a thread by three approach.

- ① By thread class
- ② By function approach
- ③ By extending thread class (over-riding + inheritance)

What do you mean by method over-riding
 $\text{run()} \rightarrow$ Inbuilt method (pre-defined method)

- ① we never change the properties and behaviours of run method.
- ② we never change the name of run()

Method overriding is a very importance concepts of oops or polymorphisms that means it is much much similar to one name with multiple form in another word we can say that method overriding refers to the method in sub class with same name as method in super class.

Class Emp: \nearrow superclass

```
def __init__(self, name, salary):
    self.name = name
    self.salary = salary
```


inc - increment

```
def getName(self):  
    return self.name
```

```
def getSalary(self):  
    return self.salary
```

```
class office (Emp):  
    def __init__(self, name, salary, inc):  
        super().__init__(name, salary)  
        self.inc = inc
```

subclass

```
def getSalary(self):  
    return self.salary + self.inc
```

```
e1 = Emp ("Ramesh", 70000)  
print(f "The salary of emp = {self.salary}  
      {e1.getName()},  
      {e1.getSalary()}")
```

```
e2 = office ("RITA", 50000)  
print(f "The salary with increment  
      = {self.salary}",  
      {e2.getSalary()})
```

Python provide inbuilt thread class
we have to create a class by
extending from the main thread
class. Inside the thread class
python provide run method that

mean run method will be inside the main thread and we have to overriding the run method. In another word we can say that the run method will override the thread execution logic. When start method is called that means we can say that run method must be invoked by another method which name is start method.

Note: A run method is a heart and soul of thread programming in case of extended class module. That means we can say that a small program or complex program must uses the run method for override the thread execution from super class to subclass.

If subclass override the constructor it must sure to invoked thread -- init -- () method. Before doing anything else to the read.

Thread communication

Queue: Queue is sequential data structure where data element can be produce at one end and data element can be consume at other end is known queue unit.

In other way we can say that Queue is a FIFO structure where data element can be inserted at one end is known a Rear of the queue.

In Queue data structure we can use queue module ~~or~~ thread communication in ~~python~~ ^{queue module} python we can implement queue. queue class we can perform two primitive operation in queue module.

(1) Insertion

(2) Deletion

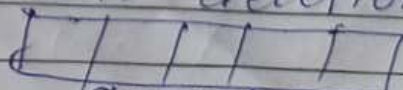
Insertion can be take place with the help of ~~put~~ ^{put function} concept and deletion can be take place in get place. In another word we can say that a queue is a linear data structure where insertion of the element at one end is known as rear and deletion of the element and other is front of the queue. It follow FIFO structure.

Note: we can implement two method in thread programming for adding and deleting the element.

(1) put()

(2) get()

In Empty queue we can insert the element. Then deletion will be performed.



queue empty.

Q Example: WA python program using thread communication or object for adding and deleting the element by the help of queue module.

```
from threading import Queue
import time
```

```
def producer (my-queue):
    num = int (input ("Enter the size of queue="))
    for i in range (1, num):
        marks = float (input ("Enter the marks of student="))
```

```
my-queue.put (marks)
my-queue (None)
```

```
def consumer (my-queue)
print ("consume the item while True:
    item = my-queue.get()
```

Note

get(): In python it will be delete 1st element at array

None: Last element of array queue
stop

→ if item is none:

```
break
print(f "The item is {item}")
```



```
print(f "consumer End")
```

```
my-queue = Queue()
```

```
thread1 = Thread(target = producer  
arg = my-queue)
```

```
thread2 = Thread(target = consumer,  
arg = my-queue)
```

```
thread1.start()
```

```
thread2.start()
```

```
thread1.join()
```

```
thread2.join()
```

28/4/25 ARRAY in python

It is special variable which hold the multiple value in same datatype.

↳ An array is a special variable which can hold more than one value that means we can say that array in python is a data structure that store collection of element of some data type. Array are mainly design for numerical operation and offer memory efficiency. In another word we can say that

an array can be defined as storage for containers of similar item place in continuous memory allocation.

In ~~per~~ python we can use or create array into two way

- ① Array module and
- ② Numerical Python (Numpy)

① Array module : for array modules we can use API package or inbuilt library of python.

Syntax of array

array-name = array (data type, [items])

↓
user defined

Ex: Suppose we want to store 5 integers
value in array module.

arr - num = array, ('i' [1, 2, 3, 4, 5])
↓ integer datatype

There are several type of array in array module.

- (1) zero dimension
- (2) 1 - dimension
- (3) 2 - dimension