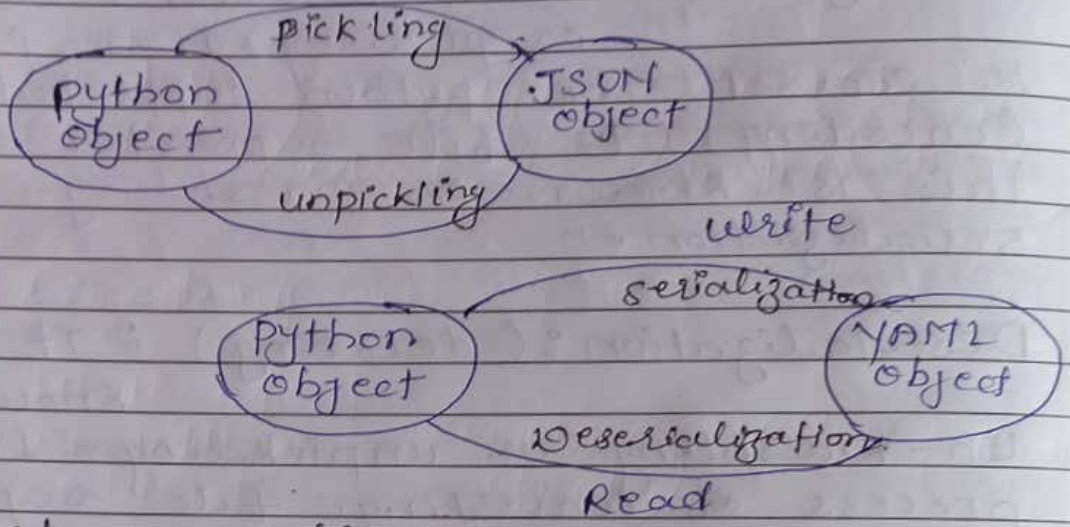
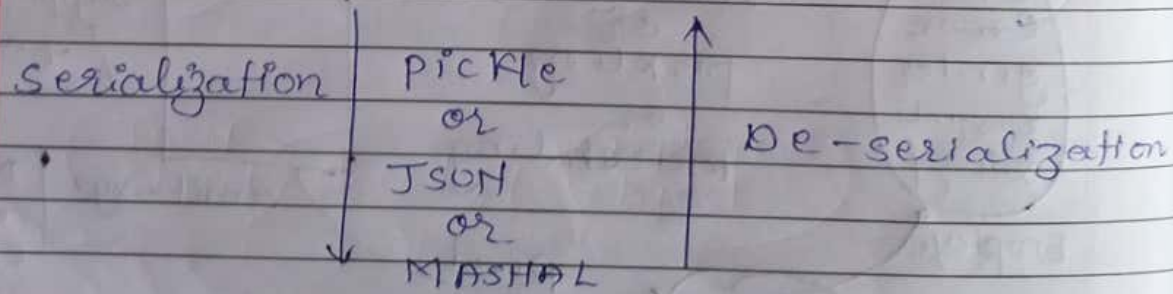
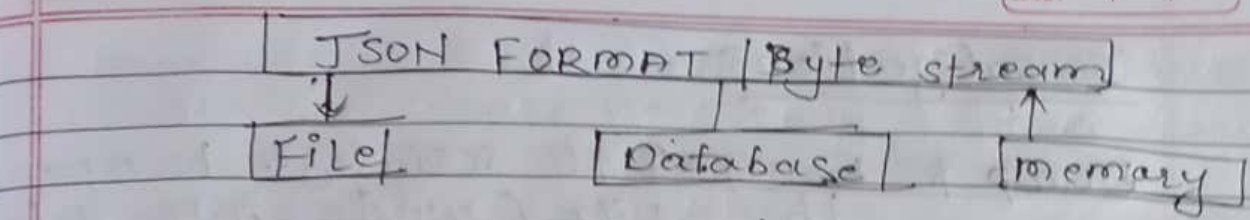
In another word we can say that the process of converting an object form python to file supported form or network supported form either be a JSON or YAML (yet another markup language)



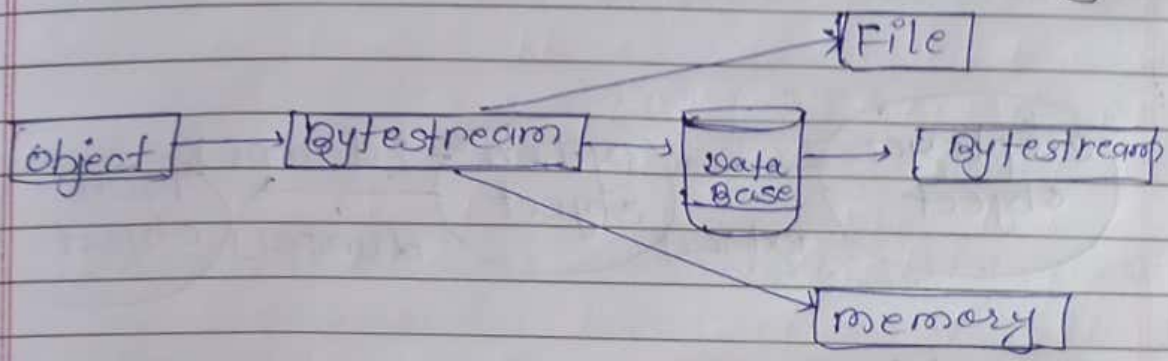
Object serialization

15/4/25 PYTHON SERIALIZATION





process of pickling and unpickling



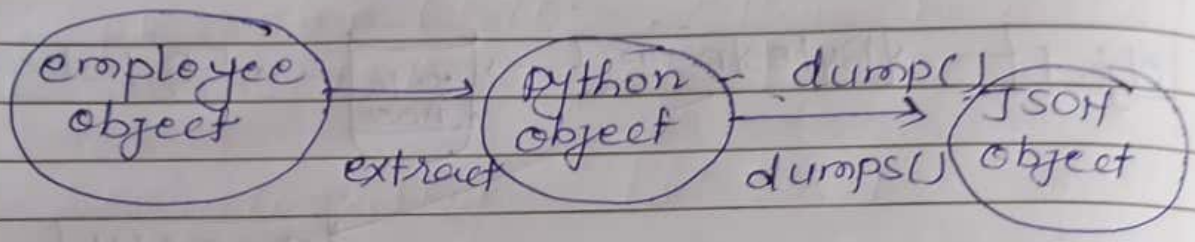
pickling is a process where a python object into bytestream to written into file and memory this phenomena is known as picking (pickle). whereas unpickling is a process where a object stored & bytestream and converted into a python object is known as unpickling.

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()

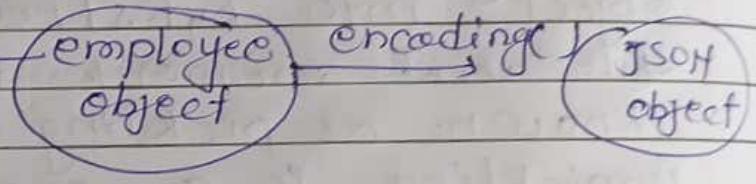
Serialization:

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

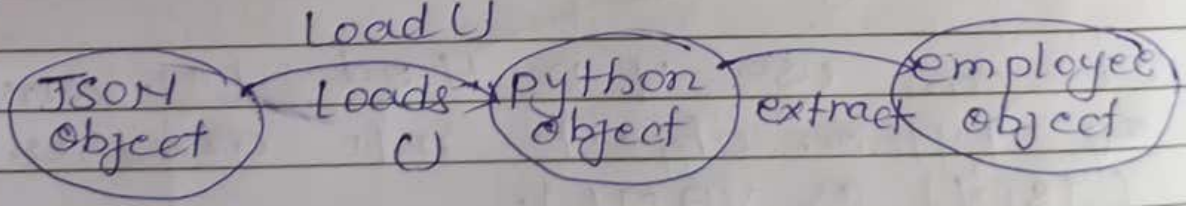


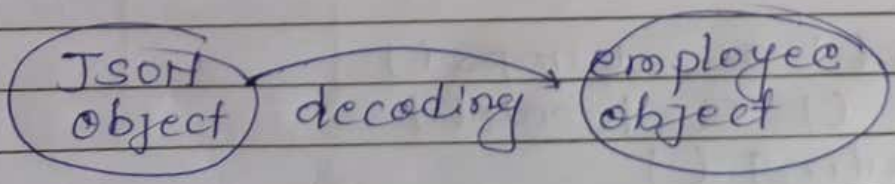
Encoding

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.ecity}
        Employee mobile = {self.emobile}
        Employee salary = {self.esal}")

empt = Emp (5768, "Ramesh", "Delhi",
            567898, 12000)
```

Serialization to JSON

```
JSON-string = json.pickle.emode(empt)
print ("After conversion from emp object
to JSON object = ", JSON-string)
```

PAGE NO.:
DATE: 16/4/25

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

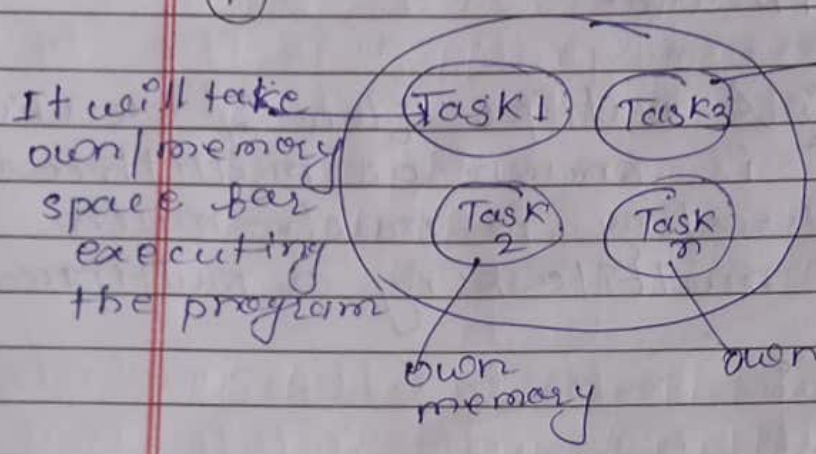
1. single tasking: At a time user can perform single process or job.

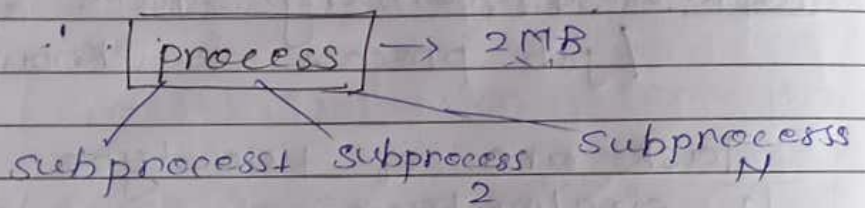
2. multitasking: At a time more than one job or process. multitasking means executing more than one task at the same time.

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

PAGE NO.:
DATE: / /

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

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

Ex: suppose we have hundred student in coaching center each batch has 10 student and 4th batch going on by teaches that is based on time slot if company hire 3 more teacher and batch distribute among them than all batch is going parallely that means every teacher is a strength and this performance is known as thread based multitasking or parallisim

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

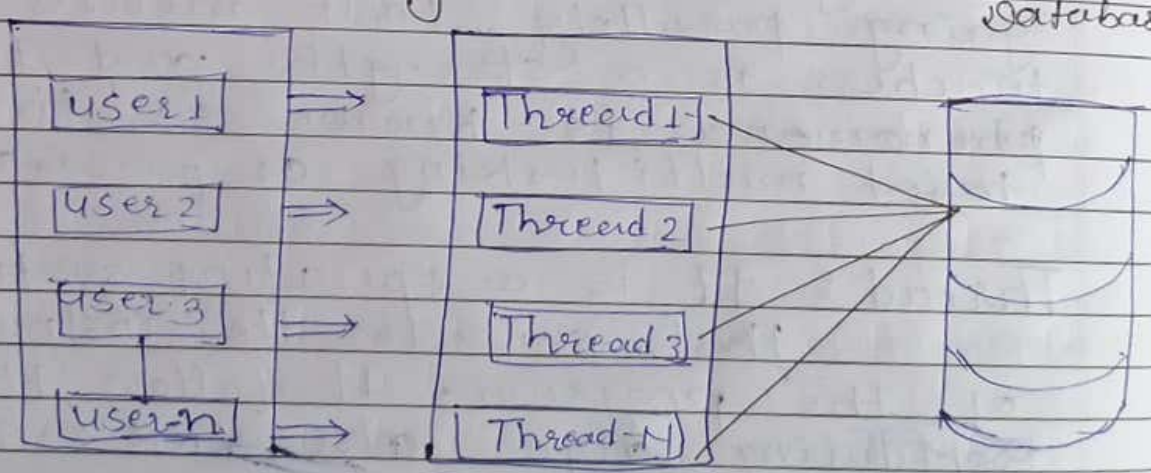
main thread: when we start to create a python program and one thread running automatically immediatly is known as main thread of the program.

PAGE NO.:
DATE: / /

Multithreading: It is a conceptual programming where a program is divided into more than one or two or n sub programs and all program will be execute concurrently or simultaniously. This phenomena 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



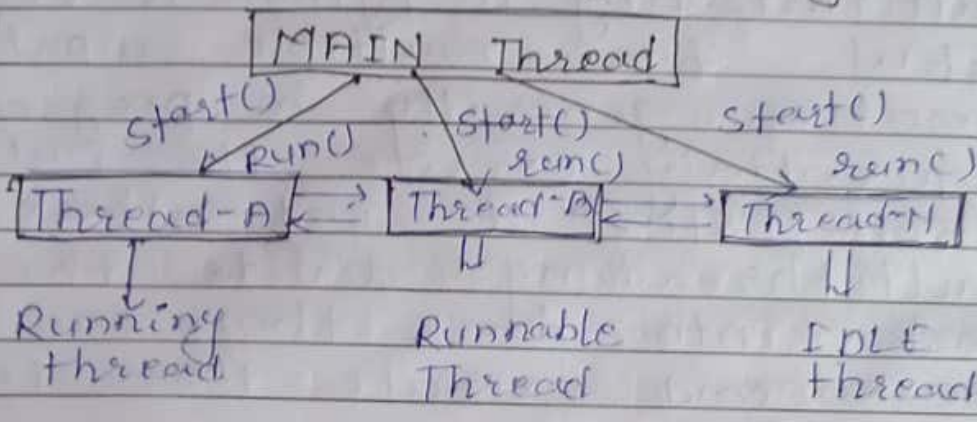
Example of Multithreading:

① pubgi (In pubgi user can perform

Several task at the same time concurrently

(i) Image displaying
(ii) multi media 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 process actually processor is busy in execution of another thread.

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

PAGE NO.:
DATE: / /

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 overlaping manner.

Creating thread:

How to create thread in python)

We can create a thread into two way

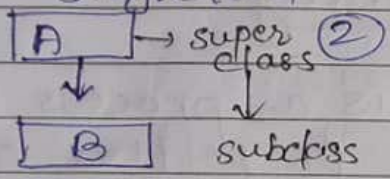
1. By thread class present in thread module.
2. By Extending thread class. (Inheritance)

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

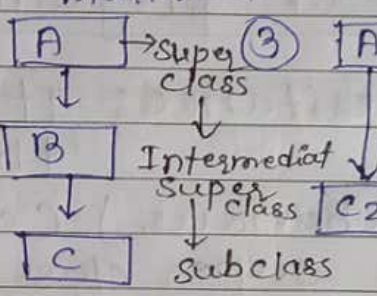
that means we can say that it is mechanism to allow the subclass or child class or derived acquire the attribute / property / method / behaviour of parent class or super class.

Types of inheritance

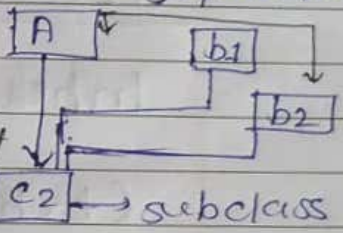
① single level



② multilevel

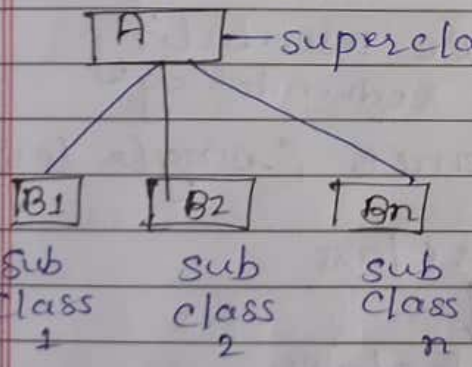


③ superclass

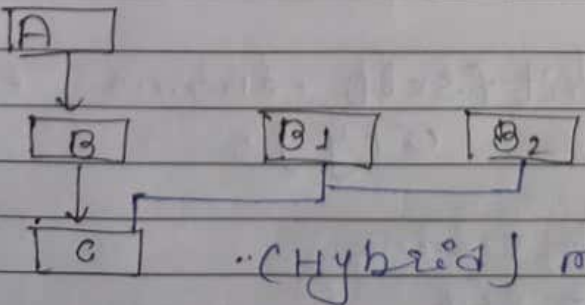


(multiple inheritance

④ Hierarchical



⑤



(Hybrid) multilend. + multiple

PAGE NO. :
DATE : / /

Note : The inheritence allows the user bar code reuse ~~that means~~ and reducing ~~data~~ data redundancy as well as ~~from~~ promoting the ~~hierarchical~~ hierarchical structure of oops.

17/4/25

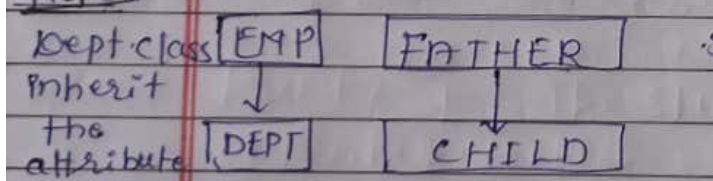
Thread programming

Inheritance: It is a process to inherit the properties and method (behaviour) (attribute) ~~from~~ 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}
 Identification = {self.eid} salary {self.sala
 city {self.city}                          }

super().def--init--(ename, eid, salary, city

 self.dept-name = dept.name;
 self.sub = sub
 def emp-info1(self):
     super().emp-info()

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

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

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

Q WA 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

Multithreading

→ How to create a thread class

step1 ; using API package

[from threading import *]

or

import threading

step2 - How we can create new thread with the help of thread object

object name = Threading.Thread (target = ( ), arg = ( ))

optional

or

object name = Thread (target = ( ))

step3 - using two method to execute and stop the thread.

① start thread

object-name.start( )

② 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 Even Number:
    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 numbers")
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 "

PAGE NO.:
DATE: 22/4/25

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-ridding + inheritance)

What do you mean by method over-ridding

run() → 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 overridding is a very importance concepts of oops or poly morphism 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.

→ superclass

```
Class EMP:
    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):          # subclass
    def __init__(self, name, salary, inc)

super().__init__(name, salary)
    self.inc = inc

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 inbuild thread class we have to create a class by excending 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 overridding 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 envocked by another method which name is start method.

Note: A run method is a heart and soul of thread programming in case of extented 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 envocked thred.--init -- () method. Before doing anything else to the read.

PAGE NO.:
DATE: 25/4/25

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.