Ⓗ Pass by reference :- Ⓗ When we do $x=5; y=x;$ In the scenario another value is not created for y, another reference of the memory allocation 5 is created.

Ⓗ When we pass argument to function, in case of immutable objects value is not changed for the obj outside the function. But In case of mutable objects, object value is changed outside the function on changing it inside the function.

Ⓗ Pass by value : Ⓗ When we do $x=5; y=x;$ In this scenario another memory allocation is created for y.

Ⓗ When we pass argument to a function and changed value, that objects value is not changed outside the function in any case.

Ⓗ Namespace : Ⓗ Namespaces are like dictionary where key are the object names and values are the object themselves.

Ⓗ Types of name spaces : (i) Built-in (ii) Global (iii) Enclosing (iv) Local.

Ⓗ Built in Namespaces contain all the built in objects i.e, objects from standered library. It is available all the times throughout the prog

Ⓗ Global Namespace is at the module level. It contains all the functions and variables at the module level.

Ⓗ Local Namespace is at the function level. It contains all the variables inside the function.

Ⓗ Enclosing Namespace is when we define a inner () function inside a function say outer (), then all the variables in outer() function is at enclosing namespace.

DataStructure & Algorithm for memory management:

OS virtual memory manager creates a chunk of virtual memory for the pvm.

Pvm's object allocator allocates memory whenever a new object is created or deallocates when a object is deleted.

Pvm's memory allocation strategy has 3 components: arena, pool, block.

Blocks are chunks of 8 bytes of memory which is called a single size class. **Pools** are chunks of 64 blocks. **Areana** is generally of size 256kb of memory, i.e, chunck of 50 pools.

Blocks: Blocks can be in three states:

untouched ⇒ These blocks are never allocated with data.

free ⇒ These blocks are once allocated but now free to store data.

allocated ⇒ These blocks are filled with allocated data

Pools: Pools has three states:

used ⇒ these have available blocks of data to be stored. A used pools list tracks all the pools in used states.

full ⇒ These do not have available blocks of data. If these pools frees some data then it added to the list of used pools.

empty ⇒ These pools have never allocated with data. Freepools list keeps track of all the empty pools.

Arenas: Arenas does not have states. These are organised into double linked list and keeps track of the no. of free pools available. If all the pools in a arena is free, then the arena chunck is truly freed, i.e, that chunck becomes available for the OS use.

Celery

Task schedular build on top of message broker such as Redis, Rabbitmq.

It extends the functionality of message broker for specific use case such as task queing.

Celery beat is a component used to send periodic or cron-like task at certain intervals.

Flower is a component which is a monitering tool for celery tasks.

Celery worker : Consumer of the message queue.
Celery client : Publisher of the message queue.

#

$$\text{app} = \text{Celery}(\underbrace{\text{'ins1'}}_{\text{Name of celery instance}}, \underbrace{\text{broker: 'amqp://'}}_{\text{message broker to use for delivring messages}}, \underbrace{\text{backend: "rpc://"}}_{\text{Result store to use for storing message state or returning value}}, \underbrace{\text{include: ["tasks.mod1"]}}_{\text{modules to include when searching for a method in this instance}})$$

) File Handling ⇒ # File Stream # File Object # Text Stream, Binary Stream

Class hierarchy ⇒

$$\begin{tikzcd}
 & \text{IO Base} \arrow[dl] \arrow[d] \arrow[dr] & \\
\text{RawIoBase} & \text{BufferedIO Base} & \text{Text IOBase}
\end{tikzcd}$$

Text Streams ⇒ text Based files , excel files .

Binary Streams ⇒ Executable files.

Unix / Linux Based files ends with LF | "/n".

Windows based files ends with CRLF | "/r/n".
These creates problem while file manupulation.

To resolve this issue from windows , after the file is opened another process called 'translation of new line characters" occur in the class level. This transformes the CRLF characters to LF characters and vice versa for writing operation.

$$\text{open}(\underbrace{\text{fileName.txt}}_{\text{fileName}}, \underbrace{\text{mode = rt}}_{\text{Mode according to the operation on file}}, \underbrace{\text{encoding = None}}_{\text{Encoding in which the file should open}}) \rightarrow \text{File Stream Object}$$

Some streams which are automatically opened when the program starts.
sys. stdIn () stream ⇒ Input Stream ^ This is used by input () .
sys. std out () stream ⇒ output stream ^ This is used by print ().
sys. stderr () stream ⇒ Error stream.

File encoding ⇒ We can explicitely mention file encoding in a python script.
Example: #! bin/python.
-*- coding : <encoding name> -*-
The encoding defination line must be in 1st/2nd line. Any mistake will raise an error during compilation.
By default encoding is UTF-8. Other encodings can be latin-1,