Multithreading and Multi Processing (threading module)

Applications are two types

1. Single Tasking Applications
2. Multi tasking Applications

Single Tasking applications

An application which allows performing one operation at a time is called single tasking application.

Multi Tasking applications

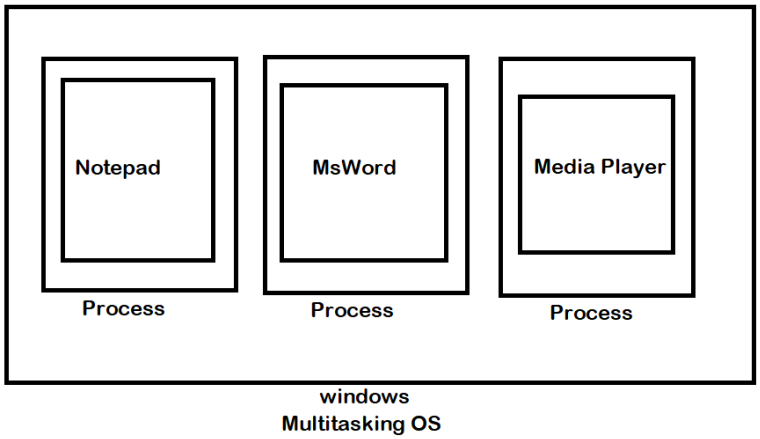
An application which allows performing more than one operation concurrently or simultaneously is called multi tasking applications.

Multitasking applications are two types

1. Process based multitasking
2. Thread based multitasking

Process based multitasking

A process is nothing but program under execution.
A process is nothing an instance of program.
Simultaneous execution of more than one program is called process based multitasking. This is suitable for developing operating systems or network servers but not suitable at application level.



Thread based multitasking

A thread is an instance of process

A thread performs an operation independent of other operations.

A thread is nothing process under execution or program under execution.

A thread is child process.

A thread is an independent path of execution within program.

Adv of Multitasking

1. Utilization of CPU resources or CPU Idle Time
2. Increasing efficiency of programs or applications

For developing thread based multitasking applications python provides “threading” module. It is a default module which comes with python software.

Example:

```
def fun1(a,b): # Keyword and Position only required argument
    print("function1")

def fun2(*,a,b): # Keyword only required arguments
    print("function2")
```

```
def fun3(a,b,/): # Position only required arguments
    print("function3")

def fun4(a,b,/,*,x,y):
    print("function4")

fun1(10,20)
fun1(b=20,a=10)
fun2(a=10,b=20)
fun3(100,200)
fun4(1000,2000,x=10,y=20)
```

How to create thread?

Thread is created in two ways

1. Inheriting Thread class (OR) class based
2. Callable object (OR) function based

Thread class or data type

The Thread class represents an activity that is run in a separate thread of control. There are two ways to specify the activity: by passing a callable object to the constructor, or by overriding the run() method in a subclass. No other methods (except for the constructor) should be overridden in a subclass. In other words, only override the __init__() and run() methods of this class.

Once a thread object is created, its activity must be started by calling the thread's start() method. This invokes the run() method in a separate thread of control.

class threading.Thread(group=None, target=None, name=None, args=(), kwargs={}, *, daemon=None)

Callable object or Function Based

Basic steps for creating function based thread

1. Develop function which is executed by thread
2. Create thread object with target as function
3. Execute or invoke thread by calling start method

Example

```
import threading
def even():
    for num in range(1,21):
        if num%2==0:
            print(f'EvenNo {num}')

def odd():
    for num in range(1,21):
        if num%2!=0:
            print(f'OddNo {num}')


# Callable object
t1=threading.Thread(target=even)
t2=threading.Thread(target=odd)

t1.start()
t2.start()
```

Output

```
E:\python7amdec23>python ttest2.py
EvenNo 2
EvenNo 4
EvenNo 6
EvenNo 8
EvenNo 10
OddNo 1
OddNo 3
OddNo 5
EvenNo 12
OddNo 7
EvenNo 14
OddNo 9
OddNo 11
EvenNo 16
OddNo 13
OddNo 15
EvenNo 18
EvenNo 20
OddNo 17
OddNo 19
```