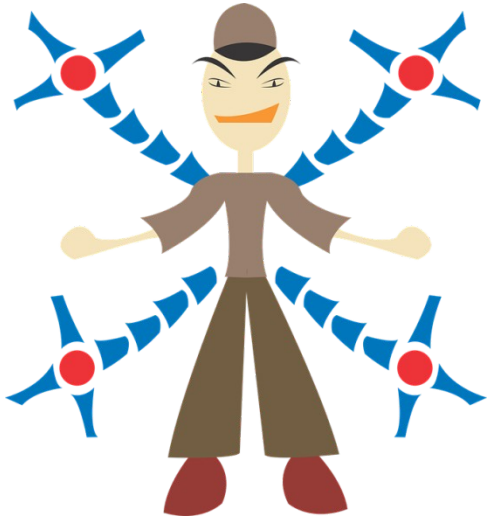
Multitasking

Executing multiple task at the same time.

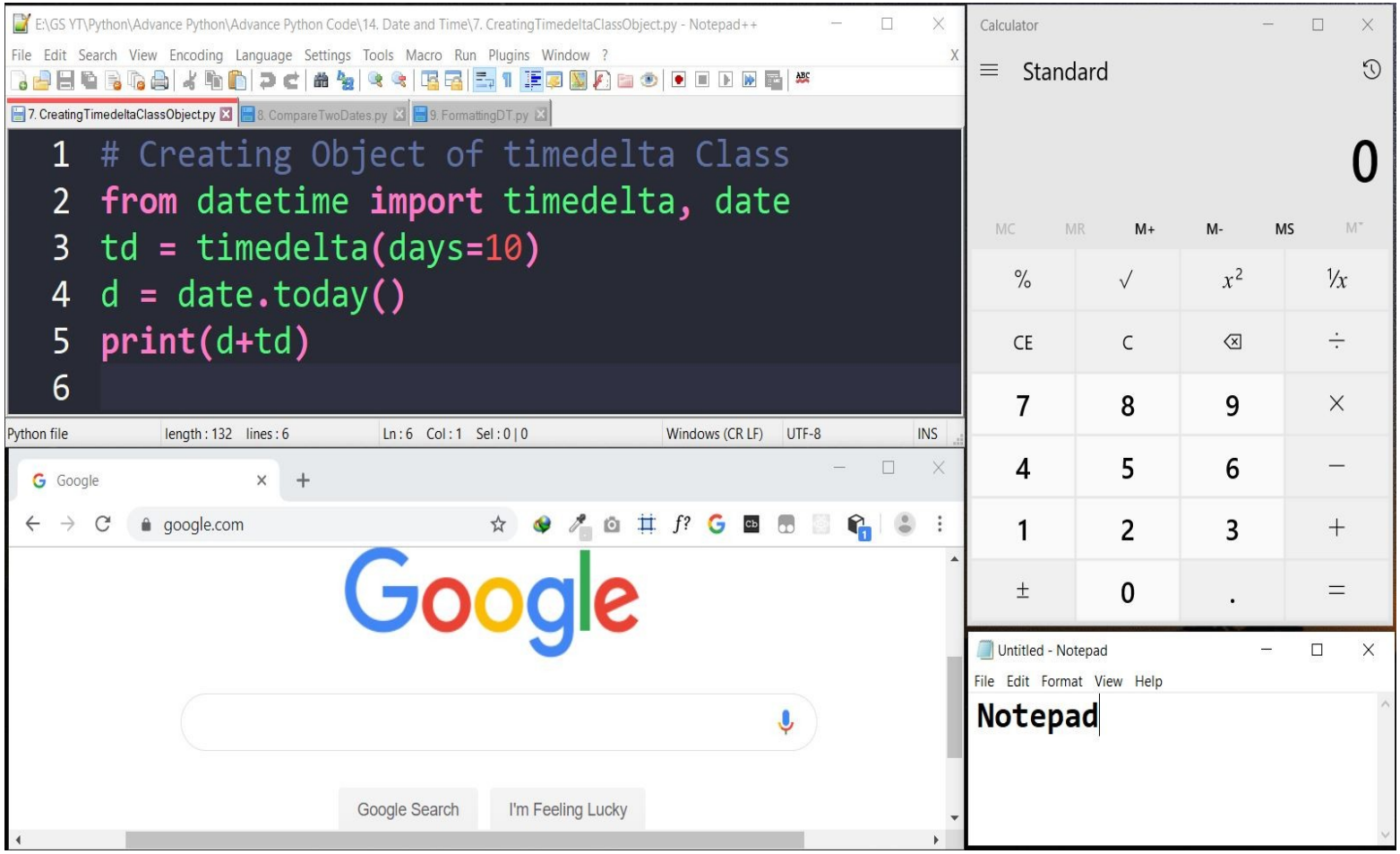


Type of Multitasking

- Process based Multitasking
- Thread based Multitasking

Process Based Multitasking

Executing multiple task at the same time where each task is a separate independent program(process), is called process based multitasking. It is suitable for Operating System level.


E:\GS YT\Python\Advance Python\Advance Python Code\14. Date and Time\7. CreatingTimedeltaClassObject.py - Notepad++
File Edit Search View Encoding Language Settings Tools Macro Run Plugins Window ?
7. CreatingTimedeltaClassObject.py
8. CompareTwoDates.py
9. FormattingDT.py
1 # Creating Object of timedelta Class
2 from datetime import timedelta, date
3 td = timedelta(days=10)
4 d = date.today()
5 print(d+td)
6
Python file
length : 132 lines : 6
Ln : 6 Col : 1 Sel : 0 | 0
Windows (CR LF)
UTF-8
INS
Google
google.com
Google Search
I'm Feeling Lucky
Calculator
Standard
0
MC MR M+ M- MS M^
%
√
x²
1/x
CE
C
÷
7 8 9 ×
4 5 6 −
1 2 3 +
± 0 . =
Untitled - Notepad
File Edit Format View Help
Notepad

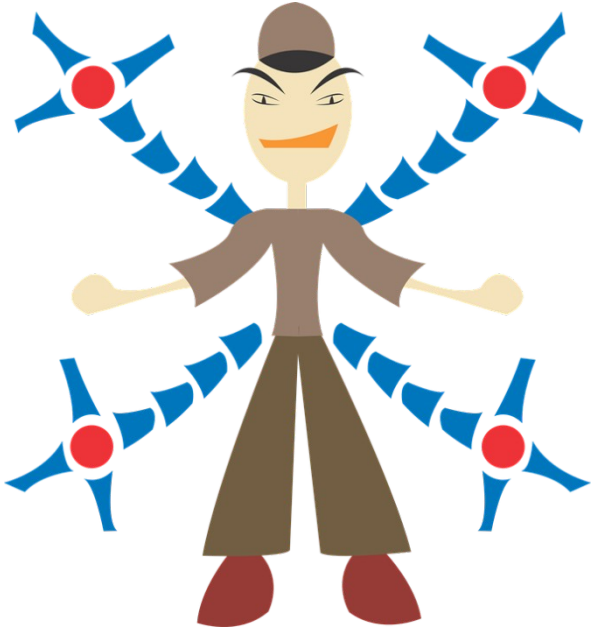
Thread Based Multitasking

Executing multiple task at the same time where each task is a separate independent part of the same program(process), is called Thread based multitasking and each independent part is called Thread. It is suitable for Programmatic level.

Ex: - MS Word

Thread

Thread is a separate flow of execution. Every thread has a task.



- Flying Thread
- CallAuntyMay Thread
- Watching MJ Thread
- Doc Thread

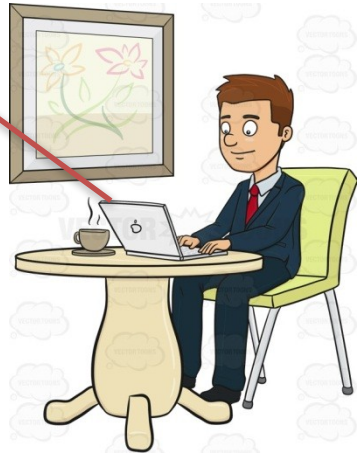
Multithreading

Using Multiple Threads in program or process

The main important application areas of multi threading are:

- Multimedia Graphic
- Animations
- Video Games
- Web Servers
- Application Servers


भारतीय स्टेट बैंक
State Bank of India
With you - all the way
Online SBI



Main Thread

- When we start any Python Program, one thread begins running immediately, which is called Main Thread of that program created by PVM.
- The main thread is created automatically when your program is started.

```
import threading
t = threading.current_thread().getName()
print(t)
```

Creating a Thread

Thread class of *threading* module is used to create threads. To create our own thread we need to create an object of Thread Class.

Following are the ways of creating threads:-

- Creating a thread without using a class
- Creating a thread by creating a child class to Thread class
- Creating a thread without creating child class to Thread class

Creating a thread without using a class

from threading import Thread

thread_object = Thread(target=function_name, args=(arg1, arg2, …))

thread_object – It represents our thread.

target – It represents the function on which the thread will act.

args – It represents a tuple of arguments which are passed to the function.

Ex:-

t = Thread(target=disp, args=(10,20))

How to Start Thread

Once a thread is created it should be started by calling start() Method.

```
from threading import Thread
def disp(a, b):
    print("Thread Running:", a, b)
t = Thread(target=disp, args=(10, 20))
t.start()
```

Starting Thread

```
from threading import Thread
def disp(a, b):
    print("Thread Running:", a, b)
for i in range(5):
    t = Thread(target=disp, args=(10, 20))
    t.start()
```

Starting Thread

```
from threading import Thread
def disp():
    for i in range(5):
        print("Child Thread")
t = Thread(target=disp)
# upto here there is only one thread – Main Thread
# All the above code executed within Main Thread
t.start()
# Once we start Child thread, there are now Two Threads – Main Thread and Thread-1
# Child Thread is responsible to run disp method
# and below code will be run by Main thread
for i in range(5):
    print("Main Thread")
```

Main thread is responsible to create and Start Child Thread, once the child thread has started both the thread behave separately.

Set and Get Thread Name

- current_thread() – This function return current thread object.
- getName() – Every thread has a name by default, to get the name of thread we can use this method.
- setName(name) – This method is used to set the name of thread.
- name Property – This property is used to get or set name of the thread.

  Ex:-

  ```
  thread_object.name = ‘String’
  print(thread_object.name)
  ```

Creating a thread by creating a child class to Thread class

We can create our own thread child class by inheriting *Thread* Class from *threading* module.

```
class ChildClassName(Thread):
        statements
Thread_object = ChildClassName ()
```

Ex:-

```
class Mythread(Thread):
         pass

t = Mythread()
```

Thread Class's Methods

- start ( ) – Once a thread is created it should be started by calling start() Method.
- run( ) – Every thread will run this method when thread is started. We can override this method and write our own code as body of the method. A thread will terminate automatically when it comes out of the run( ) Method.
- join ( ) – This method is used to wait till the thread completely executes the run ( ) method.

Thread Child Class with Constructor

```
from threading import *

Class Mythread(Thread):
        def __init__(self, a):
            Thread.__init__(self)
            self.a = a


t = Mythread(10)
```

Thread Class as Parent Class

Calling Thread Class Constructor

Creating a thread w/o creating a child class to Thread class

We can create an independent thread child class that does not inherit from *Thread* Class from *threading* module.

```
class ClassName:
        statements
object_name = ClassName ()
Thread_object = Thread(target=object_name.function_name, args=(arg1, arg2,…))
```

Ex:-

```
class Mythread:
         def disp (self, a, b): print(a, b)
myt = Mythread()
t = Thread(target=myt.disp, args=(10, 20))
t.start()
```

Single Tasking using a Thread

When multiple tasks are executed by a thread one by one, then it called single tasking.

Writing Examination

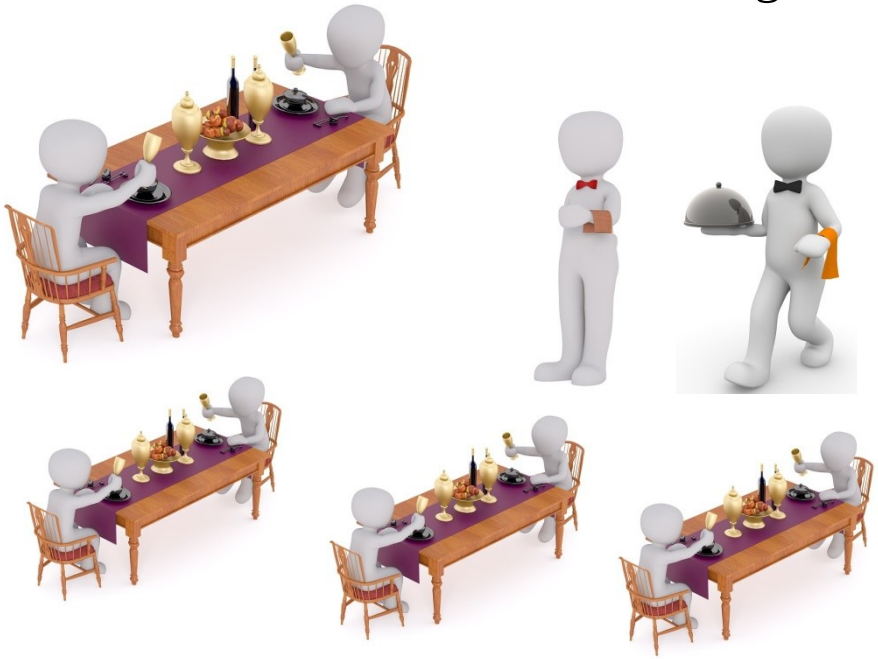
- Question 1
- Question 2
- Question 3

Multitasking using Multiple Thread

When multiple tasks are executed at a time, then it is called Multi-tasking. For this purpose we need more than one thread and when we use more than one thread, it is called multi threading.

Multitasking using a Multiple Thread

When multiple tasks are executed at a time, then it is called Multi-tasking. For this purpose we need more than one thread and when we use more than one thread. it is called multi threading.



Race Condition

Race condition is a situation that occurs when threads are acting in an unexpected sequence, thus leading to unreliable output.

This can be eliminated using thread synchronization.

Thread Identification Number

Every thread has an unique identification number which can be accessed using variable ident.

Syntax:- Thread_object.ident

Ex:- t.ident