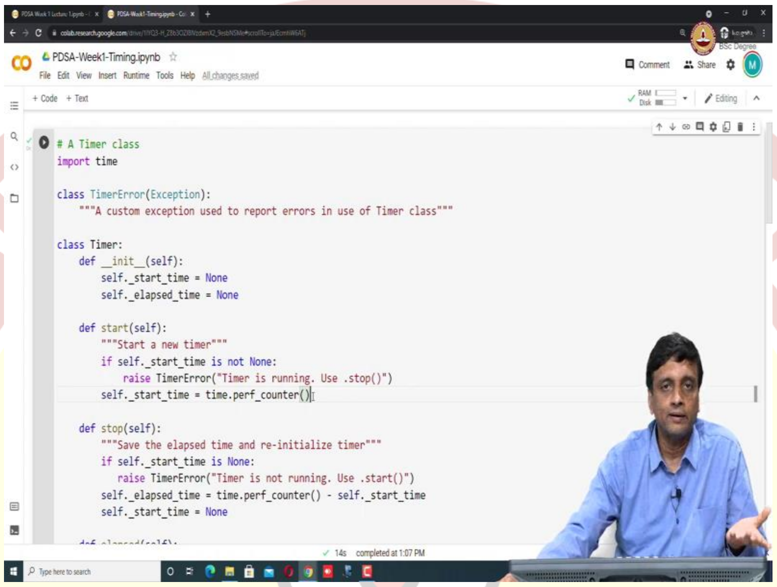
IIT Madras

ONLINE DEGREE

Programming, Data Structures and Algorithms using Python
Professor Madhavan Mukund
Implementation of Python Codes (Part 3)

We had created in the lecture a class which would represent the time that a piece of code takes, so a timer class.

(Refer Slide Time: 0:18)



```python
# A Timer class
import time

class TimerError(Exception):
    """A custom exception used to report errors in use of Timer class"""

class Timer:
    def __init__(self):
        self._start_time = None
        self._elapsed_time = None

    def start(self):
        """Start a new timer"""
        if self._start_time is not None:
            raise TimerError("Timer is running. Use .stop()")
        self._start_time = time.perf_counter()

    def stop(self):
        """Save the elapsed time and re-initialize timer"""
        if self._start_time is None:
            raise TimerError("Timer is not running. Use .start()")
        self._elapsed_time = time.perf_counter() - self._start_time
        self._start_time = None
```

So, here is a more elaborate version of the class. So, let us go through the code just to understand what this elaborate version is doing. So, first of all if you remember code can throw exceptions. So, what we are going to do is make this kind of this timer sort of little bit proof again except, misuse by saying that you must stop it only after you start it and so on.

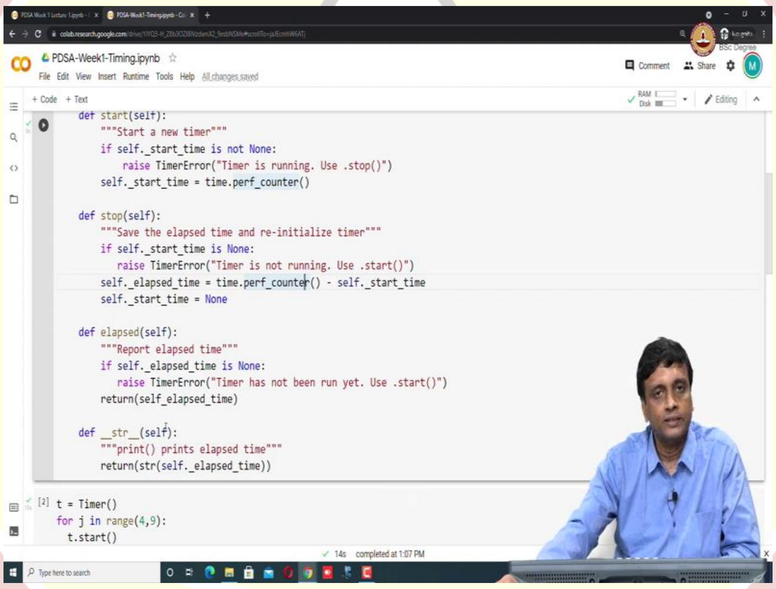
So, we will have a new error that we are creating and this is just a way of creating a new error. So, we say that timer error is a new class which is of type exception, it inherits from exception. So, just, so these quotes inside these double, triple quotes, these are what are called documentation string. So, they are the things that you get when you hover over a function.

So, let me see if we can see that. Let me just execute this and then if I go and for instance if I sometimes if I hover over something, I might see it. So, it is loading something, but anyway we will come back to that, but anyway. So, the main thing is now we have this timer error exception. So, when we start the timer instead of setting it to 0, we will put the start time and elapsed time to none to say that it is not set.

So, now when you start the timer, it must be a timer which is not running. So, that is the difference now. We can check that this timer is not running by saying is the start none or not, if the start time is not none, it means the time is already running, so we can raise an error saying timer is running, please use stop. So, do not restart the timer because it is already running, this is a mistake.

Otherwise, if there is no error, if you do not raise. This word ‘raise’ is basically to create an exception. So, if you do not raise an exception, you do what we had seen in the slide which is you set the start time to the current value of performance counter. Remember the current value performance counter has no independent meaning; it is just used as a reference to subtract later.

(Refer Slide Time: 2:36)



```
    def start(self):
        """Start a new timer"""
        if self._start_time is not None:
            raise TimerError("Timer is running. Use .stop()")
        self._start_time = time.perf_counter()

    def stop(self):
        """Save the elapsed time and re-initialize timer"""
        if self._start_time is None:
           raise TimerError("Timer is not running. Use .start()")
        self._elapsed_time = time.perf_counter() - self._start_time
        self._start_time = None

    def elapsed(self):
        """Report elapsed time"""
        if self._elapsed_time is None:
           raise TimerError("Timer has not been run yet. Use .start()")
        return(self_elapsed_time)

    def __str__(self):
        """print() prints elapsed time"""
        return(str(self._elapsed_time))
```

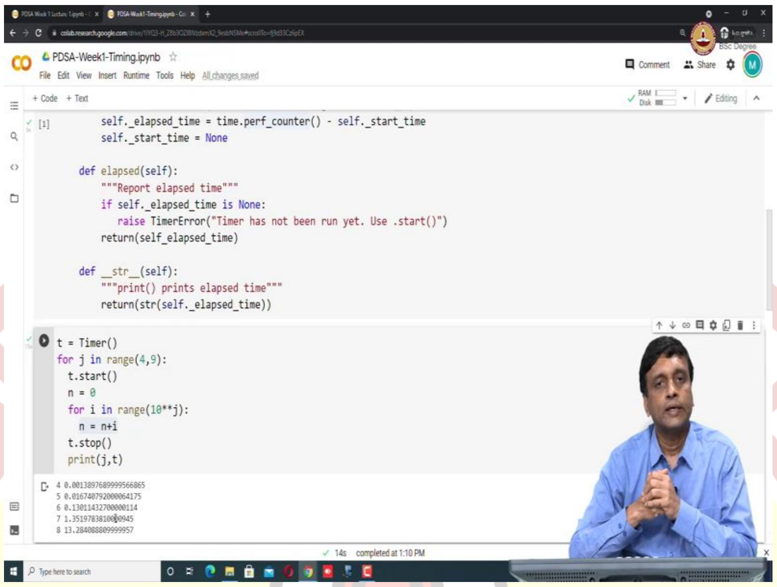
```
[2] t = Timer()
    for j in range(4,9):
      t.start()
```

14s completed at 1:07 PM

Similarly, when you stop the timer you check whether the timer is running. So, if the timer is start time is none, it means the timer is not running and you give an error saying timer is not running, please use start. Otherwise, you create, you set the elapsed time to the current value of the performance counter minus the start time and you reset the start time to none. So, now the timer can be run again because you have stopped it, but you have saved the time elapsed time.

So, you can look up the elapses time, but the timer is not running, and it is ready to run again. And finally, elapsed as we said, will check that actually that the elapses time is something meaningful, it is not the first time when you never set it. So, if the elapses time is not none, it will give you the elapses time, otherwise it will throw an error. And finally, we have if you just want to dump the value of the timer, then you get the elapsed time. So, that is the string function.

(Refer Slide Time: 3:25)



```
        self._elapsed_time = time.perf_counter() - self._start_time
        self._start_time = None

    def elapsed(self):
        """Report elapsed time"""
        if self._elapsed_time is None:
          raise TimerError("Timer has not been run yet. Use .start()")
        return(self_elapsed_time)

    def __str__(self):
        """print() prints elapsed time"""
        return(str(self._elapsed_time))
```

```
t = Timer()
for j in range(4,9):
  t.start()
  n = 0
  for i in range(10**j):
    n = n+i
  t.stop()
  print(j,t)
```

```
4 0.0013897689999566865
5 0.016740792000064175
6 0.13011432700000114
7 1.3519783810000945
8 13.284088809999957
```

So, now this is the piece of code that I wanted to show you. So, this is just a trivial piece of code. It is just a loop, which is running this n equal to n plus i for i ranging from 1 to some large numbers. So, what is that large number, I am doing it for 10 to the power j, where j ranges from 4 to 9. So, I am doing it for 10 to the power 4, which is 1 with four zeros.

So, it is 10,000 100,000 that is 10 to the power 5, 1 lakh, 10 to the power 6, and 10 to the power 7, and then 10 to the power 8 because 4 to 9 will be 4 5 6 7 8. So, it is running it for each of these values. And what are we doing before we start the loop, we are running, starting the timer and when we finish the loop, we are stopping the timer.

So, this is just a simple loop to tell us h ow long Python takes to execute something as trivial as n equal to n plus i that many times. So, if you run this, and you can see that the first time, the time when it takes 10,000 it runs in 0.01 it is probably difficult to see on the screen but I will share this code with you and you can run it, you can trust me to say when j equal to four the output of the timer is 0.001 that is one thousandth of a second.

When this is 10 to the power 5, it is 0.01 is one hundredth of a second, when it is 10 to the power 6 it is 0.13, it is about 0.1 seconds, when it is n to the power 7 and this is what I promised you in the lecture that 10 to the power 7 operations per second it takes 1.3 seconds and 10 to the power 8 it takes 13 seconds so you can see that there is an clear 10 factors slow down when I increase the number of operations by a factor of 10.

So, when I go from 1000 to 10,000 from 10,000 to 1 lakh, from one lakh to 10 lakhs, 10 lakhs to 1 crore each time, this loop takes 10 times as much. And it is useful to calibrate now our expectations when we run code, we will see later on that when we are trying to, for instance, if you are trying to sort a list or you are trying to do some other complicated operation on a large structure, then you might actually end up taking some 10 to the power some number of steps as we will see.

And it will be useful to estimate how long it is going to take without actually running the code, we have a kind of a clear idea that if it is going to be, if I have to traverse a list, for example, whose size is 10 to the power 7 is going to take me 1 second just to go from this end to that end because power 8 is going to be 10 seconds just to go from this end to that end without even doing anything.

So, we will come back to this later on. But it is useful to have this kind of timer to give us an idea about how long things actually take when they run because we can do a theoretical calculation but there is nothing like seeing it in kind of black and white to know exactly how long your code is taking.