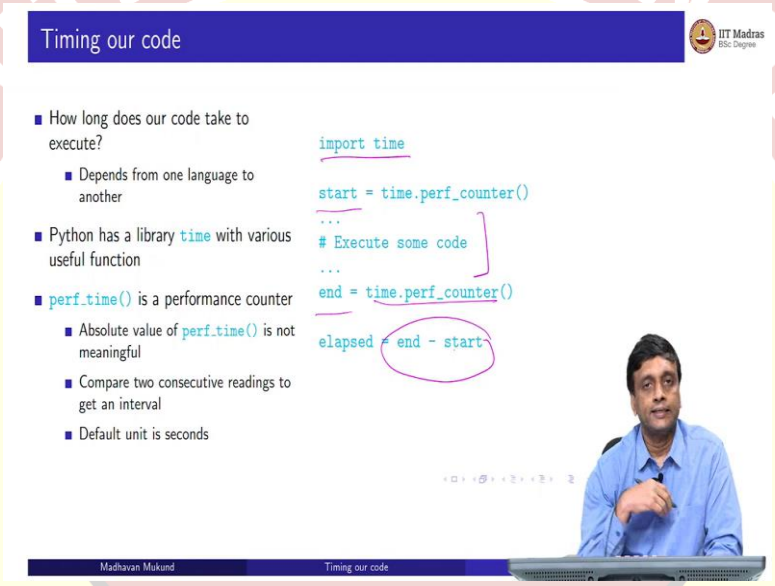
IIT Madras

ONLINE DEGREE

Programming, Data Structures and Algorithms using Python
Professor Madhavan Mukund
Timing our code

So, in this course, we are going to be focusing on efficiency. And one of the things that is important in efficiency is how long it takes code to run. So, we will be doing mostly theoretical calculations, but it is sometimes useful to do practical computations to understand how long Python actually takes to execute code. So, let us look at a way of measuring how long a piece of Python code takes to run in Python itself.

(Refer Slide Time: 0:33)



So in general, the running time of code will vary from one language to another. So, one way of measuring these things is from outside, you can take your operating system, and you can call a function or call a program and measure the time it takes for that program to run. But this will not be useful if you want to know how much part of the program.

Suppose you want to know how much time each function that you call inside the program takes, you might be sorting something in the middle of it, how long did the sorting, so then you actually need to embed this timing into your program. So, Python has a library called time, which gives us some useful functions for doing this.

So, one of them that we will use now is something called the performance counter, which is invoked by this function called perf time. So, perf time does not give us a useful value in itself. If I call this function perf time, it will give me some number; this number has no meaning in itself. But if I call it twice, if I call it now when I call it a little later, then I will get to values whose difference is meaningful.

So, if I take two consecutive readings of perf time, I will get a difference. So, this is like having a watch. So, if I have this watch, but the watch is not showing correct time, so maybe it is showing time in a different time zone, I have just travelled from some other country and I have not reset my watch.

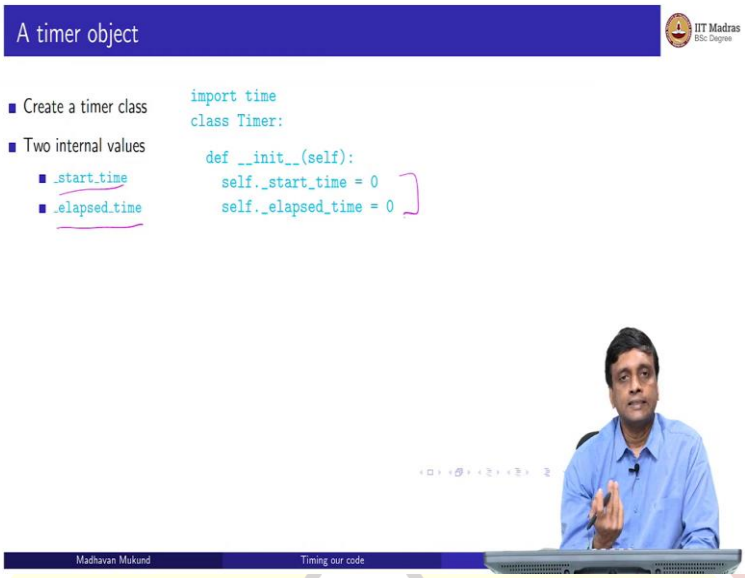
So, if I look at the watch, and I look outside, they will be mismatched, my watch may be showing 12 o'clock, it might be 6:30 outside. But if I look at it, 12 o'clock, and then I look at it five minutes later, and it shows 12:05, I know that 5 minutes have elapsed between the time I saw the watch the previous time, and that time I saw the watch now.

So, the absolute value that the time is showing on my watch is not useful. But the relative difference between the time I read now, so if I want to do something after 10 minutes or 15 minutes, I can measure that with my watch, My watch is not set correctly. So, perf time is like that. So, perf time you would think of as a clock that is not set correctly. But if you call it twice the difference will tell you the interval between the two calls.

And by default, this is in seconds. So, how would you use it? Well, typically, you will like any other library, you will import it. And now you want to measure the time that a certain piece of code takes. So, before you start this code, you call this perf counter function and store the return value in a variable say let us call it start after your code, you store it another variable, let us call it end.

And finally, this is the meaningful thing, you can compare the difference, you can compute the difference of n minus start. And this will actually tell you how much time elapsed. So, this is the way in which you can use a performance counter. So, we can now embed this call into a class and create a timer object.

(Refer Slide Time: 3:15)

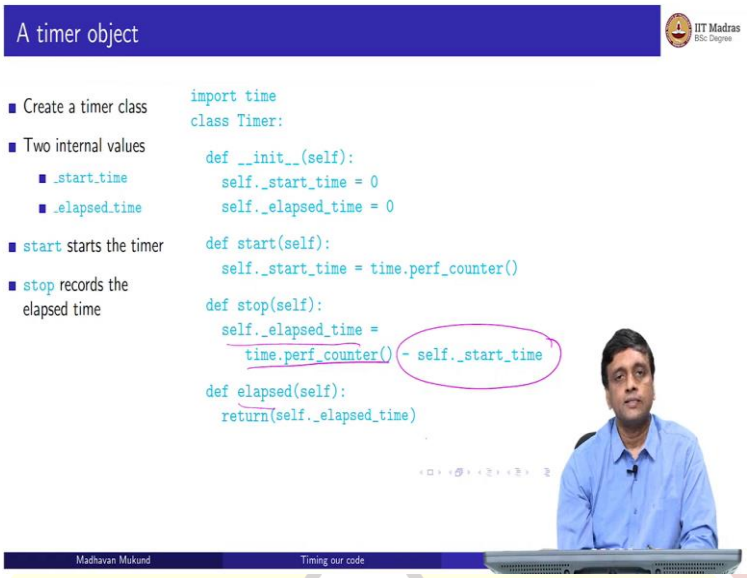


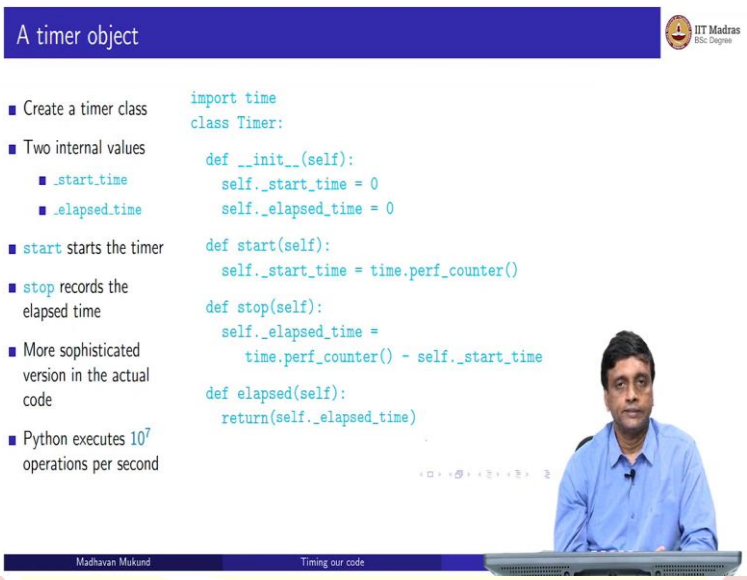
So, we want a timer object so that we do not have to explicitly call this perf counter ourselves, but we want to say start the timer stop the timer. So, this is like if you have a phone or if you have something else you can have a stopwatch you can say start and stop. And then you know how much time it is taken.

So, this is how we measure for instance performances and athletics when the person starts, you start the timer when the person finishes, you finish it press the timer, and you know how much time so we want to create a timer class. So, obviously, we have to import the time library to do that. And we will keep track of two things right when the timer started and how much time I measured the last time when I stopped it.

So, I will start and stop and I will have to. I would like to store the thing. So, when I start the clock, I remember when I started the first call to perf meter when I finish, I, am not interested anymore in when I started, when I finished. I am interested in the time that elapsed. So, I will store that. So, this is a simplistic version there is a more elaborate version, which we will look at when we look at the code, but initially let us just assume that when the timer is created, we set both of these values to 0. Now, the functions that I need are start and stop.

(Refer Slide Time: 4:23)





So, what will start do, so start will start the timer? So, it will basically call this performance counter in the time library and assign it to my start time. So, now my timer is running implicitly I have set the starting time. Now when I say stop, I call this performance timer again, it is not important to me what that value is as such, I do not need to store it, what I want is the difference between that and the time that I started and I will store that in this elapsed time thing.

So, that when I call this function elapsed, I will get the elapsed time right, so this gives us a timer class, I will show you separately a more elaborate version of this and how to use this. And with this, we can actually measure the time that Python takes to run and it will turn out that Python actually executes something like 10 to the power 7 operations in a second.

This is considerably slower than languages like C++ and C, which usually do ten to the eight or more, so it is at least, so pythons is at least a factor of 10 slower than other languages. It is sometimes it matters sometimes it does not matter. But this 10 to the 7 is useful just to calibrate for ourselves, how long things are going to take and to understand why things need to be done more efficiently in certain cases.