Example 1: Measuring Time Complexity

In this lesson, we are going to learn how to compute the running time complexity of an algorithm that involves loops.

We'll cover the following

In the previous lesson, we calculated the running time complexity of a very basic Python program. Lets now calculate the running time complexity of a more complex program. We will split the code into individual operations and then compute how many times each is executed.

Simple For Loop of Size `n`

Here is an example of a simple loop of size ***n***:

```
1  n = 10  # just as an example, n can be anything
2  sum = 0
3  for var in range(n):
4      sum += 1
5
6  print(sum)
7
```

Operation	Number of executions
`n = 10`	1
`sum = 0`	1
`range(n)`	1
`var=0`	1
`var=1`	1
`var=2`	1

Operation	Number of executions
...	
`var=n-1`	1
`sum+=1`	$3 \times n$
`print(sum)`	2

> Note that while `range(n)` executes only once, its execution cost is n. This is because it creates a list of values from 0 to n - 1.

minimize

Note that `range(10)` returns numbers from 0 to 9.

```
for i in range(10):
  print(i) # note that range n returns a list of values from 0 to 9
```

RUN

Output 0.691s

```
0
1
2
3
4
5
6
7
8
9
```

Running Time Complexity

After counting how many times each operation is executing, we will just add all of these counts to get the time complexity of this program.

$$\text{Time complexity} =$$

$$1 + 1 + \mathrm{n} + (1 + 1 + 1 + \ldots + 1) + 3\mathrm{n} + 2$$

$$\Rightarrow\ 2 + \mathrm{n} + \mathrm{n} + 3\mathrm{n} + 2$$

$$\Rightarrow 5n + 4$$

In the next lesson, we will look at another example of a program containing nested loops and compute its running time complexity.

Back

Comparing Algorithms

Next

Example 2: Measuring Time Complexity

Completed