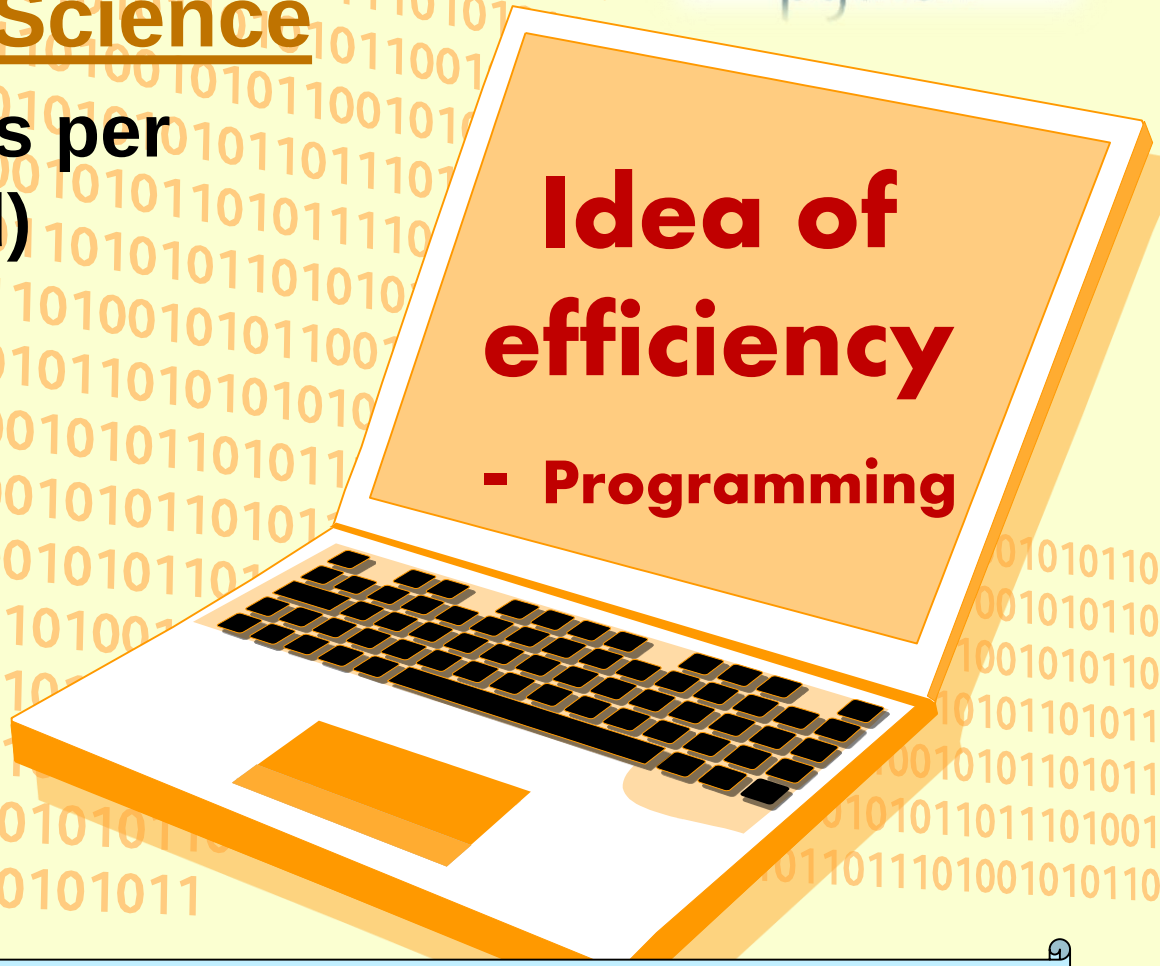
Chapter 6 :

Computer Science

Class XII (As per CBSE Board)



New Syllabus 2019-20

Visit : python.mykvs.in for regular updates

Idea of efficiency - Programming

Efficient programming is a manner of programming that, when the program is executed, it uses a low amount of overall resources pertaining to specially computer hardware. A program is designed by a human being, and different human beings may use different algorithms, or sequences of codes, to perform particular tasks, so the efficiency of such different programs varies, depend upon the number of resources being used. Practicing to create a low size(number of line of codes) and low resource algorithm results in an efficient program.

Idea of efficiency - Programming

Performance defined as inversely proportional to the wall clock time:-

- **Wall clock time/elapsed time: time to complete a task as seen by the user. In wall clock timing all kind of time is included ,e.g. operating system overhead or potentially interfering other applications etc.**

- **CPU time: does not include time slices introduced by external sources (e.g. running other applications).**

Idea of efficiency - Programming

Performance defined as inversely proportional to the wall clock time:-

To maximize performance, minimize execution time

performance = 1 / execution_timeX

“X is n times faster than Y”

– Execution time on Y is n times longer than on X

$$\frac{Performance_x}{Performance_y} = \frac{Executiontime_y}{Executiontime_x} = n$$

Idea of efficiency - Programming

Performance defined as inversely proportional to the wall clock time:-

E.g.

If a particular desktop runs a program in 60 seconds and a laptop runs the same program in 90 seconds, how much faster is the desktop than the laptop?

= Performancedesktop/ Performancelaptop

= (1/60)/(1/90) = 1.5

So, the desktop is 1.5 times faster than the laptop

Idea of efficiency - Programming

Count the number of operations a piece of code is performing:-

To compute the number of operations in a piece of code,then simply count the number of arithmetic operations+other operation that code is performing. All operations (addition, subtraction, multiplication, and division) are usually counted to be the same, which is not exactly true, since multiplication includes several additions and division includes several multiplications when actually executed by a computer. However, we are looking for an estimate here, so it is reasonable to assume that on average, all operations count in the same manner.

Here is an example (just for illustration):

```
r=0
for i in range(4):
    for n in range(4):
        r = r+(i*n)
print(r)
```

For each r there is 1 multiplications, 1 addition and 1 assignment resulting in 3 operations. This loop is executed 4X4 times, so there are (4X4)r operations. This is the the order of the code. In this example, its is $O(4^2r)$.

Idea of efficiency - Programming

Measure the time taken by a Python Program

To measure the script execution time is simply possible by using *time* built-in Python module. time() function is used to count the number of seconds elapsed since the epoch.

e.g.program

```
import time
start = time.time()
r=0
for i in range(400):
    for n in range(400):
        r = r+(i*n)
print(r)
end = time.time()
print(end - start)
```

OUTPUT

```
6368040000
0.12480020523071289 #TIME TAKE TO EXECUTE THE PYTHON SCRIPT
```

Idea of efficiency - Programming

Compare programs for time efficiency

With the help of time() function,we can compare two/more programs with different algo for same problem that which one take less time.Below two code scripts are for prime no time efficiency purpose.

```
import time
start = time.time()
a=int(input("Enter number: "))
k=0
for i in range(2,a//2+1):
    if(a%i==0):
        k=k+1
if(k<=0):
    print("Number is prime")
else:
    print("Number isn't prime")
end = time.time()
print(end - start)
```

OUTPUT

```
Enter number: 5
Number is prime
1.689096450805664
```

```
import time
start = time.time()
number = int(input("Enter any number: "))
if number > 1:
    for i in range(2, number):
        if (number % i) == 0:
            print(number, “Not a prime no")
            break
    else:
        print(number, "is a prime number")
else:
    print(number, "is not a prime number")
end = time.time()
print(end - start)
```

OUTPUT

```
Enter any number: 5
5 is a prime number
1.909109115600586
```