All Divisors of a Number

Given a natural number n, print all distinct divisors of it.

Examples:

```
Input : n = 10
Output: 1 2 5 10

Input:  n = 100
Output: 1 2 4 5 10 20 25 50 100

Input:  n = 125
Output: 1 5 25 125
```

A **Naive Solution** would be to iterate all the numbers from 1 to n, checking if that number divides n and printing it. Below is a program for the same:

Python

```
# Python implementation of Naive method
# to print all divisors

# method to print the divisors
def printDivisors(n) :
    i = 1
    while i <= n :
        if (n % i==0) :
            print (i,end=" ")
        i = i + 1

# Driver method
print ("The divisors of 100 are: ")
printDivisors(100)
```

« Prev
Next »

Output:

```
The divisors of 100 are:
1 2 4 5 10 20 25 50 100
```

Time Complexity : O(n)
Auxiliary Space : O(1)

Can we improve the above solution?
If we look carefully, all the divisors are present in pairs. For example if n = 100, then the various pairs of divisors are: (1,100), (2,50), (4,25), (5,20), (10,10)
Using this fact we could speed up our program significantly.
We, however, have to be careful if there are two equal divisors as in the case of (10, 10). In such case, we'd print only one of them.

Below is an implementation for the same:

Python

```python
 # A Better (than Naive) Solution to find all divisors
import math

# method to print the divisors
def printDivisors(n) :

    # Note that this loop runs till square root
    i = 1
    while i <= math.sqrt(n):

        if (n % i == 0) :

            # If divisors are equal, print only one
            if (n / i == i) :
                print (i,end=" ")
            else :
                # Otherwise print both
                print (i , int(n/i), end=" ")
        i = i + 1

# Driver method
print ("The divisors of 100 are: ")
printDivisors(100)
```

Output:

```
The divisors of 100 are:
1 100 2 50 4 25 5 20 10
```

Time Complexity: O(sqrt(n))
Auxiliary Space : O(1)