NPTEL Course: Programming, Data Structures and Algorithms in Python (*by* Prof. Madhvan Mukund)

Tutorial (Week 1)

Presented by: Jivesh Dixit

Problem 1: Listing all prime numbers below a given number ‘N’

Approach: Following the *Eratosthenes* algorithm,

1. for ‘*every number <=N’,*
2. if it is divisible by any numbers below from 2 to $\sqrt{N}$, it's not a *‘prime number’.*
3. Or else, it is a prime number.
4. We can either print directly all the numbers that turn out to be prime <= *N,* or we can also store it in an empty list and print the list.

```python
def prime_num(N):
  '''integers below or equal to one are out of scope
  for the definition of prime numbers.'''

  if(N<=1):
    return False

  #using Eratosthenes algorithm
  for i in range(2,int(N**(1/2))+1):
    #if divisible by i, then n is not a prime number.
    if(N%i==0):
      return False

  #otherwise, N is prime number.
  return True
```

```python
num = int(input("Input the value of N:"  ));
#checking for all number: 1 to num
for i in range(1,num+1):
  #if i is prime
  if(prime_num(i)):
    print(i)
    # print(i ,end=" ")
```

Problem 2: Calculating sum of cubes of the digits of a given number.

Approach: Following the algorithm,

1. First we define a function that calculates any exponent of a number.
2. Then we carefully only allow integers >=0, numbers below that or non-numerics must throw a warning and terminate the program.
3. Now, we proceed with calculating the last digit of the number using modulo function, add its cube (calculated using function in 1st point) to a variable having value ‘0’, then we calculate the greatest integer after dividing the number by 10.
4. We repeat the above-mentioned procedure until the cubic sum of all digit is calculated.

Problem 3: (Extension of Problem 2) Armstrong number

Armstrong number: *An ‘Armstrong number’ is the number if the sum of its digits raised to an exponent equal to the number of digits returns the number itself.*

Approach: Following the algorithm,

1. We carefully only allow integers >=0, numbers below that or non-numerics must throw a warning and terminate the program.
2. We define a function that calculates the number of digits in the number, let's call order hereafter.
3. Now, we proceed with calculating the last digit of the number using modulo function, add after raising it to the power (=) order to a variable having value ‘0’, then we calculate the greatest integer after dividing the number by 10.
4. We repeat the above-mentioned procedure until the desired sum of all digit is calculated.
5. We also check whether the number fits the definition of *‘Armstrong number’.*

Problem 4: Find number of zeros at the end of factorial of a number

<u>*Approach:*</u>

- One simple approach can be, calculate the factorial of the number, then count all the zeros at the end by keep dividing by 10 until modulo becomes non-zero. Count the number of divisions. But this method could be computationally costly for large numbers, given the definition of the factorial.

factorial(n) or n! = n x n-1 x n-2 x ………….. x 3 x 2x 1

- A more efficient approach can be, writing factorial in terms of prime factors, i.e. *prime factorization of 24 =* 2^3 *x 3, of 55 = 11 x 5, of 7! =* 2^4 *x* 3^2 *x 5 x 7* etc.
- Number of zeros in a factorial(n) will depend upon the exponent of *'5'* and *'2'* in its prime factorization.

1. However, intuitively, exponent of *'5'* will be smaller than the exponent of *'2'*. Hence number of zeros will be equal to the exponent of *'5'* in its prime factorization.
2. exponent of prime *'p'* in factorial of a number *'n'* is given by the formula:

exponent of prime 'p' = $[n / p] + [n / p^2] + [n / p^3] + [n / p^4] + \ldots\ldots.$

Number of zeros in factorial (n) = exponent of prime '5' = $[n / 5] + [n / 5^2] + [n / 5^3] + [n / 5^4] + \ldots\ldots.$

Problem 5: Number of days in February?

__Approach:__ Following the algorithm,

1. Common belief is that year that is divisible by 4 is a *'leap year'*. However it is not sufficient condition for the leap year. For example: 1900, 2100, 2200, divisible by 4 are not a leap year, but 2000 is.
2. To complete the sufficient condition for a *'leap year'*, for century years such that 100, 200, .., 1000, … 1800, .., 2000 etc., year should also be divisible by 400.
3. Hence, if year is not a century year *year%4 = 0,* and if year is a century year year%400 = 0, for the year to be a *'leap year'.*