Basic Maths for dsa

digit concept

example ⟶ 7789 # Extraction of digit?

every where we need concept ⟶ modulus ⟶ Remainder

7789 / 10 ⟶ (9) ⟶ # Remainder

Now we need ⟶ 778 ⟶ possible base of divide

$\frac{7789}{10}$ = 7789 Take int of 7789 ⟶ will get 778

$\frac{7789}{10} / 10 = 9$

$\frac{778}{10} / 10 = 8$ ⟶ Extracted all digit in reverse order

$\frac{77}{10} / 10 = 7$

7 / 10 = 7

psuedo code

problem statement

```
while n>0:
    lastdigit = n%10
    print(lastdigit)
    n = n / 10
```

Count digit ⟶ evenly divide n and leaves no remainder

353 353/3 ⟶ if zero than count+=1

digit ⟶ 3

Another way of counting digits

7789 ⟶ Take a log with base 10

So over here will get 3.899 something

now will add 1 in 3.899 than take int will get 4

More

whenever the iteration depends on division ⟶ logarithm comes into picture

⟶ Time complexity

Reverse of digit

4100 ⟶ Reverse ⟶ 0014 ✗

→ Output 14

reverse = (reverse x 10) + last digit

0 x 10 + 0 = 0

0 x 10 + 0 = 0

0 x 10 + 1 = 1

1 x 10 + 4 = 14

palindrom number

Original number and its reverse both are same ⟶ palindrome number

Example ⟶ 121 ⟶ 121

Both same called palindrome number

rev : 0

rev = (rev * 10) + digit

if N == rev:

return palindrome

Armstrong Number

Example ⟶ 371

Taking cube of each digit than sum it whatever result comes

if result == original number

than it is armstrong number

pseudo code

sum = 0

sum = digit * digit * digit

if sum == n:

return armstrong number

print all divisors

Example ⟶ 36 divisors which gives you remainder zero

what are those divisors? 1 2 3 4 6 9 12 18 36

Total 9 ⟶ divisors

Note

All divisors lies in 1 to n itself

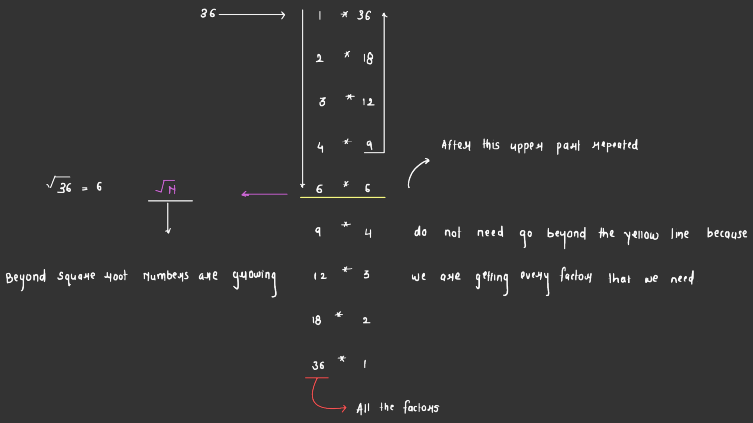
Pseudo Code

```
for i in range(n+1)
    if n % i == 0:
        print(i)
```

Time complexity for this ⟶ O(n)

To high

More Optimized way



pseudo code

```
for i in range(sqrt(n)):
    if n % i == 0:
        print(i)
        if (n/i) != i
            print(n/i)
```

Prime Number

Exactly has two factors 1 and itself

Example ⟶ $\frac{3}{1}$, $\frac{3}{3}$ ⟶ 1, 3

3 ⟶ prime number

Brute force approach

```
Count = 0

for i in range(n+1):          Time Complexity ⟶ O(n)

    if (n%i) == 0:

        count += 1

    if count == 2:

        print prime number
```

GCD and HCF

Greatest common divisor and Highest common factor

Example ⟶ $n_1 = 9$, $n_2 = 12$

1, 3, 9 1, 2, 3, 4, 6, 12

Greatest common divisor for 9, 12 ⟶ 3

$n_1 = 9$, $n_2 = 12$

Numbers between 1 to 12 if any number divide both

1, 3

pseudo code

```
for (i=1, i<=min(n1,n2), i++)

    if n1%i==0 and n2%i==0:

        gcd=i
```

Taking linear time complexity not optimised so will another approach

Euclidean Algorithm

n_1, n_2

$Gcd(n_1, n_2)$ ⟶ $Gcd(n_1-n_2, n_2)$

$Gcd(a, b)$ ⟶ $Gcd(a-b, b)$ $a > b$

15, 20

⟶ Euclidean Algorithm

$gcd(20, 15) = gcd(5, 15) = gcd(10, 5) = gcd(5, 5) = gcd(0, 5)$

5 = 5

Both are equal

As you can see one number became zero ⟶ Another number is our gcd

15, 20

⟶ Euclidean Algorithm

gcd(20,15) = gcd(5,15) = gcd(10,5) = gcd(5,5) = gcd(0,5)

Can we reach at final output by skiping

To make this possible can we say

gcd(a,b) = gcd(a%b, b)

psuedo code

```
while a>0 and b>0:
    if a>b:
        a = a%b
    else
        b = b%a
if a==0
    print a
else
    print b
```