JANUARY
10 '24
02nd Week
010-356
WEDNESDAY

Programs of Mathematics

S	M	T	W	T	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

JAN 2024

Leap year (or) Not

```
08 if (n%4==0 and n%100!=0 or n%400==0):
       print ('Yes')
09 else:
       Print ('No')
10
```

Reverse A No.

(a) `Print (str(num)[::-1])`

(b)
```
data = int(input())
n = int(i) for i in str(data)
n.reverse()
Print(n)
```

Add No. w/o '+' operator.

```
03 Print (abs(-a-b))
```

Sum of natural No.

(a)
```
n = int(input())
print(n*(n+1)/2)
```

(b)
```
def fn(n):
    if n==0:
        return 0
    return n+fn(n-1)
```

#4 Count No. of digits.

```
n = int(input())
05 rev = 0
   while n!=0:
06     d = n%10
       rev = rev*10 + d
07     n = n//10
   print(rev)
```

(b)
```
def fn(n):
    count = 0
    while n!=0:
        d = n%10
        count = count+1
        n = n//10
    return count.
```

(c) ✓
```
n = int(input())
res = 0
while n>0:
    n = n//10
    res = res+1
return res.
```

(d)
```
n = str(n)
return len(n)
```

Just count, No. of division operations happens. //

Terniary operator. # for maximum of 3 no.

```
def fn():
    return a if a>b and a>c else b if b>c else c.
```

FEB 2024

S	M	T	W	T	F	S
				1	2	3
4	5	6	7	8	9	10
11	12	13	14	15	16	17
18	19	20	21	22	23	24
25	26	27	28	29		

(for int)

```
def Pal(n):
    rev = 0
    temp = n
    while temp != 0:
        ld = temp % 10
        rev = rev * 10 + ld
        temp = temp // 10
    return rev == n
```


Palindrome Number

(a)
```
n = input()
l = list(n)
rev = list(reversed(l))
rev = ''.join(rev)
```

(b)
```
num = 1234
reverse = int(str(num)[::-1])
if num == reverse:
    print('yes')
else:
    print('no')
```

(c)
```
return n == n[::-1]
```

Factorial

(a)
```
import math as m
def f1(n):
    return m.factorial(n)
```

(b)
```
def f2(n):
    m = 1
    for i in range(1, n+1):
        m = m * i
    return m
```

(c)
```
def f3(n):
    if(n==0):
        return 1
    return n*f3(n-1)
```

Swap two No:—

(a)
```
def f1(a,b):
    t = a
    a = b
    b = t
    print('a{a} & b{b}')
a = int(input())
b = int(input())
f1(a,b)
```

(b)
```
def f2(a,b): # using add & sub
    a = a + b
    b = a - b
    a = a - b
```

(c)
```
def f3(a,b): # using mul & div
    a = a * b
    b = a // b
    a = a // b
```

(d)
```
def f5(a,b):
    a, b = b, a
```

(e)
```
def f4(a,b): # using bitwise
    a = a ^ b
    b = a ^ b
    a = a ^ b
```

JANUARY 12 '24 FRIDAY

02nd Week 012-354

LCM

{Naive Approach}

```
def lcm(a,b):
    res = max(a,b)
    while True:
        if res % a == 0 and res % b == 0:
            return res
        res += 1
    return res
```

(a)
```
import math as m
n1 = int(input())
n2 = int(input())
Hcf = m.gcd(n1, n2)
Lcm = int((n1 x n2) / Hcf)
print(Lcm)
```

HCF / GCD

(a)
```
import math as m
m.gcd(n1,n2)
```

(b)
```
def gcd(a,b):
    while a != b:
        if a > b:
            a = a - b
        else:
            b = b - a
    return a
print(gcd(a,b))
```

Euclidean Algorithms

(c)
```
def gcd(a,b):
    if b == 0:
        return a
    return gcd(b, a%b)
print(gcd(a,b))
```

(b) Whether Even (or) Odd

Every Even No ends with 0 (LSD) —
Every odd No ends with 1 (LSD) -

n&1 → 0 Even
n&1 → 1 odd

In binary
0 — 0000
3 — 0011
4 — 0100

```
def f(n):
    return (n&1) == 0
```

Reverse a No. —

To extract last digit, if you divide a number by 10, then the remainder will be last digit eg 123 % 10 → (3)

To reduce the number by one digit from end, simply divide the number by 10. eg 123 to 12 → 123/10 → 12 ✓

```
reverse = 0
while n != 0:
    digit = N % 10
    reverse = revese x 10 + digit
    N = N // 10
return (reverse)
```

A number which is divisible by one and itself.

Check Prime

```
def prime(n):
    if n <= 1:
        return False
    for i in range(2,n):
        if n%i == 0:
            return False
    return True
```

O(n)

(a) (2, int(math.sqrt(n)) + 1)

(b) (2, int(n/2) + 1)

Prime numbers in Given range:

```
l = []
for i in range(l,u+1):
    if i>1:
        for j in range(2, int(i**0.5)+1):
            if (i%j) == 0:
                break
        else:
            l.append(i)
print(l)
```

faces

O(n^2)

Sum of digits

(a)
```
n = input()
s = 0
for i in n:
    s = s + int(i)
print(s)
```

(b)
```
n = 12345
s = 0
while n! = 0:
    digit = int(n%10)
    s = s + digit
    n = n//10
print(s)
```

Efficient solution of LCM.

```
def gcd(a,b):
    if b == 0:
        return a
    return gcd(b, a%b)
def lcm(a,b):
    return a*b//gcd(a,b)
print(lcm(a,b))
```

Life is long if you know how to use it

formula → a*b = gcd(a*b) * lcm(a*b)

JANUARY
14
SUNDAY
'24
03rd Week
014-352

S	M	T	W	T	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

JAN 2024

Prime factorization.

```
def isPrime(n):
    for i in range(2,n):
        if n%i==0:
            return False
    return False

def printPfactors(n):
    for i in range(2,n+1):
        if isPrime(i):
            n=i
            while n%n==0
                print(i)
                n=n*i
n=int(input())
printPfactors(n)
```

No. of divisors

```
n=int(input())
l=[]
for i in range(1,n+1):
    if n%i==0:
        l.append(i)
print(len(l))
```

Perfect No.

```
n=int(input())
s=0
for i in range(1,n+1):
    if n%i==0:
        s=s+1
print('yes' if s==n else 'no')
```

Check whether the given digit is present in num (or) not.

(a)
```
def f1(n,key):
    while n!=0:
        d=n%10
        if key==d:
            return True
        n=n//10
    else: return False
```

(b)
```
def f2(n,key)
    return(str(key) in
           str(n))
```

Count No. of digit.

M-4

```
import math as m
def Cont(n):
    return (int(m.log10(n)+1))
```

Cont(n) # 5438 , o/p = 4 ✓

177715 , o/p = 6 ✓

eg $\log_{10}(5438) = 3.745$

$\log_{10}(177715) = 5.249$

$\log_{10}(973) = 2.988$

$3.745 + 1 = 4.745 \Rightarrow 4$

$5.249 + 1 = 6.249 \Rightarrow 6$

Check Palindrom No.

```
def Pal(n):
    rev = 0
    temp = n.
    while n != 0:
        d = n%10
        rev = rev*10 + d
        n = n//10
    return rev == n
Print(Pal(n))
n = int(input())
```

→ TC = $O(\log_{10}(n))$

→ SC = $O(1)$

We are taking a new variable because if we take the original one then that data will already change, we will always get wrong Result !

Check Armstrong No.

```
n = int(input())
num = n
total = 0
nod = len(str(n))
while num > 0:
    id = num%10
    total = total + (id ** nod)
    num = num//10
return total == n.
```

→ Here we can use (a) log method
(b) while loop method.

→ TC = $O(\log_{10}(N))$

SC = $O(1)$

All factors of a Give No. / All Divisors.

M-1

```
n = int(input())
l = []
for i in range(1, n+1):
    if n%i == 0:
        l.append(i)
print(l)
```

TC = O(n)
SC = O(k) → k would be total No. of factors.

This is because that for loop won't include last No.

M-2

eg 10

1, 2, 3, 4, 5 / 6, 7, 8, 9, 10
✓ ✓ x x ✓ / x x x x ✓

means जहां तक ले जाओ वो No. के/

eg 20

1, 2, 3, 4, 5, 6, 7, 8, 9, 10 / 11, 12, 13, 14, 15, 16, 17, 18, 19, 20
✓ ✓ x ✓ ✓ x x x x ✓ / x x x x x x x x x ✓

```
n = int(input())
l = []
for i in range(1, n//2+1):
    if n%i == 0:
        l.append(i)
l.append(n)
print(l)
```

TC = O(n)
SC = O(k)

Optimal Solution

eg 36.

① 36/1 = 36, 0	1, 36
② 36/2 = 18	2, 18
③, 36/3 = 12	3, 12
4, 36/4 = 9	4, 9
5, 36/5 = x	x
6, 36/6 = 6	6, 6

```
n = int(input())
import math as m
l = []
for i in range(1, int(m.sqrt(n))+1):
    if n%i == 0:
        l.append(i)
        if n//i != i:
            l.append(n//i)
print(sorted(l))
```

⇒ TC = O(n lg n)

No. of trailing zeros

```
import math
n= int(input())
f= math.factorial(n)
c= 0
while f!=0 # f>0
    if f%10!=0:
        break
    c= c+1
    f= f//10
return c
```

0 नहीं आया, तो आगे बढ़ो।

Armstrong No.

```
def fun(n):
    s=0
    while n!=0:
        d= n%10
        s= s+d**3
        n= n//10
    return s
```

Strong No.

153 = 5! + 1! + 3! = 153 = false

```
import math as m
def fun(n):
    s=0
    while (n!=0):
        d= n%10
        s= s+math.factorial(d)
        n=n//10
    return s
```

Find x to power of y

(a)
```
import math
def f1(x,y):
    m=1
    for i in range(y):
        m=m*x
    return m
```

(b)
```
def f2(x,y)
    return int(math.pow(x,y))
```

(c)
```
def f3(x,y)
    return x**y
```

(d)
```
def f4(x,y)
    pow(x,y)
```

Trailing Zeros

```
import math as m
n= int(input())
f= m.factorial(i)
t= f
c=0
while t!=0:
    if t%10!=0:
        break
    c= c+1
    t= t//10
print(c)
```

JANUARY
16
TUESDAY
'24
03rd Week
016-350

S	M	T	W	T	F	S
	1	2	3	4	5	6
7	8	9	10	11	12	13
14	15	16	17	18	19	20
21	22	23	24	25	26	27
28	29	30	31			

Fibonacci Number

(a)
```
def fib(n):
    if n in [0,1]:
        return n
    else:
        return fib(n-1) + fib(n-2)
```

(b)
```
def fun(n):
    l = []
    a = -1
    b = +1
    for i in range(n):
        c = a+b
        l.append(c)
        a = b
        b = c
    return l.
```

Computing power

(a)
```
def power(num):
    if n==0:
        return 1
    temp = power(x, n//2)
    temp = temp*temp
    if (n%2==0):
        return temp
    else:
        return temp*x
```

(b)
```
def Power(num):
    res = 1
    while n>0:
        if n%2!=0:
            res = res*n
        n = n*n
        n = n//2
    return res
```

Sieve of Eratosthenes

→ We have to find all the prime No: upto that No:

eg 1 - 20

multiple of : 2: 4, 6, 8, 10, 12, 14, 16, 18, 20
: 3: 6, 9, 12, 15
: 5: 10, 15, 20

TC = O(log(n))

This method works well when is relatively small, allowing us to determine whether any natural no. less than (or) equal to is prime (or) composite.

```
def Sieve(n):
    if n <= 1:
        return
    is_prime = [True] * (n+1)
    i = 2
    while i*i <= n:          # i <= n
        if is_prime[i]:      # Print(i, end = " ")
            for j in range(2*i, n+1, i):   # for j in range(i*i, n+1, i)
                is_prime[j] = False
    i = i+1  # i += 1
    for i in range(2, n+1):
        if is_prime[i]:
            Print(i, end = " ")
```

} you may omit this as well.

```
Sieve(n)
n = 18
```

Reverse a list

(a) l.reverse()

(b) nl = l[::-1]

(c) nl = list(reversed(l))

(d) → two pointer approaches

```
l = eval(input())
lindex = 0
rindex = len(l) - 1
while (lindex < rindex):
    l[lindex], l[rindex] = l[rindex], l[lindex]
    lindex += 1
    rindex -= 1
print(l)
```