04 - Iteration Control Structures

For example:

Input	Result
20	1 2 4 5 10 20

Ex. No. : 4.1 **Date:**

Register No.: **Name:**

Factors of a number

Determine the factors of a number (i.e., all positive integer values that evenly divide into a number).

```
n=int(input())
        for i in range(1,n+1):
                if n%i==0:
 print(i,end=" ")
```

For example:

Input	Result
292	1
1015	2
108	3
22	0

Ex. No. : 4.2 **Date:**

Register No.: **Name:**

Non Repeated Digit Count

Write a program to find the count of non-repeated digits in a given number N. The number will be passed to the program as an input of type int.
Assumption: The input number will be a positive integer number >= 1 and <= 25000.
Some examples are as below.
If the given number is 292, the program should return 1 because there is only 1 non--repeated digit '9' in this number
If the given number is 1015, the program should return 2 because there are 2 non-repeated digits in this number, '0', and '5'.
If the given number is 108, the program should return 3 because there are 3 non-repeated digits in this number, '1', '0', and '8'.
If the given number is 22, the function should return 0 because there are NO non--repeated digits in this number.

```
n=input()
        m={}
        for i in n :
                if i in m :
                        m[i]+=1
        p=0
        for i in m.values():
                if i==1:
                        p=p+1
         print(p)
```

Example1: if the given number N is 7, the method must return 2

Example2: if the given number N is 10, the method must return 1

For example:

Input	Result
7	2
10	1

Ex. No. : 4.3 **Date:**

Register No.: **Name:**

Prime Checking

Write a program that finds whether the given number N is Prime or not. If the number is prime, the program should return 2 else it must return 1.

Assumption: 2 <= N <=5000, where N is the given number.

```
def is_prime(N):
    if N <= 1:
        return 1
    for i in range(2, int(N**0.5) + 1):
        if N % i == 0:
            return 1
    return 2
N = int(input("Enter a number between 2 and 5000: "))
print(is_prime(N))
```

Input Format:

Integer input from stdin.

Output Format:

Perfect square greater than N.

Example Input:

10

Output:

16

Ex. No. : 4.4 **Date:**

Register No.: **Name:**

Next Perfect Square

Given a number N, find the next perfect square greater than N.

```
import math


def next_perfect_square(N):
    root = math.sqrt(N)
    next_int = math.ceil(root)
    next_perfect_square = next_int ** 2
    return next_perfect_square
N = int(input("Enter a number: "))
print(next_perfect_square(N))
```

NOTE: Fibonacci series looks like –

0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, . . . and so on.

i.e. Fibonacci series starts with 0 and 1, and continues generating the next number as the sum of the previous two numbers.

- first Fibonacci number is 0,
- second Fibonacci number is 1,
- third Fibonacci number is 1,
- fourth Fibonacci number is 2,
- fifth Fibonacci number is 3,
- sixth Fibonacci number is 5,
- seventh Fibonacci number is 8, and so on.

For example:

Input:

7

Output

8

Ex. No. : 4.5 **Date:**

Register No.: **Name:**

Nth Fibonacci

Write a program to return the nth number in the fibonacci series. The value of N will be passed to the program as input.

```
def fibonacci(n):
    if n <= 0:
        return "Invalid input. Please enter a positive integer."
    elif n == 1:
        return 0
    elif n == 2:
        return 1

    a, b = 0, 1
    for _ in range(2, n):
        a, b = b, a + b
    return b
N = int(input("Enter the value of N: "))
print(f"The {N}th number in the Fibonacci series is: {fibonacci(N)}")
```

Input Format:

Single Integer Input from stdin.

Output Format:

Yes or No.

Example Input:

175

Output:

Yes

Explanation

1^1 + 7^2 +5^3 = 175

Example Input:

123

Output:

No

For example:

Input	Result
175	Yes
123	No

Ex. No. : 4.6 **Date:**

Register No.: **Name:**

Disarium Number

A Number is said to be Disarium number when the sum of its digit raised to the power of their respective positions becomes equal to the number itself. Write a program to print number is Disarium or not.

```
num = int(input("Enter a number: "))
num_str = str(num)
length = len(num_str)
sum_of_powers = 0
for i in range(length):
    digit = int(num_str[i])
    sum_of_powers += digit ** (i + 1)
if sum_of_powers == num:
    print(f"{num} is a Disarium number.")
else:
    print(f"{num} is not a Disarium number.")
```

Sample Test Cases

Test Case 1

Input

4

Output

1234

Explanation:

as input is 4, have to take 4 terms.

1 + 11 + 111 + 1111

Test Case 2

Input

6

Output

123456

For example:

Input	Result
3	123

Ex. No. : 4.7 **Date:**

Register No.: **Name:**

Sum of Series

Write a program to find the sum of the series 1 +11 + 111 + 1111 + . . . + n terms (n will be given as input from the user and sum will be the output)

```
n=int(input())

sum = 1

 C = 1

for i in range (1, n) :

        sum =sum * 10+1

         C = C + sum

print(C)
```

For example:

Input	Result
292	2
1015	3

Ex. No. : 4.8 **Date:**

Register No.: **Name:**

Unique Digit Count

Write a program to find the count of unique digits in a given number N. The number will be passed to the program as an input of type int.
Assumption: The input number will be a positive integer number >= 1 and <= 25000.
For e.g.
If the given number is 292, the program should return 2 because there are only 2 unique digits '2' and '9' in this number
If the given number is 1015, the program should return 3 because there are 3 unique digits in this number, '1', '0', and '5'.

```
n=int(input())

a =[ ]

while n >0:

        if n%10 not in a:

                a.append(n%10)

                n =n//10

print(len(a))
```

Input Format:

Single Integer input.

Output Format:

Output displays Yes if condition satisfies else prints No.

Example Input:

14

Output:

Yes

Example Input:

13

Output:

No

Ex. No. : 4.9 **Date:**

Register No.: **Name:**

Product of single digit

Given a positive integer N, check whether it can be represented as a product of single digit numbers.

```
n =int(input())

if( n%2 ==0 or n%3 ==0 or n%5 ==0 or n%7 ==0) :

        print (" Yes ")

else:

        print (" No ")
```

Input Format:

Single integer input.

Output Format:

Yes or No.

Example Input:

24

Output:

Yes

Example Input:

26

Output:

No

For example:

Input	Result
24	Yes

Ex. No. : 4.10 **Date:**

Register No.: **Name:**

Perfect Square After adding One

Given an integer N, check whether N the given number can be made a perfect square after adding 1 to it.

```
n=int(input())+1
a = 0
if(n==0 or n ==1) :
        a = 1
        for i in range ( 2 ,(n//2)) :
if (n==i*i) :
        a = 1
        break
if a ==1 :
        print("Yes")
else:
        print (" No ")
```