
Factorial: Calculate the factorial of a given number using a loop.

{ A factorial is a mathematical operation that you write like this: n! . It represents the multiplication of all numbers between 1 and n.
So if you were to have 3! , for example, you'd compute 3 x 2 x 1 (which = 6) }

```
def factorial(n):
    result = 1
    for i in range(1, n+1):
        result *= i
    return result
```

Reverse a String: Reverse a given string using a loop.

```
def reverse_string(s):
    reversed_s = ""
    for char in s:
        reversed_s = char + reversed_s
    return reversed_s
```

Palindrome Check: Write a function to check if a given string is a palindrome (reads the same forwards and backwards).

{ A palindromic number is a number (such as 16461) that remains the same when its digits are reversed. }

```
def is_palindrome(s):
    return s == s[::-1]
```

Fibonacci Series: Generate the first 10 numbers in the Fibonacci series.

{ Fibonacci sequence is a sequence in which each number is the sum of the two preceding ones. The sequence commonly starts from 0 and 1 . The first few values in the sequence are:
0, 1, 1, 2, 3, 5, 8, 13, 21, 34, 55, 89, 144. }

```
a, b = 0, 1

for _ in range(10):

    print(a, end=" ")

    a, b = b, a + b
```

Prime Numbers: Print all prime numbers between 1 and 50.

{ A prime number is a whole number greater than 1 whose only factors are 1 and itself. The first few prime numbers are 2, 3, 5, 7, 11, 13, 17, 19..... }

```
def is_prime(n):

    if n ⩽ 1:

        return False

    for i in range(2, int(n**0.5) + 1):

        if n % i == 0:

            return False

    return True


primes = [num for num in range(2, 51) if is_prime(num)]
```

Multiplication Table: Print the multiplication table of a given number (e.g., 7).

```
num = 7

for i in range(1, 11):

    print(f"{num} x {i} = {num*i}")
```

Remove Duplicates: Remove duplicates from a list while preserving the order of elements.

```
def remove_duplicates(lst):
    unique_lst = []
    for item in lst:
        if item not in unique_lst:
            unique_lst.append(item)
    return unique_lst
```

Check for Anagrams: Write a function to check if two strings are anagrams of each other.

{ An anagram is a word or phrase formed by rearranging the letters of a different word or phrase, typically using all the original letters exactly once. For example, the word *anagram* itself can be rearranged into *nag a ram* }

```
def are_anagrams(str1, str2):
    return sorted(str1) == sorted(str2)
```

Find Maximum Element: Find the maximum element in a list without using the **max** function.

```
def find_max_element(lst):
    max_element = lst[0]
    for item in lst:
        if item > max_element:
            max_element = item
    return max_element
```

Check for Armstrong Number: Write a program to check if a given number is an Armstrong number

{ A number is thought of as an Armstrong number if the sum of its own digits raised to the power number of digits gives the number itself. For example, 0, 1, 153, 370, 371, 407 are three-digit Armstrong numbers and, 1634, 8208, 9474 are four-digit Armstrong numbers and there are many more.A number is thought of as an Armstrong number if the sum of its own digits raised to the power number of digits gives the number itself. For example, 0, 1, 153, 370, 371, 407 are three-digit Armstrong numbers and, 1634, 8208, 9474 are four-digit Armstrong numbers and there are many more. }

```
def is_armstrong(n):
    num = n
    num_of_digits = len(str(n))
    total = 0
    while n > 0:
        digit = n % 10
        total += digit ** num_of_digits
        n //= 10
    return total == num
```

Print Pascal's Triangle: Print the first n rows of Pascal's triangle.

```
def generate_pascals_triangle(n):
    triangle = []
    for i in range(n):
        row = [1] * (i + 1)
        if i > 1:
            for j in range(1, i):
                row[j] = triangle[i-1][j-1] + triangle[i-1][j]
        triangle.append(row)
    return triangle
```

Check for Palindrome Number: Write a program to check if a given number is a palindrome.

```
def is_palindrome_number(n):
    return str(n) == str(n)[::-1]
```

Check for Leap Year: Write a program to check if a given year is a leap year.

```
def is_leap_year(year):
    return (year % 4 == 0 and year % 100 != 0) or (year % 400 == 0)
```

Find the Nth Prime Number: Write a function to find the Nth prime number.

```
def nth_prime(n):
    primes = []
    num = 2
    while len(primes) < n:
        is_prime = True
        for p in primes:
            if num % p == 0:
                is_prime = False
                break
        if is_prime:
            primes.append(num)
        num += 1
    return primes[-1]
```