Name- Bhavya Mahawar
batch -80
Sapid-590029516

1. Find a factorial of given number.

```
def factorial(n):
    if n < 0:
        return "Factorial is not defined for negative numbers"
    elif n == 0:
        return 1
    else:
        fact = 1
        for i in range(1, n + 1):
            fact = fact * i
        return
number = 5
print(f"The factorial of {number} is {factorial(number)}")

number = 0
print(f"The factorial of {number} is {factorial(number)}")

number = -3
print(f"The factorial of {number} is {factorial(number)}")
```

2. Find whether the given number is Armstrong number.

```
def is_armstrong_number(number):
    str_number = str(number)
    num_digits = len(str_number)

    sum_of_powers = 0
    for digit in str_number:
        sum_of_powers += int(digit) ** num_digits

    if sum_of_powers == number:
        return True
    else:
        return False

# Example :
print(f"Is 15:3 an Armstrong number? {is_armstrong_number(153)}") # Expected: True
print(f"Is 9 an Armstrong number? {is_armstrong_number(9)}")    # Expected: True (single digit numbers are Armstrong numbers)
print(f"Is 370 an Armstrong number? {is_armstrong_number(370)}") # Expected: True
print(f"Is 371 an Armstrong number? {is_armstrong_number(371)}") # Expected: True
print(f"Is 407 an Armstrong number? {is_armstrong_number(407)}") # Expected: True
print(f"Is 1634 an Armstrong number? {is_armstrong_number(1634)}") # Expected: True
print(f"Is 123 an Armstrong number? {is_armstrong_number(123)}") # Expected: False
```

3. Print Fibonacci series up to given term.

```
def fibonacci_series(n_terms):
    a, b = 0, 1
    count = 0
    series = []

    if n_terms <= 0:
        return "Please enter a positive integer"
    elif n_terms == 1:
        series.append(a)
        return series
    else:
        while count < n_terms:
            series.append(a)
            nth = a + b

            a = b
            b = nth
            count += 1
        return series
#examples
terms = 10
print(f"Fibonacci series up to {terms} terms: {fibonacci_series(terms)}")

terms = 1
print(f"Fibonacci series up to {terms} term: {fibonacci_series(terms)}")
print(f"Fibonacci series up to {terms} term: {fibonacci_series(terms)}")

terms = 0
print(f"Fibonacci series up to {terms} terms: {fibonacci_series(terms)}")
```

4. Write a program to find if given number is prime number or not.

```
def is_prime(number):
    if number <= 1:
        return False
    for i in range(2, int(number**0.5) + 1):
        if number % i == 0:
            return False
    return True

# Example usage:
print(f"Is 17 a prime number? {is_prime(17)}")
print(f"Is 4 a prime number? {is_prime(4)}")
print(f"Is 2 a prime number? {is_prime(2)}")
print(f"Is 1 a prime number? {is_prime(1)}")
print(f"Is 29 a prime number? {is_prime(29)}")
print(f"Is 100 a prime number? {is_prime(100)}")
```

5. Check whether given number is palindrome or not.

```
def is_palindrome(number):
    str_number = str(number)
    if str_number == str_number[::-1]:
        return True
    else:
        return False

# Example :
print(f"Is 121 a palindrome? {is_palindrome(121)}")
print(f"Is 12321 a palindrome? {is_palindrome(12321)}")
print(f"Is 12345 a palindrome? {is_palindrome(12345)}")
print(f"Is 1001 a palindrome? {is_palindrome(1001)}")
print(f"Is -121 a palindrome? {is_palindrome(-121)}")
```

6. Write a program to print sum of digits

```
def sum_of_digits(number):

    number = abs(number)

    sum_digits = 0

    for digit in str(number):
        sum_digits += int(digit)
    return sum_digits

# Example usage:
print(f"The sum of digits for 123 is: {sum_of_digits(123)}")
print(f"The sum of digits for 4567 is: {sum_of_digits(4567)}")
print(f"The sum of digits for 0 is: {sum_of_digits(0)}")
print(f"The sum of digits for 9 is: {sum_of_digits(9)}")
print(f"The sum of digits for 10 is: {sum_of_digits(10)}")
print(f"The sum of digits for -123 is: {sum_of_digits(-123)}")
```

7. Count and print all numbers divisible by 5 or 7 between 1 to 100.

```
divisible_numbers = []
count = 0

for number in range(1, 101):
    if number % 5 == 0 or number % 7 == 0:
        divisible_numbers.append(number)
        count += 1

print(f"Numbers divisible by 5 or 7 between 1 and 100 are: {divisible_numbers}")
print(f"Total count of such numbers: {count}")
```

8. Convert all lower cases to upper case in a string.

```
def convert_to_uppercase(input_string):
    return input_string.upper()

# Example usage:
string1 = "Hello World"
print(f"Original string: '{string1}' -> Uppercase: '{convert_to_uppercase(string1)}'")

string2 = "python programming"
print(f"Original string: '{string2}' -> Uppercase: '{convert_to_uppercase(string2)}'")

string3 = "MiXeD CaSe 123!"
print(f"Original string: '{string3}' -> Uppercase: '{convert_to_uppercase(string3)}'")

string4 = "ALL CAPS"
print(f"Original string: '{string4}' -> Uppercase: '{convert_to_uppercase(string4)}'")
```

9. Print the table for a given number: 5 * 1 = 5 5 * 2 = 10...........

```
def print_multiplication_table(number, limit=10):
    print(f"Multiplication Table for {number}:")
    for i in range(1, limit + 1):
        print(f"{number} * {i} = {number * i}")

# Example usage:
print_multiplication_table(5)
print("\n")
print_multiplication_table(7, limit=12)
print("\n")
print_multiplication_table(3, limit=5)
```

10. Write a program to print the following pattern 123454321 1234 *4321 123 * * 321 12 * * * 21 1 * * * * 1

```
def print_pattern(N):
    for row in range(1, N + 1):
        line = ""
        for j in range(1, N - row + 2):
            line += str(j)
        if row > 1:
            line += " " * (row - 1)
            middle_stars_part = ""
            for k in range(row - 1):
                if k > 0:
                    middle_stars_part += " "
                middle_stars_part += "*"
            line += middle_stars_part


        start_num_right = N - row + 1
        if row == 1:
            start_num_right = N - 1

        for j in range(start_num_right, 0, -1):
            line += str(j)

        print(line)


print_pattern(5)
```

11. Write a program to print the sum of the following series 1+ ½ + 1/3 + ¼ +....+1/n

```
def sum_harmonic_series(n):
    if n <= 0:
        return "Please enter a positive integer for n"

    series_sum = 0
    for i in range(1, n + 1):
        series_sum += 1 / i
    return series_sum
```

```
# Example usage:
print(f"Sum of series up to n=1: {sum_harmonic_series(1)}")
print(f"Sum of series up to n=2: {sum_harmonic_series(2)}")
print(f"Sum of series up to n=5: {sum_harmonic_series(5)}")
print(f"Sum of series up to n=10: {sum_harmonic_series(10)}")
print(f"Sum of series up to n=0: {sum_harmonic_series(0)}")
```

Double-click (or enter) to edit

github repo link-https://github.com/mahawarbhavya/pythonexperiments.git