UNIT-3

Create a pair of functions to convert Fahrenheit to Celsius temperature values and vice versa. Where C = (F - 32) * (5 / 9)

In [1]:
```
def fahrenheit_to_celsius(fahrenheit):
    return (fahrenheit - 32) * (5/9)

def celsius_to_fahrenheit(celsius):
    return (celsius * (9/5)) + 32
print("fahrenheit_to_celsius",fahrenheit_to_celsius(98.3))
print("celsius_to_fahrenheit",celsius_to_fahrenheit(36))
```

```
fahrenheit_to_celsius 36.833333333333336
celsius_to_fahrenheit 96.8
```

Write a Python function to calculate the factorial of a given number.

In [2]:
```
def factorial(n):
    if n == 0 or n == 1:
        return 1
    else:
        return n * factorial(n - 1)
n=int(input("enter number: "))
print(f"factorial of {n}: {factorial(n)} ")
```

```
enter number: 5
factorial of 5: 120
```

Write a Python function to check whether a number is in a given range.

In [4]:
```
def check_in_range(num, start, end):
    return start <= num <= end
n=int(input("enter number: "))
start=int(input("enter number: "))
end=int(input("enter number: "))
if check_in_range(n,start,end):
    print("in range")
else:
    print("not in range")
```

```
enter number: 7
enter number: 1
enter number: 6
not in range
```

Write a Python function to display the Fibonacci sequence till the given user input

n.

In [5]:
```
def fibonacci_sequence(n):
    a, b = 0, 1
    print("Fibonacci sequence:")
    while a < n:
        print(a, end=" ")
        a, b = b, a + b

# Taking user input
num = int(input("Enter a number to generate Fibonacci sequence up to that number: "))
fibonacci_sequence(num)
```

```
Enter a number to generate Fibonacci sequence up to that number: 5
Fibonacci sequence:
0 1 1 2 3
```

Write a Python function to find the Max of TWO numbers.

In [6]:
```
def find_max(a, b):
    if a>b:
        return a
    else:
        return b
print(f"max number is {find_max(5,7)}")
```

```
max number is 7
```

Write a Python program to accept two numbers and check it for odd or even number

In [9]:
```
def odd_or_even(num):
    if num % 2 == 0:
        return "Even"
    else:
        return "Odd"
n=int(input("enter number"))
print("number is "+ odd_or_even(n))
```

```
enter number7
number is Odd
```

Write a Python program to check whether the given no is Armstrong or not using user defined function.

In [10]:
```
def is_armstrong(num):
    order = len(str(num))
    temp = num
    sum = 0
    while temp > 0:
        digit = temp % 10
        sum += digit ** order
```

```
        temp //= 10
    return num == sum
if is_armstrong(153):
    print("number is armstrong")
else:
    print("number is armstrong")
```

```
number is armstrong
```

Write a python program to demonstarte a function to print the number 1 to 5.

In [11]:

```
def print_numbers(n):
    for i in range(1, n+1):
        print(i)
n=int(input("enter number: "))
print("print number",print_numbers(n))
```

```
enter number: 5
1
2
3
4
5
print number None
```

Write a Python Program to demonstarte a Simple Calculator using python functions

In [12]:

```
def add(a, b):
    return a + b

def subtract(a, b):
    return a - b

def multiply(a, b):
    return a * b

def divide(a, b):
    if b == 0:
        return "Cannot divide by zero"
    else:
        return a / b
a=int(input("enter number: "))
b=int(input("enter number: "))
op=input("enter sign + for add - for sub * for multipication / for divison")
if op=="+":
    print(add(a,b))
elif op=="-"  :
    print(subtract(a, b))
elif op=="*":
    print(multiply(a, b))
elif op=="/" :
    print(divide(a, b))
```

```
enter number: 4
enter number: 2
enter sign + for add - for sub * for multipication / for divison/
2.0
```

Write a Python program to demonstrate the function of finding sum and average of first n natural numbers.

In [13]:
```python
def sum_and_average(n):
    if n <= 0:
        return "Please enter a positive integer."

    total = 0
    count = 0
    for i in range(1, n + 1):
        total += i
        count += 1

    average = total / count
    return total, average

# Taking user input
num = int(input("Enter a number to find sum and average of first n natural numbers: "))
result = sum_and_average(num)
print(result)
```

```
Enter a number to find sum and average of first n natural numbers: 10
(55, 5.5)
```

Write a Python program to demonstrate the function of finding multiplication of first n natural numbers.

In [14]:
```python
def multiplication_of_natural_numbers(n):
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result
n=int(input("enter number: "))
print(f"multiplication_of_natural_number ({n}): {multiplication_of_natural_numbers(n)}")
```

```
enter number: 5
multiplication_of_natural_number (5): 120
```

Write a Python program to find reverse of given number using user defined function.

In [15]:
```python
def reverse_number(num):
    reverse = 0
    while num > 0:
        remainder = num % 10
        reverse = (reverse * 10) + remainder
        num = num // 10
    return reverse
num=int(input("enter number: "))
```

```
print("reverse number is",reverse_number(num))
```

```
enter number: 458925
reverse number is 529854
```

"Write your own python program for computing square roots that implements Newton's Method. Use of inbuilt function, math library or x**0.5 is not allowed.

- Newton Method is a category of guess-and-check approach. You first guess what the square root might be and then see how close your guess is. You can use this information to make another guess and continue guessing until you have found the square root (or a close approximation to it). Suppose x is the number we want the root of and guess is the current guessed answer. The guess can be improved by using (guess+ x/guess)/2 as the next guess.
- The program should -

1. Prompt the user for the value to find the square root of (x) and the number of times to improve the guess.
2. Starting with a guess value of x/2, your program should loop the specified number of times applying Newton's method and report the final value of guess. "

In [16]:
```
def square_root_newton(x, iterations):
    guess = x / 2
    for i in range(iterations):
        guess = (guess + x / guess) / 2
    return guess

# Taking input from the user
x = float(input("Enter the number to find the square root of: "))
iterations = int(input("Enter the number of iterations: "))

result = square_root_newton(x, iterations)
print(f"The square root of {x} using Newton's method is approximately: {result}")
```

```
Enter the number to find the square root of: 9
Enter the number of iterations: 3
The square root of 9.0 using Newton's method is approximately: 3.000015360039322
```

In [ ]:
```

```