def countdown(n):

if n <= 0:

print('Blastoff!')

else:

print(n)

countdown(n-1)

def countup(n):

if n >= 0:

print('Blastoff!')

else:

print(n)

countup(n+1)

num = int(input("Enter a number: "))

if num > 0:

countdown(num)

elif num < 0:

countup(num)

else:

print('Blastoff!')

***Output for input 3:***

Enter a number: 3

3

2

1

Blastoff!

***Output for input -3:***

Enter a number: -3

-3

-2

-1

Blastoff!

***Output for input 3:***

Enter a number: 0

Blastoff!

For input of zero, I have called the print statement to output "Blastoff!"

because zero is neither positive nor negative, and it's the starting point

from which we count either up or down.

2. Code of the program implementing a runtime error:

def factorial(n):

if n == 0:

return 1

else:

return n \* factorial(n - 1)

num = int(input("Enter a number: "))

print("The factorial of", num, "is", factorial(num))

Expected Output:

Enter a number: -3

Traceback (most recent call last):

File "c:\Users\User\Documents\GitHub\University-of\_the-people\T3 22\23\week3\runtime Error.py", line 11, in <module>

print("The factorial of", num, "is", factorial(num))

File "c:\Users\User\Documents\GitHub\University-of\_the-people\T3 22\23\week3\runtime Error.py", line 7, in factorial

return n \* factorial(n - 1)

File "c:\Users\User\Documents\GitHub\University-of\_the-people\T3 22\23\week3\runtime Error.py", line 7, in factorial

return n \* factorial(n - 1)

File "c:\Users\User\Documents\GitHub\University-of\_the-people\T3 22\23\week3\runtime Error.py", line 7, in factorial

return n \* factorial(n - 1)

[Previous line repeated 995 more times]

File "c:\Users\User\Documents\GitHub\University-of\_the-people\T3 22\23\week3\runtime Error.py", line 4, in factorial

if n == 0:

RecursionError: maximum recursion depth exceeded in comparison

The maximum recursion depth has been exceeded, according to the error notice. When the input is negative, the recursion continues indefinitely, exhausting the program's memory.

We must include a check to see if the input is a positive integer in order to correct this mistake. We can print an error message and go back if it's not.

The error can be attended to as follows:

def factorial(n):

if n == 0:

return 1

elif n > 0:

return n \* factorial(n - 1)

else:

print("Error: Enter a positive integer.")

return

num = int(input("Enter a number: "))

if num >= 0:

print("The factorial of", num, "is", factorial(num))

else:

print("Error: Enter a positive integer.")

The output for a negative integer input is now:

Enter a number: -3

Error: Enter a positive integer.

This program illustrates a typical runtime issue that can happen in

recursive functions if the base case is incorrectly defined or if the function calls itself endlessly. It's crucial to thoroughly evaluate the

base case and the circumstances in which the function should terminate if you want to prevent these kinds of mistakes.