# **Programming Assignment Unit 3**

Computer Science, University of the People

CS 1101-01 Programming Fundamentals - AY2024-T1

Instructor, Jose Alvarado

September 28, 2023

## Q1: Countdown using recursion.

For this exercise, we were asked to write a program that asks the user to input a numeric value. If the user inputs a positive value, we should display a countdown starting from the input down to 0 as performed in the function supplied in the book. If the user provides us with a negative value, we will perform a count up starting from the input value up to 0. At the end of each action, we will display a “blastoff!’ confirmation:



**Explanation:**   
The code above has three main parts each represented in a method.

1. **Method ‘countdown’ –** this method is based on the method provided in the book, but I have made changes to it, so the final ‘blastoff’ command is in a more central location. The method takes a single parameter which is used to break the recursion logic. Each time the method is called the method prints out the current value of the parameter and then passes the next value of the countdown to the following recursion step.
2. **Method ‘countup’ –** this method is basically identical to the countdown method but with two changes, the first difference is the condition logic, instead of checking if the value is greater than zero, we check if the value is less than zero, this is because we are counting up from a negative number. The second difference is the value we pass to the next recursion method is the incremented value that was passed into the method.
3. **Method ‘main’ –** this is the main method of our application. It first outputs some instructions to our user to tell him what we want him to do. After that take the input from the user and test its value. If the value is positive, we pass the value to the recursion method head called ‘countdown’, otherwise we check if the value is less than zero and if so, we pass the value to the recursion method called ‘countup’. In the hidden case of if the value entered is equal to zero, we don’t call either method but instead, immediately skip to the ‘blastoff’ output.

**Testing the code:**



**Explanation of results:**

The results above demonstrate the three separate use cases for the code:

1. A positive input – this shows how the code handles a positive input which invokes the ‘countdown’ method.
2. A negative input – this shows how the code handles a negative input which invokes the ‘countup’ method.
3. A zero-value input – this shows how the code handles a zero-value input which skips the ‘countup’/’countdown’ methods.

## Q2: Catalog with discounts

For this exercise, we were asked to write a Python program that first has no error handling and intentionally throws a non-user-friendly error in the case of a divide by zero circumstance the following is the error-prone code:



**Explanation:**

In the code above we ask the user to input two numbers. Then we print out the result of dividing the first number by the second number. As we can see in the output, our user entered a value of five for the first number and a value of zero for the second number. This resulted in the runtime error of ‘ZeroDivisionError: float division by zero’ since we tried to divide a number by zero and had no error handling in place.

**Fixing the code:**

In the following code, we have added error handling to support a more user-friendly experience to our application.



**Explanation:**

In the code above we again ask the user to input two numbers. Then we print out the result of dividing the first number by the second number. As we can see in the output, our user entered a value of five for the first number and a value of zero for the second number. This time, however, we had error handling and prevention in place. The first thing we added is a condition that checks if the second number provided is indeed zero. If the value is indeed zero, we will throw a system runtime error as required but this time we will provide it with a more user-friendly message. The second thing we have added is an error-catching safety net in the form of a try-catch code block. This block catches any “ZeroDivisionError” runtime errors that arise inside of it and performs the required actions defined in the ‘except’ code block. In this case, the block tells the interpreter to print out the message contained in the error that was thrown. This results in the output we see above of a much more user-friendly error informing the user of his mistake.

## Conclusion

In this assignment, we learned and practised two vital parts of programming. The first is recursion. Which is the practice of methods calling themselves from within themselves creating a type of loop. It's important that when working with recursion we always have a break/escape condition that will break the recursion and prevent and endless recursion loop. The second thing we did was error handling and prevention. This is very important since as programmers we build programs for the use of users, and we don’t always know what the users are going to do so we need to try and handle all eventualities and make sure that they are handled with a more graceful way then just letting the interpreter throw a generic system exception.

## References

Python 3 Documentation

<https://docs.python.org/3/>

Think Python - How to Think Like a Computer Scientist (2nd Edition, Version 2.4.0)

Chapter 5 – Conditionals and recursion

Source Scripts:

