# Learning Journal Unit 3

Godfrey Ouma

University of the People

CS 1101: Programming Fundamentals

Janice Block

July 6, 2023

# Learning Journal Unit 3

1.

**(a) Recursive function coundown**

```
In [17]: def countdown(n):
    ...:     if n <= 0:
    ...:         print('Blastoff!')
    ...:     else:
    ...:         print(n)
    ...:         countdown(n - 1)
    ...: countdown(3)
3
2
1
Blastoff!
```

**(b) Recursive function countup**

```
In [19]: def countup(n):
    ...:     if n>=0:
    ...:         print("Blastoff!")
    ...:     else:
    ...:         print(n)
    ...:         countup(n+1)
    ...:
    ...: countup(-3)
-3
-2
-1
Blastoff!
```

(c) **Function to call (countdown or countup) for input of zero**

Calling coundown function for an input of zero

```
In [15]: def countdown(n):
    ...:     if n <= 0:
    ...:         print('Blastoff!')
    ...:     else:
    ...:         print(n)
    ...:         countdown(n - 1)
    ...: countdown(0)
Blastoff!
```

Calling countup function for an input of zero

```
In [16]: def countup(n):
    ...:     if n>=0:
    ...:         print("Blastoff!")
    ...:     else:
    ...:         print(n)
    ...:         countup(n+1)
    ...:
    ...: countup(0)
Blastoff!
```

**Explanation:**

Even the countup functions print "Blastoff!" upon handling zero, thus I opted to utilize the countdown function instead. To keep the software consistent, only one function—countdown—was chosen. It is important to remember that this choice is arbitrary and that calling the countup function for zero is an option as well.

2.

**(a) Runtime error!**

```
In [8]: a = 59
   ...: b = 0
   ...:
   ...: result = a/b
   ...:
   ...: print(result)
Traceback (most recent call last):

  Cell In[8], line 4
    result = a/b

ZeroDivisionError: division by zero
```

Runtime error refers to an error that does not occur until the program has started to execute but that prevents the program from continuing, and its purpose is to indicate that something exceptional (and bad) has happened (Downey, 2015). Some of the common types of runtime error include logic error, memory leak error, division by zero error, unidentified object error, input and output error, and encoding error (JOSNA, 2021). Therefore, the type of runtime error produced in the example given above is “division by zero error,” which is printed in the output after placing zero in the denominator when dividing 59 by 0.

**Fixing the “division by zero error”**

By determining whether the denominator is zero before to conducting the division, it is possible to prevent the "division by zero error" from occurring. However, the problem can be resolved by utilizing a "if statement" to manage this situation and conduct alternative actions or display the proper message if the denominator is zero (Pierce, 2023). Here is a newer, error-handling version of the program:

```
In [9]: a = 59
   ...: b = 0
   ...: if b == 0:
   ...:     print("Cannot divide by zero")
   ...: else:
   ...:     print(a/b)
Cannot divide by zero
```

When running the above code, the correct out “Cannot divide by zero” is produced as expected, thus avoiding the running time error. If the value of the denominator is unknown in advance, "try_except” block can also be used to detect and handle this error (Pierce, 2023). See the new version below:

```
In [11]: try:
    ...:     a = 59
    ...:     b = 0
    ...:     print(a/b)
    ...: except ZeroDivisionError as e:
    ...:     print("Error: Cannot divide by zero")
Error: Cannot divide by zero
```

It is clear from the example above that enclosing the code in “try-except” blocks enables the program to carry on running even after an exception is encountered.

# References

Downey, A. (2015). *Think Python: How to think like a computer scientist.* Green Tree Press.

JOSNA. (2021, October 11). *What is a runtime error?* ElectronicsHub. https://www.electronicshub.org/runtime-error/

Pierce, D. (2023, March 8). *How to fix ZeroDivisionError in Python*. Rollbar. https://rollbar.com/blog/python-zerodivisionerror/