The Project:

Take in a string (from a user or a file), compute the total number of occurrences of each character in the ASCII set. Print out the top three characters (with the most occurrences). Print out how many characters in the ASCII set you didn’t detect. Then print out the string in reverse order.

Definition of Done:

I will be done when I have exhausted all the possible ways of improving my implementation. I will write a design document and follow it. I will also think up several test cases for entering in my program to see if the program can handle several exceptional inputs. While I code, I will add comments as if I am telling someone who is reading my code how my code works. I will follow some coding standards that I create in this document. After testing all the test cases and making sure the program works, I will be done.

Coding Standards:

* Functions will be block commented with three single quotes
* Global variables will be declared at the top of the file with a comment that explains what it will be used for in the future
* Functions will be declared at the top of the file, under the global variables
* Variables should be named in all lowercase with an underscore in between words, i.e this\_is\_a\_variable
* The main portion of the script should be encased in this statement
  + If \_\_name\_\_ == ‘\_\_main\_\_’:
* In a for loop, if the loop is looping through an iterable, name the variable something intuitive that explains what the variable is referring to
  + For char in string
  + For num in list
* If a for loop is looping through a range, it is ok to name the variable a single letter
  + For i in range(500)
  + For x in range(10)

The Design:

1. Take in string from somewhere
   1. Use a menu
   2. Take in a string from user
      1. Input(‘Enter a string>’)
      2. Set a string variable equal to the data read from stdin
   3. Take in string from file
      1. Open file
      2. Read file
      3. Set a string variable equal to the data read from file
2. Compute total number of occurrences of each character that appears in ASCII
   1. Create a dictionary with keys of numbers ranging from 0 – 127
   2. Loop through 1 – 127 and set all the dictionary entries equal to 0
   3. Loop through the string and find the ASCII value of each character in the string (ord(char))
   4. Add 1 to the dictionary entry with the number that is found from getting the ASCII value as its key
3. Print top three characters
   1. Max\_key = Max(dict, key=dict.get)
   2. Dict[max\_key] = -1
4. Print out how many characters in the ASCII set you didn’t detect
   1. Number = 0
   2. For key, value in dict.items():
      1. If value == 0:
         1. Number += 1
5. Print out string in reverse order
   1. Print(string[::-1])

Test Cases:

1. Menu
   * Enter in ‘y’ and ‘Y’ for a file
   * Enter in ‘n’ and ‘N’ for stdin
   * Enter in ‘A’, ‘c’, and ‘Z’ to see if the menu catches those errors
   * Enter in CTR-C to see if the program will exit gracefully
2. File
   * Enter in “test.xml” (a file I made in the directory)
   * Enter in “test” (not a file in my directory)
   * Enter in an empty string
   * Enter in CTR-C
   * Enter in “test” and then “test.xml” to see if the program works after being given invalid data
3. STDIN
   * Enter in “hello world”
   * Enter in “the quick fox jumped over the lazy brown dog”
   * Enter in “abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ”
   * Enter in an empty string
   * Enter in CTR-C
   * Enter in an empty string and then “hello world" to see if the program works after being given invalid data

The Code:

*#will hold the string*

string = ''

*#will hold the number of occurrences of each character. initially all chars are set to 0*

chr\_dict = {key: 0 for key in range(128)}

*#will hold the number of characters not found in the string*

number = 0

*#will hold the menu choice from the user*

menu = ''

'''

Gets the choice that a user would like, whether they want to read from a file or from

stdin. If Control C is pressed, it exits the program

'''

**def** menu\_choice():

    try:

        menu = input('Would you like to read from a file [y/N]: ')

        if menu.lower() != 'y' and menu.lower() != 'n':

            print('Enter the character y to read from a file or n to read from stdin.')

            return menu\_choice()

        else:

            return menu

    except KeyboardInterrupt:

        print('Good Bye.')

        exit(0)

'''

Gets the name of a file and then attempts to read from the file. If the file

doesn't exist, it recursively calls itself until the file does exist

'''

**def** read\_file():

    try:

        filename = input('Enter in the name of the file you would like to read in: ')

        try:

            with open(filename, 'r') as file:

                return file.read()

        except FileNotFoundError:

            print('That file does not exist. Try again.')

            return read\_file()

        except UnicodeDecodeError:

            print('File must be a text file.')

            return read\_file()

    except KeyboardInterrupt:

        print('Good Bye.')

        exit(0)

'''

Similar to read\_file, this function gets input from the user and if it is empty or null,

it recursively calls itself until the input is not empty or null

'''

**def** read\_stdin():

    try:

        return\_string = input('Enter in the string: ')

        if len(return\_string) == 0 or return\_string is None:

            print('You must enter a string to continue.')

            return read\_stdin()

        else:

            return return\_string

    except KeyboardInterrupt:

        print('Good Bye.')

        exit(0)

if \_\_name\_\_ == '\_\_main\_\_':

    menu = menu\_choice()

*#if the user wants to read from a file, set string to read\_file*

*#if the user wants to read from stdin, set string to input()*

    if menu.lower() == 'y':

        string = read\_file()

    elif menu.lower() == 'n':

        string = read\_stdin()

*#increment the dictionary values for the characters in string*

    for char in string:

        chr\_dict[ord(char)] += 1

*#print out the three most common letters*

    for i in range(3):

*#get the key with the maximum value*

        max\_key = max(chr\_dict, key=chr\_dict.get)

*#set the value of this key to -1 so on the next loop it doesn't get picked up again*

        chr\_dict[max\_key] = -1

*#print the*

        print(**f**'ASCII value: {max\_key} - Character value: {chr(max\_key)}')

    for key, value in chr\_dict.items():

        if value == 0:

            number += 1

    print(**f**'There were {number} ASCII character(s) that were not found in this string.')

*#print out the string backwards*

    print(string[::-1])