Marco Seman

Professor Jie Shen

November 4, 2018

Assignment #2

Machine Type: Windows 8

Interpreter Type:

Perl: Active Perl, Notepad++

Python: JetBrains PyCharm Community Edition

Table of Contents

[Question 1: 2](#_Toc529132517)

[Question 2: 3](#_Toc529132518)

[Question 3: 5](#_Toc529132519)

[Question 4: 7](#_Toc529132520)

[Question 5: 8](#_Toc529132521)

[Question 6: 9](#_Toc529132522)

[Question 7: 10](#_Toc529132523)

[Question 8: 12](#_Toc529132524)

# Question 1:

Write a program that reads a string from the standard input, and uses a Perl regular expression to test whether the string looks like a valid IP address.

CODE:

use strict;

use warnings;

print "Enter a Valid IP Address: ";

my $entry = <STDIN>;

chomp $entry;

print "You have entered '$entry'\n";

# test whether the IP address is whithin range

if($entry =~ /(\d{1,3}\.\d{1,3}\.\d{1,3}\.\d{1,5}\:\d{1,5})/) {

print "This is a valid IP Address: $entry";

}

#if test is not within range, let user know

else{

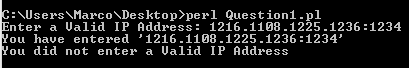
print "You did not enter a Valid IP Address"

}

Description of your code design and implementation:

For the first question, we needed to use regular expression to test if the string looks like a valid IP address. I allowed the user to enter in a value and created an if statement with the condition that the user must enter in valid entry of an IP Address. If the value is not within range, let user know with an else statement.

Test Cases:





# Question 2:

Write a program that reads a string from the standard input, and match the following patterns:

a) an odd digit followed by an even digit (eg. 12 or 74)

b) a letter followed by a non-letter followed by a number

c) a word that starts with an upper case letter

d) the word "yes" in any combination of upper and lower cases letters

e) one or more times the word "the"

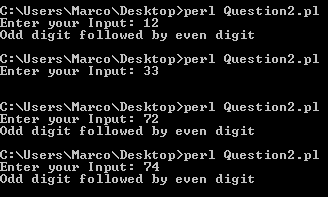
f) a date in the form of one or two digits, a dot, one or two digits,

a dot, two digits

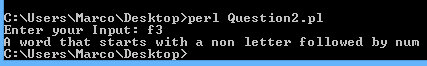
g) a punctuation mark

Test Cases:

a)



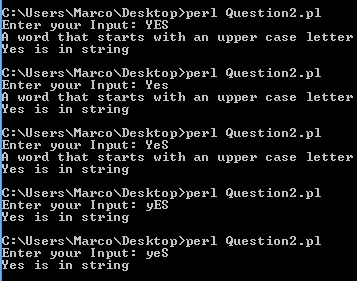
b)



c)



d)



e)



f)



g)



Code:

use strict;

use warnings;

print 'Enter your Input: ';

my $valInput = <STDIN>;

#my @StringVal = (['Yes'], ['yEs'], ['yeS'], ['YEs'], ['YeS'], ['yes'], ['yES'], ['YES']);

my $StringVal2 = "the";

# Letter Patter for a - odd followed by even

if ($valInput =~/^-?\d+$/){

# use split the value into smaller sections if its a two digit value

my @char = split //, $valInput;

# If the value's first double digit value remainder is one when the module

# is divisable by two. the single digit must also be divisable by 2 and have the mode of 0

if($char[0] %2 == 1 && $char[1]%2 == 0){

print "Odd digit followed by even digit";}}

# Letter Patter for b - letter followed by non letter

if ($valInput =~/[a-z]/ and $valInput =~/\W/ and $valInput =~ /^-?d+$/){

print "A word that starts with a non letter followed by num";}

# Letter Patter for c - word starting with uppercase

if ($valInput =~/^[A-Z]/){

print "A word that starts with an upper case letter \n";}

# Letter Patter for d - "yes" is in string

if ($valInput =~ "Yes" or $valInput =~ "Yes" or $valInput =~ "yEs" or $valInput =~ "yeS" or $valInput =~ "YEs" or $valInput =~ "YeS" or $valInput =~ "yes" or $valInput =~ "yES" or $valInput =~ "YES"){

print "Yes is in string";}

# Letter Patter for e - "the" was used one or more times

if ($valInput =~ /\Q$StringVal2\E/) # The{

print "the was found";}

# Letter Patter for f - Date

if ($valInput =~ /[0-9]{2}.[0-9]{2}/){

print "Youre in date\n";}

# Letter Patter for g - Punctuation Mark

if ($valInput =~ s/[[:punct:]]//g){

print "punctuation found\n";}

Description of your code design and implementation:

For Question 3, the plan was to test the user’s standard input to match the pattern necessary. The design idea was to create multiple if statements to ensure that the program will able to discover what types of pattern was from the user’s entry.

# Question 3:

Que Write a program that takes as input text file (input.txt) and identifies all the email addresses that are embedded in this file, and prints all the email addresses out to an output file called “output.txt”. In the output file, the email addresses should be separated by comma and sorted alphabetically. The screenshot should display at least first twenty email addresses, together with a line number. The first two email addresses will be displayed as

1: xxx@xxx

2: yyy@yyy

, and so on.

Description of Design and Implementation:

In my design, I decided to install Email:Valid to pull email addresses from the input document. I then used a while loop to take those email addresses and split them up with a space and a new line. I then sorted all the email addresses and pasted them into an output document.

Code:

# Installed Email::Valid to read email from txt document

use Email::Valid;

open (MYFILE, 'input.txt');

my $email\_file = "output.txt"; #Export file

my $found = 0;

open(FH,'>',$email\_file);

open my $outputEmail, ">",$email\_file;

while (<MYFILE>) {

chomp;

my @words = split(' ');

foreach my $word (@words) {

if(Email::Valid->address($word))

{

$found++;

print FH "$word\n";

}

}

}

open(FH, '<output.txt');

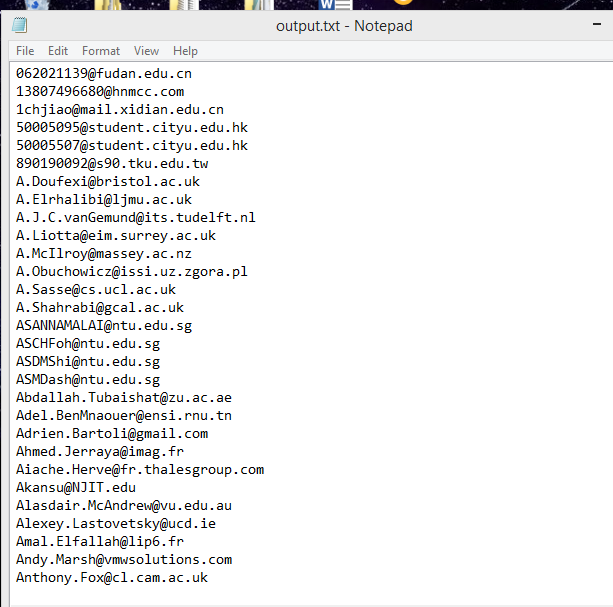
my @lines = <FH>;

@lines = sort @lines;

open(FILE, "> $email\_file");

print FILE @lines;

Test Case:



# Question 4:

Write a program that reads a number, adds up all the digits, and

displays the result. Run your code with the following test number:

176298

It should give 33.

Code:

print "Please Enter Your Values: ";

my $input = <>;

chomp $input;

my @strings = ($input);

for my $string (@strings) {

my $sum;

$sum += $\_ for split(//, $string);

print "$sum\n";

}

Description of your code design and implementation:

For Question 4, the goal was to take a value entered by the user with not spaces and split those values into single digits. Once the values have been split up, I set up a for loop that that add the values together one by one. Once all values are added together, the result is displayed.

Test Case:



# Question 5:

Write a program that reads a series of digits and it displays

the digits in ascending order, together with their frequency. E.g.,

for the input:

1 7 6 2 9 8 2 0 2 0

the program will display:

0 2

1 1

2 3

6 1

7 1

8 1

9 1

Code:

my @values = (1,7,6,2,9,8,2,0,2,0);

my %count;

foreach my $num (@values) {

$count{$num}++;

}

foreach my $num (sort keys %count) {

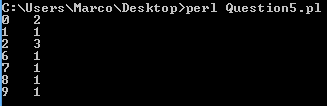
printf "%-3s %s\n", $num, $count{$num};

}

Description of your code design and implementation:

For question 5, the design is to place a series of digits inside an array. I then generate a for loop that counts the amount of times each character is displayed until the loop reaches the end of the array. I then create another for loop that sorts the numeric values in order and displays each of the numbers from lowest to highest and and next to them the count of amount of times each value was used with appropriate spacing.

Test Case:



# Question 6:

In cryptography, Caesar cipher is one of the simplest encryption techniques. The key idea of this method is to replace each plaintext letter with one fixed number of places down the alphabet. Below is an example with a shift of three:

Plain: ABCDEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyz

Cipher: DEFGHIJKLMNOPQRSTUVWXYZabcdefghijklmnopqrstuvwxyzABC

To cipher a string, ‘A’ is replaced by ‘D’, ‘B’ is substituted by ‘E’, and so on. To decode a string, ‘A’ is replaced by ‘x’, etc.

Write a program that first ask user to do encoding or decoding. Then it reads an input sentence and a number for shift. The output of the program is an encoded sentence or decoded sentence, depending upon user’s choice.

Test your program with a sentence: “Why write in blue or black when you can make your words invisible”. Demonstrate the encoded sentence and the deciphering of the encoded sentence.

CODE:

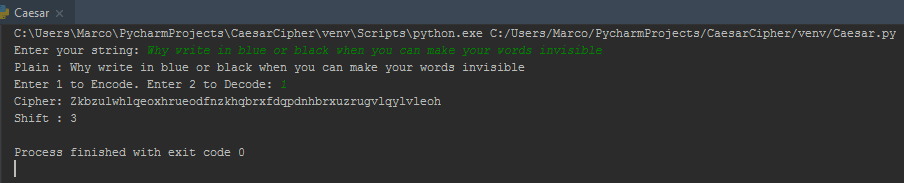
def encrypt(subString, x):  
 result = ""  
 UserInput = input("Enter 1 to Encode. Enter 2 to Decode: ")  
  
 # If the user decides to Encrypt message  
 if (UserInput == "1"):  
 for i in range(len(subString)):  
 character = subString[i]  
 # This if statement will Encrypt uppercase characters  
 if (character.isupper()):  
 result += chr((ord(character) + x - 65) % 26 + 65)  
 # else it will Encrypt lowercase characters  
 else:  
 if (character == ' '):  
 character = ' '  
 else:  
 result += chr((ord(character) + x - 97) % 26 + 97)  
 return result  
  
 # If the user decides to decrypt message  
 if (UserInput == "2"):  
 for i in range(len(subString)):  
 character = subString[i]  
  
 # This if statement will Decrypt uppercase characters  
 if (character.isupper()):  
 result += chr((ord(character) - x - 65) % 26 + 65)  
  
 # else it will Decrypt lowercase characters  
 else:  
 if (character == ' '):  
 character = ' '  
 else:  
 result += chr((ord(character) - x - 97) % 26 + 97)  
  
 return result  
  
# check the above function  
# Read input and display results  
subString = input("Enter your string: ")  
  
x = 3  
print ("Plain : " + subString)  
print ("Cipher: " + encrypt(subString, x))  
print ("Shift : " + str(x))

Description of your code design and implementation:

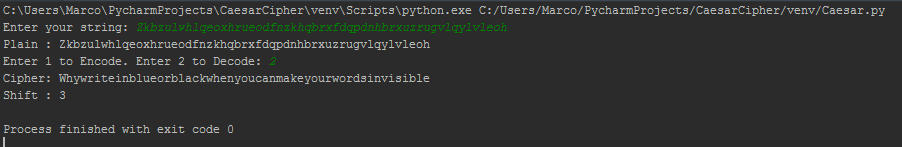
For the ceasar cipher program, the first step was to take a user input. I then determined that im going to generate a for loop and take the string of characters entered by the user and look through the letters one by one. When it takes the first character [0], the if statement will determine whether the character is uppercase using isupper command. If the character isn’t uppercase, the system determines it’s a lower case by default and ignores any whitespaces. When determining the shifting, I decided to use “char((ord(----) + character shift – numeric value in ASCII table” to shift the character. I did that for both the upper case and lower case, then return the values to result. For deciphering, I copied exactly what I had for the encryption and changed the value so it shifts to the left in the ASCII table. The program will then reveal the plan text, and the encrypted/decrypted message.

Test:

Encrypt:



Decrypt:



# Question 7:

Write a program that takes a URL as an input. It then fetches the web page of that URL, and strips out all of the HTML tags. We assume that a tag is anything between a pair of angular brackets: < … > . Sort all the tags alphabetically, and print out the result on computer screen. Each tag is printed out only once together with its frequency (# of occurrences).

Test URL is https://www.tandfonline.com/toc/tjms20/current.

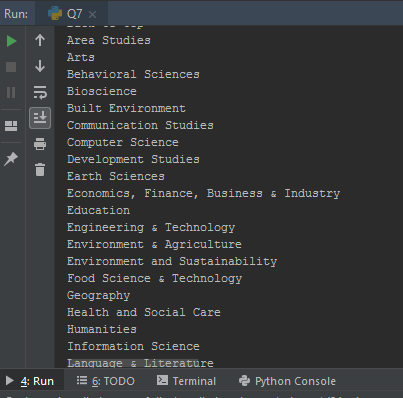
Description of Design and Implementation:

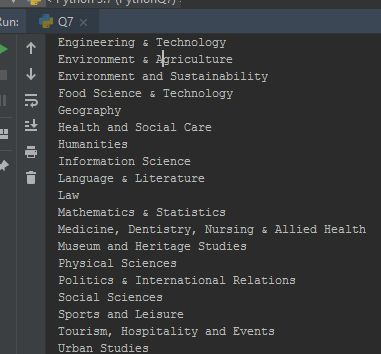
For question 7, I needed to download an HTML parser for Python 3 so I installed urllub and beautifulsoup4. Once I enabled the feature to read html tags, I then created a string for the URL then used the library feature “urllib.request” to read and store the url and then soup to read and store html tags. I then created a for loop to remove any scripts or styles that were in the html tags. Then, I retrieved the information of the tags and placed it into the string text. I split the strings and displayed them one line at a time and sorted them alphabetically.

Code:

import urllib.request  
from bs4 import BeautifulSoup  
  
# URL:  
URL = "https://www.tandfonline.com/toc/tjms20/current"  
  
#Extract HTML Tags from website using beutiful soup  
htmlTags = urllib.request.urlopen(URL).read()  
soup = BeautifulSoup(htmlTags)  
  
# Remove scripts and css  
for script in soup(["script", "style"]):  
 script.extract()  
  
# retrieve text from the website and place it in text  
text = soup.get\_text()  
  
# Take the lines, and delete them that have spacces between them  
lines = (line.strip() for line in text.splitlines())  
  
# Display in multiple lines using split  
split = (phrase.strip() for line in lines for phrase in line.split(" "))  
# remove any blank lines  
text = '\n'.join(splitting for splitting in split if splitting)  
sorted(text, reverse=True)  
print(text)

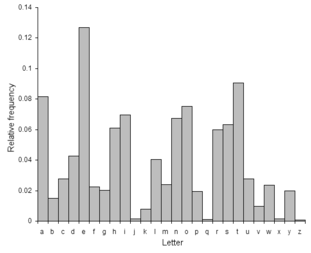
Test Case:





# Question 8:

The distribution of letters in a document has a distinctive and predictable shape, as shown in the following figure. This distribution could be effective information to break a cipher. Write a program that reads an input file, and outputs the count or frequency of all 26 letters to an output file: count.dat. Case sensitivity is not considered, i.e., ‘a’ is treated as the same as ‘A’. Test file is “constitution.txt” at Canvas.



Code:

def Question8():  
 import string  
  
 fileName = ("C:/Users/Marco/Desktop/constitution.txt")  
 File = open(fileName)  
 # Creating a file to export my documentation to my local machine  
 OutputFile = open("C:/Users/Marco/Desktop/count.dat","w")  
 # Attempt to remove specific characters  
 OtherValues = ['—', '”', '“']  
  
 # Creating three arrays. the 1st array gathers the letters, the second array is for counting the amount  
 # of times the character is used. the third array is to calculate the fequency and sort them  
 list = []  
 letterCount = []  
 frequency = []  
  
 # The total count of of the character used  
 TotalCount = 0  
  
 countCharacters = {}  
 # A nested for loop that removes any whitespaces, digits, or punctuations  
 for i in File: # search through the file and look at every character  
 i = i.translate(str.maketrans('', '', string.punctuation)) # ignores punctuations: ()-'".:;  
 i = i.translate(str.maketrans('', '', string.whitespace)) # ignores spaces  
 i = i.translate(str.maketrans('', '', string.digits)) # ignores numeric values  
 #i = i.translate(str.maketrans('', '', string.OtherValues)) # failed attempt  
 # there wasnt a library for the other values which made it difficult to remove other values  
 # such as: — ””  
 i = i.lower() # doesn't matter if they are lower case or upper case  
  
 # for loop that loops by each character, takes that letter and assign a number to it  
 # based on the amount of times it was used.  
 for letterCharacter in i:  
 if letterCharacter in countCharacters:  
 countCharacters[letterCharacter] += 1  
 else:  
 countCharacters[letterCharacter] = 1  
 # For loop that counts the amount of times a specific character is used  
 for characters, x in countCharacters.items():  
 # Place zero as a place holder theen square the brackets to use the list that'll be modified  
 list.append([x, characters, 0])  
 letterCount.append(x)  
 # calculates to total amount of times a character is used  
 totalcount = sum(letterCount)  
 #take the amount of times the character is used, then sort it by the amount of times how many times its used  
 for y in letterCount:  
 LetterFrequency = (y/totalcount) \* 100  
 frequency.append(LetterFrequency)  
 frequency.sort(reverse=True)  
 list.sort(reverse=True)  
  
 # For loop that counts the entire character count so that i can divide it in the next loop  
 for characters in list:  
 TotalCount += characters[0]  
  
 # Calculate the amount of times the character is used divide it by the total count  
 for letter in list:  
 decimalResult = (letter[0] / TotalCount)  
 #Proper formatting of the decimal  
 letter[2] += float(format(decimalResult, '.6f'))  
# A for loop to print out all of the letters and frequency with the appropriate formatting  
 for i in list:  
 print('{} is {} '.format(i[1], i[2]))  
 # Write all information to the dat file  
 OutputFile.write('\n{} is {} '.format(i[1], i[2]))  
 # Close out of the count.dat file  
 OutputFile.close()  
Question8()

Description of your code design and implementation:

When setting up my design, I know I was going to need multiple for loops. I needed a nested for loop to read through each of the characters of a file and count the frequency of each letter. I decided an array would be useful to assign each letter to a list. My first for loop counted the amount of times each letter was used, if there wasn’t a letter already, assign it one in order to generate the total count of the character. This ended up working fine, but then I needed to create another for loop to ignore any other values such as special characters, digits, and spaces. I then created a for loop that counts the frequency each letter is used. Another for loop takes each character that’s in an array and divides it by the total character count to get a decimal of the average time the character is used along with sorting based on the frequency of each character from highest to lowest. Finally I created a for loop to read through all the characters and display the results on both the terminal and output file (count.dat).

Test Case:

