¥ *Python for Everybody Exercises* ¥

* *Chapter 1 :- WHY SHOULD YOU LEARN TO WRITE PROGRAMS?*

*Exercise 1 :-*

***1} what is the function of the secondary memory in a computer?***

1. ***Execute all of the computation and logic of the program.***
2. ***Retrieve web pages over the Internet.***
3. ***Store information for the long term, even beyond a power cycle.***
4. ***Take input from the user.***

***Ans :-*** *Storing Information for Long Term, Even Beyond The Power Cycle.*

***2} What is a program?***

***Ans :-*** *A program is a Combination of Statements Ordered in a Structured manner to perform a Specific Task. It is a set of instructions that speciﬁes a computation.*

***3} What is the difference between a compiler and an interpreter?***

***Ans :-*** *A Compiler Compiles Entire Program at Once whereas Interpreter Checks each statement Line By line & Compiles Them.*

* An **interpreter** reads the source code of the program as written by the programmer, parses the source code, and interprets the instructions on the ﬂy. Python is an interpreter and when we are running Python interactively, we can type a line of Python (a sentence) and Python processes it immediately and is ready for us to type another line of Python.
* A ***compiler*** needs to be handed the entire program in a ﬁle, and then it runs a process to translate the high-level source code into machine language and then the compiler puts the resulting machine language into a ﬁle for later execution.

***4} Which of the following contains “machine code”?***

***a) The Python interpreter***

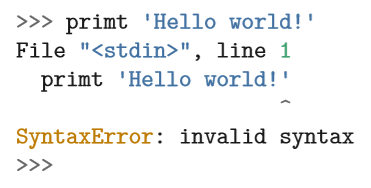
***b) The keyboard***

***c) Python source ﬁle***

***d) A word processing document***

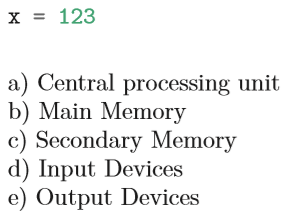
***Ans :-*** *Python Source Code.*

***5} What is wrong with the following code:***



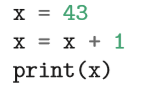
***Ans :-*** *It’s a Syntax Error as The keyword “Print” is Typed incorrectly as “primt”.*

***6} Where in the computer is a variable such as “x” stored after the following Python line finishes?***



***Ans :-*** *Main memory.*

***7} What will the following program print out:***



***Ans :-*** *44.*

***8} Explain each of the following using an example of a human capability: (1) Central processing unit, (2) Main Memory, (3) Secondary Memory, (4) Input Device, and (5) Output Device. For example, “What is the human equivalent to a Central Processing Unit”?***

***Ans :-***

*1} CPU :- Cortex.*

*2} Main Memory :- Brain.*

*3} Secondary Memory :- Thalamus.*

*4} Input Devices :- Sensory Organs.*

*5} Output Devices :- Reflexes & Response Organs*

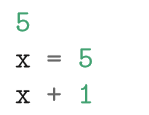
***9} How do you fix a “Syntax Error”?***

***Ans :-*** *You can Mitigate the Syntax error by checking Out for Syntax Mistakes That You Might have made in your Code and Correcting the Syntax with Right keywords or Statement.*

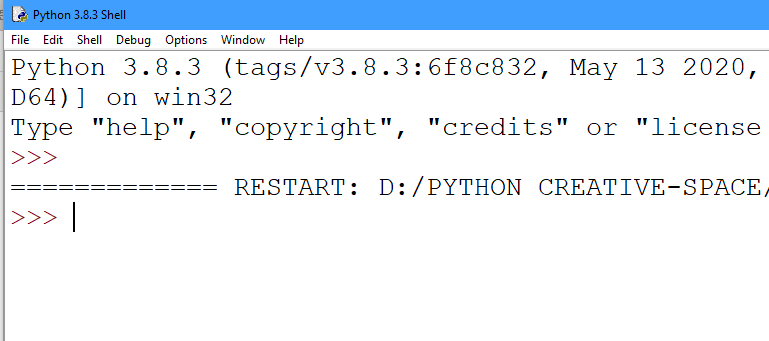
* *Chapter 2 :- VARIABLES, EXPRESSIONS, AND STATEMENTS.*

*Exercise 2 :-*

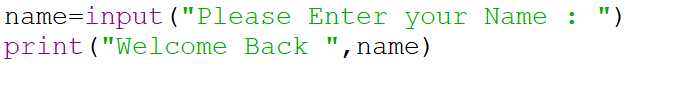
***1} Type the following statements in the Python interpreter to see what they do:***



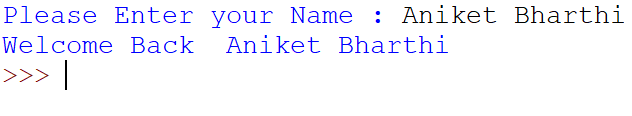
***Code :-***



***2} Write a program that uses input to prompt a user for their name and then welcomes them.***

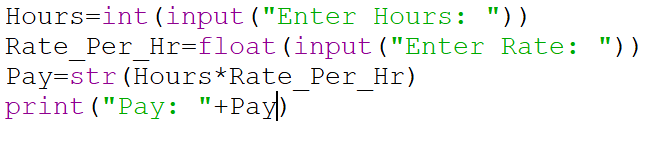
***Code :- ***

***Output :-***

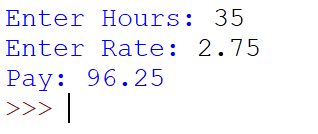
******

***3} Write a program to prompt the user for hours and rate per hour to compute gross pay.***

***Code :-***

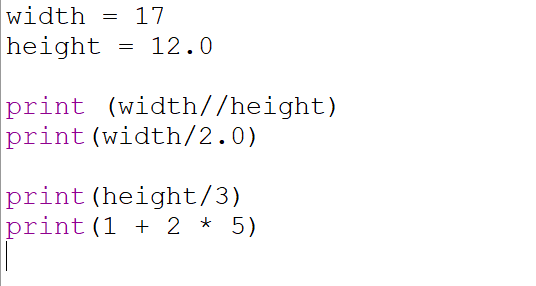


***Output :-***

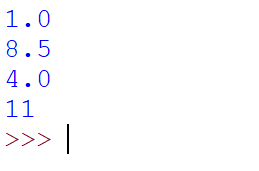


***4} Assume that we execute the following assignment statements:***

***Code :-***

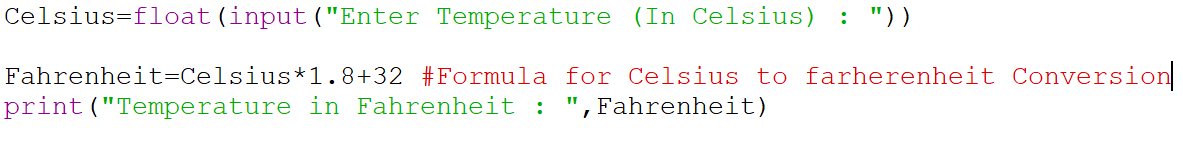
******

***Output :-***

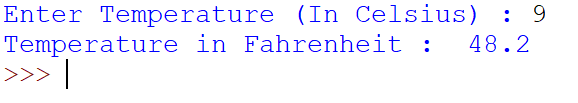
******

***5} Write a program which prompts the user for a Celsius temperature, convert the temperature to Fahrenheit, and print out the converted temperature.***

***Code :-***



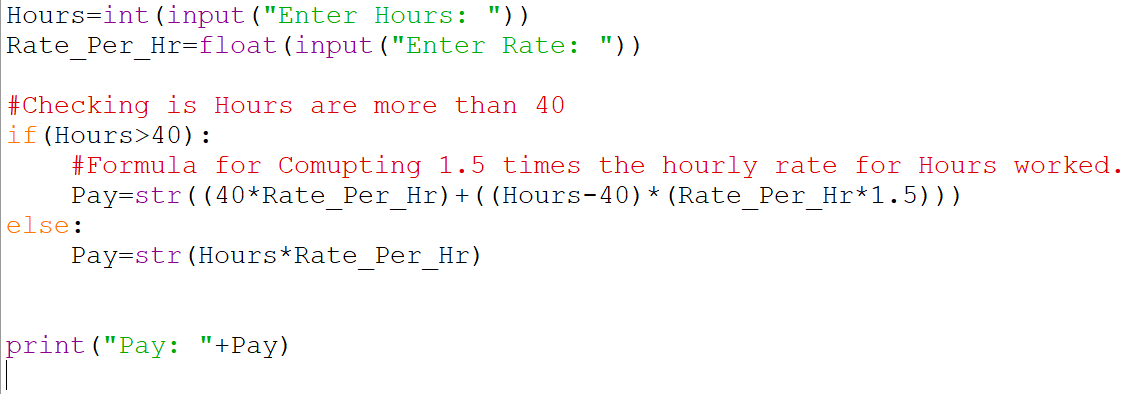
***Output :-***



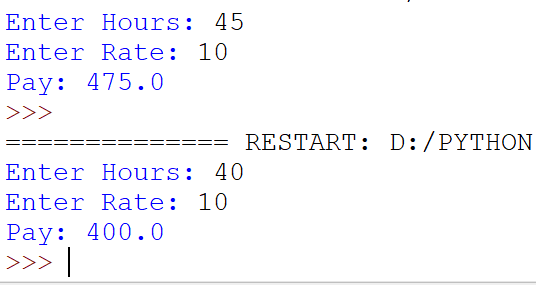
*Chapter 3 :- CONDITIONAL EXECUTION.*

***1} Rewrite your pay computation to give the employee 1.5 times the hourly rate for hours worked above 40 hours.***

***Code :-***

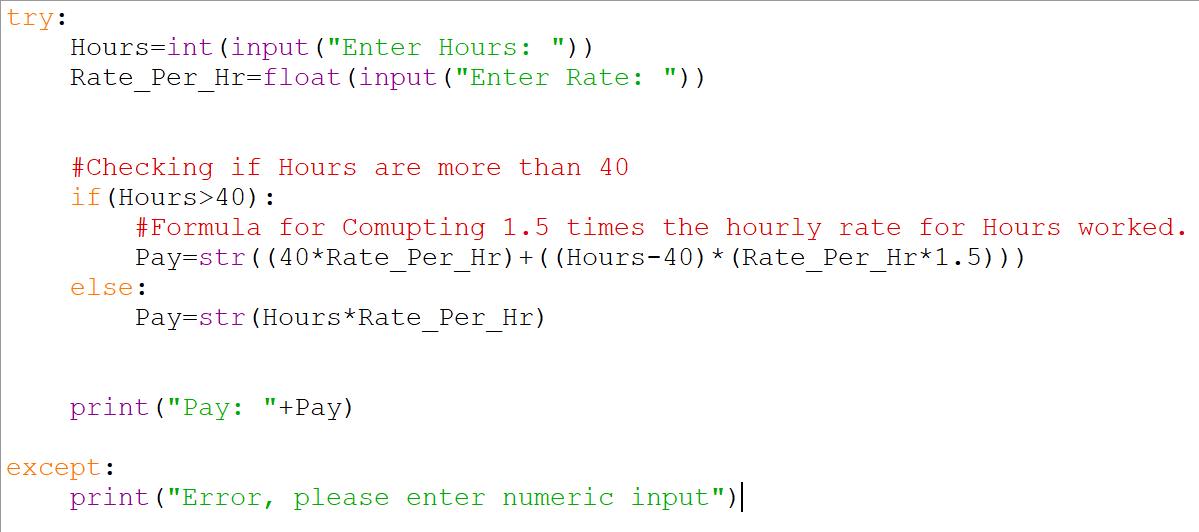


***Output :-***

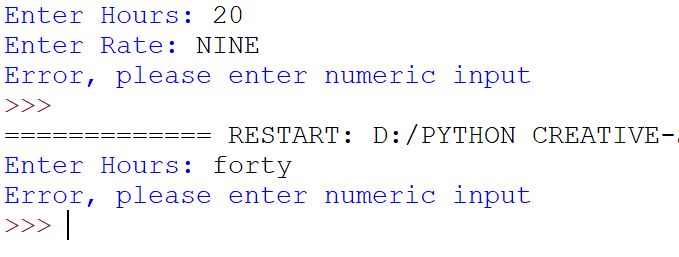


***2} Rewrite your pay program using try and except so that your program handles non-numeric input gracefully by printing a message and exiting the program. The following shows two executions of the program:***

***Code :-***

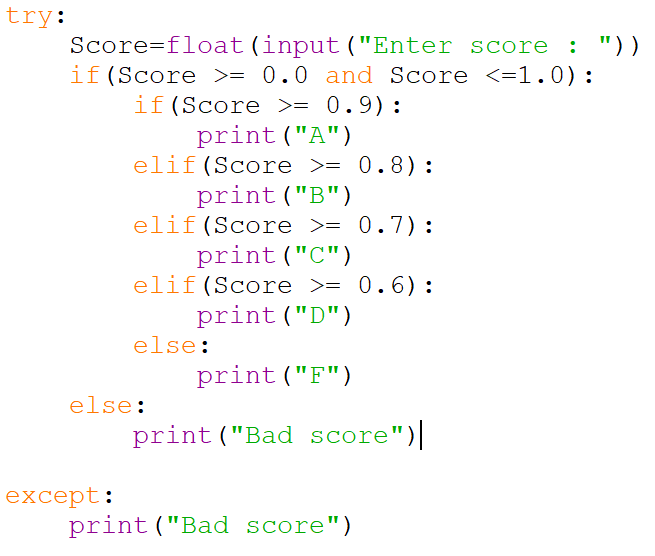


***Output :-***

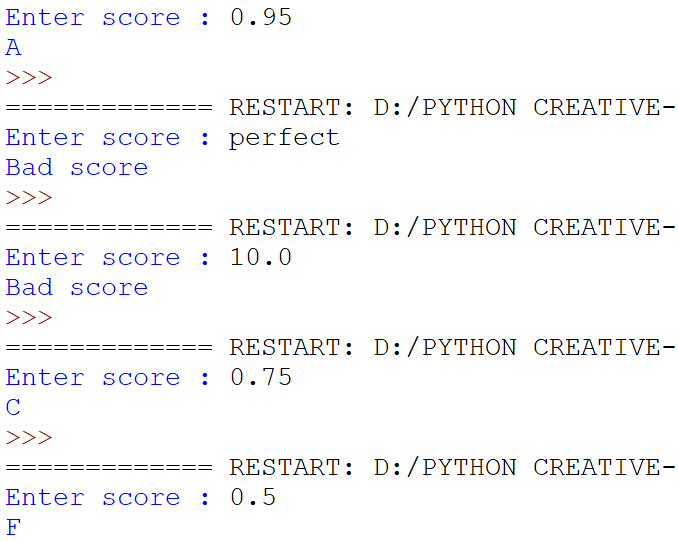


***3} Write a program to prompt for a score between 0.0 and 1.0. If the score is out of range, print an error message. If the score is between 0.0 and 1.0, print a grade using the following table:***

***Code :-***



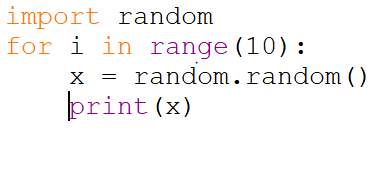
***Output :-***



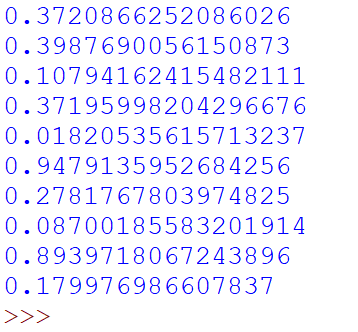
* *Chapter 4 :- FUNCTIONS.*

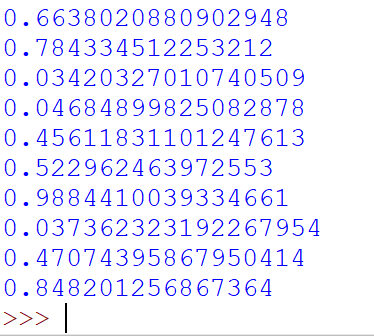
Exercise 1: *Run the program on your system and see what numbers you get. Run the program more than once and see what numbers you get.*

***Code :-***



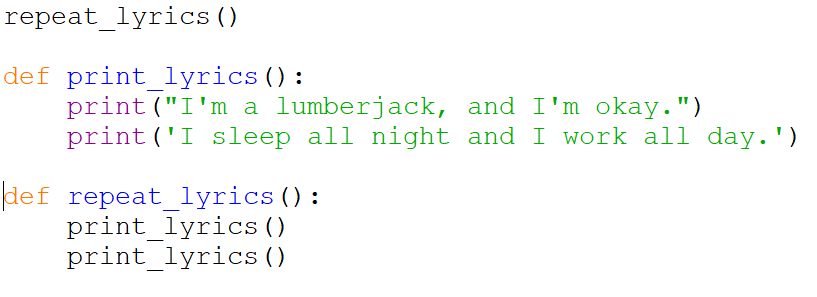
***Output :-***



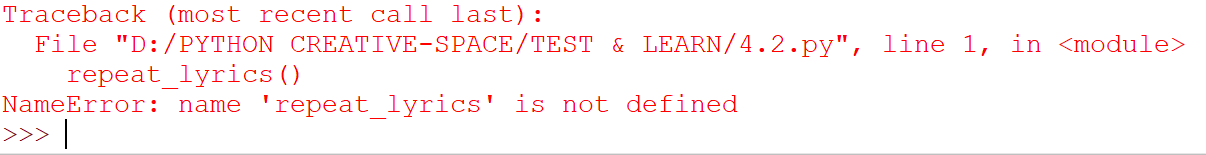


Exercise 2: *Move the last line of this program to the top, so the function call appears before the definitions. Run the program and see what error message you get.*

***Code :-***

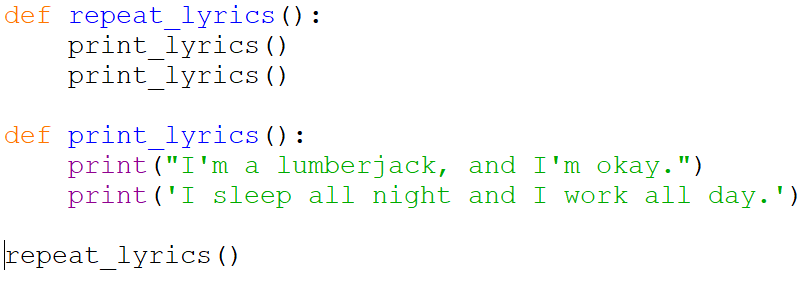


***Output :-***

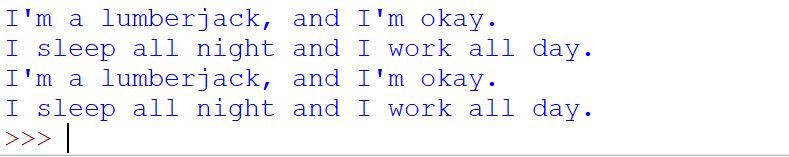


Exercise 3: *Move the function call back to the bottom and move the definition of print\_lyrics after the definition of repeat\_lyrics. What happens when you run this program?*

***Code :-***



***Output :-***



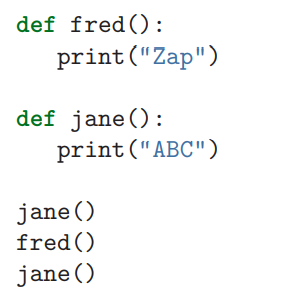
Exercise 4: *What is the purpose of the “def” keyword in Python?*

***Ans :-*** *b and c are both true :-*

*b) It indicates the start of a function*

*c) It indicates that the following indented section of code is to be stored for later.*

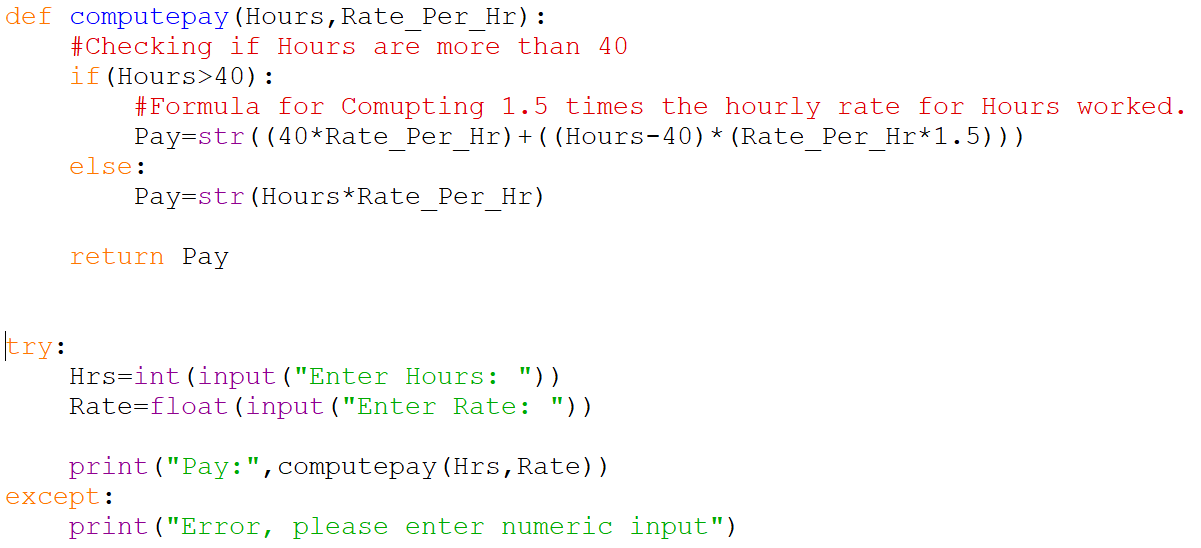
Exercise 5: *What will the following Python program print out?*



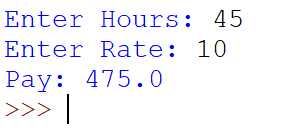
***Ans :-*** *ABC Zap ABC*

*Exercise 6: Rewrite your pay computation with time-and-a-half for overtime and create a function called computepay which takes two parameters (hours and rate).*

***Code :-***

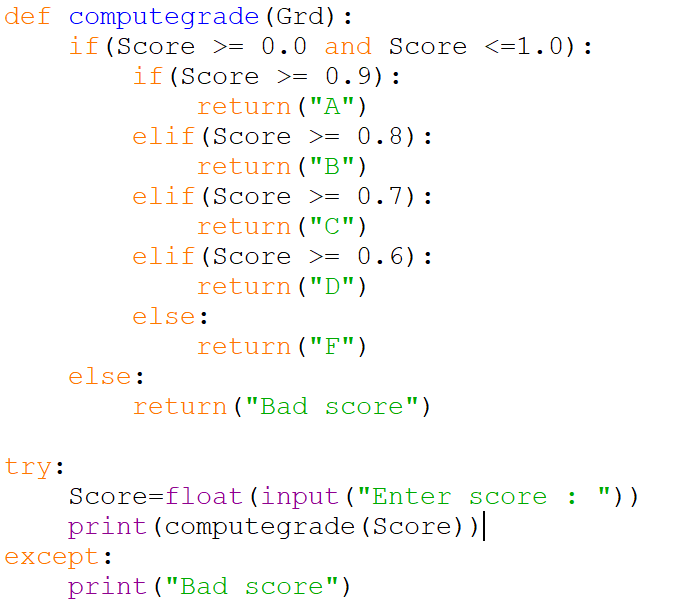


***Output :-***

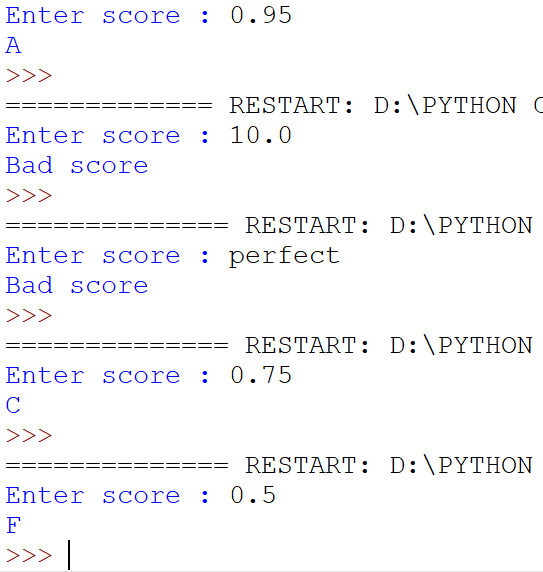


*Exercise 7: Rewrite the grade program from the previous chapter using a function called computegrade that takes a score as its parameter and returns a grade as a string.*

***Code :-***



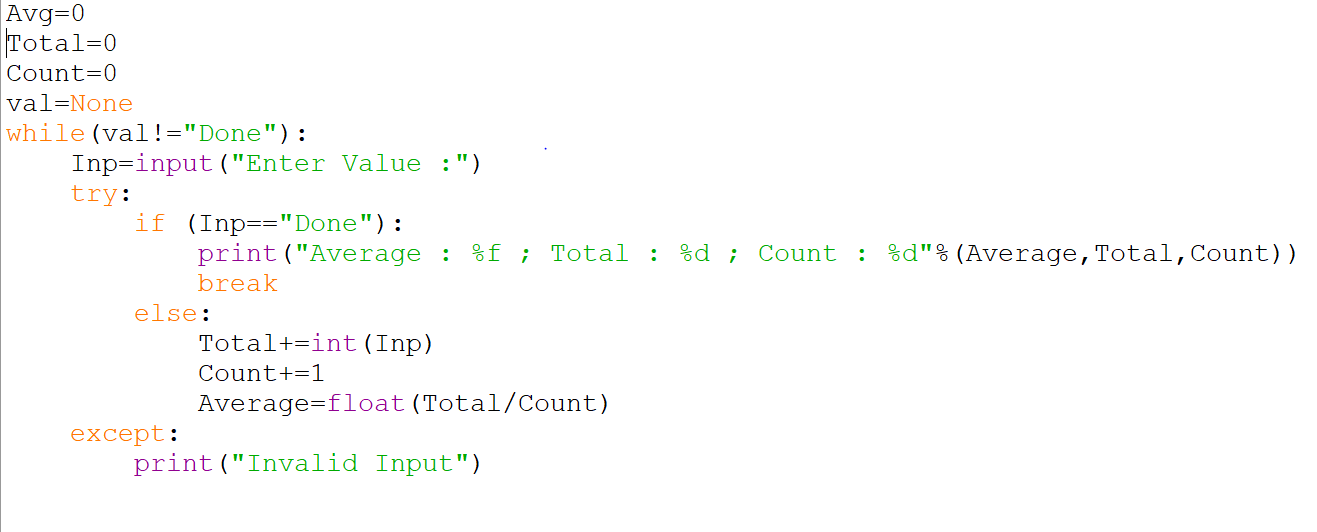
***Output :-***



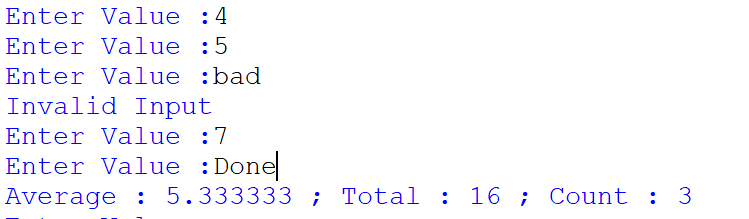
* *Chapter 5 :- ITERATION*

*Exercise 1: Write a program which repeatedly reads numbers until the user enters “done”. Once “done” is entered, print out the total, count, and average of the numbers. If the user enters anything other than a number, detect their mistake using try and except and print an error message and skip to the next number.*

***Code :-***

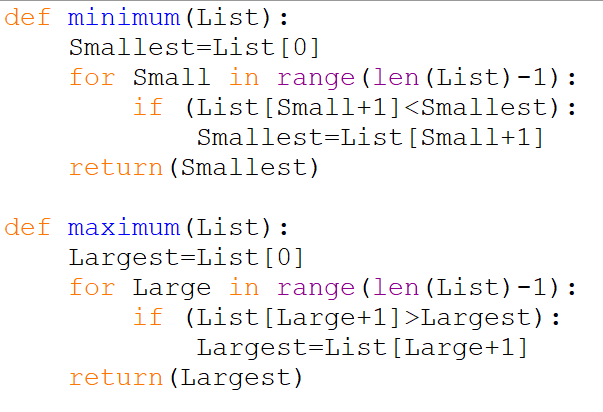


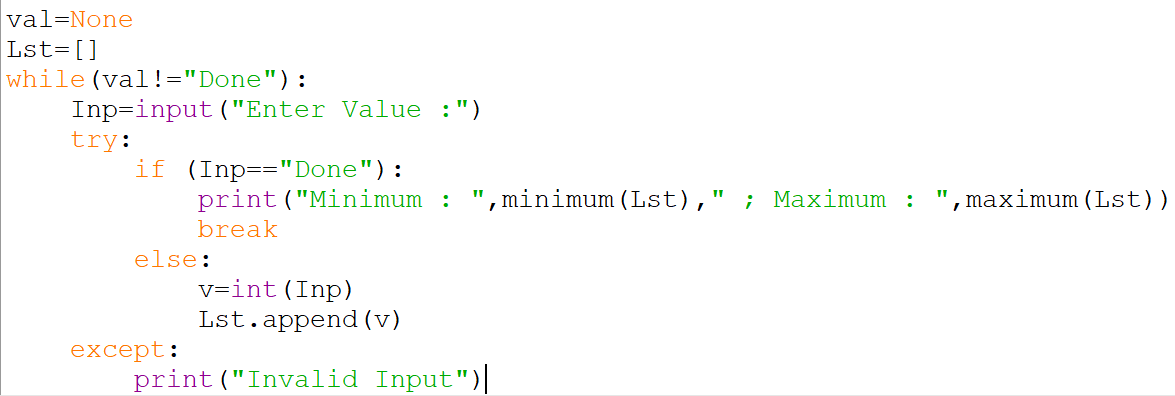
***Output :-***



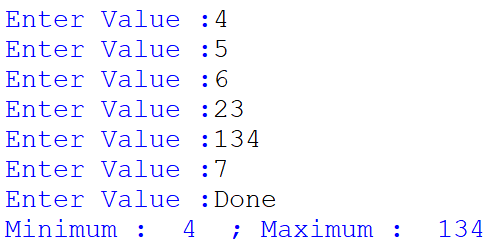
*Exercise 2: Write another program that prompts for a list of numbers as above and at the end prints out both the maximum and minimum of the numbers instead of the average.*

***Code :-***





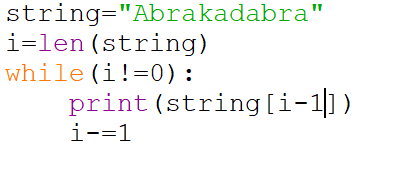
***Output :-***



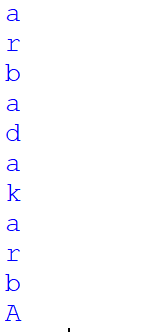
* *Chapter 6 :- Strings*

*Exercise 1: Write a while loop that starts at the last character in the string and works its way backwards to the ﬁrst character in the string, printing each letter on a separate line, except backwards.*

***Code :-***

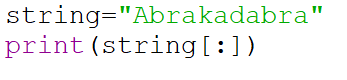


***Output :-***



*Exercise 2: Given that fruit is a string, what does fruit[:] mean?*

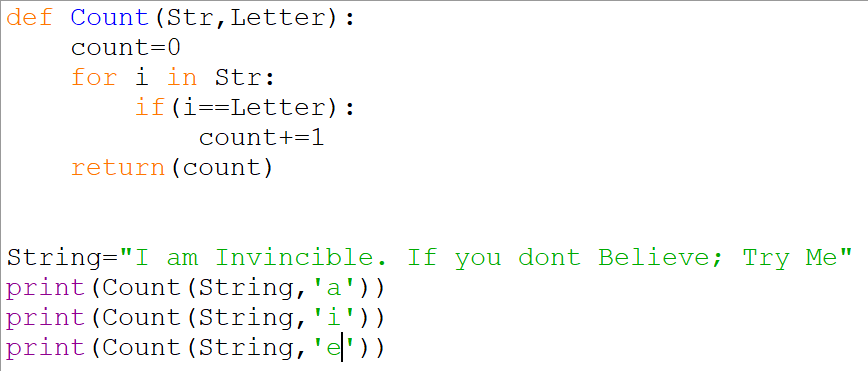
*Ans :- It will Print The Entire String*



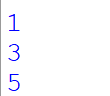


*Exercise 3: Encapsulate this code in a function named count, and generalize it so that it accepts the string and the letter as arguments.*

***Code :-***



***Output :-***

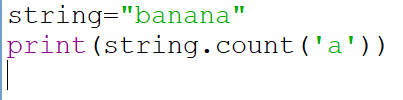


*Exercise 4: There is a string method called count that is similar to the function in the previous exercise. Read the documentation of this method at:*

*https://docs.python.org/library/stdtypes.html#string-methods*

*Write an invocation that counts the number of times the letter a occurs in “banana*

***Code :-***



***Output :-***

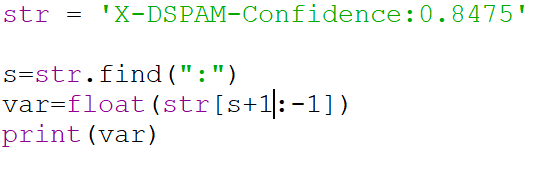


*Exercise 5: Take the following Python code that stores a string:*

*str = 'X-DSPAM-Confidence:0.8475'*

*Use find and string slicing to extract the portion of the string after the colon character and then use the float function to convert the extracted string into a ﬂoating point number.*

***Code :-***

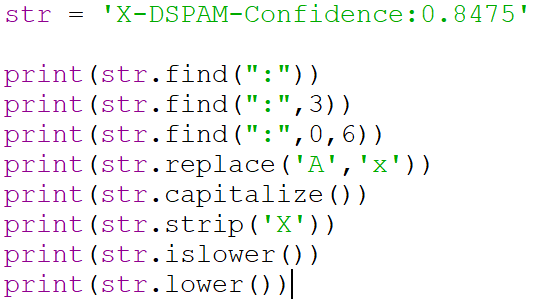


***Output :-***

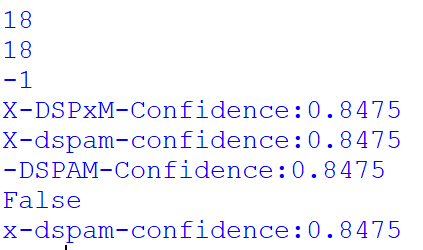


*Exercise 6: Read the documentation of the string methods at https://docs.python.org/library/stdtypes.html#string-methods You might want to experiment with some of them to make sure you understand how they work. strip and replace are particularly useful.*

***Code :-***



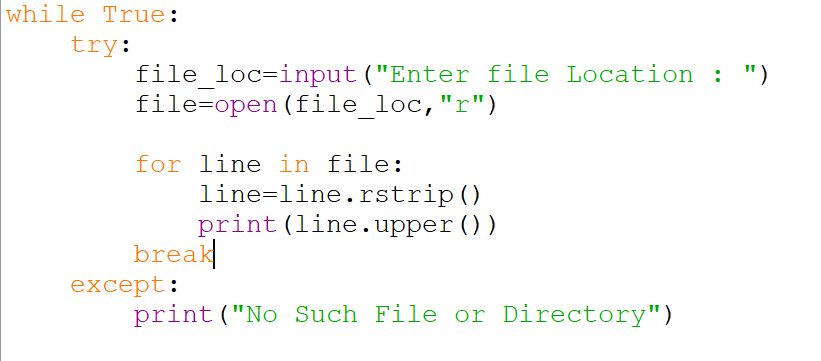
***Output :-***



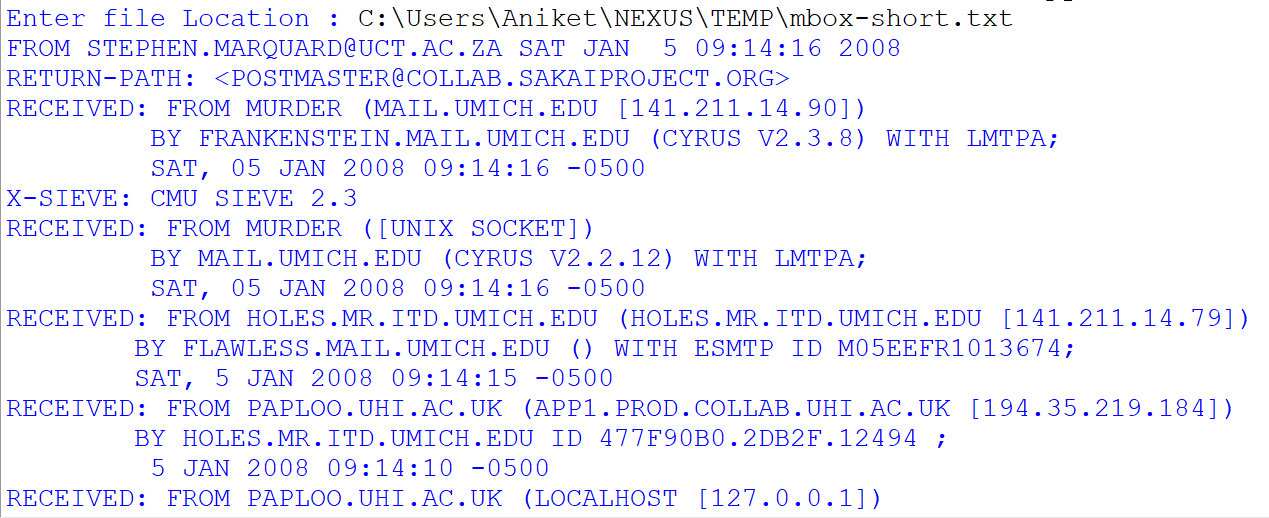
* *Chapter 7 :- Files*

*Exercise 1: Write a program to read through a ﬁle and print the contents of the ﬁle (line by line) all in upper case.*

***Code :-***

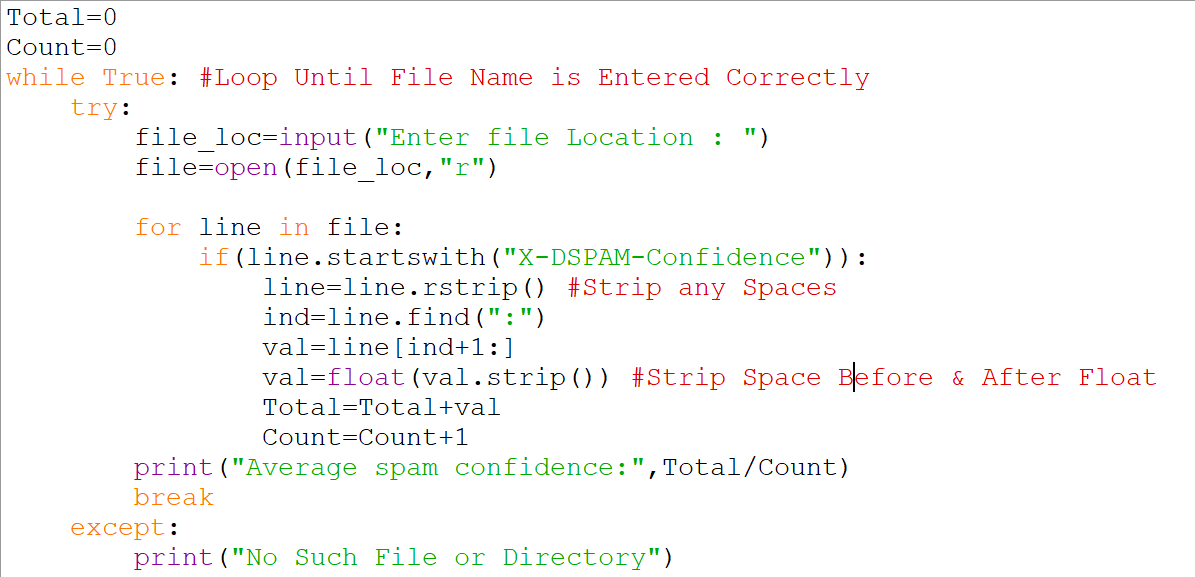


***Output :-***

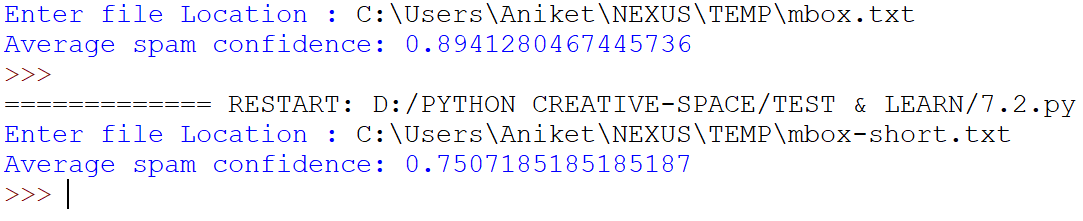


*Exercise 2: Write a program to prompt for a ﬁle name, and then read through the ﬁle and look for lines of the form:*

***Code :-***

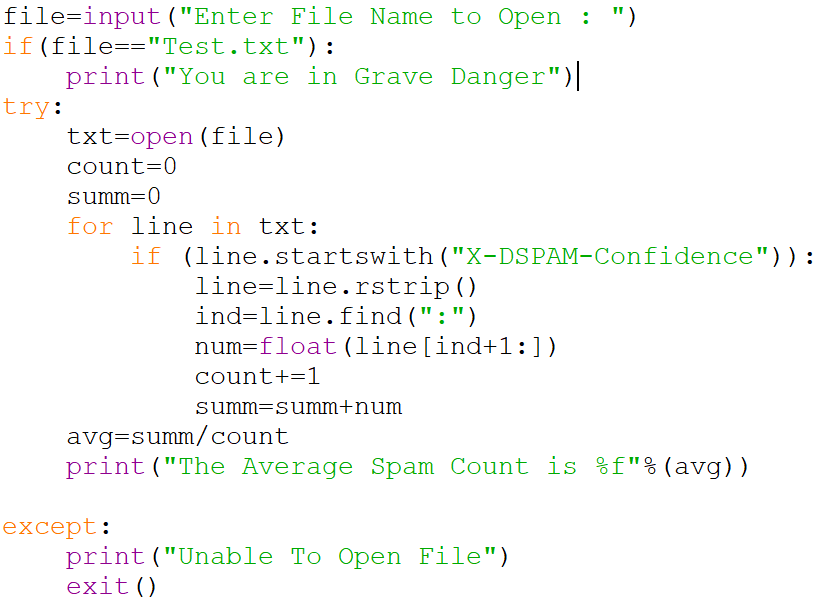


***Output :-***

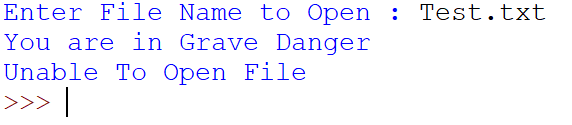


*Exercise 3: Sometimes when programmers get bored or want to have a bit of fun, they add a harmless Easter Egg to their program. Modify the program that prompts the user for the ﬁle name so that it prints a funny message when the user types in the exact ﬁle name “na na boo boo”. The program should behave normally for all other ﬁles which exist and don’t exist.*

***Code :-***



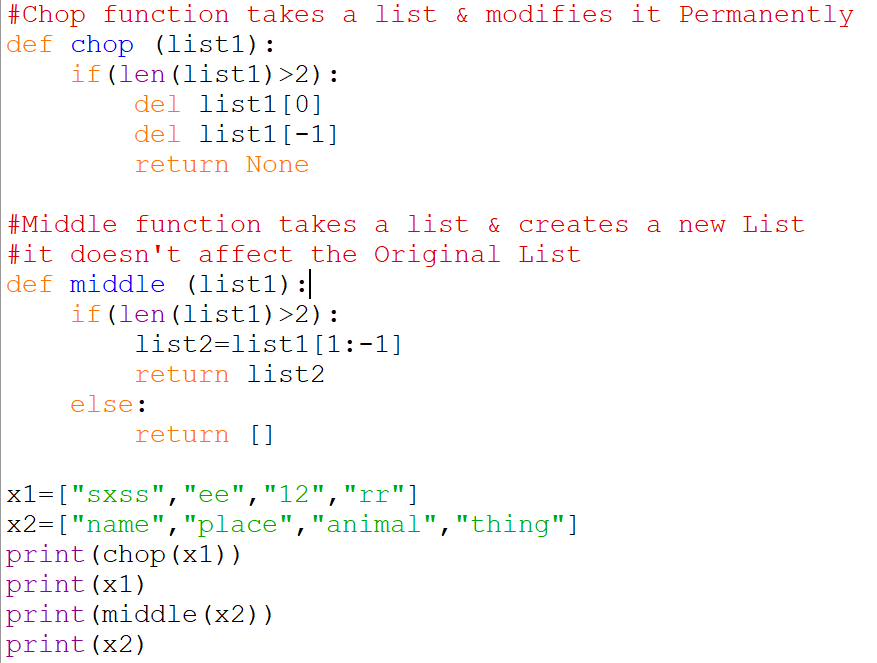
***Output :-***



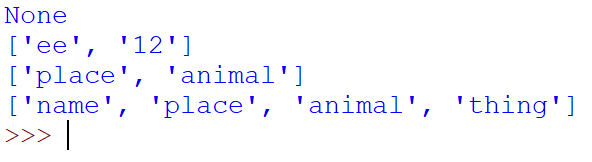
* *Chapter 8 :- Lists*

*Exercise 1: Write a function called chop that takes a list and modiﬁes it, removing the ﬁrst and last elements, and returns None. Then write a function called middle that takes a list and returns a new list that contains all but the ﬁrst and last elements.*

***Code :-***

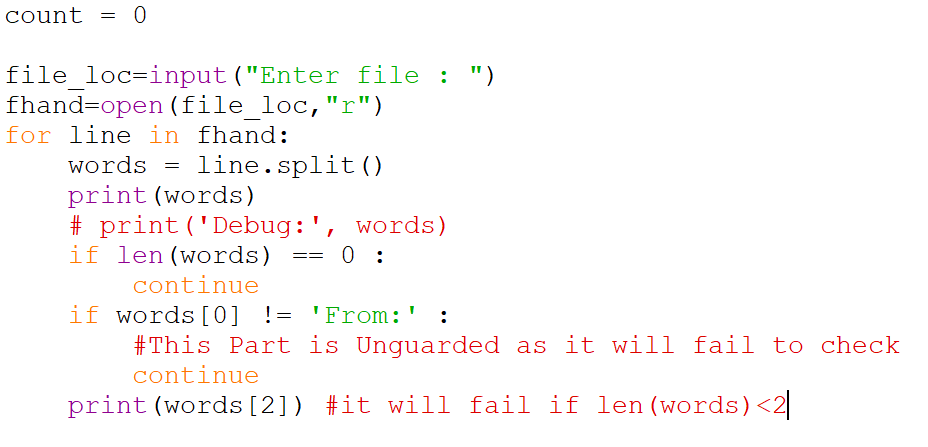


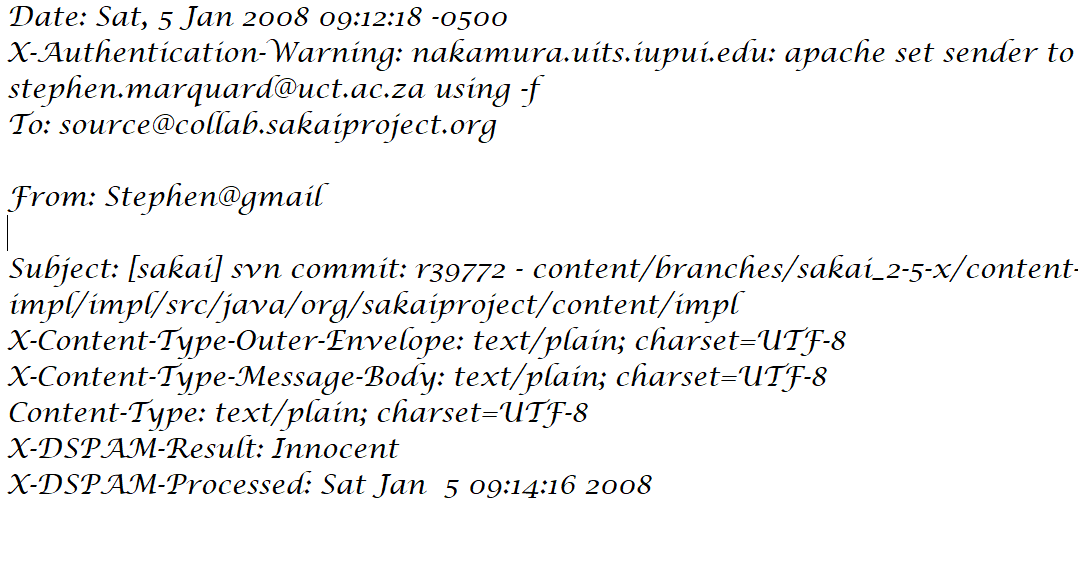
***Output :-***



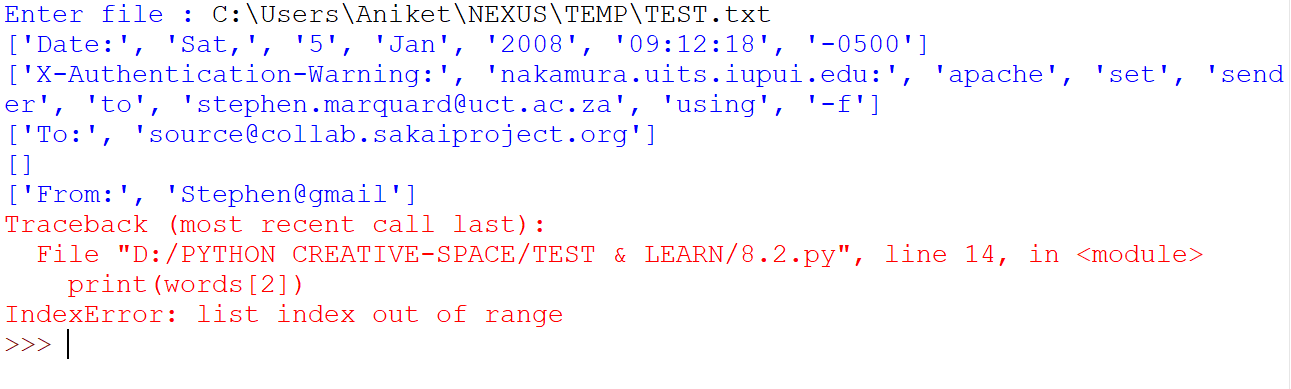
*Exercise 2: Figure out which line of the above program is still not properly guarded. See if you can construct a text ﬁle which causes the program to fail and then modify the program so that the line is properly guarded and test it to make sure it handles your new text ﬁle.*

***Code :- This Code is not Properly Guarded.***

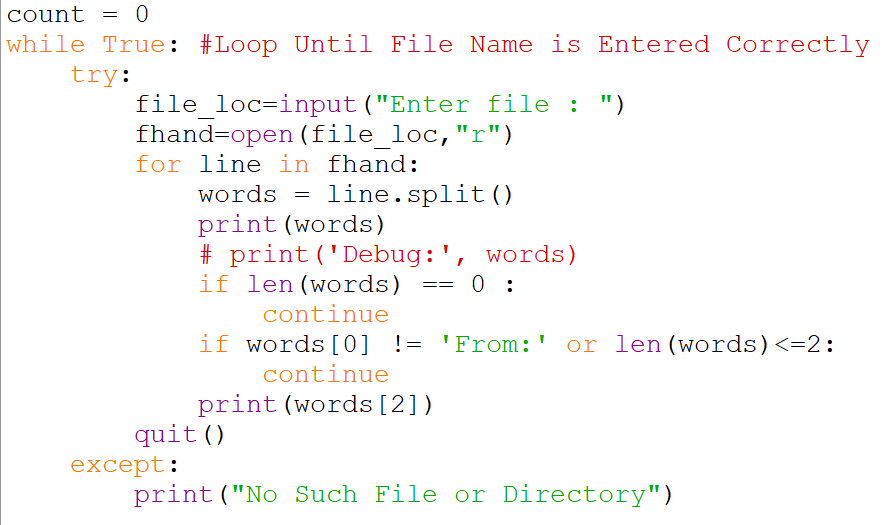




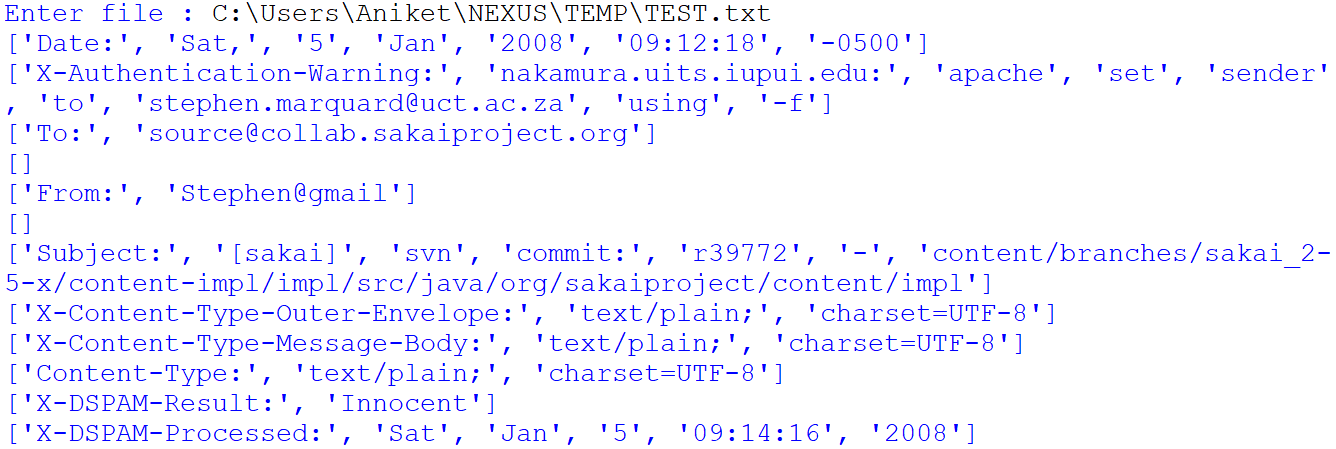
***Output :-***



***Code :-***

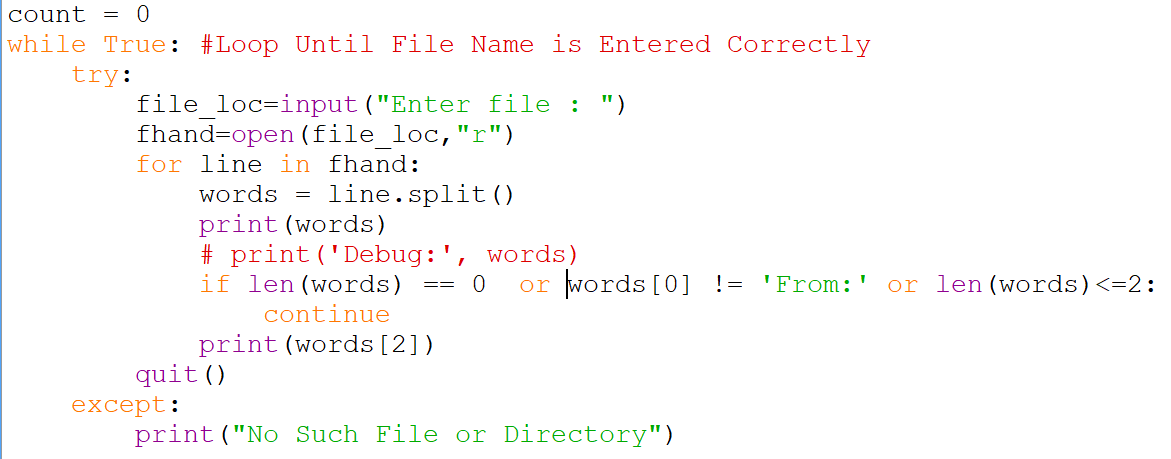


***Output :-***



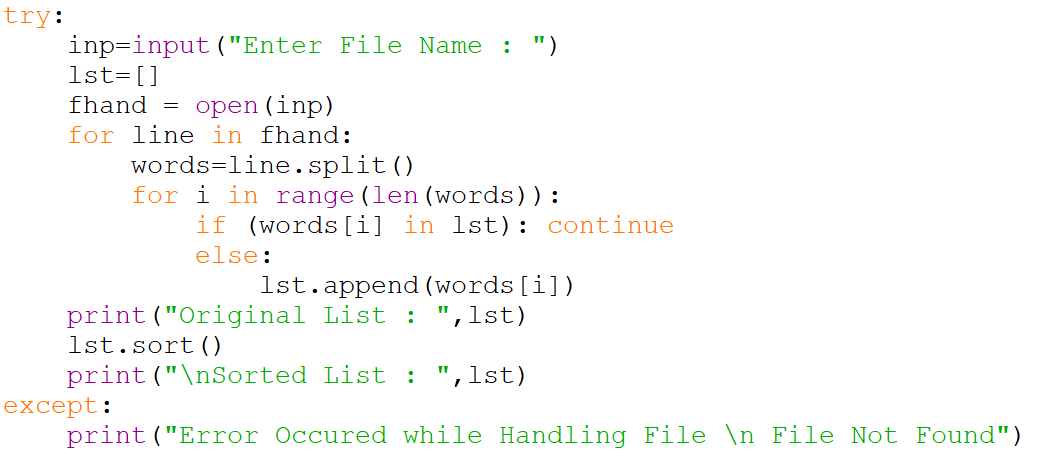
*Exercise 3: Rewrite the guardian code in the above example without two if statements. Instead, use a compound logical expression using the or logical operator with a single if statement.*

***Code :-***

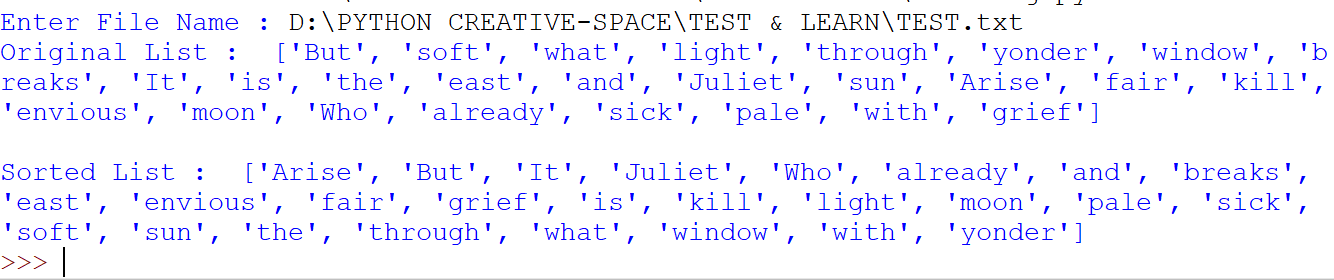


*Exercise 4: Download a copy of the ﬁle www.py4e.com/code3/romeo.txt. Write a program to open the ﬁle romeo.txt and read it line by line. For each line, split the line into a list of words using the split function. For each word, check to see if the word is already in a list. If the word is not in the list, add it to the list. When the program completes, sort and print the resulting words in alphabetical order.*

***Code :-***

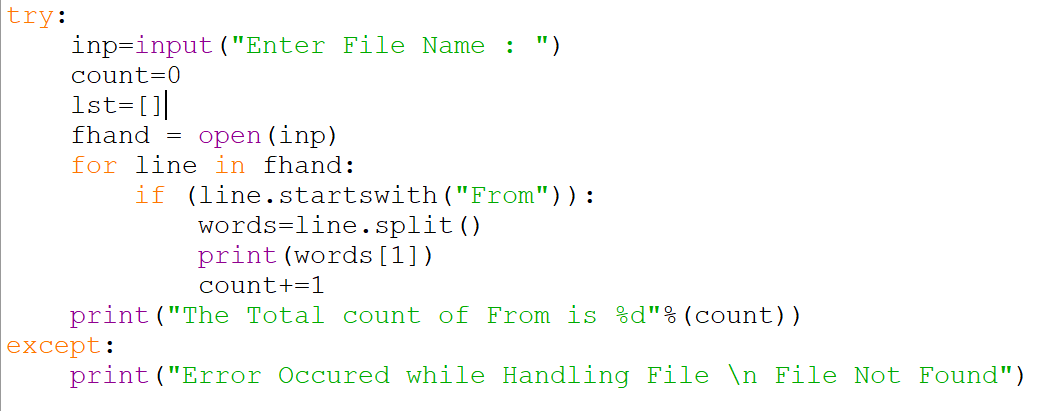


***Output :-***

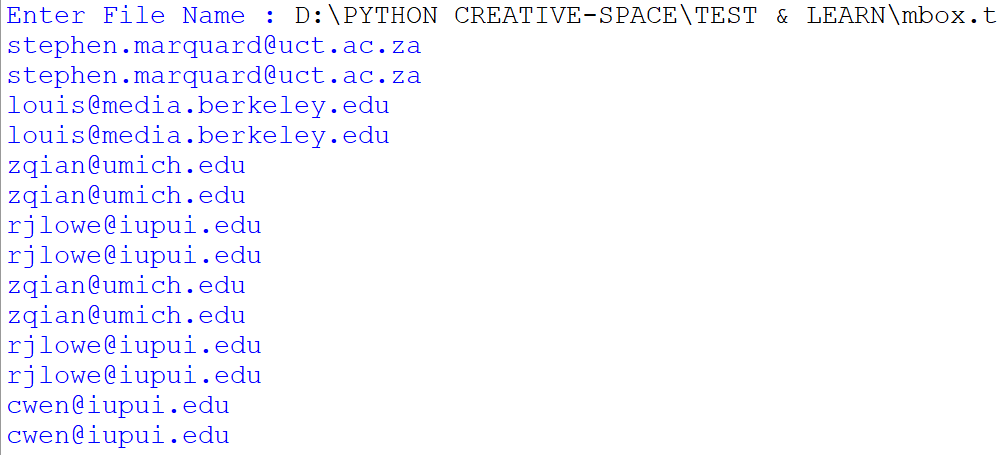


*Exercise 5: Write a program to read through the mail box data and when you ﬁnd line that starts with “From”, you will split the line into words using the split function. We are interested in who sent the message, which is the second word on the From line.*

***Code :-***



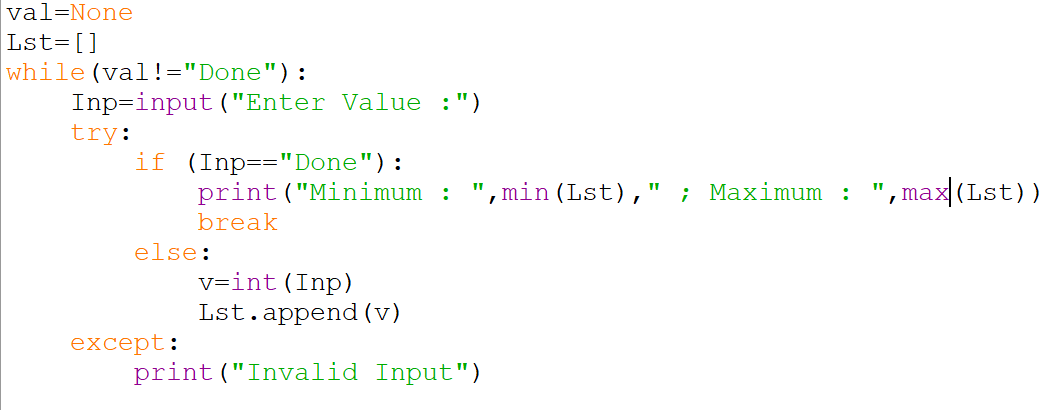
***Output :-***



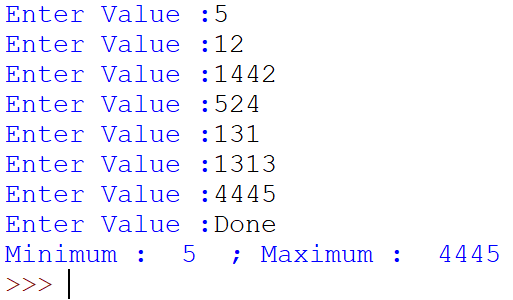


*Exercise 6: Rewrite the program that prompts the user for a list of numbers and prints out the maximum and minimum of the numbers at the end when the user enters “done”. Write the program to store the numbers the user enters in a list and use the max() and min() functions to compute the maximum and minimum numbers after the loop completes.*

***Code :-***



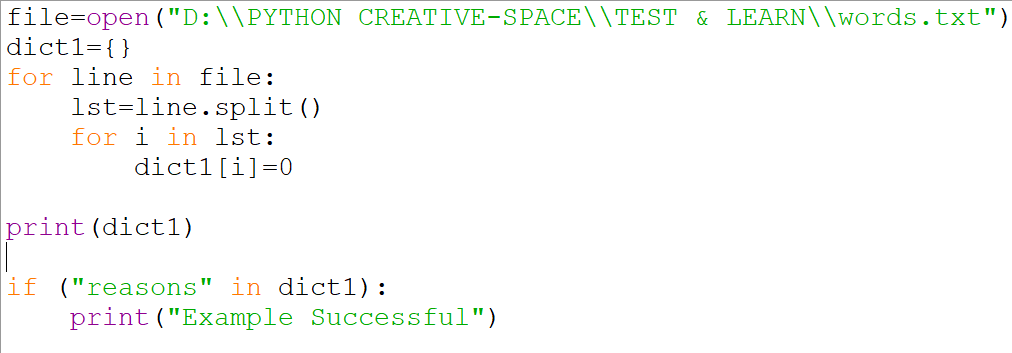
***Output :-***



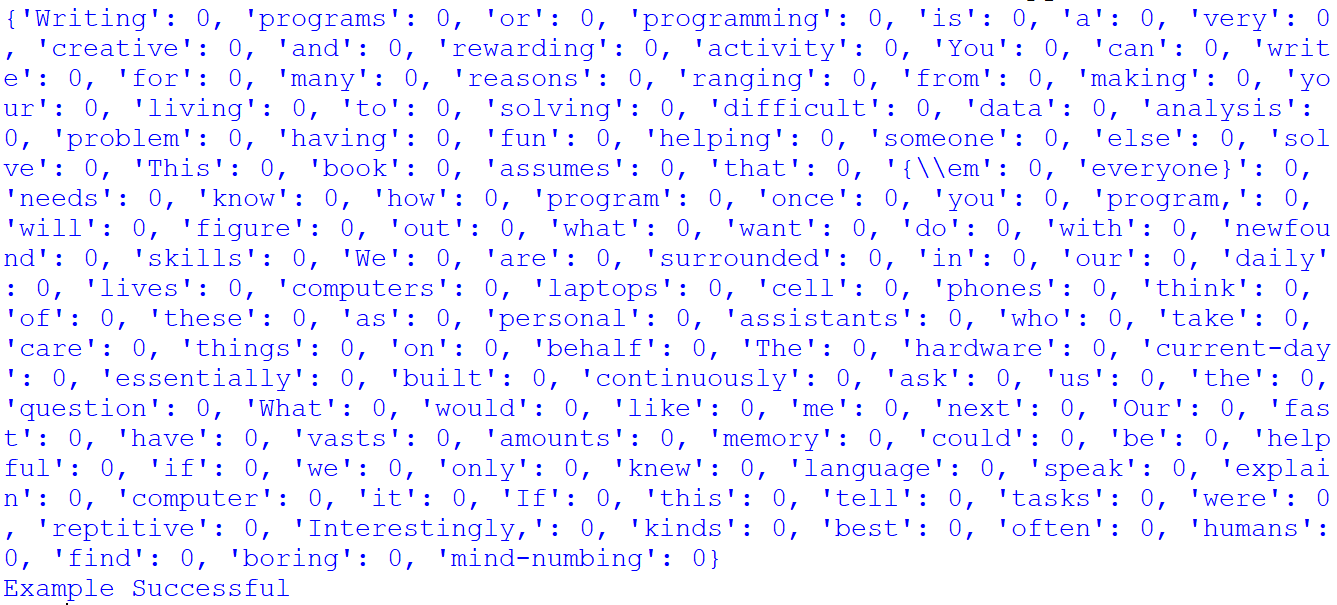
* *Chapter 9 :- Dictionary*

*Exercise 1: Download a copy of the ﬁle* [www.py4e.com/code3/words.txt](http://www.py4e.com/code3/words.txt)*. Write a program that reads the words in words.txt and stores them as keys in a dictionary. It doesn’t matter what the values are. Then you can use the in operator as a fast way to check whether a string is in the dictionary.*

***Code :-***

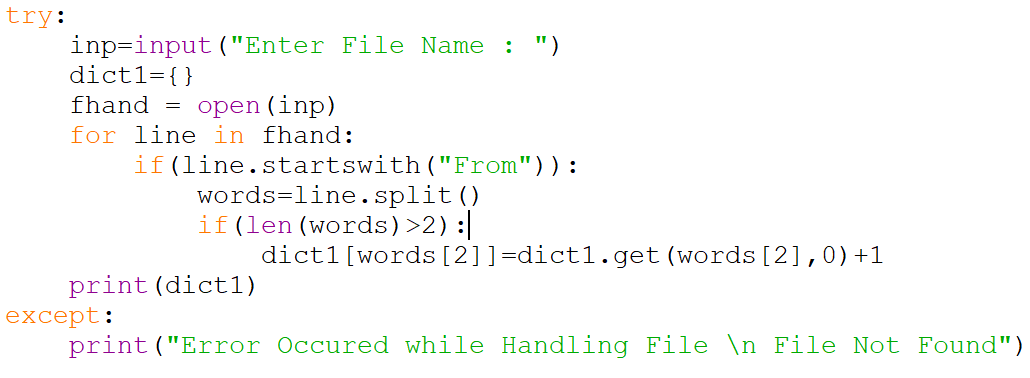


***Output :-***

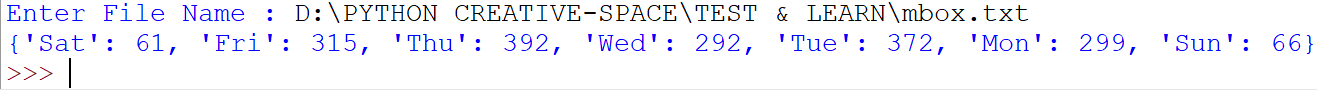


*Exercise 2: Write a program that categorizes each mail message by which day of the week the commit was done. To do this look for lines that start with “From”, then look for the third word and keep a running count of each of the days of the week. At the end of the program print out the contents of your dictionary (order does not matter).*

***Code :-***

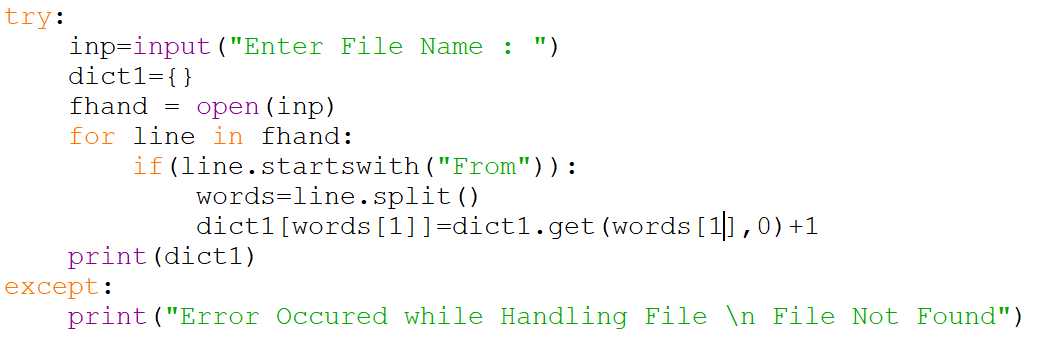


***Output :-***

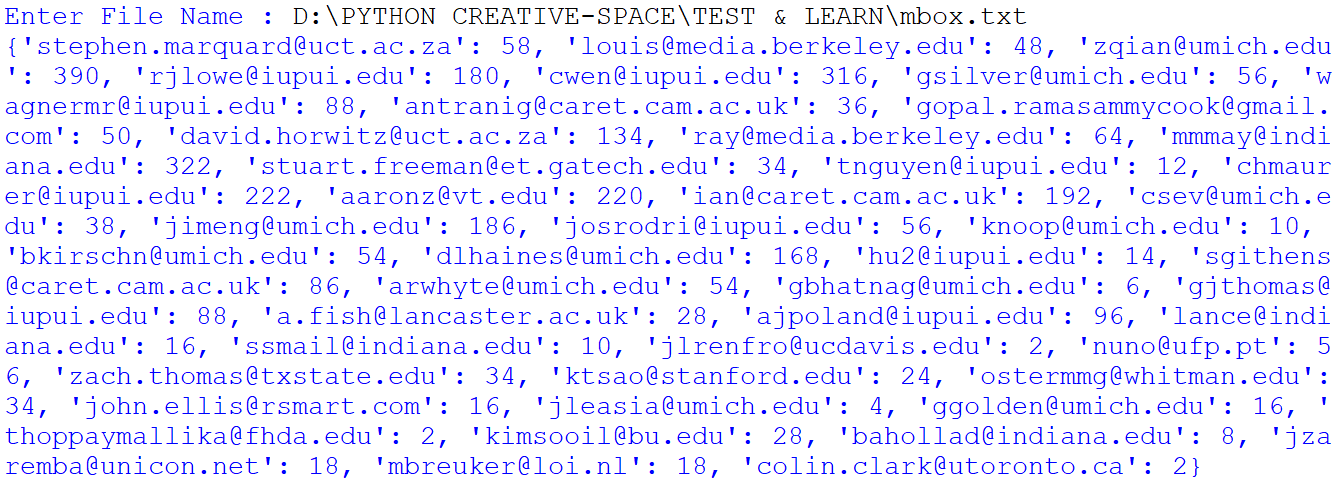


*Exercise 3: Write a program to read through a mail log, build a histogram using a dictionary to count how many messages have come from each email address, and print the dictionary.*

***Code :-***



***Output :-***



*Exercise 4: Add code to the above program to ﬁgure out who has the most messages in the ﬁle. After all the data has been read and the dictionary has been created, look through the dictionary using a maximum loop (see Chapter 5: Maximum and minimum loops) to ﬁnd who has the most messages and print how many messages the person has.*

***Code :-***

*def minimum(dict1):*

*init=None*

*key\_ind=0*

*for key,val in dict1.items():*

*if (init==None):*

*Smallest=val*

*key\_ind=key*

*init=0*

*elif(val<Smallest):*

*Smallest=val*

*key\_ind=key*

*return(key\_ind)*

*def maximum(dict1):*

*init=None*

*key\_ind=0*

*for key,val in dict1.items():*

*if (init==None):*

*Largest=val*

*key\_ind=key*

*init=0*

*elif(val>Largest):*

*Largest=val*

*key\_ind=key*

*return(key\_ind)*

*try:*

*inp=input("Enter File Name : ")*

*dict1={}*

*fhand = open(inp)*

*for line in fhand:*

*if(line.startswith("From")):*

*words=line.split()*

*dict1[words[1]]=dict1.get(words[1],0)+1*

*maxi=maximum(dict1)*

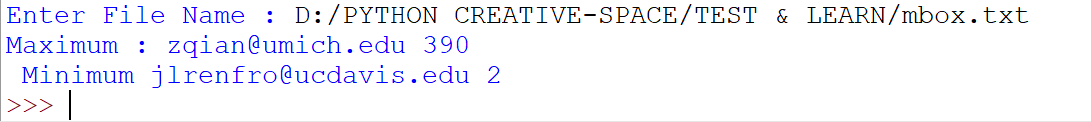
*mini=minimum(dict1)*

*print("Maximum : %s %d\n Minimum %s %d"%(maxi,dict1[maxi],mini,dict1[mini]))*

*except:*

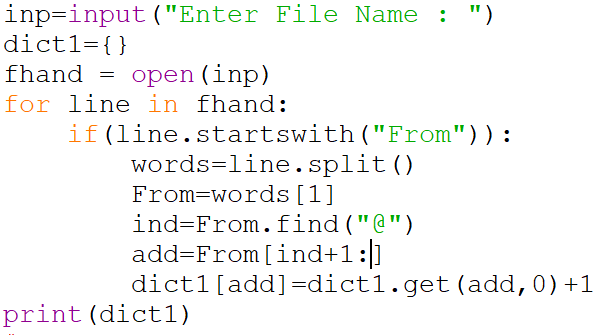
*print("Error Occured while Handling File \n File Not Found")*

***Output :-***

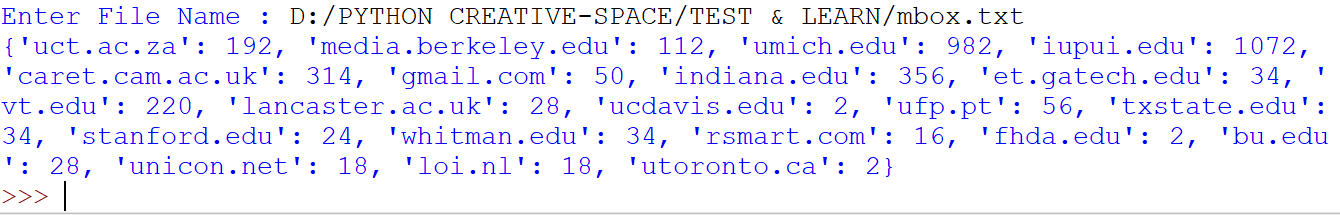


*Exercise 5: This program records the domain name (instead of the address) where the message was sent from instead of who the mail came from (i.e., the whole email address). At the end of the program, print out the contents of your dictionary.*

***Code :-***



***Output :-***

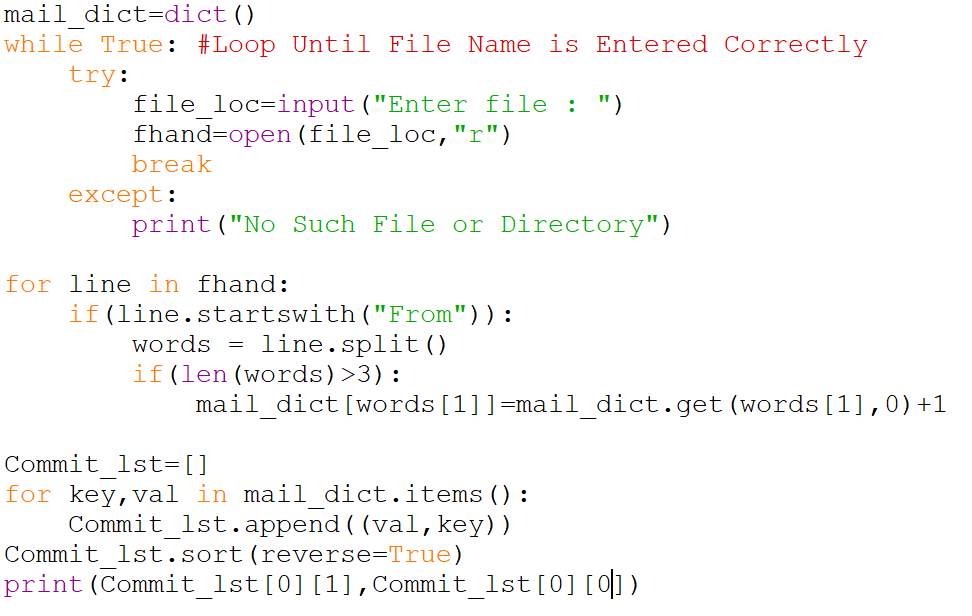


* *Chapter 10 :- Tuples*

*Exercise 1: Revise a previous program as follows: Read and parse the “From” lines and pull out the addresses from the line. Count the number of messages from each person using a dictionary.*

*After all the data has been read, print the person with the most commits by creating a list of (count, email) tuples from the dictionary. Then sort the list in reverse order and print out the person who has the most commits.*

***Code :-***

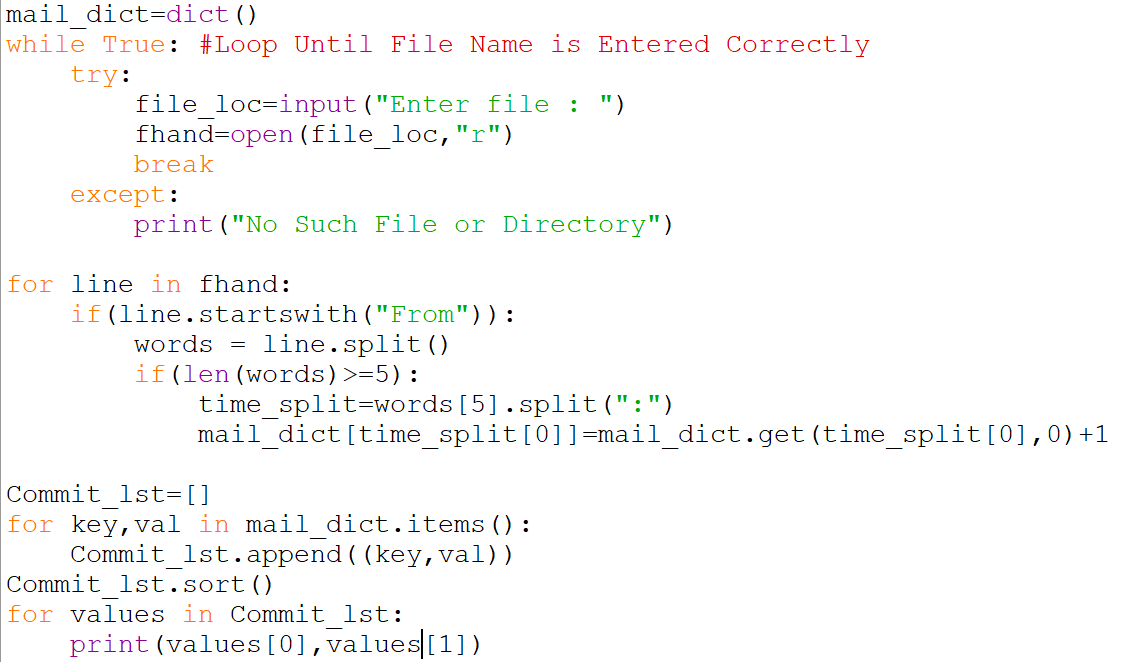


***Output :-***

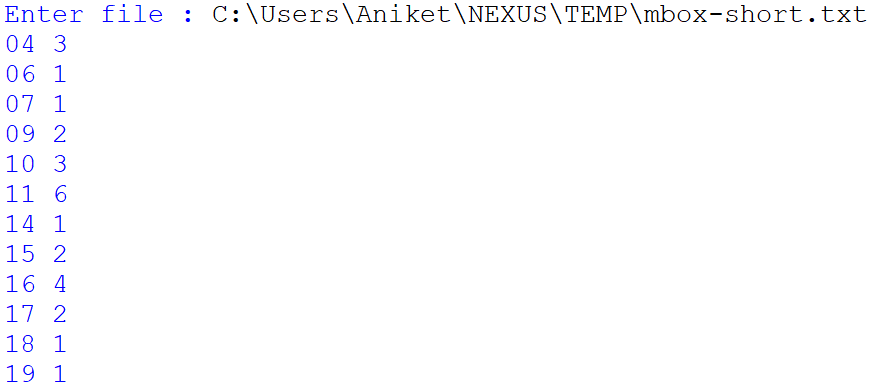


*Exercise 2: This program counts the distribution of the hour of the day for each of the messages. You can pull the hour from the “From” line by ﬁnding the time string and then splitting that string into parts using the colon character. Once you have accumulated the counts for each hour, print out the counts, one per line, sorted by hour as shown below.*

***Code :-***

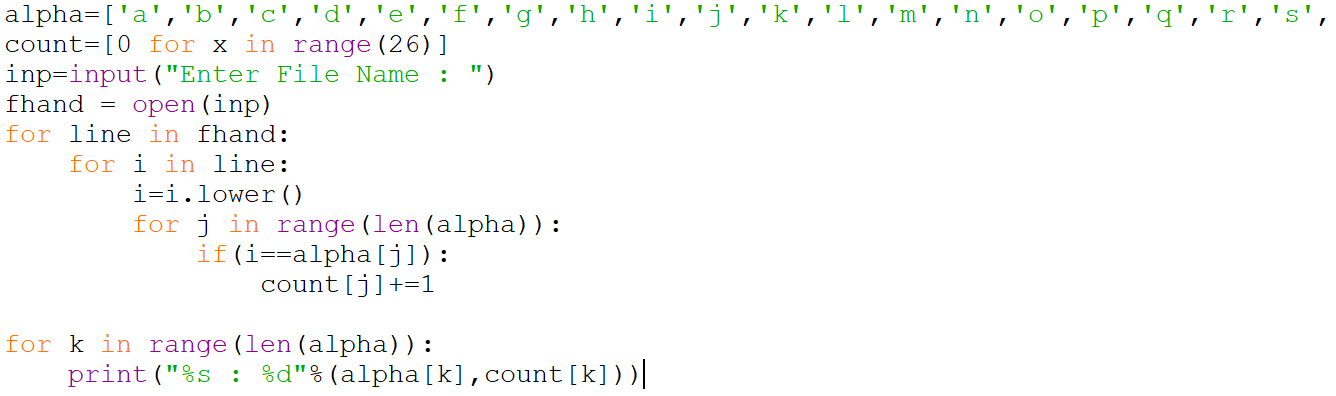


***Output :-***

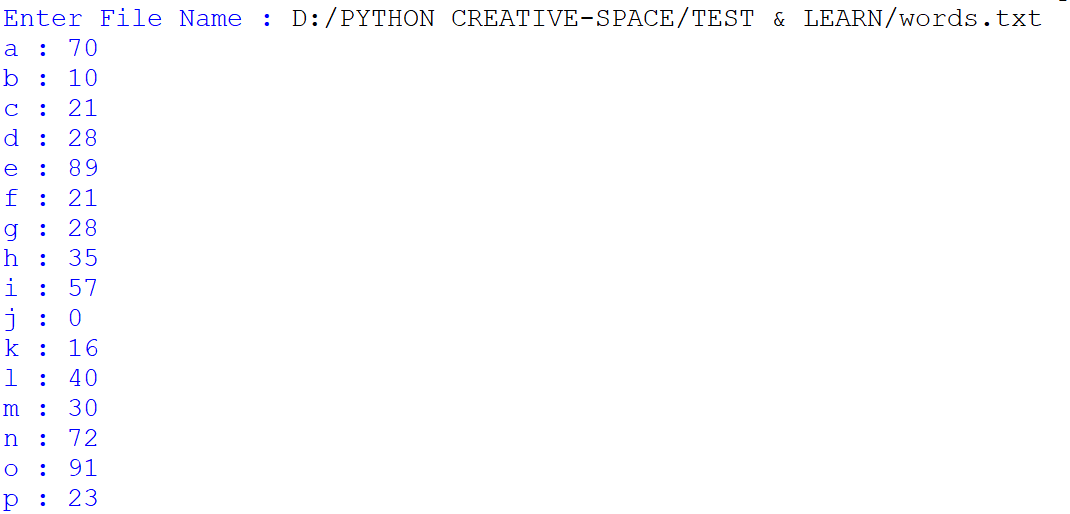


*Exercise 3: Write a program that reads a ﬁle and prints the letters in decreasing order of frequency. Your program should convert all the input to lower case and only count the letters a-z. Your program should not count spaces, digits, punctuation, or anything other than the letters a-z. Find text samples from several diﬀerent languages and see how letter frequency varies between languages.*

***Code :-***



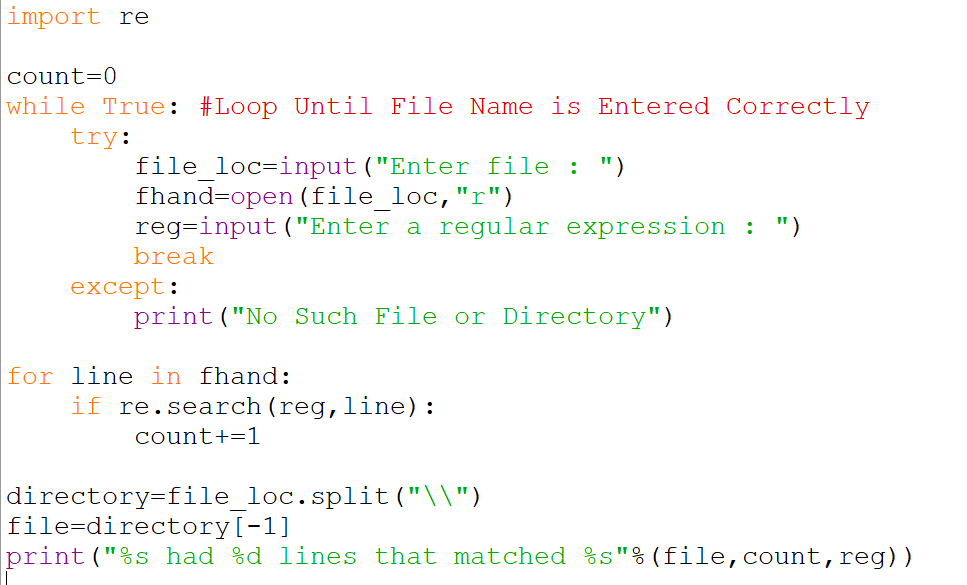
***Output :-***



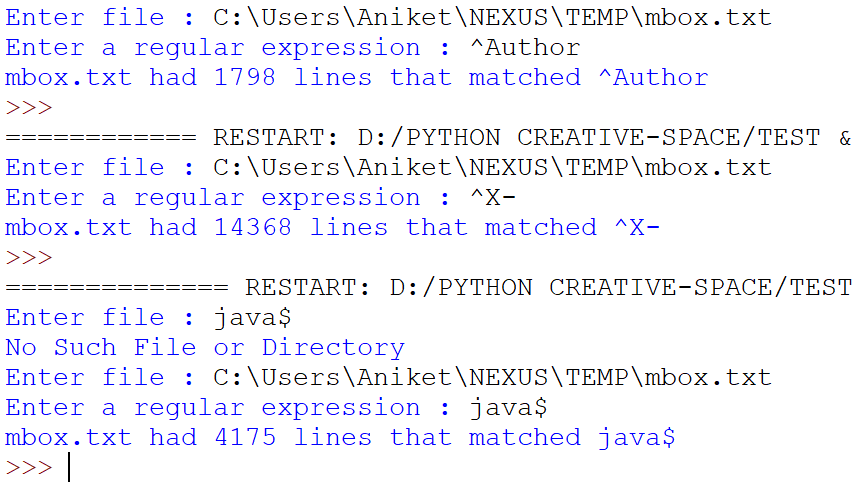
* *Chapter 11 :- Regular Expressions*

*Exercise 1: Write a simple program to simulate the operation of the grep command on Unix. Ask the user to enter a regular expression and count the number of lines that matched the regular expression:*

***Code :-***



***Output :-***

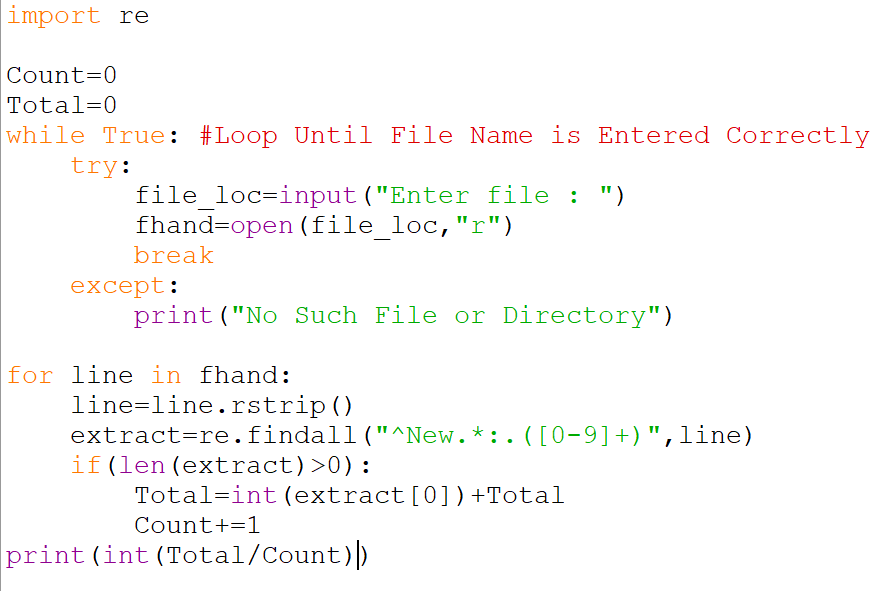


*Exercise 2: Write a program to look for lines of the form:*

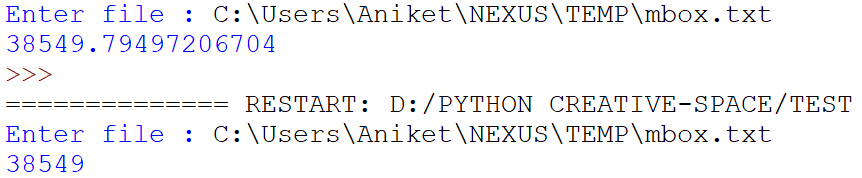
***New Revision: 39772***

*Extract the number from each of the lines using a regular expression and the findall() method. Compute the average of the numbers and print out the average as an integer.*

***Code :-***



***Output :-***



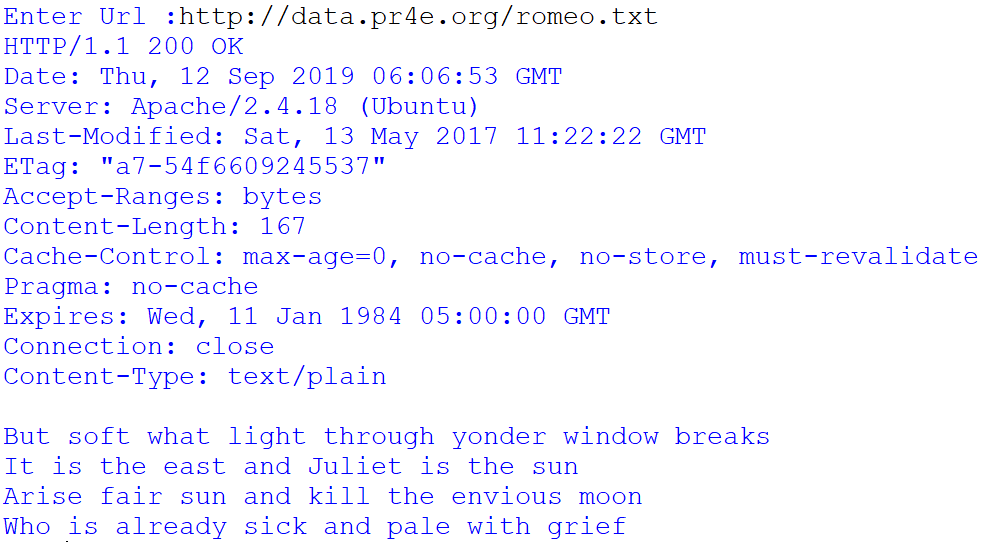
* *Chapter 12 :- Networked Programs*

*Exercise 1: Change the socket program socket1.py to prompt the user for the URL so it can read any web page. You can use split('/') to break the URL into its component parts so you can extract the host name for the socket connect call. Add error checking using try and except to handle the condition where the user enters an improperly formatted or non-existent URL.*

***Code :-***

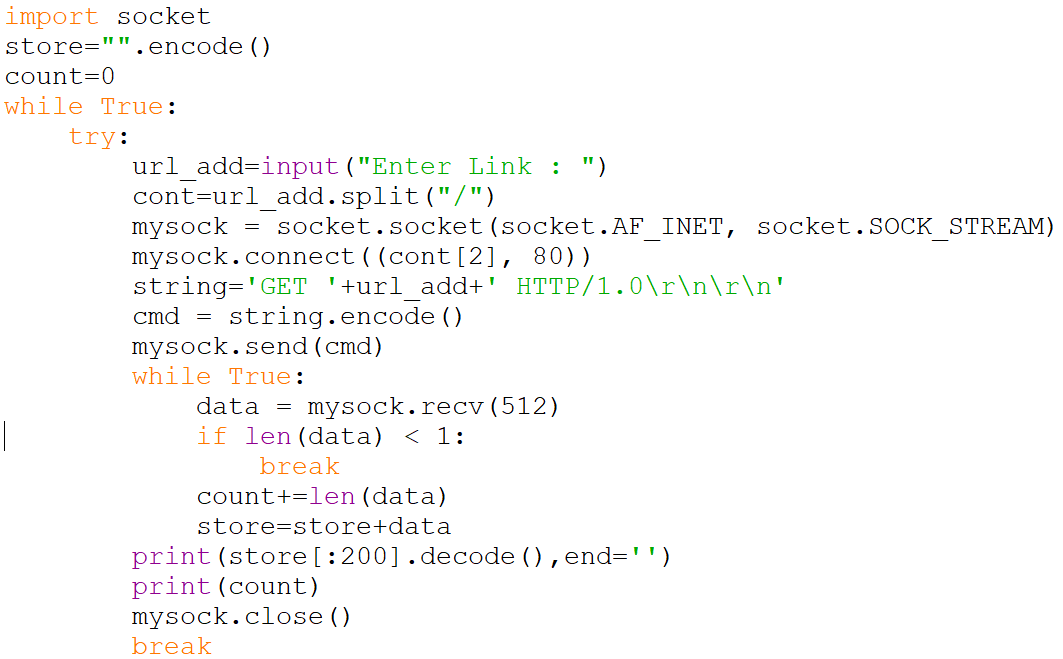


***Output :-***

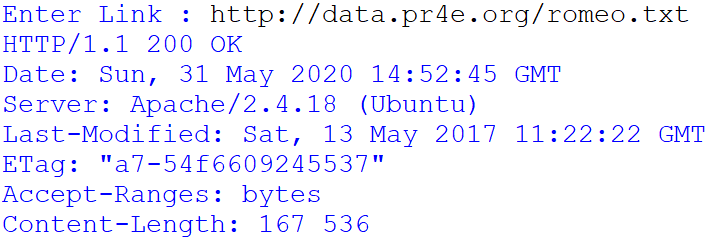


*Exercise 2: Change your socket program so that it counts the number of characters it has received and stops displaying any text after it has shown 3000 characters. The program should retrieve the entire document and count the total number of characters and display the count of the number of characters at the end of the document.*

***Code :-***

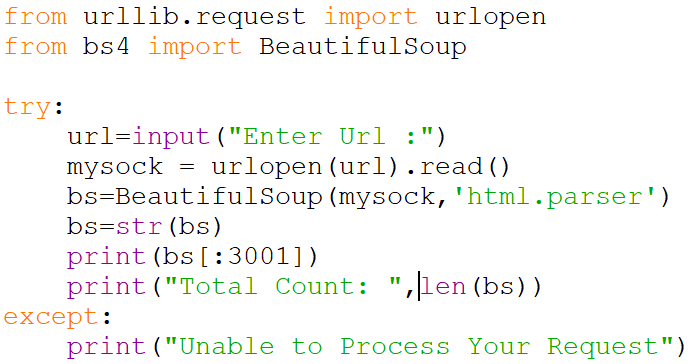


***Output :-***

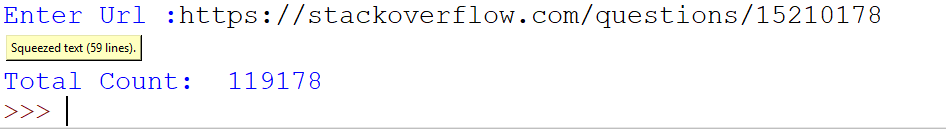


*Exercise 3: Use urllib to replicate the previous exercise of (1) retrieving the document from a URL, (2) displaying up to 3000 characters, and (3) counting the overall number of characters in the document. Don’t worry about the headers for this exercise, simply show the ﬁrst 3000 characters of the document contents.*

***Code :-***

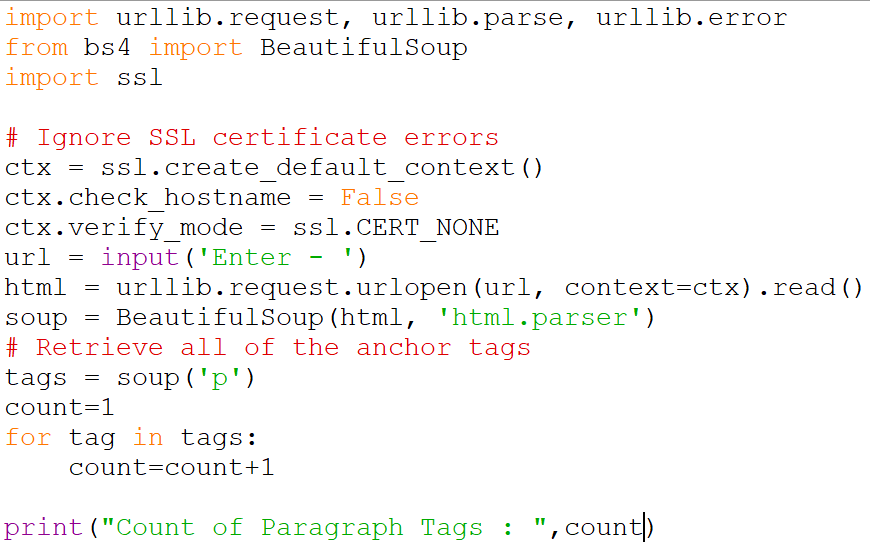


***Output :-***



*Exercise 4: Change the urllinks.py program to extract and count paragraph (p) tags from the retrieved HTML document and display the count of the paragraphs as the output of your program. Do not display the paragraph text, only count them. Test your program on several small web pages as well as some larger web pages.*

***Code :-***

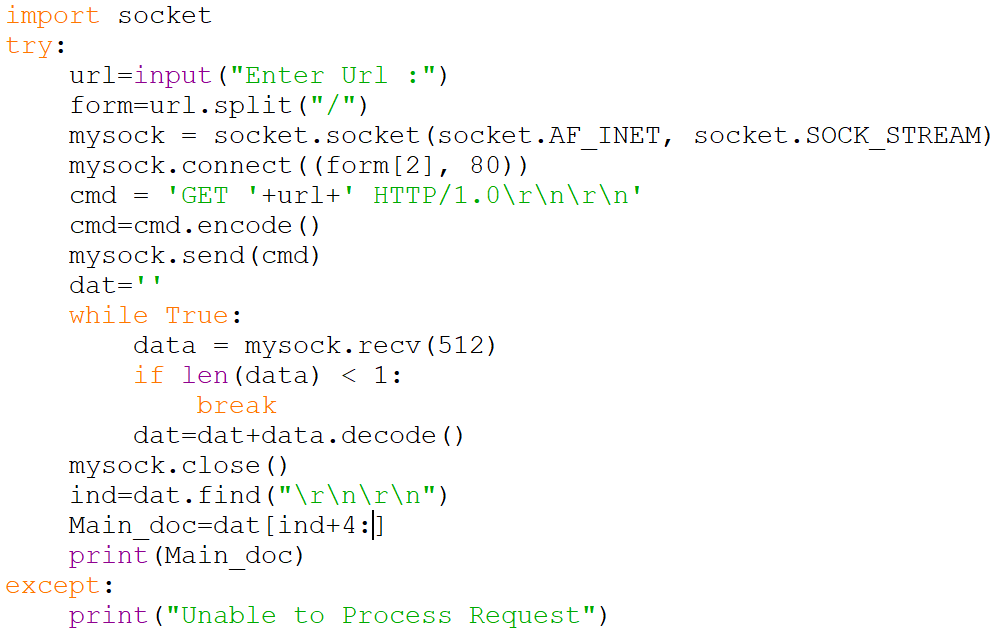


***Output :-***

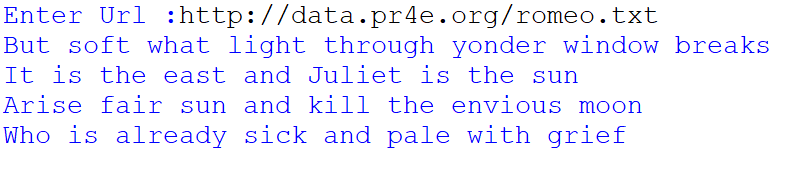


*Exercise 5: (Advanced) Change the socket program so that it only shows data after the headers and a blank line have been received. Remember that recv receives characters (newlines and all), not lines.*

***Code :-***



***Output :-***



* *Chapter 13 :- Using Web Services*

*Exercise 1: Change either geojson.py or geoxml.py to print out the twocharacter country code from the retrieved data. Add error checking so your program does not traceback if the country code is not there. Once you have it working, search for “Atlantic Ocean” and make sure it can handle locations that are not in any country.*

***Code :-***

*import urllib.request, urllib.parse, urllib.error*

*import json*

*import ssl*

*api\_key = False*

*Main\_Data=''*

*# If you have a Google Places API key, enter it here*

*# api\_key = 'AIzaSy\_\_\_IDByT70'*

*# https://developers.google.com/maps/documentation/geocoding/intro*

*if api\_key is False:*

*api\_key = 42*

*serviceurl = 'http://py4e-data.dr-chuck.net/json?'*

*else :*

*serviceurl = 'https://maps.googleapis.com/maps/api/geocode/json?'*

*# Ignore SSL certificate errors*

*ctx = ssl.create\_default\_context()*

*ctx.check\_hostname = False*

*ctx.verify\_mode = ssl.CERT\_NONE*

*ans="y"*

*while (ans=="y"):*

*address = input('Enter location: ')*

*if len(address) < 1:*

*break*

*parms = dict()*

*parms['address'] = address*

*if api\_key is not False:*

*parms['key'] = api\_key*

*url = serviceurl + urllib.parse.urlencode(parms)*

*print('Retrieving', url)*

*uh = urllib.request.urlopen(url, context=ctx)*

*data = uh.read().decode()*

*print('Retrieved', len(data), 'characters')*

*try:*

*js = json.loads(data)*

*except:*

*js = None*

*if not js or 'status' not in js or js['status'] != 'OK':*

*print('==== Failure To Retrieve ====')*

*print(data)*

*continue*

*Main\_Data=json.dumps(js, indent=4)*

*print(json.dumps(js, indent=4))*

*lat = js['results'][0]['geometry']['location']['lat']*

*lng = js['results'][0]['geometry']['location']['lng']*

*print('lat', lat, 'lng', lng)*

*location = js['results'][0]['formatted\_address']*

*print(location)*

*Main\_Data=json.loads(Main\_Data)*

*cou=Main\_Data["results"]*

*lst=cou[0]["address\_components"]*

*print("Country Code :", lst[-1]['short\_name'])*

*ans=input("Do You Want to Continue (y/n): ")*

***Output :-***

