PROG12583 Assignment 2

1249

This assignment must be done as individual work. You should not use any resources that generate code, such as ChatGPT, in the completion of this assignment.

When asked to name files with <username>, please replace <username> with your SLATE/Sheridan College username. It is often composed of your last name (or at least the first couple of characters) plus some letters from your first name and/or some numbers. For example mine is “forestke”.

Part 1:

Create a file called <username>_a2_p1.py. Write a program that asks the user to input a number between 0 and 500 (exclusive). If the user enters a value that is not a number, or a number that lies outside that range, the program should re-prompt the user for another input. If the user enters more than 5 invalid inputs in a row, the program should ask the user if they want to continue. If the user enters ‘n’ or ‘N’, the program should end. If the user enters ‘y’, ‘Y’, or just hits enter, the program should continue by asking the user to input a number. The number represents a value, in cents, for which the program will calculate the minimum number of coins required to make that amount. For example, if the user enters the value of 16, the program should calculate that making that amount of change will require 1 dime (10 cents), 1 nickel (5 cents), and 1 penny (1 cent). The program should ask the user if they want to continue. If the user enters ‘n’ or ‘N’, the program should end. If the user enters ‘y’, ‘Y’, or just hits enter, the program should continue by asking the user to input a new number.

```
Welcome to the change calculator

Enter the value in cents (0-500): 345

Twoonies:  1
Loonies:   1
Quarters:  1
Dimes:     2
Nickles:   0
Pennies:   0

Would you like to continue? (Y/n): n
Ok, see you later!
```

Feel free to come up with your own spin on the messages, especially the message you see

when the program ends. You can find a more detailed running version of this program on the course website.

Part 2:

Create a file called <username>_a1_p2.py. Write a program that accepts a number from the user. The number must be positive, and must lie in the range of 0 to 10000. Once you get a valid number, calculate all of the factors for that number. A factor is a number that divides evenly into another number (that is, it doesn't leave a remainder). For example, the factors of 10 are 1, 2, 5, and 10. Numbers like 3 and 8 aren't factors of 10, because they don't divide evenly into 10 (10 / 3 has a remainder of 1, while 10 / 8 has a remainder of 4).

Once you have calculated all of the factors, print a brief message to the user that tells them how many factors have been found. If the number is a prime number (that is the only factors of that number are 1 and itself), you should also print a message notifying the user that the number is prime.

Once you've printed that brief message to the user, ask the user if they want to see a list of all the factors. If the user enters 'y', 'Y', or just hits enter, print the list of factors. If the user instead enters 'n' or 'N', don't print anything. Keep prompting the user until they enter one of the valid values.

Part 3:

Create a file called <username>_a2_p3.py. Write a program that accept a list of strings, each of which is separated from the one before it by a space character ' '. The user should enter the words all in one go and only hit enter at the very end.

Once you've got the list of words your program should split it into a list of individual strings. Print out a brief message letting the user know how many strings they've entered.

Now comes the fun part. Create a menu that gives the users the following options:

1. Clear list
2. Print list
3. Alphabetize list
4. Add word(s) to list
5. Delete word(s) from list
6. Remove duplicate words from list
7. Exit program

For each menu option, implement the necessary functionality to your program. You are going to have to do some research online on how to figure out how to do some of these things (or check out the textbook), but that is all part of the learning!

Your program should sit in the menu and respond to user input. When the user enters a number that corresponds to one of the menu options you should do the necessary actions (including getting more information from the user, modifying the list, printing things, etc…), after which your program should return to the main menu.

For the menu items 1. Clear list, 6. Remove duplicate words from list, and 7. Exit program, you should ask the user if they are certain they want to perform the selected action. For example, you could use a prompt asking "Are you sure? Enter yes/no".

For menu item 3. Alphabetize list, you should also set an "alphabetized" flag that lets the program know the list has been alphabetized. This flag will be used when the user tries to add new words to the list.

For menu item 5. Delete words(s) from list, allow the user to enter a list of words they want to delete from the list as a string, with each word separated by a space. Remove any instances of those words that exist in the current list. As you remove words, keep track of how many words you removed. When you are done the removal option, print a message to the user letting them know how many words got removed from the list. Ask the user if they want more information. If they do, give them a detailed breakdown of which words got removed and which words couldn't be found in the list. Once you are done removing the words, go back to the main menu.

For menu item 4. Add word(s) to list, allow the user to enter the list of words they want as a list of strings, with each word separated by a space. Once the user has added the strings, check if the "alphabetized" flag is set (you will set the flag if the user previously alphabetized the list). If the flag is set, ask the user if they want to keep the list alphabetized after it adds the new words, or if the user wants to add the new words to the end of the list. If the user wants to keep the list alphabetized make sure that after you add the new words the list stays alphabetized (and keep the "alphabetized" flag set). If the user doesn't want to keep the list alphabetized, add the new words to the end of the list and clear the "alphabetized" flag. Once you are done adding new word(s), ask the user if they want to add more words to the list, or if they want to return to the main menu.

Submission Requirements:

Create a .zip file named <username>_assignment_2.zip that contains your 3 programs. The programs should be in the root of the .zip file (the .zip file should not contain any directories). Submit the .zip file to the assignment dropbox.

Grading Scheme:

/3 Part 1
/5 Part 2
/10 Part 3
/2 Submission requirements met
/2 Coding conventions and comments

/22 Total