While Loops Challenge 26: Factor Generator App

Description:
You are responsible for writing a program that generates all factors of a given number. Your program will display the factors individually and give a mathematically summary of how different pairs of factors can be multiplied together to get the given number.

Step By Step Guide:

- Print a welcome message.
- Create an active flag variable to control a while loop and set it to True.

- Use this flag to run a while loop:
    - Get user input for a number to determine the factors of.

    - Write an algorithm to determine the factors of the given number.
        - Create a blank list called factors.
        - Use a for loop to loop through the numbers 1 up to and including the given number:
            - If the given number is divisible by the current number in the loop, the current number is a factor:
                - Add the current number to the list factors.
                - Modulo division will help you here.

    - Print all factors of the number as formatted below.

    - Print a summary which shows how the factors multiply together to get the users number using a for loop.
        - The first element in your list will pair with the last element.
            - factors[0] an factors[-1]
        - The second element will pair with the second to last element.
            - factors[1] and factors[-2]
        - The third element will pair with the third to last element.
            - factors[2] and factors[-3]
        - Try to generalize this pattern using an iterable variable i.
        - You only need to loop through half of your list to accomplish this.
            - for i in range(int(len(factors)/2))

    - Get user input for if they would like to continue the program.
    - If the user does not want to continue:
        - Set your flag variable to False
        - Print a goodbye message thanking the user.

- Use at least 2 comments to describe sections of your code.
- “Chunk” your code so that is readable.

- Use appropriate and informative variable names.
- Format your output as below.

Example Output 1:

Welcome to the Factor Generator App

Enter a number to determine all factors of that number: 44

Factors of 44 are:
1
2
4
11
22
44

In summary:
1 * 44 = 44
2 * 22 = 44
4 * 11 = 44

Run again (y/n): n
Thank you for using the program. Have a great day.

Example Output 2:

Welcome to the Factor Generator App

Enter a number to determine all factors of that number: 100

Factors of 100 are:
1
2
4
5
10
20
25
50
100

In summary:
1 * 100 = 100
2 * 50 = 100
4 * 25 = 100
5 * 20 = 100

Run again (y/n): y

Enter a number to determine all factors of that number: 33

Factors of 33 are:
1
3
11
33

In summary:
1 * 33 = 33
3 * 11 = 33

Run again (y/n): n
Thank you for using the program. Have a great day.

While Loops Challenge 27:
Even Odd Number Sorter App

Description:
You are responsible for writing a program that sorts a list of comma separated numbers as either even or odd. Upon sorting the numbers into two groups, your program will then sort each group numerically and display the results.

Step By Step Guide:

- Print a welcome message.
- Create an active flag variable that will control a while loop and set it to True.

- Use this flag to run a while loop:
    - Get user input for a series of numbers separated by a comma.
        - This series of numbers will be a string, don't change this for now.
        - Take precaution to remove any spaces the user may have put into the number using the .replace() method for strings.
        - Replace " " a space with "" a blank string.
        - The user may enter "1, 2, 3, 4," or they may enter "1,2,3,4".
        - You want to have the results of the program be the same regardless.

    - Turn the string into a list using the .split() method for strings.
        - Each element in our list should be a number.
        - The string should be split using a comma.
        - For example, if i have a list values = ['1,2,3,4,5'] calling values.split(",") would return a list ['1', '2', '3', '4', '5'].
        - Store this return value in a variable called num_list.

    - Create a blank list called evens.
    - Create a blank list called odds.

    - For every number in your list of numbers:
        - Cast the number to an integer.
        - Recall a number is even if it is divisible by 2.
        - If the number is even:
            - Append the number to the evens lists.
            - Print a statement that the number is even.
        - Else:
            - Append the number to the odds lists.
            - Print a statement that the number is odd.

    - Permanently sort your even numbers in numerical order.
    - Permanently sort your odd numbers in numerical order.

    - Display all the even numbers in numerical order as formatted below.
    - Display all odd numbers in numerical order as formatted below.

    - Get user input for if they would like to continue the program.
    - If the user does not want to continue:
        - Set your flag variable to False
        - Print a goodbye message thanking the user.

- Use at least 2 comments to describe sections of your code.
- “Chunk” your code so that is readable.
- Use appropriate and informative variable names.
- Format your output as below.

Example Output:

Welcome to the Even Odd Number Sorter App

Enter in a string of numbers separated by a comma (,) : 1, 2, 3, 6, 18, 9, 4, 13, 22

---- Result Summary ----

1 is odd!
2 is even!
3 is odd!
6 is even!
18 is even!
9 is odd!
4 is even!
13 is odd!
22 is even!

The following 5 numbers are even:

2
4
6
18
22

The following 4 numbers are odd:

1
3
9
13

Would you like to run the program again (y/n): y

Enter in a string of numbers separated by a comma (,) : 1, 2, 3, 4

```
---- Result Summary ----
        1 is odd!
        2 is even!
        3 is odd!
        4 is even!

The following 2 numbers are even:
        2
        4

The following 2 numbers are odd:
        1
        3

Would you like to run the program again (y/n):  n
Thank you for using the program.  Goodbye.
```

While Loops Challenge 28: Prime Number App

Description:
You are responsible for writing a program that will either determine if a given number is prime or display all prime numbers within a given range of values. When determining all prime numbers within a given range, your program will time the process and report how long the calculations took to the user. It is important to time certain processes within our programs so we can so how efficient our code is.

Step By Step Guide:

- Print a welcome message.
- Create an active flag variable to control a while loop and set it to True.

- Use this flag to run a while loop:
    - Display the users options for the program formatted as below.
    - Get user input for which option the program should run.
        - The program has 3 possibilities.
        - First, the user entered 1 and wants to calculate if a number is prime.
        - Second, the user entered 2 and wants all prime numbers within a set range of numbers.
        - Third, the user entered some foolish input other than 1 or 2.

    - If the user entered 1:
        - Get user input for their number to check.
        - Run an algorithm to determine if the number is prime or not.
            - A number is prime if it is only divisible by 1 and itself. Another way to think of this is a number is prime if it is not divisible by any number from 2 to itself minus 1.
            - Begin your algorithm by assuming the given number is prime.
            - Create a variable prime_status and set it equal to True
            - Using a for loop, loop through the numbers from 2 up to but not including the given number. Each iteration you should:
                - Check if the given number is divisible by the current number in the loop. If the given number is divisible by the current number in the loop:
                    - Set prime_status to False
                    - Break out of the for loop as there is no need to check other numbers.
                    - You can check divisibility using the modulus operator.
        - If, after the for loop is done prime_status is True:
            - Print a message stating the number is prime.
        - Else:
            - Print a message stating the number is not prime.

- Elif the user entered 2:
    - Get user input for the lower bound of the numerical range.
    - Get user input for the upper bound of the numerical range.
    - Create a variable primes and set it equal to a blank list. This will hold all prime numbers between the lower bound and upper bound.

    - Time how long the upcoming calculations are going to take.
        - Timing processes is outside the scope of basic Python. To perform this action we will have to import an extra library of code.
        - Type import time as the first line of code in your program.
        - We want to use the time library to measure how long the upcoming calculations take.
        - Prior to running the upcoming loop, call the time library to capture the current time and store it in a variable start_time.
        - Reference google of python documentation on how to use the time library.

    - For every number in between the lower bound and upper bound determine run an algorithm to determine if the number is prime.
        - Use a for loop to loop through a range of numbers between lower bound and upper bound. Each iteration you should:
            - Check if the current number is greater than 1. If it is:
                - Run your previous prime checking algorithm.
            - Else:
                - Set prime_status equal to False as 1 is not prime.
            - If prime_status is True:
                - Add the current prime candidate number to the list primes.

    - Once the algorithm is finished for every number between the lower bound and upper bound, call the time library again to capture the current time and store it in a variable end_time.
    - Create a variable delta_time and set it equal to the difference of end_time and start_time.
        - This is how long your calculation took.
        - Round this value to 4 decimal places.

    - Print a summary of the amount of time the calculations took.
    - Prompt the user to press enter to continue
    - Print all of the found prime numbers.

- Else:
    - The user entered a value other than 1 or 2.
    - Print a message letting them know their choice was not a valid option.

        - Get user input for if they would like to continue the program.
        - If the user does not want to continue:
            - Set your flag variable to False
            - Print a goodbye message thanking the user.

- Use at least 2 comments to describe sections of your code.
- “Chunk” your code so that is readable.
- Use appropriate and informative variable names.
- Format your output as below.

Example Output:

```
Welcome to the Prime Number App

Enter 1 to determine if a specific number is prime.
Enter 2 to determine all prime numbers within a set range.
Enter your choice 1 or 2: 1

Enter a number to determine if it is prime or not: 55
55 is not prime!
Would you like to run the program again (y/n): y

Enter 1 to determine if a specific number is prime.
Enter 2 to determine all prime numbers within a set range.
Enter your choice 1 or 2: 1

Enter a number to determine if it is prime or not: 11
11 is prime!
Would you like to run the program again (y/n): y

Enter 1 to determine if a specific number is prime.
Enter 2 to determine all prime numbers within a set range.
Enter your choice 1 or 2: 2

Enter the lower bound of your range: 1
Enter the upper bound of your range: 100

Calculations took a total of 0.0003 seconds.
The following numbers between 1 and 100 are prime:
Press enter to continue.
2
3
5
7
11
13
```

17
19
23
29
31
37
41
43
47
53
59
61
67
71
73
79
83
89
97
Would you like to run the program again (y/n): y

Enter 1 to determine if a specific number is prime.
Enter 2 to determine all prime numbers within a set range.
Enter your choice 1 or 2: 51

That is not a valid option.
Would you like to run the program again (y/n): n

Thank you for using the program. Have a nice day.

While Loops Challenge 29: Guess My Word App

Description:
You are responsible for writing a program that plays a word guessing game with a user. Your program will provide a category of words to the user and a string of dashes “-----” that represent the length of the word. The user will guess the word and with each incorrect guess, your program will reveal a letter at random, “-a---”. Upon guessing the word correctly, your program will then inform the user how many guesses they took.

Step By Step Guide:

- Print a welcome message.
- Create a dictionary called game_dict that has a minimum of four keys.
    - Each key should be a category such as "sports", "colors", or "fruits".
    - Each subsequent value should be a list.
    - In the list you should have a minimum of 6 strings that are in that category.
    - For instance if the key is sports, the subsequent value should be a list containing the words "basketball", "baseball", "tennis", etc...

- Create a blank list called game_keys.
- Use a for loop to loop through the keys of game_dict:
    - Append the current key in the dictionary to the list game_keys.

- Create an active flag variable to control a while loop and set it to True.

- Use this flag to run a while loop:
    - To implement the randomness of the game, extra code will be needed that is outside the scope of basic Python.
        - Type import random as the first line of code in your program.

    - Randomly pick a key from the list game_keys and store it in a variable named game_category.
        - Generate a random integer that correspond to the indices of the list game_keys.
        - game_category = game_keys[?????]
    - Randomly pick a value from the dictionary that corresponds to that key and store it in a variable named game_word.
        - Generate a random integer that corresponds to the indices of the list associated as the value to the chosen key from above.
        - game_word = game_dict[game_category][?????]

    - Create an empty list called blank_word.
    - For every letter that appears in game_word:
        - Append a dash ‘-’ to blank_word

        - For instance if the game_category was fruits and the game_word was apple, you should append five "-"to the list.

- Print a clue statement informing the user what category of words they are to guess from and how many letters there are in the word as formatted below.

- Create an empty string called guess that will eventually hold the users guess.
- Create a variable guess_count and set it equal to zero.

- Use a second while loop nested inside the first to allow the user to continue guessing as long as their guess is not equal to the word they are trying to guess, which would imply they won the game. This second loop will simulate playing a single round of the game while the first loop will control playing the game or quitting.
    - Print blank_word, which is a list, as a string using the .join() method.
    - Get user input for their guess and update the value of the variable guess.
    - Increment guess_count by 1.

    - If the guess is correct:
        - Inform the user they won the game and in how many tries.
        - End the game by breaking out of the second while loop.
    - Else:
        - The guess is not correct.
        - Inform the user and reveal a letter at random to the user.
            - Make sure to reveal a different letter each time.
            - You should check that the letter you are choosing to reveal is currently represented by a "-" in the blank_word list.
            - If it is, to reveal it, replace the “-” with the corresponding letter such that when you print blank_word it will be a combination of “-” and letters.
            - If it is not, continue to pick a random letter until you find one that is.
            - To do this, create an active flag variable to control a while loop and set it to True.
            - Use this flag to run a while loop:
                - Create a variable letter_index and set it equal to a random integer that corresponds with a random index of your list blank_word.
                - If blank_word[letter_index] is equal to a “-”:
                    - Set blank_word[letter_index] equal to the actual letter of the game_word.
                    - Set your flag variable to False.

- Once a game round is over, get user input for if they would like to play again.
- If the user does not want to play again:

        - Set your flag variable to False
        - Print a goodbye message thanking the user.

- Use at least 2 comments to describe sections of your code.
- “Chunk” your code so that is readable.
- Use appropriate and informative variable names.
- Format your output as below.

Example Output:

Welcome to the Guess My Word App

Guess a 7 letter word from the following category: Classes

Enter your guess: history
That is not correct. Let us reveal a letter to help you!
---l---

Enter your guess: science
That is not correct. Let us reveal a letter to help you!
---li--

Enter your guess: english

Correct! You guessed the word in 3 guesses.
Would you like to play again (y/n): y

Guess a 6 letter word from the following category: Fruits

Enter your guess: apples
That is not correct. Let us reveal a letter to help you!
---a--

Enter your guess: orange
That is not correct. Let us reveal a letter to help you!
---a-a

Enter your guess: bananas
That is not correct. Let us reveal a letter to help you!
-a-a-a

Enter your guess: banana

Correct! You guessed the word in 4 guesses.
Would you like to play again (y/n): n

Thank you for playing our game.

While Loops Challenge 30: Power Ball Simulator App

Description:
You are responsible for writing a program that simulates the Power Ball Lottery. The traditional power ball is played by randomly choosing 5 white balls numbered 1 through 69 then randomly choosing 1 red ball numbered 1 through 26. The traditional power ball has astronomically low odds of winning. Therefore, your program will allow users to adjust the odds by setting how many balls the lottery will choose from. Your program will then calculate the odds the user has of winning. The user will purchase tickets in a set interval, without purchasing repeated losing tickets, until they either win the lottery or choose to give up.

Step By Step Guide:

- Print a welcome message as formatted below.

- The normal Power Ball Lottery chooses 5 random white balls numbered 1 through 69.
- Once a ball is chosen, it cannot be chosen again.
- Using all 69 balls will make winning the lottery nearly impossible. Therefore, we want to give the user the option to set how many white balls they will pick from.
- Get user input for the number of white balls to use.
    - Store this user input in a variable called white_balls.

- If the user entered a number less than 5, set the value of white_balls to five.
    - This ensures that we have enough numbers to choose from.

- Similarly, the red Power Ball is normally chosen from red balls numbered 1 through 26.
- We want to give the user the option to set how many red balls they will pick from.
- Get user input for the number of red balls to use.
    - Store this input in a variable called red_balls.

- If the user entered a number less than 1, set the value of red_balls to one.
    - This ensures we have at least one power ball.

- Generate the odds the user will win this specific lottery.
- Create a variable called odds and set it equal to 1.
    - You will use this variable to perform repeated multiplication.
- Use a for loop to loop an appropriate number of times to perform the correct multiplication. Your for loop should be a numerical for loop such as for i in range(5):
    - For example, if the user wanted 69 white balls and 26 red balls the odds would be calculated as follows: (69*68*67*66*65)*26/120.
    - For example, if the user wanted 20 white balls and 4 red balls the odds would be calculated as follows: (20*19*18*17*16)*4/120.
    - Try to determine the relationship between the number of white balls used, the value of i in the for loop, and the terms being multiplied together to get the odds.
- Determine the odds for any specific lottery given and print the odds to the user.

- We will allow users to purchase tickets in a set interval.
- Get user input for how many tickets they would like to purchase in each interval.
    - Store this input in a variable called ticket_interval.

- Next, we need to simulate creating the winning lotto numbers.
- Each lottery ball will be represented by a random integer.
    - Type import random as the first line of code in your program.
- Create a blank list called winning_numbers.

- Generate the numbers to represent the white balls.
    - While there are less than 5 values in winning_numbers:
        - Generate a random integer from 1 up to the users value of white_balls.
        - If the integer does not already appear in the list winning_numbers:
            - Append the integer to the list winning_numbers. We cannot have repeated values.

- Permanently sort the list winning_numbers.

- Generate the number to represent the red power ball.
    - Generate a random integer from 1 up to the users value of red_balls.
    - Append this to the end of the list winning_numbers.
    - Do not sort the list as the red power ball is always at the end of the winning numbers.

- You now have a list of numbers, winning_numbers that represent the winning Power Ball numbers; 5 numbers that are ordered and then one final Power Ball number added at the end.
    - For example [10, 22, 24, 54, 67, 3]

- Now, simulate the actual Power Ball drawing.
- Print a welcome message as formatted below.
- Display the winning power ball numbers all on one line as formatted below.
    - Use the end= argument.
- Prompt the user to press enter to being purchasing tickets.

- Create a variable called tickets_purchased and set it equal to 0.
    - This will keep track of how many tickets have been created.
    - Every time your while loop runs you should increment this by 1.
- Create a flag variable called active and set it to True.
    - This variable will stay True as long as the user wants to continue to purchase tickets. After, the number of tickets stored in ticket_interval have been created, ask the user if they want to purchase more. If they do not, set this variable to False.
- Create a blank list called tickets_sold.
    - This will keep track of every ticket that has been sold.

    - We will use this list to make sure that we don't purchase a losing ticket more than once.

- Use a while loop that will continue to run until a winning ticket is picked or user wants to stop wasting their money. This loop needs to check two conditions: The winning ticket has not been sold and the user still wants to play.
    - Create a blank list called lottery_numbers which will hold the current lottery numbers of a ticket that is being purchased.
        - Once the ticket is created and determined to be a loser, you can use this same variable for the next ticket.
    - Simulate picking a lottery ticket.
        - You should do this in a similar manner to picking the actual power ball numbers. Pick 5 numbers, sort them, and then add a 6th.
    - If this current ticket is not in your list of tickets sold:
        - Increase the number of purchased tickets by 1.
        - Add it to the list of tickets sold,
        - Print the ticket as a list.
    - Else:
        - Print a message that a losing ticket was generated that has already been purchased and that the ticket is being disregarded.

    - After picking the number of tickets in ticket_interval, which can be checked using modulo division:
        - Print the total number of tickets purchased.
        - Get user input for if they would like to continue purchasing tickets.
        - If the user does not want to continue:
            - Set your flag variable to False

- The while loop will end either by purchasing the winning ticket or by giving up.
- If the current lottery ticket is equal to the winning ticket:
    - You won the lottery!
    - Print the winning ticket numbers.
    - Print the total number of tickets purchased.
- Else:
    - You didn’t win the lottery.
    - Print how many tickets were purchased.

- Use at least 2 comments to describe sections of your code.
- “Chunk” your code so that is readable.
- Use appropriate and informative variable names.
- Format your output as below.

Example Output 1:

----------Power-Ball Simulator----------

How many white-balls to draw from for the 5 winning numbers (Normally 69): 7
How many red-balls to draw from for the Power-Ball (Normally 26): 1
You have a 1 in 21.0 chance of winning this lottery.
Purchase tickets in what interval: 5

---------Welcome to the Power-Ball----------

Tonight's winning numbers are: 1 2 3 5 7 1
Press 'Enter' to begin purchasing tickets!!!
[2, 3, 5, 6, 7, 1]
[1, 3, 4, 5, 6, 1]
[1, 2, 4, 5, 6, 1]
[1, 3, 4, 5, 7, 1]
Losing ticket generated; disregard...
[1, 2, 5, 6, 7, 1]
5 tickets purchased so far with no winners...

Keep purchasing tickets (y/n): y
[1, 4, 5, 6, 7, 1]
[1, 3, 5, 6, 7, 1]
[1, 2, 3, 5, 7, 1]

Winning ticket numbers: 1 2 3 5 7 1
Purchased a total of 8 tickets.

Example Output 2:

----------Power-Ball Simulator----------

How many white-balls to draw from for the 5 winning numbers (Normally 69): 69
How many red-balls to draw from for the Power-Ball (Normally 26): 26
You have a 1 in 292201338.0 chance of winning this lottery.
Purchase tickets in what interval: 20

---------Welcome to the Power-Ball----------

Tonight's winning numbers are: 31 37 55 57 61 12
Press 'Enter' to begin purchasing tickets!!!
[11, 36, 40, 50, 52, 10]
[8, 20, 45, 49, 51, 14]
[35, 42, 57, 60, 67, 3]
[19, 21, 28, 53, 67, 12]
[5, 17, 22, 59, 67, 18]

[1, 3, 12, 15, 50, 17]
[18, 44, 45, 54, 66, 24]
[6, 33, 47, 55, 61, 16]
[28, 35, 36, 47, 52, 5]
[7, 9, 37, 42, 57, 24]
[12, 14, 21, 56, 60, 18]
[20, 52, 55, 62, 69, 16]
[11, 38, 41, 61, 63, 17]
[5, 24, 25, 33, 37, 7]
[36, 38, 44, 51, 62, 15]
[20, 23, 59, 60, 66, 20]
[4, 9, 23, 29, 33, 21]
[11, 14, 35, 36, 60, 3]
[7, 19, 48, 50, 68, 26]
[6, 24, 27, 37, 58, 18]
20 tickets purchased so far with no winners...

Keep purchasing tickets (y/n): y
[4, 8, 14, 22, 68, 16]
[3, 24, 34, 41, 57, 12]
[30, 36, 39, 40, 59, 16]
[6, 23, 33, 34, 50, 26]
[1, 19, 54, 60, 61, 23]
[1, 35, 43, 52, 54, 13]
[3, 6, 7, 37, 50, 13]
[2, 6, 22, 28, 47, 25]
[6, 19, 46, 55, 60, 5]
[8, 44, 50, 52, 61, 2]
[39, 44, 52, 54, 64, 17]
[16, 19, 24, 27, 49, 13]
[18, 24, 27, 34, 68, 12]
[15, 19, 27, 44, 57, 23]
[40, 42, 47, 55, 65, 10]

[13, 30, 36, 45, 59, 17]
[4, 16, 22, 28, 57, 2]
[17, 27, 51, 65, 66, 14]
[7, 15, 22, 43, 53, 4]
[11, 18, 20, 30, 53, 21]
40 tickets purchased so far with no winners...

Keep purchasing tickets (y/n): n

You bought 40 tickets and still lost!
Better luck next time!