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.
```