Python Projects

Solved by Michael Zolotarenko

Prime Number Finder ✎ Lists

from the Codecademy Intermediate Python 3 course (view Certificate of completion)

The Prime Number Finder project is one of the engaging assignments within Codecademy's Code Challenges, focusing on Python list operations. It entails creating a prime_finder() function to detect prime numbers within a defined range. In my solution, I implemented additional functionality through user input and visual output enhancements, enriching the interactive experience for exploring prime numbers.

Example:

```
Enter the upper limit to find prime numbers up to (an integer greater than 1), or 'q' to quit: 123
Enter the number of columns to print, or 'q' to quit: 6
There is a total of 30 prime numbers in the range 2 to 123.
+-----------------------------+
|  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 | 101 | 103 | 107 | 109 | 113|
+-----------------------------+
```

Task (view at codecademy.com)

Create a prime_finder() function that takes in a number, n, and returns all the prime numbers from 1 to n (inclusive). As a reminder, a prime number is a number that is only divisible by 1 and itself.

For example, prime_finder(11) should return [2, 3, 5, 7, 11].

Solution

prime_finder.py (download robot-race.tar.gz from GitHub Portfolio repository)

```
# Prime number finder function.
# Returns a list of prime numbers in a range 2 to n (inclusive).

def prime_finder(n):
    primes = []
    for i in range(2, n + 1):
        prime_num = True
        for j in range(2, int(i**0.5) + 1):
            if i % j == 0:
                prime_num = False
                break  # No need to continue checking if it's not a prime
        if prime_num:
            primes.append(i)
    return primes

# Present the set of prime numbers in the form of a list or matrix
def create_matrix(primes, num_columns=1):
    if num_columns == 1:
        print("Prime numbers:", primes)
    else:
        num_rows = (len(primes) + (num_columns - 1)) // num_columns
```

```
        max_width = max(len(str(prime)) for prime in primes)
        border = "+" + "-" * (max_width + 2) * num_columns + "+"
        print(border)
        for i in range(num_rows):
            row = primes[i*num_columns : (i+1)*num_columns]
            row_str = " | ".join(str(prime).center(max_width) for prime in row)
            print("|" + row_str + "|")
        print(border)

# User prompt block
while True:  # Infinite loop
    try:
        user_input_n = input("Enter the upper limit to find prime numbers up to (an integer
greater than 1), or 'q' to quit: ")
        if user_input_n.lower() == 'q':
            print("Exiting the program.")
            break  # Exit the loop and end the program

        n = int(user_input_n)
        if n <= 1:
            raise ValueError("Please enter an integer greater than 1.")

        user_input_col = input("Enter the number of columns to print, or 'q' to quit: ")
        if user_input_col.lower() == 'q':
            print("Exiting the program.")
            break  # Exit the loop and end the program

        if not user_input_col:
            num_columns = 1
        else:
            try:
                num_columns_candidate = int(user_input_col)
                if num_columns_candidate <= 1:
                    num_columns = 1
                else:
                    num_columns = num_columns_candidate
            except ValueError:
                num_columns = 1  # Set default value if conversion fails

        primes = prime_finder(n)
        print(f"There is a total of {len(primes)} prime numbers in the range 2 to {n}.")

        # Call create_matrix to display primes in either list or matrix format
        create_matrix(primes, num_columns)

    except ValueError as ve:
        # Handle the case where the input is invalid
        print(ve)
    except Exception as e:
        # Handle other exceptions, if any
        print("An error occurred:", e)
```