

Prime Number Calculator

Textbook

Prime Number Calculator!

Recommended time to do this project: After the **Python Lists Continued** lesson.



Instructions

Imagine you just found a safe and the only clue to opening it is that the password is a prime number. Create a tool to check whether or not a number is a prime number so you can open the safe and get the treasure!

A prime number must be a positive integer greater than one that has no other factors besides 1.

Prompt the user to enter a number.

If it is not a prime number, print the list of all its factors. (Hint, you will need to loop through each number until you get to the number inputted by the user to see if it is divisible by anything other than 1.)

Note: A reminder that 1 and the inputted number are considered factors. But for this challenge they should not be included in the factor list.

If it is a prime number, print that it is a prime number in this format:

"x is a prime number"

For example:

Input: 12

Output: [2, 3, 4, 6]

Another example:

Input: 11

Output: "11 is a prime number"

Decomposing Large Tasks with Small Code Segments

When you're faced with a big, complex problem in programming, trying to solve it all at once can feel overwhelming. A powerful strategy is to decompose the problem, which means breaking it down into smaller, more manageable pieces. Each of these smaller pieces can then be handled by its own dedicated code segment, making the overall problem much easier to tackle.

Imagine you need to build a program that handles customer orders for an online store. This is a big problem! You could decompose it into smaller code segments: one segment to take the customer's order, another to calculate the total cost, a third to process payment, and a final one to send a confirmation email. Each of these smaller tasks can be written and tested independently.

By defining these new code segments (which often become functions or methods, as you'll learn more about), you make your code more organized, readable, and easier to debug. If there's an issue with how the payment is processed, you know exactly which code segment to examine, rather than sifting through a massive block of code.

This approach also makes your code reusable; once you have a "calculate total cost" segment, you can use it in other parts of your program or even in entirely different projects. Decomposing problems into smaller, focused code segments is a fundamental skill for any programmer, allowing you to build complex and robust applications more effectively.

Mathematical Thinking

As you work through programming challenges, remember that discussing your mathematical thinking with classmates is incredibly valuable. This isn't just about sharing answers; it's about explaining *how* you arrived at those answers, clearly communicating the mathematical ideas, vocabulary, and methods you used.

By listening to how others approached the same problem, you can analyze their thinking, perhaps discover more efficient methods, and even recognize errors in your own or their logic, offering constructive suggestions for correct solutions. Being able to justify your results by explaining your processes, and constructing arguments based on evidence from your code, will deepen your understanding and help you learn from every step of the journey, making errors an opportunity for growth.

Checkpoint

Prime Number Calculator

Imagine you just found a safe and the only clue to opening it is that the password is a prime number. Create a tool to check whether or not a number is a prime number so you can open the safe and get the treasure!

A prime number must be a positive integer greater than one that has no other factors besides 1.

1. Prompt the user to enter a number.
2. If it is not a prime number, print the list of all its factors. (Hint, you will need to loop through each number until you get to the number inputted by the user to see if it is divisible by anything other than 1.)
3. *Note: A reminder that 1 and the inputted number are considered factors. But for this challenge they should not be included in the factor list.*
4. If it is a prime number, print that it is a prime number in this format:

```
"x is a prime number"
```

For example:

Input: 12

Output: [2, 3, 4, 6]

Another example:

Input: 11

Output: "11 is a prime number"

Hint: You will use `.append` for this project.

Hint: Consider using the following concept:

```
if len(factors) > 0:
```

```
    print( )
```

Answer Keys & Solutions

Checkpoint Solutions

Prime Number Calculator

```
1 number = int(input("What number do you want to know?"))
2
3 # print(range(number))
4 factors = []
5 for x in range(2, number):
6     if number % x == 0:
7         factors.append(x)
8
9
10 if len(factors) > 0:
11     print(factors)
12 else:
13     print(str(number) + " is a prime number")
```