Ifs, loops, and function homework

1. A function to reverse a string

Write and test a function that reverses a string entered by a user. This function will have one input value (a string) and one output value (also a string).

Test your function on, among other things, Napoleon's quote 'able was i ere i saw elba'

In [5]:
```
user_input = input("Enter a string: ")
print("Reversed string:", user_input[::-1])
```

```
Enter a string: able was i ere i saw elba
Reversed string: able was i ere i saw elba
```

In [6]:
```
user_input = input("Enter a string: ")
print("Reversed string:", user_input[::-1])
```

```
Enter a string: rand
Reversed string: dnar
```

Optional challenge: run the above on "race car" and then fix the resulting string.

In [22]:
```
user_input = input("Enter a string: ")
print("Reversed string:", user_input[::-1])
```

```
Enter a string: race car
Reversed string: rac ecar
```

2. Determine if a number is prime

Write some code to test whether a number is prime or not, a prime number being an integer that is evenly divisible only by 1 and itself.

Hint: another way to think about a prime number is that, if the smallest number (other than 1) that divides evenly into a number *is* that number, than the number is a prime.

The easiest solution involves one `while` loop and one `if` test.

In [11]:
```
def is_prime(number):
    return number > 1 and all(number % i for i in range(2, int(number*

user_input = int(input("Enter a number: "))
print(f"{user_input} is {'a prime' if is_prime(user_input) else 'not a
```

```
Enter a number: 2
2 is a prime number.
```

3. Find the first 10 primes

Extend your code above to find the first 10 prime numbers. This will involve wrapping your existing code in another "outer" loop.

In [34]:
```
def is_prime(number):
    return number > 1 and all(number % i for i in range(2, int(number*

count_primes, current_number = 0, 2

while count_primes < 10:
    if is_prime(current_number):
        print(current_number, end=' ')
        count_primes += 1
    current_number += 1
```

```
2 3 5 7 11 13 17 19 23 29
```

In [1]:
```
def is_prime(number):
    return number > 1 and all(number % i for i in range(2, int(number*

count_primes, current_number = 1, 2

while count_primes < 10:
    if is_prime(current_number):
        print(current_number, end=' ')
        count_primes += 1
    current_number += 1
```

```
2 3 5 7 11 13 17 19 23
```

4. Make a function to compute the first n primes

Functionalize (is that a word?) your above code. A user should be able to call your code with one integer argument and get a list back containing that number of primes. Make sure your function handles inputs of an incorrect type gracefully. You should also warn the user if they enter a really big number (which could take a long time...), and give them the option of either bailing or entering a different number.

In [2]:
```
def is_prime(number):
    return number > 1 and all(number % i for i in range(2, int(number*

def get_primes(n):
    try:
        n = int(n)
        if n <= 0:
            return "Enter a positive integer."

        primes, current_number = [], 2
        while len(primes) < n:
            if is_prime(current_number):
                primes.append(current_number)
            current_number += 1

        return primes

    except ValueError:
        return "Enter a valid integer."

user_input = input("Enter the number of primes you want to generate: "
result = get_primes(user_input)
print(result)
```

```
Enter the number of primes you want to generate: y
Enter a valid integer.
```

In [3]:
```
def is_prime(number):
    return number > 1 and all(number % i for i in range(2, int(number*

def get_primes(n):
    try:
        n = int(n)
        if n <= 0:
            return "Enter a positive integer."

        primes, current_number = [], 2
        while len(primes) < n:
            if is_prime(current_number):
                primes.append(current_number)
            current_number += 1

        return primes

    except ValueError:
        return "Enter a valid integer."

user_input = input("Enter the number of primes you want to generate: "
result = get_primes(user_input)
print(result)
```

```
Enter the number of primes you want to generate: -2
Enter a positive integer.
```

In [4]:
```
def is_prime(number):
    return number > 1 and all(number % i for i in range(2, int(number*

def get_primes(n):
    try:
        n = int(n)
        if n <= 0:
            return "Enter a positive integer."

        primes, current_number = [], 2
        while len(primes) < n:
            if is_prime(current_number):
                primes.append(current_number)
            current_number += 1

        return primes

    except ValueError:
        return "Enter a valid integer."

user_input = input("Enter the number of primes you want to generate: "
result = get_primes(user_input)
print(result)
```

```
Enter the number of primes you want to generate: 7
[2, 3, 5, 7, 11, 13, 17]
```