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 [10]:
```
def rev_str(user_str):
    store_str = []                      # empty list to store new reversed
    join_str = ''                       # empty string to join the elements
    for i in reversed(user_str):
        store_str.append(i)             # append elements to empty list
    store_str = join_str.join(store_str) # join the elements into one string
    return store_str
```

In [11]:
```
rev_str('able was i ere i saw elba')
```

Out[11]:
```
'able was i ere i saw elba'
```

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

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 [9]:
```
num = int(input("Enter a number to check if it is a prime number or not:"))


if num <= 1:
    print(f"{num} is not a prime number.")
elif num == 2:
    print(f"{num} is a prime number!")
else:
    for i in range(2, num):
        if num % i == 0:
            print(f"{num} is not a prime number.")
            break
        else:
            print(f"{num} is a prime number!")
            break
```

```
5 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 [64]:
```
num = 0
num_of_primes = 0
primes = []

while num_of_primes < 10:
    if num <= 1:
        num += 1
    elif num == 2:
        primes.append(num)
        num += 1
        num_of_primes += 1
    else:
        for i in range(2, num):
            if num % i == 0:
                break
        else:
            primes.append(num)
            num_of_primes += 1
        num += 1

print(primes)
```

```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29]
```

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 [59]:
```
def first_n_primes(some_num):
    n_too_large = 100  # too large of a number to compute n primes

    # warnings to user

    if some_num > n_too_large:
        warning = input("Warning: Computing a large number of primes may tal

        while (warning =='n') or (some_num > n_too_large):    # keeps asking

            if (warning == 'y') or (some_num <= n_too_large): # user wants
                break

            elif warning == 'n':
                some_num = int(input("Enter a new number: "))

                if some_num >= n_too_large:                   # warns user
                    warning = input("This number may still be too large. Do

            else:                                             # checks that
                warning = input("Invalid input. Please enter 'y' or 'n'.")
```

```
    # compute the first n primes

    num = 0                                 # counts numbers to goes through all
    num_of_primes = 0                       # records how many prime numbers are
    primes = []                             # stores the prime numbers

    while num_of_primes < some_num:         # runs while the number of primes in
        if num <= 1:
            num += 1
        elif num == 2:
            primes.append(num)
            num += 1
            num_of_primes += 1
        else:
            for i in range(2, num):
                if num % i == 0:            # not prime number
                    break
            else:                           # is prime number. else is out of t
                primes.append(num)
                num_of_primes += 1
            num += 1
    return primes
```

In [60]:
```
first_n_primes(11)
```

Out[60]:
```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31]
```

In [62]:
```
first_n_primes(120)
```

Out[62]:
```
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 7
1]
```