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 [11]:
```
a_string = 'able was i ere i saw elba'
separator = ''
separator.join(list(reversed(a_string)))
```

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

In [16]:
```
def reverse_a_string(a_string) :
    separator = ''
    reversed_string = separator.join(list(reversed(a_string)))
    return reversed_string
```

In [14]:
```
a_string = 'able was i ere i saw elba'
new_string = reverse_a_string(a_string)
print(f'This is my new reversed string: {new_string}')
```

```
This is my new reversed string: able was i ere i saw elba
```

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

In [25]:
```
a_string = 'race car'
new_string = reverse_a_string(a_string)
print(f'This is my new reversed string: {new_string}')

new_string_list = list(new_string)
new_string_list[3] = 'e'
new_string_list[4] = ' '
separator = ''
fixed_new_string = separator.join(new_string_list)
print(f'This is my new, fixed reversed string: {fixed_new_string}')
```

```
This is my new reversed string: rac ecar
This is my new, fixed reversed string: race car
```

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 [1]:
```
def prime_number(a_num):
    trueOrFalse = False
    if a_num == 1:
        trueOrFalse = False
    else :
        i = 2
        while i < a_num :
            if a_num%i == 0 :
                trueOrFalse = False
                break
            else :
                trueOrFalse = True
                i += 1
    return trueOrFalse
```

In [2]:
```
prime_number(4)
```

Out[2]:
```
False
```

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 [8]:
```
primeNumbers = []
a_num = 1

while len(primeNumbers) < 10 :
    trueOrFalse = prime_number(a_num)
    if trueOrFalse == True :
        primeNumbers.append(a_num)
    a_num += 1

print(primeNumbers)
```

```
[3, 5, 7, 11, 13, 17, 19, 23, 29, 31]
```

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 [9]:
```
def listPrimeNumbers(numberOfPrimes) :
    primeNumbers = []
    a_num = 1

    while len(primeNumbers) < numberOfPrimes :
        trueOrFalse = prime_number(a_num)
        if trueOrFalse == True :
            primeNumbers.append(a_num)
        a_num += 1

    return primeNumbers
```

In [10]:
```
listPrimeNumbers(15)
```

Out[10]:
```
[3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53]
```

In [11]:
```
def listPrimeNumbers(numberOfPrimes) :
    try:
        primeNumbers = []
        a_num = 1

        while len(primeNumbers) < numberOfPrimes :

            trueOrFalse = prime_number(a_num)
            if trueOrFalse == True :
                primeNumbers.append(a_num)
            a_num += 1

        return primeNumbers
    except TypeError:
        print(f'Please try again by entering a number, either a float or integer, instead of {type(numberOfPrimes)}')
```

In [13]:
```
listPrimeNumbers('hi')
```

```
Please try again by entering a number, either a float or integer, instead of <class 'str'>
```

In [3]:

```python
def listPrimeNumbers(numberOfPrimes) :

    try:
        primeNumbers = []
        a_num = 1
        tooManyNumbers = 100
        while len(primeNumbers) < numberOfPrimes :

            if numberOfPrimes > tooManyNumbers :
                print('Oops! That\'s a lot of numbers...maybe do less')
                stop = input('Are you sure you want to continue? [Y/N]')
                if stop == 'N' :
                    break
                if stop == 'Y' :
                    tooManyNumbers = numberOfPrimes + 1


            else :
                trueOrFalse = prime_number(a_num)
                if trueOrFalse == True :
                    primeNumbers.append(a_num)
                a_num += 1

        return primeNumbers
    except TypeError:
        print(f'Please try again by entering a number, either a float or integer, instead of {type(numberOfPrimes)}')
```

In [4]:

```
listPrimeNumbers(101)
```

```
Oops! That's a lot of numbers...maybe do less
Are you sure you want to continue? [Y/N]Y
```

```
Out[4]: [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,
         127,
         131,
         137,
         139,
         149,
         151,
         157,
         163,
         167,
         173,
         179,
         181,
         191,
         193,
         197,
         199,
         211,
         223,
         227,
         229,
         233,
         239,
         241,
         251,
         257,
         263,
         269,
         271,
         277,
         281,
         283,
         293,
         307,
         311,
         313,
         317,
         331,
         337,
         347,
         349,
         353,
         359,
         367,
         373,
         379,
         383,
         389,
         397,
         401,
         409,
         419,
         421,
         431,
         433,
         439,
         443,
         449,
 457,
 461,
 463,
 467,
 479,
 487,
 491,
 499,
 503,
 509,
 521,
 523,
 541,
 547,
 557]
```

In [5]:

```
listPrimeNumbers(17)
```

Out[5]:

```
[3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61]
```

In [6]:

```
listPrimeNumbers('hi')
```

```
Please try again by entering a number, either a float or integer, instead of <class 'str'>
```

In [7]:

```
listPrimeNumbers(200)
```

```
Oops! That's a lot of numbers...maybe do less
Are you sure you want to continue? [Y/N]N
```

Out[7]:

```
[]
```

In [ ]:

```
```