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 [ ]:
```
for ch in x[::-1]:
    print(ch)
```

```
e
l
c
i
s
p
o
p
```

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

In [ ]:
```

```

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 [8]:
```
prime = int(input("input an integer greater than two to see if it is prime"))

i = 2
while i < prime:
    if prime % i == 0:
        print(f"{prime} is not a prime number")
        break
    i += 1
else:
    if prime > 1:
        print(f"{prime} is a prime number")
```

```
999 is not 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 [ ]:
```
howmany = int(input("how many prime numbers would you like listed"))
num = 2
primesFound = 0
primes = []

while primesFound < howmany:
    i=2
    is_prime = True

    while i < num:
        if num % i == 0:
            is_prime = False
            break
        i += 1
    if is_prime:
        primes.append(num)
        primesFound +=1
    num +=1

print(primes)
```

```
  Cell In[14], line 19
    return(primes)
    ^
SyntaxError: 'return' outside function
```

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 [22]:
```
def primelist(n):
    if n > 10000:
        confirm = input(
            "Warning: calculating more than 10000 primes may take some time.
            "Type YES to continue:")
        if confirm != "YES":
            print("Cancelled.")
            return []

    primes = []
```

```
    num = 2

    while len(primes) < n:
        i = 2
        is_prime = True

        while i < num:
            if num % i == 0:
                is_prime = False
                break
            i += 1

        if is_prime:
            primes.append(num)

        num += 1

    return primes
```

In [24]:
```
primelist(5)
```

Out[24]:
```
[2, 3, 5, 7, 11]
```