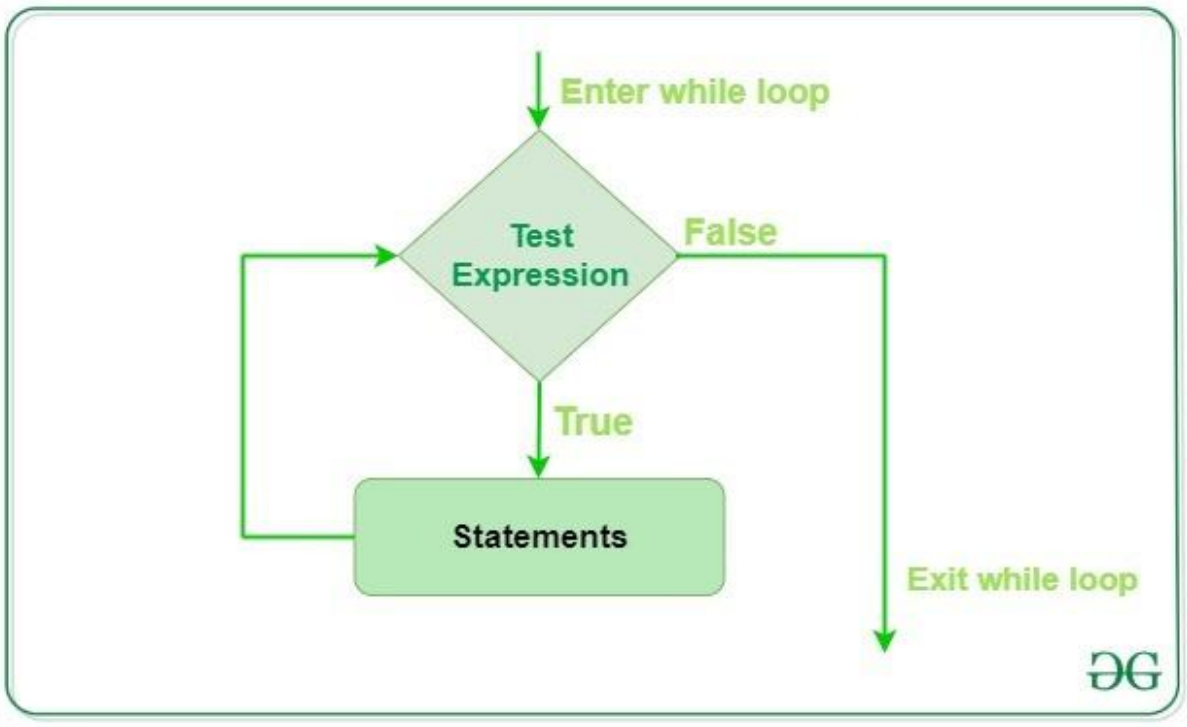
Lesson 7: While Loops

Introduction to Loops

Let's say you wanted to print numbers 1-10. How would we do this as of now?

```
print(1)
print(2)
print(3)
print(4)
print(5)
print(6)
print(7)
print(8)
print(9)
print(10)
```

That is A LOT of print statements. Is there any way we can shorten this piece of code? Well, we're programming, so of course there is. Loops are a way to repeat code until a certain condition is met. The most basic form of a loop is called a while loop. It works like this:



In code, while loops look something like this:

```
while condition:
    # insert looped code here
```

Now, let's shorten our program to print numbers 1-10.

```
counter = 1
while counter <= 10:
    print(counter)
    counter += 1
```

1. First, we can define a counter to help us count numbers 1 - 10
2. Then, we can check if counter is less than or equal to 10 (we don't want to print out any more than the first 10 numbers)
3. If the condition is met, we'll print out our counter and increase its value by one (or *increment* it)

What's very important here is the idea of using a counter. Counters are often used in programming to iterate through lists, data, etc.

Break and Continue

There are a few keywords we should get familiar with when using while loops: **break** and **continue**.

break - the break keyword allows us to exit a while loop. When it is used, the while loop will immediately end without completing any of the code after (within the loop).

For example,

```
while True:
    print("Hi!")
    break
    print("Bye!")
```

The code above will print "Hi" and then immediately exit the loop. It will not print "Bye!"

continue - the continue keyword continues to the next iteration of the while loop without completing any of the code after it (within the loop).

For example,

```
i = 1
while i <= 5:
```

```
    print("Hi!")
    i += 1
    continue
    print("Bye!")
```

The code above will print “Hi!” 5 times without ever printing “Bye!”. The continue keyword will immediately skip to the next iteration (notice that we’re still adding 1 to i before we reach the continue keyword).

While Loop Exercises

Exercise 1
Write a while loop that adds up the first 100 numbers (1-100).

Exercise 2
Write a while loop that prints the first three numbers backwards (3, 2, 1).

Review: Exercise 3
Write a while loop that prints the first 5 numbers on the same line (12345).

Exercise 3, Part 2
Write a program that asks the user for a number n, then prints the first n numbers on the same line (12…n).

Exercise 4
Write a while loop that finds the average of the first ten numbers.

Exercise 5
Write a program that computes the factorial of a number inputted by the user.

As a reminder, factorial (!) means the product of all positive numbers less than or equal to a number. For example, 6! = 6 * 5 * 4 * 3 * 2 * 1

Exercise 6
Write a program that takes in input, asking if the program should shut down. If the user responds “y”, end the program. If the user responds “n”, ask again. Otherwise, print “Invalid input”.

Challenge Problem: Exercise 7

Write a while loop that prints the first 5 numbers in the following pattern.

1
1 2
1 2 3
1 2 3 4
1 2 3 4 5