Loop and a Half

REEdundancy

```
my_num = int(input("Enter a positive number: "))
while my_num <= 0:
    print("Number must be positive!")
    my_num = int(input("Enter a positive number: "))
```

REEdundancy

```
my_num = int(input("Enter a positive number: "))
while my_num <= 0:
    print("Number must be positive!")
    my_num = int(input("Enter a positive number: "))
```

REEdundancy

```
while my_num <= 0:
    print("Number must be positive!")
    my_num = int(input("Enter a positive number: "))
```

REEdundancy

```
while my_num <= 0:
    print("Number must be positive!")
    my_num = int(input("Enter a positive number: "))
```

```
Traceback (most recent call last):
  File "main.py", line 1, in <module>
    while my_num <= 0:
NameError: name 'my_num' is not defined
```

REEdundancy

```
my_num = 0
while my_num <= 0:
    print("Number must be positive!")
    my_num = int(input("Enter a positive number: "))
```

REEdundancy

```
my_num = 0
while my_num <= 0:
    print("Number must be positive!")
    my_num = int(input("Enter a positive number: "))
```

```
Number must be positive!
Enter a positive number:
```

REEdundancy

```
my_num = 0
while my_num <= 0:
    my_num = int(input("Enter a positive number: "))
    print("Number must be positive!")
```

REEdundancy

```
my_num = 0
while my_num <= 0:
    my_num = int(input("Enter a positive number: "))
    print("Number must be positive!")
```

```
Enter a positive number: -1
Number must be positive!
```

REEdundancy

```
my_num = 0
while my_num <= 0:
    my_num = int(input("Enter a positive number: "))
    print("Number must be positive!")
```

```
Enter a positive number: -1
Number must be positive!
Enter a positive number: 5
Number must be positive!
```

REEdundancy

```
my_num = 0
while my_num <= 0:
    my_num = int(input("Enter a positive number: "))
    print("Number must be positive!")
```

Wouldn't it be convenient if we could stop the `while` loop in the middle?

Good news: We can!

Using `break`!

`break` is a command that will instantly end a loop, skipping the remaining body, whether it's a `while` loop **or** a `for` loop.

```
my_num = 0
while my_num <= 0:
    my_num = int(input("Enter a positive number: "))
    if my_num > 0:
        break
    print("Number must be positive!")
print("Nice number!")
```

Breaking is Good

Using `break`!

`break` is a command that will instantly end a loop, skipping the remaining body, whether it's a `while` loop **or** a `for` loop.

```
my_num = 0
while my_num <= 0:
    my_num = int(input("Enter a positive number: "))
    if my_num > 0:
        break
    print("Number must be positive!")
print("Nice number!")
```

Breaking is Good

Because `my_num` can't change in between the `if` statement and the top of the `while` loop, there's really only one place the loop's going to end - at the `if`. That means that we can actually get rid of the `condition` for the `while` loop entirely!

```
my_num = 0
while my_num <= 0:
    my_num = int(input("Enter a positive number: "))
    if my_num > 0:
        break
    print("Number must be positive!")
print("Nice number!")
```

Breaking is Good

Because `my_num` can't change in between the `if` statement and the top of the `while` loop, there's really only one place the loop's going to end - at the `if`. That means that we can actually get rid of the `condition` for the `while` loop entirely!

```
my_num = 0
while True:
    my_num = int(input("Enter a positive number: "))
    if my_num > 0:
        break
    print("Number must be positive!")
print("Nice number!")
```

Breaking is Good

We can also get rid of the initial value for `my_num`, because it doesn't need to exist before the `while` loop anymore!

```
while True:
    my_num = int(input("Enter a positive number: "))
    if my_num > 0:
        break
    print("Number must be positive!")
print("Nice number!")
```

Breaking is Good

Because the loop is ending in the **middle**, the condition can be `True` and the loop will not become infinite.

This is called the *loop and a half* approach!

```
while True:
    my_num = int(input("Enter a positive number: "))
    if my_num > 0:
        break
    print("Number must be positive!")
print("Nice number!")
```

Continuing On

Where `break` will skip the rest of the body of the loop and jump to after the loop, `continue` jumps to the top of the loop instead.

This allows you to skip part of your loop, but keep looping afterward.

When used with a `while` loop, it jumps back up to the `condition` check for the loop.

Break vs. Continue - While Loops

```
# Before Loop
while condition:
    # Do Stuff
    if other_condition:
        break
    # Other Stuff

# After Loop
```

Skips the rest of the loop

```
# Before Loop
while condition:
    # Do Stuff
    if other_condition:
        continue
    # Other Stuff

# After Loop
```

Jumps to the top of the loop
Checks `condition`

Break vs. Continue - For Loops

```
# Before Loop
for i in range(3):
    # Do Stuff
    if condition:
        break
    # Other Stuff

# After Loop
```

Skips the rest of the loop

```
# Before Loop
for i in range(3):
    # Do Stuff
    if condition:
        continue
    # Other Stuff

# After Loop
```

Jumps to the top of the loop
The **iterator** `i` takes its next value