While Loops

If we need to check something multiple times, we can end up with a lot of “if” statements

- E.g., when Upsorn searches for someone with glasses, we would need a new “if” statement for everyone in the class

We can shorten this with a while loop

When a computer encounters a while loop, it checks the condition

- If the condition is true, it runs the code within the loop and checks again
- If it is still true, it runs the code again and checks a third time
- The while loop will continue to run until the condition is not true
  - This means we could end up with an infinite loop!
  - Infinite loops crash computers (or Archimedes)
    - If you submit code with an infinite loop, Archimedes crashes
    - This makes no one get their code back in 2 hours
    - Please don’t do this
  - Always ensure that your condition will be broken at some point before you run the program

Example from powerpoint:

```
ctr = 0
experts = ["wear", "no", "no", "wear", "wear", "no"]
while ctr < len(experts):
   expert = experts[ctr]
   if expert == "wear":
       print("shake")
   else:
       print("no")
   ctr += 1
#must be < len(experts) bc if experts has 6 items and we ask for the 7th item (at
location 6), we will get an error
```

Plug this into Visualize Python for a detailed explanation of what the computer is doing:

- At start, ctr = 0 and len(experts) = 6 so ctr < len(experts), so the code runs
- Experts[ctr] is the value at location 0, i.e., “wear”
- Bc it is “wear”, we print “shake”
- Ctr → 1

- Ctr = 1 and len(experts) = 6 so ctr < len(experts), code runs
- Experts[ctr] is the value at location 1, i.e., “no”
- Bc it is “no”, we print “no”
- Ctr → 2

- Following this, we print “no” “shake” “shake” and “no”

- After the last print of “no”, we increment ctr → 6

Ctr = 6 and len(experts) = 6 so ctr < len(experts) is False
We therefore exit the while loop here

If we wanted to express this with “if” statements, we would need many more lines of code
In addition, if we did not know the length of experts, we would not know how many times to check if there’s another value
This while loop can handle a list with 1 value or 100
Cannot do that with an “if” statement

While is not the only looping keyword
These use a Boolean expression as the condition
If we want to run code a certain number of times, we can use a variable:

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

OR

```
index = 1
while True:
   if index > 10:
       break
   print(index)
   index += 1
```

Both of which print:

```
1
2
3
4
5
6
7
8
9
10
```

For is also used (it looks like `for i in range(#):`)
We will not be discussing for loops today, but they take a set of values and apply the loop to each of the values
this is my words, not Upsorn’s While loops are generally more useful if we don’t know how many times we want the loop to run, but have a specific condition we want to stop it at; For loops are more useful if we have a specific number of times we want the code to run or list to apply the code to, especially if we don’t have a specific condition to stop at

Two more examples:

```
while(input("Do you want to continue (Y/N) ? ")) == Y:
   print("let's continue")

done = ""
while done != "quit":
   print("let's continue -- not done yet")
   done = input("Continue? (type quit to quit)")
```

An infinite loop:

```
cnt = 1
while infinite > -1:
   # infinite is undefined here, so we get an error
   # we assume we are getting infinite from some other code or file
   # cnt cannot get us out of the loop bc it is not in the condition
   # always "take full control of your loop", know that it will break eventually
   print(cnt)
   cnt += 1
```

Loop Examples:

(Not available online)

(If what is being printed doesn't make sense, can always put in Visualize Python)

Code:

```
x = 10
while x > 5:
   print(x)
   x -= 1
```

Prints:

```
10
9
8
7
6
```

Code:

```
#write a loop that takes a word from the user and prints it until the user types "stop"
word = input("Give me a word (type stop to quit) ")
while word != "stop":
   print(word)
   word = input("Give me a word (type stop to quit) ")
```

Prints:

```
Give me a word (type stop to quit) one
one
Give me a word (type stop to quit) two
two
Give me a word (type stop to quit) three
three
```

```
Give me a word (type stop to quit) four
four
Give me a word (type stop to quit) five
five
Give me a word (type stop to quit) gonna stop now
gonna stop now
Give me a word (type stop to quit) stop
***my inputs are in red***
```

Code:

```
#now a while loop that prints "stop" when we stop, and is shorter
word = ""
while word != "stop":
    word = input("Give me a word (stop to stop) ")
    print(word)
```

Prints:

```
Give me a word (stop to stop) test
test
Give me a word (stop to stop) tes
tes
Give me a word (stop to stop) test
test
Give me a word (stop to stop) stop
stop
```

Code:

```
number = -1
while number < 1 or number > 100:
    number = int(input("Give me a number between 1 and 100: "))
    print(number)
```

Prints:

```
Give me a number between 1 and 100: -5
-5
Give me a number between 1 and 100: 500
500
Give me a number between 1 and 100: 50
50
```

In other words, we can use logical operators in the conditions of our while loops. Here's the table again:

x	y	x and y	x or y	not x	not y
False	False	False	False	True	True

False	True	False	True	True	False
True	False	False	True	False	True
Ture	True	True	True	False	False