LOOPS

QUICK INVENTORY

- This week has been about prioritizing *computational thinking* over specific Python syntax; starting next week, it'll swing back a bit in the other direction
- Thanks for being good sports while we try out some new activities
- HW0 Due last night, or today by 11:59pm with a late token
- HW1 Practice & Project released now
  - Practice due by Tuesday, Sep 8
  - Project due by Monday, Sep 14
  - Presentations on Sep 16 & Sep 17

ANY QUESTIONS: LOOPS

REVIEW: FOR VISITS EACH ITEM IN A SEQUENCE.

```
for character in "Hello!":
    print(character)
```

```
'H'
'e'
'l'
'l'
'o'
'!'
```

The loop will:

- Run's the code in the loop once for each item in the sequence
- Each time we run the "body" (code *in* the loop), our loop variable (`character` in this example) will store the next element from the sequence.

REVIEW: LOOP OVER RANGES FOR REPEATING STEPS

Instead of:

```
print("Hello!!!!")
print("Hello!!!!")
...
print("Hello!!!!")
```

We can do something better:

```
for i in range(0, 10): # loop 10 times (0, 1, 2, 3, 4, 5, 6, 7, 8, 9)
    print("Hello!!!!")
    # note how we aren't required to use i
```

REVIEW: FILTER VALUES OUT OF A SEQUENCE

We can use a loop to only print that meet a certain condition. This is called **filtering.**

```
for value in sequence:        # For each value in the source sequence,
  if condition(value):        # if that value meets some condition...
    print(value)              # print it!
```

`condition()` is a placeholder here to represent some boolean expression that helps decide whether or not to include `value`.

```
for num in range(30):
  if num % 3 == 0 or num % 5 == 0:
    print(num)
```

NUM	NUM % 3 == 0	NUM % 5 == 0	CONDITION HOLDS?	PRINTS
0	True	True	True	0
1	False	False	False	
2	False	False	False	
3	True	False	True	3
4	False	False	False	
5	False	True	True	5
(skip a few rows...)	...	...	...	...
29	False	False	False	

REVIEW: AGGREGATION WITH LOOPS

Sometimes, we only want to learn some *property* of a sequence instead of creating a whole new sequence.

- Commonly accomplished with an **accumulator variable:**
  - a variable that has its value updated over successive iterations of the loop
  - important to declare accumulator variables *outside* of the loop so we don't overwrite its value each time.

Consider the following loop: what is the final value of `aggregator`?

```
aggregator = 1
for i in range(1, 6):
    aggregator = aggregator * i
print(aggregator)
```

I	AGGREGATOR (BEFORE UPDATE)	AGGREGATOR (AFTER UPDATE)
1	1	1 * 1 ➡️ 1
2	1	1 * 2 ➡️ 2
3	2	2 * 3 ➡️ 6
4	6	6 * 4 ➡️ 24
5	24	24 * 5 ➡️ 120

🖨️ ➡️ 120

🔍🔍🔍 Let's solve a few puzzles. 🔍🔍🔍

Somewhere in this building, there's a (small) prize to be had. Unfortunately, the location is hidden—but there are clues!

- Take a worksheet and plan to work with some folks sitting near you.
- We'll solve a loop-related puzzle in each part, and each one will give us a piece of a clue.
- We'll put together all the clues in the end to find the location of the prize, which someone will go retrieve.

PUZZLE 1

You have intercepted a coded message stored in a Python variable called `secret_string`.

```
secret_string = "WXARSTTAMAJZLAIRNIMDNIRHPALVZZABANCIEEENML
```

Hints:

- Remember that you can iterate over a sequence of numbers by iterating over a `range()`
- Remember that you can check if a number is a multiple of another by using `%`
- You can access a character at a specific position `i` in a string `s` by writing `s[i]`.

PUZZLE 1: SOLUTION

The clue is hidden in every 9th character: positions 8, 17, 26

```
secret_string = "WXARSTTAMAJZLAIRNIMDNIRHPALVZZABANCIEEENMLTHAS"

clue = ""                                   # accumulator, declared outside th
for i in range(len(secret_string)):         # i takes on every position in the
    if i % 9 == 8:
        print(secret_string[i])
```

MILES

PUZZLE 2

You have intercepted a secret transmission stored in a variable called `jammed_signal`. The enemy tried to destroy the message by jamming it full of `x`s, and `m`s.

```
jammed_signal = "xxxmmmxmdxmxmmmyxmxmmlmxmxmxmxammxmnmx"
```

Hints:

- Remember that you can iterate over characters in a string directly with a for loop.

PUZZLE 2: SOLUTION

Filter out the jamming characters and keep everything else.

```
jammed_signal = "xxxmmmxmdxmxmmmyxmxmmlmxmxmxmxammxmnmx"

message = ""                                # accumulator, declared outside
for character in jammed_signal:             # visit each character in the st
    if character != "x" and character != "m":   # skip the jamming chara
        print(character)
```

PUZZLE 3

You have intercepted a broadcast frequency stored in a variable called `beacon_signal`. To find the next clue, you must count exactly how many times a key letter appears in the broadcast.

```
beacon_signal = "KZKQAKBKQKKZCKDQKEKKQFKZGKHKQKJ"
```

Hints:

- Remember that you can declare an *aggregator* variable outside of a loop that you can update within the loop

PUZZLE 3: SOLUTION

The key letter is `Z`. Count it with an accumulator.

```
beacon_signal = "KZKQAKBKQKKZCKDQKEKKQFKGKHKQKJ"

count = 0                              # accumulator, declared outside the loo
for character in beacon_signal:        # visit each character in the string
    if character == "Z":               # only the key letter counts
        count = count + 1
print(count)
```

🖨️ ➡️ `3`