to cover

Unit 4:
Repetition and Loops

4.1 - 4.2: 10/23/2024

to divider

A loop is a section of code that will iterate or repeat over and over again. A control variable is what helps to “control” the entire while loop.
Four steps for a while loop:

1. Initialize the control variable
2. Create a while loop with a condition that uses the control variable
3. Code that should run while the condition is true
4. Update the control variable

While loop example:

```
 name = input("Enter a name, STOP to end")

while name != "STOP":
    print("You entered: " + name)
    name = input("Enter a name, STOP to end")
print("Done.")
```

Explanation:
We first ask for a name, and store it as the variable name - this is the initialization step. Next, we have a while loop. The while loop first checks if name is something other than "STOP" using !=, a relational operator. It lets Python know that the while loop should run only when the condition is True (when name does not equal "STOP"), and that the loop should stop when the condition is False (when name equals "STOP").

Count variable: variable used to control how many times a loop runs or keep track of something in our loops
Sum variable: variable that keeps track of things each time through the loop, and can be used to add up a set of numbers

4.2 - 4.2: 10/23/2024

⬅️ BACK to divider

Count Variable Example:

```
name = input("STOP to end: ")

c = 0
while name != "STOP":
    print("You entered: " + name)
    name = input("STOP to end: ")
    c = c + 1

print("You entered " + str(c) + " words.")
```

Sum Variable Example:

```
n = int(input("Enter a number, -1 to stop: "))

sum = 0
while n != -1:
    sum = sum + n
    print("You entered: " + str(n))
    n = int(input("Enter a number, -1 to stop: "))

print("Sum of numbers entered: " + str(sum))
```

To-do:

- ✅ Edit portfolio
- ❑ Look into CFM scholarship
- ✅ Finish Assignment Unit 4

Class Notes - 10/25/2024

to divider

```
x = 10
For i in range(4):
        x = x + 1
        print(x)
```

*Loop runs 4 times
*i is a local (iterating) variable
*x is a global variable

For loops ranges:
Ranges are written in 3 different ways

- range(end)
- range(start,end)
- range(start,end,step)

```
for x in range(0,5):
    print(x)
```
output: 0,1,2,3,4

```
for x in range(1,7):
    print(x)
```
output: 1,2,3,4,5,6

```
for x in range(1,10,2):
    print(x)
```
output: 1,3,5,7,9

```
for x in range(30,10,-3):
    print(x)
```
output: 30,27,24,21,18,15,12

Class Notes - 10/29/2024

to divider

For loops vs while loops:
While loops run an infinite number of times, so long as their condition is TRUE!

```
1  i = 0
2  while i < 5:
3      print(i)
4      i = i + 1
```

Continues running until i is greater than 5
Prints 0,1,2,3,4

```
1  i = 0
2  while i < 5:
3      i = i + 1
4      print(i)
```

Follows same logic
Print 1,2,3,4,5

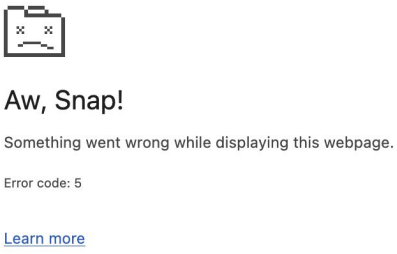
While Loop

```
1  n = 5
2
3  while n > 0:
4      print("Rinse")
5      print("Lather")
6  print("Dry off")
```

Outcome

```
Stopped after running 1000 steps.

Congratulations, you've crashed your browser!
```



*From 4.7:

A for loop is used only when we want to execute code an exact number of times, using the range() function - not user input.

A for loop cannot replace the power of a while loop when we need user input to control the loop and when it runs an indefinite amount of times.

General rule:

- If a loop runs an exact number of times, use a for loop
- When we don't know how many times it runs (user input controlled), use a while loop

4.3 - 4.5: 10/30/2024

to divider

-> Regardless of which way you choose to control a loop - with a count variable or user input - we follow the same four steps:

1. Initialize our control variable (user initialized or set by the programmer)
2. Create a while loop with a condition that uses the control variable
3. Code that should run while the condition is true
4. Update the control variable

- If we know exactly how many times something is going to run, we use a count variable
- If you don't know how many times something will run, we need user input

```
count = 0

while count < 5:
    n = int(input("Enter a number: "))
    count += 1
```

Although we are prompting the user for input, we are getting exactly 5 numbers from the user. We are going to need a loop to run exactly 5 times - so a count variable is needed.

```
n = int(input("Enter a number greater than 100: "))

while n <= 100:
    n = int(input("Too small, enter a number greater than 100: "))
```

This kind of loop will be user input controlled, because the user could input a big enough number the first time, the second time, the 100th time... we don't know how many times the loop needs to run.

The following code uses a loop to find the maximum of values the user enters:

```
n = int(input("Enter a number (-1 to STOP): "))
maximum = n

while n != -1:
    n = int(input("Enter a number (-1 to STOP): "))
    if n > maximum:
        maximum = n

if maximum == -1:
    print("There is no maximum.")
else:
    print("The maximum is: " + str(maximum))
```

- A **user-controlled loop** asks the user to input information until a test condition is met. The number of times a user-controlled loop will repeat is variable and unknown when the loop begins running.
- A **count loop** uses a count variable to count the number of times the loop will repeat. The number of times a count loop will repeat is known before the loop even begins running.

4.6 - 4.7: 10/31/2024

to divider

Range() Function:
Start, stop, and step are all parameters that change how the range function will work.

- start - an optional integer parameter that specifies which position to start; default is 0 if not specified
- stop - a required integer parameter that specifies at which position to stop
- step - an optional integer parameter specifying the increment between values; default is 1 if not specified

range(5) prints 0,1,2,3,4 since 5 is the stop
range(10) prints 0,1,2,3,4,5,6,7,8,9 since 10 is the stop
range(5,10) prints 5,6,7,8,9 since 5 is the start and 10 is the stop
range(0,10,2) prints 0,2,4,6,8 since 0 is the start, 10 is the stop, and 2 is the step

The range() function creates a sequence of numbers based on a set of parameters.

The for loop is a replacement for a while loop that uses a control variable.

Example:

```
for n in range(1, 4):
    print(n)
```

Output:
1
2
3

Summing using for loop:

```
sum = 0
for i in range(2, 6):
    sum = sum + i
print(str(sum))
```

adds 2 + 3 + 4 + 5 and prints:

14