While loops with input

Why is this important?

While loops are really important loops because they continue executing until a certain condition is met.

While loops are great for when you don't know how many times you want to loop over something.

What are we going to do?

We're going to create a program that will prompt the user for golf scores and give them an average of their scores.

Steps

1. Let's create a new file called `golf_scores.py`

2. In that file let's prompt the user for their golf score.

- Let's start by making one input and calculate the average of that input.

```
print("Golf Score Calculator")
count = 1
score = input("What was your most recent golf score? ")

average = int(score)/count
print(f"Your average golf score is {average}.")
```

- The output looks as follows:

```
$ python golf_scores.py
Golf Score Calculator
What was your most recent golf score? 90
Your average golf score is 90.
```

- Now we know how to do this with a single input. Let's try to do this with a loop.
  - We've learned about for loops which are great but we're going to use a while loop for this. We need a while loop because we can't guarantee that the user is going to put in a set number of scores.

2. Let's add a loop so that we can get as many scores as the user wants.

- We're going to add a boolean named `active` and set that to true.
  - We're going to use that boolean as our condition in our while loop to either continue iterating (executing the indented block of code) or stop.
- Let's take a look at the code.

```python
print("Golf Score Calculator")
count = 0
total_score = 0
active = True

while active:
    user_input = input("What was your most recent golf score? (enter 
'quit' to stop) ")
    if user_input == 'quit': # quit if the user enters 'quit'
        active = False
    else: # add the score to the total and increment the count
        total_score += int(user_input)
        count += 1

# use the average.

average = total_score / count
print(f"Your average golf score is {average}.")
```

- Let's take a deeper look at this.
    - the `while` loop executes the indented block of code as long as the condition to the right of it is true. For us `active = True` so we know it's going to execute at least once.
    - we're getting some `input` from the user (that is a string) and we're using our `if` statement to check if the user entered `quit`
        - if they did we're setting `active = False` which will stop the loop.
        - if the user didn't enter `quit` then we're going to add the score to the total and increment the count
    - Once the loop is complete it moves on to the next line of the code which calulates the average and prints it out.
- Note: if you want to step through it together you can add the line `breakpoint()` under the `active = True` line and run through the program with the skill that you've learned by debugging your code.
- the output looks as follows:

```
$ python golf_scores.py
Golf Score Calculator
What was your most recent golf score? (enter 'quit' to stop) 90
What was your most recent golf score? (enter 'quit' to stop) 80
What was your most recent golf score? (enter 'quit' to stop) 92
What was your most recent golf score? (enter 'quit' to stop) 89
What was your most recent golf score? (enter 'quit' to stop) quit
Your average golf score is 87.75.
```

- This concept is called "Using a flag" where the flag is the boolean that we're using to determine if we should continue looping or not.

3. Using break to exit a loop.

- Using flags are great but sometimes we just want to exit a loop completely. This is where the `break` keyword comes in.
    - Let's modify our code so that we remove our `active` flag and use the `break` keyword.

```
print("Golf Score Calculator")
count = 0
total_score = 0

while True:
    user_input = input("What was your most recent golf score? (enter
'quit' to stop) ")
    if user_input == 'quit': # quit if the user enters 'quit'
        break
    else: # add the score to the total and increment the count
        total_score += int(user_input)
        count += 1

# use the average.

average = total_score / count
print(f"Your average golf score is {average}.")
```

- Looking at the code here you see that we've removed the `active` flag, and we've changed the condition of the while loop to `True` which means that it will always execute.
    - If you don't have a `break` statement in the loop then it will execute forever, which in programming is something that you don't ever want to do.

Conclusion

Here we've done a couple of examples that show how to use while loops. We've also shown how to use a flag to exit a loop and how to use the `break` keyword to exit a loop.

We also learned how to step through a while loop using the `breakpoint()` function.