print("Hello, World!")

NCSS Challenge - Beginners
Week 4 Part 1
What will we cover?
- Looping with conditions;
- Counters;
- Counting with while loops.
What does this cover?

- Design, modify and follow simple algorithms involving sequences of steps, branching, and **iteration (repetition)** (ACTDIP019)
- Implement digital solutions as simple visual programs involving branching, **iteration (repetition)**, and user input (ACTDIP020)
- Implement... **iteration** and functions in a **general-purpose programming language** (ACTDIP030)
Looping with conditions
Loops so far...

- So far we’ve seen for loops over:
  - Strings (looping over each character)
  - Numbers (counting between two numbers)
- These only work if we already know how many times we want to loop, or how many characters a word has.
- We might want to loop until something changes!
- For this we need a **while** loop!
Using a while loop, we can repeat some code while a condition is true:

```python
command = input('First command: ')
while command != 'stop':
    print('You entered ' + command)
    command = input('Command: ')
print('Stopped.')```

While loops!
Let’s look in diagram form.

1. Ask user for the first command
2. Save that user input as command variable
3. Is command not equal to 'stop'? (Decision)
   - True: Print the command
   - False: Ask user for the next command
4. Save that user input as command variable
5. Print 'Stopped.'
Teacher Aside! While loops are particularly tricky!

- There are lots of ‘gotchas’ with while loops. They can be a tricky thing for students to get their heads around.
- One fun activity is to get students to design an algorithm for, say, eating a bowl of cereal, - continue until it is finished.
Some common hiccups - not starting right:

- Being finished before you even start!

```python
command = 'stop'
while command != 'stop':
    print('You entered ' + command)
    command = input('Command: ')
print('Stopped. ')
```

- The command was already 'stop', so the loop never ran!
Some common hiccups - bad setup:

- Forgetting setup

```python
while command != 'stop':
    print('You entered ' + command)
    command = input('Command: ')
print('Stopped.')
```

NameError: name 'command' is not defined
Some common hiccups - infinite loops!

```python
command = input('First command: ')
while command != 'stop':
    print('You entered ' + command)
    command = input('Command: ')
print('Stopped. ')
```

The `while` loop will keep repeating `while` the condition is `True`. In this example, we ask for the first command, but then we never ask for the next command!
Test it out!
Try the first question now!
Looping until blank lines

- A very common pattern is to loop until a line is blank:

```python
line = input('Enter line: ')  
while line:  
    print('Still running')  
    line = input('Enter line: ')  
print('Stopped')
```

*Note: We used a shortcut here, we could also use:*

```python
while line != ''
```
Test it out! (Put it together!)
Try the second question now!
Counters - keeping track
We can use a counter, a variable that starts at 0 or 1, to keep track of how many times a loop runs:

```python
counter = 1
guess = input('Guess my favourite colour: ') while guess.lower() != 'yellow':
    counter = counter + 1  # Add 1 to counter.
    guess = input('Try again: ')
print('You got it in', counter, 'tries.')```
Teacher Aside! Counting up!

- This example counts from 1, rather than 0, because by the time we are incrementing the counter, we have already made one guess!
Teacher Aside! Incrementing counters like a pro!

- There’s another way to increment the counter! In the previous example, we do:
  ```python
counter = counter + 1
  ```
- We can also use the following notation:
  ```python
counter += 1
  ```
- They both mean:
  *Take the value in `counter` and add one to it.*
Just like we add 1 to a counter each loop iteration, we can also add to a string each time the loop repeats.

```python
message = 'e'
while message != 'eeeee':
    print(message)
    message = message + 'e'
```

Building up a string
Test it out!

Try the third question now!
Numbers in while loops
Using numbers in while loops for numerical decisions

- We can use mathematical conditions in while loops:

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

While i is less than 3, print i, and add 1 to i.
Some common hiccups - not starting right:

- Not meeting the loop criteria initially:

```
i = 0
while i > 3:
    print(i)
    i = i + 1
```
Some common hiccups - not starting right:

- Not incrementing the value of `i`:
  ```python
  i = 0
  while i < 3:
    print(i)
  ```

- This will cause an infinite loop!
Instead of incrementing i by 1, we can add a bigger step! For example, we can print only odd numbers:

```python
i = 1
while i < 10:
    print(i)
    i = i + 2
```

While i is less than 10, print i, and then add 2 to i.
Loop countdown!

- Just as we can step up, we can also step down!

```
j = 10
while j > 3:
    print(j)
    j = j - 1
```

*While $j$ is more than 3, print $j$, and then reduce it by 1.*
Test it out!

Try the fourth question now!
Any Questions?

Find me at:
@groklearning
nicky@groklearning.com