print("Hello, World!")

NCSS Challenge - Beginners
Week 2 Part 1
What will we cover?

- Making decisions;
- Decisions with two options;
- Decisions about numbers;
- Complex decisions.
What does this cover?

- Define simple problems, and **describe and follow a sequence of steps and decisions** (algorithms) needed to solve them (**ACTDIP010**).
- Implement simple digital solutions as visual programs with algorithms involving **branching (decisions)** and **user input** (**ACTDIP011**).
- Design algorithms represented diagrammatically and in English, and trace algorithms to **predict output for a given input** and to identify errors (**ACTDIP029**).
Making Decisions with Code

Hello, World!
Programs without decisions get a bit boring!
Programs without decisions get a bit boring!

- Get ready to leave
- If it is raining:
  - Pick up an umbrella
- Leave the house
if statements determine True and False

- if statements work out if a condition is True or False. If the condition is True then the block controlled by the if statement will be run.

```python
raining = input('Is it raining? ')
if raining == 'yes':
    print('Pick up an umbrella.')
print('Leave the house.')
```
Indentation woes!

- Indentation makes a big difference in Python. It controls how a program flows.
- Encourage students to always use a consistent level of indentation - two spaces, for instance.
Testing equivalence with `==`

- **Assign** a value to a variable using `=`
  ```python
  flavour = 'mango'
  ```
  *Think: Let `flavour` equal `'mango'`*

- **Test** whether a value equals another value using `==`
  ```python
  if flavour == 'mango':
  ```
  *Think: Is `flavour` equivalent to `'mango'`*
Equals and Equivalent Errors!

- It’s very easy to get = and == mixed up!
Doing multiple things in an if statement block:

```python
food = input('What food do you like? ')
if food == 'cake':
    print('Wow, I love cake too!')
    print('Did I tell you I like cake?')
```
Indentation matters!

- The lines of code that are indented under the `if` statement are run *only if* that condition is true.
- If one line is indented with two spaces, and the next line with three, Python will raise an `IndentationError: unexpected indent`.
Doing multiple things in an if statement block:

- You can use a print statement to test conditional expressions:

```python
name = "Nicky"
print(name == "Nicky")
→ True
```
○ You can use the interpreter to ‘inspect’ whether a conditional expression evaluates to *True* or *False*:

```python
>>> name = "Nicky"
>>> name == "Nicky"
True
>>> name == "Sam"
False
```
Test it out!

Try the first question now!
2 Decisions with two options
Decisions with two options

- Get ready to leave
- Is it raining?
  - Yes: Pick up an umbrella
  - No: Put on a hat
- Leave the house
Decisions with two options: the `else` keyword

```python
raining = input('Is it raining? ')  
if raining == 'yes':  
    print('Pick up an umbrella.')  
else:  
    print('Put on a hat.')
```
Teacher Aside: The case of the forgotten colon.

- Watch out for those colons! Students often forget the colon at the end of a condition. One easy way for students to notice this is if automatic indentation doesn’t seem to be working correctly!
Test it out!

Try the second question now!
Comparing Things
Comparison operators for use in if statements

<table>
<thead>
<tr>
<th>Operation</th>
<th>Operator</th>
</tr>
</thead>
<tbody>
<tr>
<td>equal to</td>
<td>==</td>
</tr>
<tr>
<td>not equal to</td>
<td>!=</td>
</tr>
<tr>
<td>less than</td>
<td>&lt;</td>
</tr>
<tr>
<td>less than or equal to</td>
<td>&lt;=</td>
</tr>
<tr>
<td>greater than</td>
<td>&gt;</td>
</tr>
<tr>
<td>greater than or equal to</td>
<td>&gt;=</td>
</tr>
</tbody>
</table>
Comparison Operators, bigger, smaller, equal or not!

- Now that we have more comparison operators, we can make decisions on lots more things!

```python
x = 5
print(x <= 10)  # → True
```
Teacher Aside! Gotchas with comparisons:

- Comparing things that aren’t the same type!

```python
>>> '10' < 10
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: unorderable types: str() < int()
>>> '10' != 10
True
>>> '10' == 10
False
```
We can use the conditional operators to make decisions based on numerical input:

```python
height = int(input('How tall in cm? '))
if height > 120:
    print("Tall enough for this ride.")
else:
    print("Not tall enough for this ride.")
```
Test it out!

Try the third question now!
3  Nested decisions
Making decisions in decisions!

- The body of an `if` statement may contain another `if` statement. This is called *nesting*.

```python
x = int(input('Enter a number: '))
if x <= 3:
    print('It is less than or equal to three')
if x >= 3:
    print('It is greater than or equal to three')
```
You could nest decisions inside one another...

```python
x = int(input('Enter a number: '))
if x < 3:
    print('x is less than three')
else:
    if x == 3:
        print('x is equal to three')
    else:
        print('x is greater than three')
```

Making decisions between multiple options
Elif often makes for a much nicer option

```python
x = int(input('Enter a number: '))
if x < 3:
    print('x is less than three')
elif x == 3:
    print('x is equal to three')
else:
    print('x is greater than three')
```
Test it out!

Try the third question now!
Teacher Aside!

- Nested if statements are a tricky one to grasp. We don’t test them here - only introduce them as a concept. We’ll keep working up to them!
Any Questions?

Find me at:
@groklearning
nicky@groklearning.com