Print Statements

The `print` statement is your primary method of outputting text or any sort of value onto the console. This is helpful for two reasons:

- Debugging: You can visually check values and behaviors within your code
- Output: You can show to the user the results of your code

An example of the `print` statement can be seen below:

In [ ]:
```
# Example 1-1: the print statement
print("Howdy, World!")
```

```
Howdy, World!
```

Once you run the code above, you should have seen the sentence, "Howdy, World!", be displayed. Note that the `print` statement needs to know what to output into the screen. The "stuff" you put inside the parentheses is what ends up being outputted. In future lectures, you will be taught of different ways to format the contents that go into those parentheses.

But for now, understand that the most basic function of `print` is to display text (the characters that appear in between a pair of double quotes ("") or single quotes ('')) or display numbers (does not require using quotations as Python understands how to output numbers).

Try modifying the text in the below `print` statements if you wish to to play around with the `print` statement.

In [ ]:
```
# Example 1-2: the print statement (again)
print("CHANGE THIS TEXT WITHOUT REMOVING THE QUOTES")
print(5)
```

```
CHANGE THIS TEXT WITHOUT REMOVING THE QUOTES
5
```

Also note that if you put nothing inside the parentheses, you get a blank line:

In [ ]:
```
# Example 1-3: Blank Line
print("Howdy,")
print()
print("World!")
```

```
Howdy,

World!
```

Basic Math

Python does know some basic math. These operations include your fundemental four:

- Addition (`+`)
- Subtraction (`-`)
- Multiplication (`*`)
- Division (`/`)

and three more useful operators:

- Power (`**`)
- Floor (Integer) Division (`//`)
- Modulus (`%`)

The first five listed (fundamental four + power) are self-explanatory. Floor/integer division divides two numbers and only returns the whole part: the remainder is thrown away. Modulus is the opposite of floor/integer division: the remainder is returned and the whole part is thrown away. The below example demonstrates the seven operations in action:

In [ ]:
```
# Example 1-4: The basic math stuff
print(1 + 1)
print(1 - 1)
print(2 * 3)
print(4 / 3)

print(3 ** 4)
print(4 // 3)
print(4 % 3)
```

```
2
0
6
1.3333333333333333
81
1
1
```

Comments

You may have noticed that in my examples so far, the first line doesn't seem to do anything. It surely says something, such as `# Example 1-3: The basic math stuff`, but Python seems to ignore the lines that start with a `#`. This is what we call a comment: any line that starts with a `#`. These comments are purposefully ignored by Python and serve to let us... put comments... on our code.

Comments are integral to setting up the structure of your code and helping others understand your code. Learning how to code can lead to very messy solutions, which is completely fine! However, people may not be able to understand right away, especially with more complex solutions, so comments can help others (such as PTs like me) to understand your thought process and your code.

Multi-Line Comments

A special type of comment exists, and it's called a multi-line comment. Remember how with `print` statements, you had to surround text with a double (or single) quote on each side? If you do three quotes on each side instead, you get a multi-line (text that can go on multiple lines). If you start a line with a multi-line, then you have a multi-line comment. This distinction and definition is demonstrated in Example 1-4:

In [ ]:
```
# Example 1-5: Multi-lines
"""
This
is a
multi-line
comment
and will be ignored
"""

print("""
This is another multi-line,
but this will be outputted.
 """)
```

```
This is another multi-line,
but this will be outputted.
```