WEEK 1:
HELLO WORLD

Printing text

This week we looked at how we can use Python to write basic things to the console. We can use the `print()` function to do this. The way we tell Python what text to write is by enclosing the text with quotation marks like so:

```
print("hello there! My name is ben")
print("this is another line!")
print("and another")
print("...")
print("you get the idea!")
```

```
Python 3.8.2 (default, Feb 26 2020, 02:56:10)
> hello there! My name is ben
> this is another line!
> and another
> ...
> you get the idea!
```

(Don't forget, you can use repl.it to run and test your Python code!)

Using the print function, can you:

1. Write your name and where you're from.
2. Write the number 10 to the console
3. Draw a box: A line of equals characters (=), a message, and then another line of equals characters.
4. Using the asterisk (*) character, draw a small triangle (3 lines) and a big triangle (5 lines)

See below for some examples!

```
Hello my name is
ben,and I'm from
Lincoln, UK!
```

```
============
Hello there
============
```

```
  *
 * *
* * *

    *
   * *
  *   *
 *     *
* * * * *
```

The perils of printing

When you use the print to write out some text (we call text a “string”: a string of characters) we must make sure we use quote marks. This is different if we want to print out a variable (but we’ll talk about that next week). We can use double quote marks (“) or single quote marks (‘) but these ***MUST*** be consistent -- you can’t start with double quotes and end with single quotes.

```
print(“hello there, this is a string”)
print(‘hello there, this is a string’)

print(hello there, this is a string)
print(“hello there, this is a string’)
print(‘hello there, this is a string”)
```

Try these examples in repl.it. What happens if you run the ones which are highlighted in red? Do they give errors? What do the errors mean? Can you fix them?

```
print(5 + 10)
```
```
15
```

```
print(2.91 * 3)
```
```
8.79
```

```
print(“5 + 10”)
```
```
5 + 10
```

Calculations!

Python allows us to not only print text to the console, but also the result of some mathematical calculation. The jargon term for this is an “expression”.

So how do we do this? Well, if we write the mathematical calculation without quote marks (if there are no quotes, and numbers instead) Python will assume you mean an calculation and not some text. In this case, it will compute the calculation and show the result.

But what if we use quotes? Well, Python will just print out that text in the console!

Experiment using the print statement for mathematical expressions:

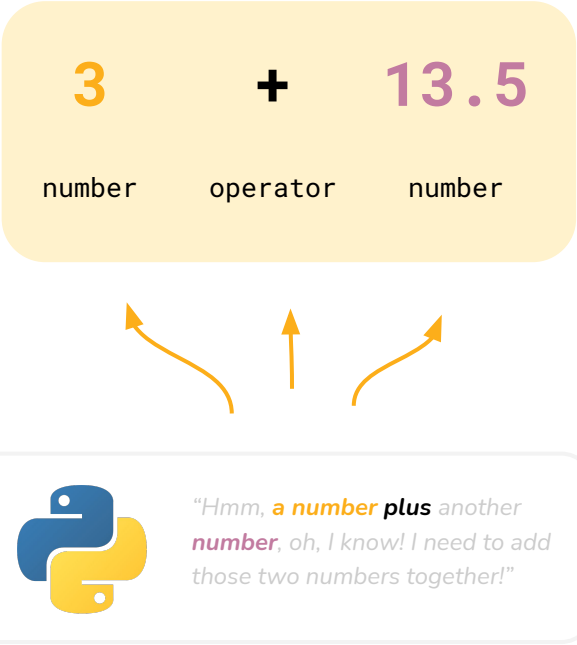
1. Can you show the result of 5 + 3? What about 5 - 3? Can you add -3.1 to say, -5 for example? (`-3.1 + -5`)
2. Can you multiply 3.141592 by 2 and show the result? Use the asterisk character (*) for multiplication.
3. Investigate the forward slash (/) and double asterisk (**) operators. What for example, does `2 ** 3` output? What about `10 / 3`?
4. Can you write out the text “`5 + 3 = `” on one line, and the result of `5+3` below it?

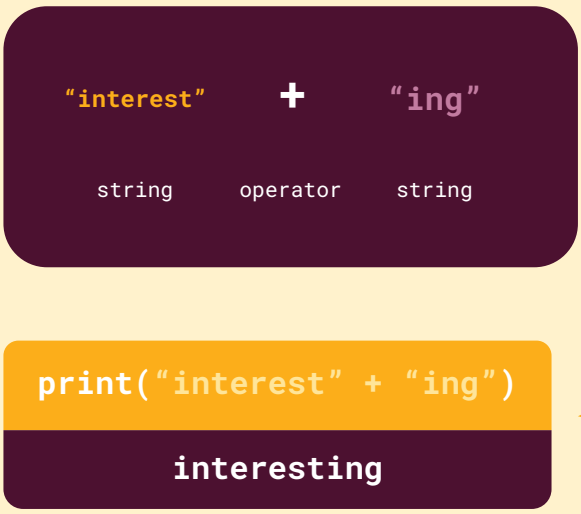
How Python "evaluates" what to do

You might wonder how Python figures out how whether to do a calculation or write some text.

What Python does is looks at the type of data on each side of the operator (operators are just a name for things like +, -, *, / and so forth) and considers what to do based on that.

For example, if Python sees numbers like 3 or 13.5 on each side, it knows you mean a calculation. If however, all of this is within quote marks (i.e. "3 + 13.5") then Python knows you mean to show that text.





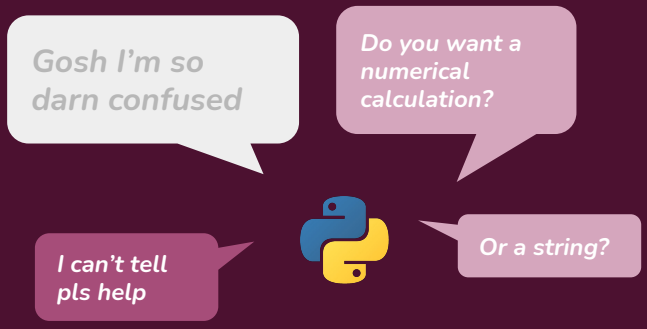
Where things get interesting

But what if Python doesn't see numbers on either side, but instead, **strings**? Well Python interprets this as a process called *"concatenation"* which is jargon for *"take these two strings, and glue them together"*.

The idea here is Python takes a look at each side of the operator, sees what it is, and from that determines what it will do. Two numbers? Well it'll compute the calculation. Two strings? Well it will glue them together.

In some cases, Python will have no idea based on what type of data is on either, and the operator. What happens if you try to subtract one string from another, like "abc" - "def"? Well, Python has no clue what to do, gets a bit confused, and throws us an error.

Another common thing you might come across is trying to print a number within a string. Python doesn't know what to do when it sees the + operator, a number on one side, and a string on the other. It doesn't know whether to try calculate a result or glue two things together. In this case, it will show an error and tell you it got a little confused.



Getting around it

If we wanted to specifically tell Python to convert a number into a string, we can use the str() function. This tells Python to specifically consider the things between the brackets as a **string,** *not* a number. This is very useful for printing out the results of mathematical calculations!

```
print("ben" + "10")
```

```
ben10
```

```
print("ben" + (5 + 5))
```

```
ERROR! Number on right, string on left!
```

```
print("ben" + str(5 + 5))
```

```
ben10
```

Experiment with string concatenation and operators:

1. Can you show the result of concatenating the string "ama" and "zing"? (use the plus operator)
2. Can you concatenate more than one string? For example, "tre" + "mend" + "ous"? Try it out!
3. What happens if you try print out the result of "hello" + 5? Do you get an error? Why do you think this is?
4. Show the result of a long sum (e.g. 1 + 2 + 3 + 4) on the **same line** as the text "Total: ". Your answer should be something like: "Total: 10". Use the str() function to do this.
5. Investigate what happens when you multiply a string by a number. For example, "abc" * 10. What does this show? How is it useful?
6. Write some code to show the word "hello" with a number of trailing "o" letters, using the steps in the previous task. For example, if I required 15 "o" letters, I would see: hellooooooooooooooo.

NEXT WEEK:

VARIABLES!