# Use Python as a Playground

Estimated Time: **1 hour**

In this assignment you are going to get your first taste of the Python programming language. You'll be learning about how you can define and work with data in Python by experimenting in the Python interpreter. The interpreter allows you to type in Python code and see the results in real-time.

## Running the interpreter

To run the interpreter you first need to start up your Cloud9 Workspace. Then in the console at the bottom type `python3` and hit enter. This will start the Python interpreter – it should look something like this:

```
Python 3.4.3 (default, Oct 14 2015, 20:28:29)
[GCC 4.8.4] on linux
Type "help", "copyright", "credits" or "license" for more information.
>>>
```

The three arrows ( `>>>` ) are your command prompt. When you type in a Python statement it will be executed and you will see the result printed back to you. For example, try typing `1 + 1` into the prompt, and hit enter. You should see the result printed.

Note

Python 2 vs Python 3: In this course you are using Python 3, the latest version of the Python language. The previous version, Python 2, is still supported and widely used, so you should expect to see Python 2 on the job. The differences between the core syntax of the two versions are fairly small, but you may find it useful to know some of the key differences so you can read and write Python 2 programs. At various points in the course these differences will be pointed out to you in note sections just like this one.

Note

Python 2 vs Python 3: Here's the first key difference: Python 2 is run using the `python2` executable rather than the `python3` executable which runs Python 3.

## Try it!

In this assignment we would like you to spend 30 minutes experimenting with the Python interpreter to try to get a feel for how the language works. To help get started there are a few questions below which you could try to answer. If you aren't sure what the code is doing then try making small changes and seeing what the outcome is. Also try making up some Python statements of your own to see whether they work as you expect.

As you work through the assignment you might be a little unsure of what everything means, make the odd typo, and see some error messages. But don't worry! This is a natural part of learning to program, and shows that you are making great progress. We will revisit the concepts covered here throughout the rest of the lesson, explaining in more detail what the code does and why.

When the 30 minutes is up jump into the next assignment, where you will be set a few small warmup challenges to help you start to think like a Python coder.

## Questions

- What is the difference between these two statements?

```
>>> 1 + 3 * 2
>>> 1 + (3 * 2)
```

- Which of these statements output the value you expect? What is the difference between them?

```
>>> 1 // 2
>>> 1 / 2
```

### Note

Python 2 vs Python 3: Try running `1 / 2` in Python 2, and compare the result with what you see in Python 3. You should notice that the `/` operator in version 2 acts like the `//` operator in version 3.

- What is happening in this code?

```
>>> width = 2.5
>>> height = 4
>>> area = width * height
>>> print(area)
```

- What is the difference between these three statements?

```
>>> Llamas in Pyjamas
>>> "Llamas in Pyjamas"
>>> # Llamas in Pyjamas
```

- What is the difference between the two print calls?

```
>>> area = 50
>>> print(area)
>>> print("area")
```

> **Note**
>
> Python 2 vs Python 3: In Python 2, `print` is a statement rather than a function. So rather than `print(area)` with parentheses, you can write `print area` .

- What happens when you run this statement, and why?

```
>>> "area" = 50
```

- What would you expect these statements to do? What is the difference between them?

```
>>> "Na" * 100 + "Batman"
>>> "Na" * 100.0 + "Batman"
```

- What do the numbers in the square brackets do?

```
>>> word = "floccinaucinihilipilification"
>>> word[10:3:-1]
```

- What does the dot do? What do the parentheses do?

```
>>> word = "bird"
>>> word.upper
>>> word.upper()
```

- What will the following expressions evaluate to, and why?

```
>>> True or False
>>> 5 > 3 and False
>>> 5 > (3 and False)
>>> True and 5 and 0
>>> True > False
>>> not 7 == 8
```

- What are these telling us?

```
>>> dir(str)
>>> help(str) # Press q to exit back to the prompt
```

☆ ☆ ☆ ☆ ☆ · Report a typo or other issue

Completed

Previous Next