Introduction to Python

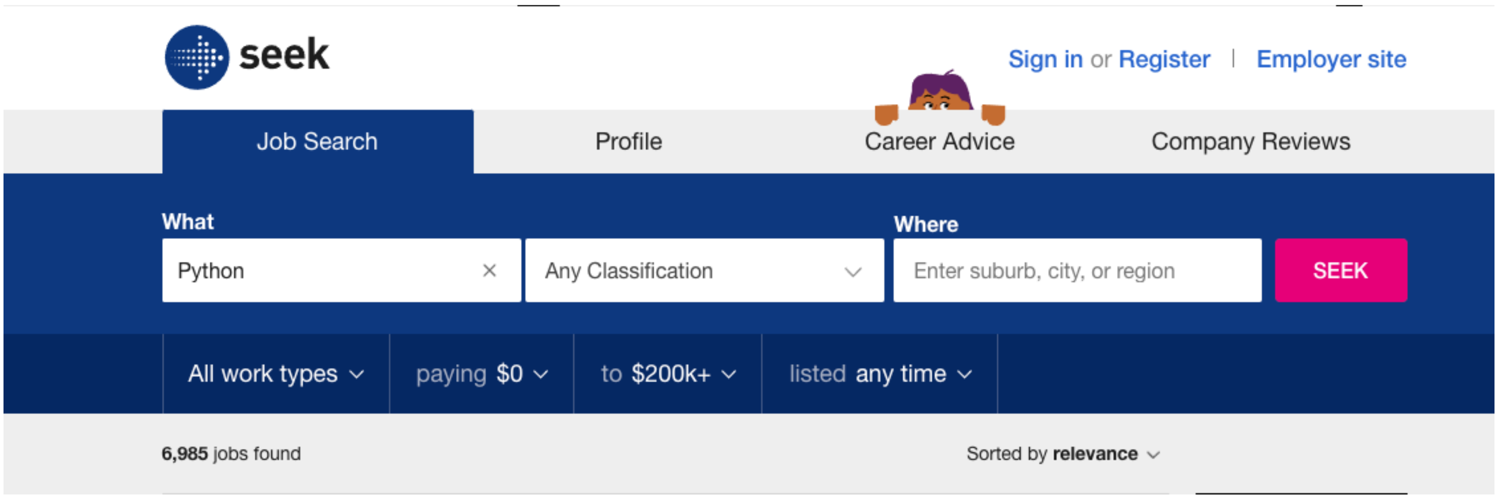
Overview

What we are learning

You're probably familiar with the idea that computers "understand" instructions in Binary. While it is theoretically possible to write programs entirely in Binary code, it would be incredibly laborious and difficult for humans to do. There are many steps involved in translating human ideas to Binary code. Generally languages that are closer to Binary or machine language are referred to as "low level" languages, whereas languages which are more abstracted - or closer to human language - are referred to as "high level" languages.

Being able to work with "low level" languages is an extremely useful and desirable skill, but that is really the domain of Computer Science. At Coder Academy we focus on a more practical application of "high level" programming languages.

One of the most in demand languages currently - in both the Web Development and Data Science space - is Python.



Python is a "multi-paradigm" language, which means you can write code in different ways. It is also highly "extensible", meaning there are many modules you can import to help solve your problem. While there are innumerable ways to write Python code the Zen of Python offers some great advice. Some wisdom from this guide includes:

- Now is better than never.
- Readability counts.

- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated
- If the implementation is hard to explain, it's a bad idea.
- If the implementation is easy to explain, it may be a good idea.

Python was initially developed in the late 1980s, but there have been several major version changes. It's important to know that these major versions are not compatible with each other. Python 2 was released in 2000, and Python 3 was released in 2008. As support for Python 2 officially ended in 2020, we will be learning Python 3.

In this lesson we will be exploring:

- Python Data types
- The Python REPL - a way of running python directly in our terminal
- Python files

Requirements

- Must have completed "Setting Up A Python Environment" Lesson
- Must have a default version of Python > 3.8

Official Documentation

- Python Documentation
- Getting Started with Python
- The Zen of Python

Other Resources

- Check Language Popularity

After this lesson

After completing this lesson you should be able to:

- Utilize the Python REPL for writing and executing snippets of Python code
- Recognize and understand Python data types
- Create and execute Python files

Data Types

One of the primary concerns in computer programming is manipulating data. While at a very low level all data being processed by a computer is Binary - a series of 1s and 0s - it is much easier for humans to classify data into different types. The different "types" of data vary across programming languages, while there is significant overlap (for example Python, JavaScript, and Ruby all have a *string* type) they are not always the same. Python has 15 types, this can be scary for beginners so we'll only look at the first couple in the list right now. The different data types in Python are:

- str
- bool
- int
- float
- complex
- list
- tuple
- range
- dict
- set
- frozenset
- bytes
- bytearray
- memoryview
- NoneType

Depending on your experience some of these may be familiar, or it might all be an alien language to you; either way let's explore some of the more basic data types now, and we'll come back to the more complicated ones later.

Lesson slides will use runnable code snippets where possible. Feel free to edit and play with any of these examples; it won't break anything. Because these are python files, we need to use the "print" function to see any output.

Strings

The `str` data type is a *string*. Strings in computer programming are groupings of letters - as well as symbols, special characters, and even numbers - that make up some text content.

There is a difference between the *int* 1 and the *str* "1" - we'll look at this later.

Unlike other languages, for example C, Python does not differentiate between single characters and collections. Regardless of length, all text in Python is `str` type.

Strings are denoted either by single quotes.

```
print('Hi there')
```

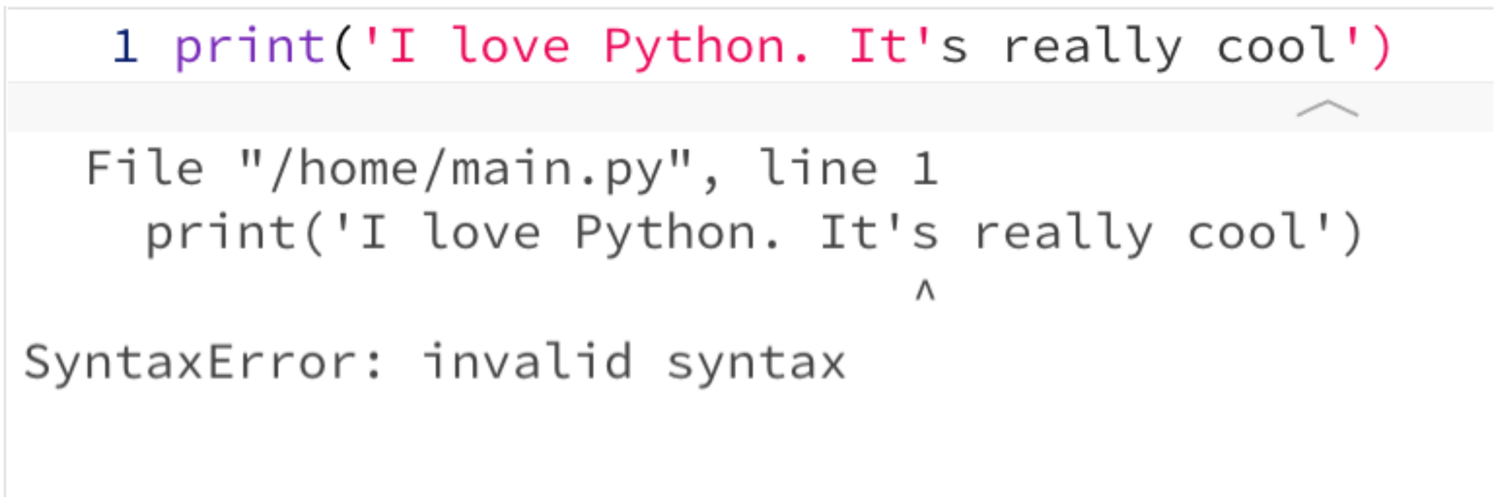
or by double quotes

```
print("Hello there")
```

What would happen if we tried something like this? `⊠ Think about it before pressing Run`.

```
print('I love Python. It's really cool')
```

An error message!



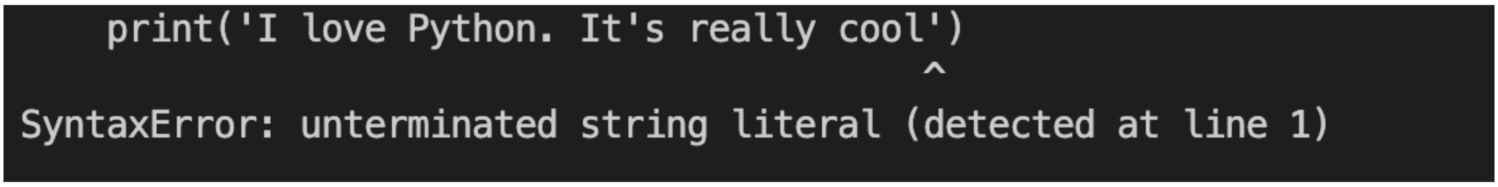
```
print('I love Python. It's really cool')

  File "/home/main.py", line 1
    print('I love Python. It's really cool')
                             ^
SyntaxError: invalid syntax
```

Error messages are extremely common when coding - even seasoned developers make mistakes and typos all the time - learning to understand and use them is what makes a great developer.

This error is very useful, although **invalid syntax** doesn't mean much the chevron `^` is pointing to exactly where our error occurs.

The error is that while we may have meant the apostrophe to indicate a contraction, Python interprets it as the end of the string. So we have the string `'I love python it'` then the `s really` `cool'` part which Python doesn't know how to process. Depending on the environment where you ran this code you may have also seen an error like this:



```
    print('I love Python. It's really cool')
                                          ^
SyntaxError: unterminated string literal (detected at line 1)
```

Here the chevron is pointing to where we start, but fail to close a string.

There are two solutions to this problem. Either *escaping* the apostrophe to indicate that we mean **the literal character** not the end of the string.

```
print('I love python it\'s really cool')
```

Or you can use double quotes to open and close your string.

```
print("I love python. It's really cool")
```

String Concatenation

One of the hardest parts of learning to code is getting your head around all the jargon. "Concatenate" is a fancy way of saying "add together", so "string concatenation" just means the ability to add strings together.

```
print("Hello " + "there" + " friend")
```

The uses for this will become a lot more clear when we look at variables.

We can also multiply strings

```
print("Hello! " * 3)
```

String Interpolation

"Interpolate" means to "insert into", so "string interpolation" is the ability to insert things - *Python Expressions* - into our strings. Again this will be much more useful when using variables, if you're comfortable with them already try them out in the following code blocks.

F Strings

String interpolation in Python has a messy history; other implementations were overly verbose and error prone. We will have a look at them another time, but Python 3 introduced F strings which quickly became the best practice for doing string interpolation. To use an F string you must use double (or triple) quotes, and place an `f` (either upper or lower case works but convention is that we use lower case) before the quotes and place the *Python expression* (which might be a variable) you want to interpolate into the string inside curly braces `{}`

```
print(f"Five times ten is: {5*10}")
```

Numbers

Python differentiates numbers into several different types:

- `int` - Integer or whole numbers
- `float` - Floating point or decimal numbers
- `complex` - Imaginary numbers

Integers

An integer is a whole number (no fractions or decimal points) which is easy for a computer to represent in Binary. Python can perform addition:

```
print(5 + 5)
```

subtraction:

```
print(10 - 5)
```

and multiplication:

```
print(10 * 50)
```

This probably behaves exactly how you expected, but what about division?

Think about what the answer will be before pressing run.

```
print(10 / 3)
```

Where is that 5 coming from?

You might remember from Math class that to represent 1/3 as a decimal we talk about 0.3 with infinite or recursive 3s. A computer can't hold an infinite amount of 3s, so at some point we have to round off. We'll discuss this more when looking at Floats, but it can be the source of bugs if you're not careful

This example ran in Python 3, in any version of Python 2 the answer would be 3.

This is one of the major differences between the two versions of Python; Python 2 assumes that when dividing two integers you must want an integer answer. In Python 3 you can perform integer division with the following

```
print(int(10 / 3))
```

So far we have looked at some mathematical operators:

- `+` for addition
- `-` for subtraction
- `*` for multiplication
- `/` for division

There are a couple more you might not be familiar with. For example `%` the modulo operator. The modulo returns the remainder that we get when performing integer division. For example we know that if we divide 10 Apples (or PCs) between 3 people everybody gets 3 and there is 1 left over. This is represented in code like this

```
print(10 % 3)
```

Floats

Floats or floating point numbers are a computer approximation of representing fractions. Most of the time this behaves exactly as expected

```
print(1 + 0.5)
print(4 * 0.25)
print(10 / 5)
print (1 / 4)
```

But sometimes it doesn't; Consider the following:

```
print(0.1 + 0.2)
```

Common sense might tell you that this is clearly `0.3` but all numbers in computers are stored in Binary, and this sometimes behaves differently in unintuitive ways. For a deeper explanation check the additional resources

Complex

As Python is one of the most popular language choices for Mathematics and Data Science - it sometimes requires the use of Imaginary numbers. The usecases of Imaginary numbers are out of the scope of this lesson. But you can make a number Imaginary in python by adding a j to it

```
print(10j)
```

Boolean

A Boolean - `bool` in Python - is a data type which can only have one of two possible values `True` or `False`; there is a strong parallel here with Binary where a value can either be a 0 or 1. This is by design as a `bool` can be represented by a single bit of Binary.

Booleans are used in a number of ways in computer programming: they might be used to represent certain attributes, for example a User might use a `bool` to represent if they are an admin; they can be used to represent the "state" of an application, for example your application might use a `bool` to represent if it is loading; they are frequency used with comparison operators to control the flow of your code, we'll look at that last one soon.

On their own Booleans don't look that special.

```
print(True)
print(False)
```

But where they start to really work well is with *comparison operators*

Comparison Operators

Comparison operators work similarly to mathematical operators. But they all result in a `bool` value. Comparison operators include:

- `==` Equality operator - checks if two values are equal
- `>` Greater than operator - checks if the value on the left is greater than the value on the right
- `<` Less than operator - checks if the value on the left is less than the value on the right
- `>=` Greater than or equals operator - checks if the value on the left is great than **or equal to** the value on the right
- `<=` Less than or equals operator - checks if the value on the left is less than **or equal to** the value on the right

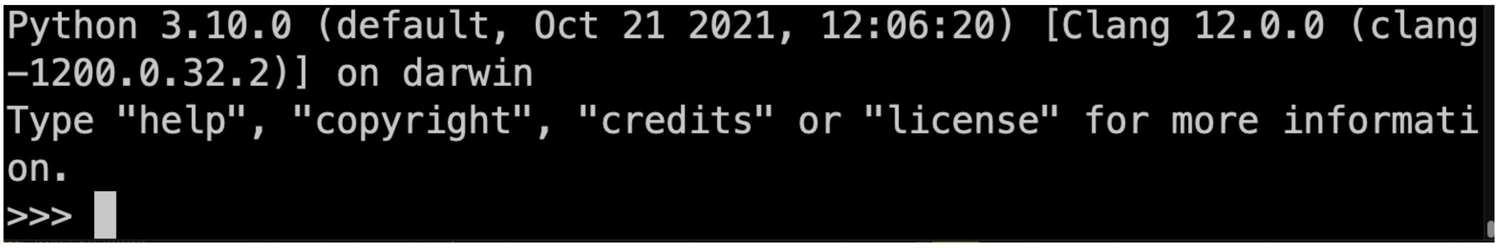
Have a play around with some of these examples. Do any of the results surprise you?

What is a REPL?

REPL stands for **R**ead, **E**val, **P**rint, **L**oop - it's basically a way for us to try out and get some immediate feedback on our code. REPLs are useful when we need to try something out, test a language feature, or reaffirm an assumption without having to create a new file.

To open the python REPL in a terminal window type `python`

You should see something like this:



```
Python 3.10.0 (default, Oct 21 2021, 12:06:20) [Clang 12.0.0 (clang
-1200.0.32.2)] on darwin
Type "help", "copyright", "credits" or "license" for more informati
on.
>>> 
```

You can run any valid python code here, try playing around with some basic math.

Once you're done you can exit the REPL by typing

`exit()`

and pressing enter.

Python Files

A Python file - often referred to as a Python script, sometimes a Python program - is a file ending in the `.py` extension In order to run, or *execute* a Python file we can use the same command to open a Python REPL

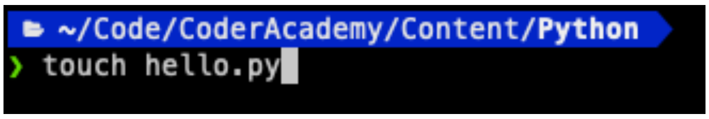
```
python
```

But we also need to pass an additional *argument* to the `python` command, *the path to the file we wish to run*.

Example

In an appropriate directory we could create a new Python file.

```
touch hello.py
```



```
~/Code/CoderAcademy/Content/Python
❯ touch hello.py
```

We could edit this file with any text editor, but remember we can easily open up an entire directory with the command

```
code .
```

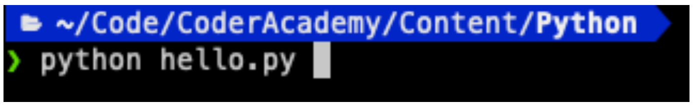
For the sake of the example let's write something very basic, but feel free to add your own flair here.

```
print("Hello from a file")
```

Unlike the REPL, we do not get immediate feedback from our Python expressions when we run a file. This is why we need to output (print) the results.

Remember to save your file, then you can *execute* this program by typing the following into your terminal:

```
python hello.py
```



```
~/Code/CoderAcademy/Content/Python
❯ python hello.py
```

We've successfully created and run our first Python file!

Quiz

This quiz is designed to help you revise the content in this lesson. Actively engaging with the content you worked through earlier helps both with remembering and understanding it.

Question 1

It is possible to write computer programs entirely in binary code

- ○ True
- ○ False

Question 2

Python is a high level language

- ○ True
- ○ False

Question 3

Which of these is a document that offers great advice on how to write Python code?

- ○ Revenge of Python
- ○ Zen of Python
- ○ Pythonistas Rejoice
- ○ Zen of the Snake

Question 4

Which version of Python will we be using?

Question 5

Reorder the words to make a correct sentence

types

python

15

has

data

Question 6

Errors are...

☐ Extremely common when coding

☐ To be avoided at all costs

☐ Your friend!

Question 7

"Concatenate" means

○ to add together

○ a cat that has learnt to con others into feeding it

Question 8

"Interpolate" means

- ◯ to pollute the code
- ◯ to insert into

Question 9

Which of these is valid in python?

- ☐ 5 + 5
- ☐ 9 / 2
- ☐ 2 - 8
- ☐ 6 * 2

Question 10

Click on the symbol that means "greater than or equal to"

`==`
`>`
`<`
`>=`
`<=`

Question 11

Click on the symbol that means "less than or equal to"

`==`
`>`
`<`
`>=`
`<=`

Question 12

Click on the symbol that is the "equality operator"

`==`
`>`
`<`
`>=`
`<=`

Print "Hello world" to the console

A "rite of passage" in programming is to write what's known as a "Hello world" program. Now it's your turn to do just that!

Print out the string "Hello world!" by adding the appropriate line of code to the index.py file. When you click on "run" you should see "Hello world!" in the console.