From JavaScript to Python

Andrew Owens (Student)

May 15, 2025

1 Basics

To get Python working in Visual Studio Code, it's a matter of simply going to Extensions in the left menu, and adding Python. VSCode can tell by the file extension how to handle the file.

2 Introduction

If you're familiar with JavaScript and wanting to learn Python, it is fairly easy – most of the differences are just syntax.

A few differences...

1. Comments are `#` rather than `//`
2. No semicolons at the end of lines
3. No curly brackets in functions and loops – more on that later
4. `print` replaces `console.log`
5. Don't need to have brackets around if, while and for conditions
6. Indentation is now compulsory, not just nice to have

But some similarities...

1. Maths operators are pretty much the same
2. Single and double quotes for strings are exactly the same
3. So is putting backslash before special characters
4. Things are still zero-indexed and still `myList[element]`

2.1 Importing libraries

In JavaScript (using CommonJS) we are used to statements like:

```
const express = require('express');  // Javascript
```

In Python, this is how you do it.

```
from mycode import *          # For *your* code - in mycode.py
import os                     # For big libraries
import numpy as np
```

2.2 Declaring and initialising variables

This is pretty easy – Python does not have `let`, `const` or `var`, so variables are initialised as follows:

```
myInt = 5
myFloat = 12.7
myString = "Hello, world"
myBool = False              # Note first letter is capitalised
```

2.3 Printing to console

Python uses `print` instead of `console.log`. It has some quirks, so we'll look at those here:

```
print("Hello, world")          # Works as expected
print(3, "x", 3, "=", 9)       # Prints "3 x 3 = 9"
print(3 + "x" + 3)             # Not allowed... error!
print(str(3) + "x" + str(3))   # Prints "3x3"
```

String literals are different in Python to JavaScript, but easy. Note Python requires 'f' before the first quotation mark, the backticks are now normal quotes, and the `$` signs are no more.

```
console.log(`My name is ${name} and I live in ${city}. (JS)`);
```

```
print(f"My name is {name} and I live in {city}. (Python)")
```

2.4 Numbers, Data Types and Operators

If you're coming from JavaScript, numbers are handled quite similarly, using almost the same operators. The most notable difference is with integer division: in Python, `10 / 5 = 2.0` , producing a float even if, as in this case, one would expect an integer.

JavaScript has both `null` and `undefined`, whereas Python uses a single equivalent: `None`, which is of type `NoneType`.

Assignment operators are the same – you can use `+=`, `-=`, etc in Python just like in JavaScript. For example, `a += 1` adds 1 to the value of `a`.

One important difference is in comparison operators. In JavaScript, `==` compares values but not types, so a strict equality operator `===` is often used to avoid bugs. In Python, however, `==` always compares both value and type, so there's no need for a separate strict equality operator.

2.5 Loops – and Python indentation

In JavaScript, this is how you code loops:

```
for (i = 0; i < max; i+=1) {
    console.log(i);
}

let done = false
while (!done) {
    // do stuff
}
```

In Python, they look like this instead:

```
for i in range(max):     # from 0 to max - 1
    print(i)

done = False
while not done:
    # do stuff
```

Notice the colon (`:`) at the end of the line and the indentation on the next line? That is not just a style choice – it is very important to how Python functions. In Python, any line ending with `:` begins a 'block', and that block must be indented consistently (typically 4 spaces). The block ends when the indent stops. There are no curly brackets like JavaScript has, and Python uses indentation to decide what to do with and when to run code.

- Even one space out of alignment can confuse Python and cause an error.
- Don't mix spaces and tabs - though a "tab" in VSCode or PyCharm is automatically translated to four spaces. Be very careful with Notepad though.

2.6 If statements

In JavaScript, this is a complex `if` statement:

```
if (grade >= 80) {
    console.log("Excellent!");
}
else if (grade >= 50) {
    console.log("A passable effort");
}
else {
    console.log("Try harder next time!");
}
```

The equivalent statement in Python looks like this. Notice a few key differences: `else if` becomes `elif`, curly brackets are removed, and a colon `:` is used to mark the start of each block. Indentation replaces curly brackets to define scope.

```python
if grade >= 80:
    print("Excellent!")
elif grade >= 50:
    print("A passable effort")
else:
    print("Try harder next time!")
```

While brackets are not required around the condition in Python, you can still use them if the logic gets complex, or simply to improve readability.

Important: In Python, `elif` *must* be followed by a condition (just like `if`), but `else` *must not* have any condition at all. This is different from JavaScript, where `else if` looks like a separate statement.

2.7 Functions

Functions in Python work almost exactly the same way as in JavaScript – only the syntax is different. The keyword `function` is replaced with `def`, but input parameters are still placed in parentheses, and the `return` statement is used in the same way to return a value.

Here's a simple example of a function that squares a number:

```javascript
// in JavaScript
function square(x) {
    return x * x;
}
```

```python
# in Python
def square(x):
    return x * x
```

Just like with `if` statements, note the colon `:` at the end of the `def` line, and the indentation underneath. There are no curly brackets in Python – indentation defines the function body.

You might have come across arrow functions in JavaScript — those short functions written using `=>`. Python doesn't have a direct equivalent, but it does offer something called `lambda` functions which serve a similar purpose for quick, one-line operations.

If you're just getting started, you don't need to worry about `lambda` at all – regular functions using `def` will cover everything you need. But if you're curious, know that `lambda` is Python's way of writing a simple, unnamed function in one line, and you'll likely come across it when working with things like `sorted()`, `filter()`, or other quick data tasks later on.

2.8 Methods

In JavaScript, when defining a method inside a class or object, we typically just write the function name without using the word `function`. For example:

```
class Student {
    sayHello() {
        console.log("Hello!");
    }
}
```

In Python, we still use the keyword `def` to define methods — even inside classes:

```
class Student:
    def say_hello(self):
        print("Hello!")
```

There are two important differences here:

- Python requires the keyword `def` before every method, even inside a class.
- The first parameter of every instance method in Python must be `self`. This refers to the object itself (similar to `this` in JavaScript), and it is passed in automatically.

2.9 Constructors and `__init__`

In JavaScript, when we create a class and want to do something as soon as a new object is created, we use a special method called a `constructor`. For example:

```
class Student {
    constructor(name) {
        this.name = name;
    }
}
let s = new Student("Alex");
```

Python has the same concept, but the special method is called `__init__` (with two underscores on each side). It is run automatically whenever a new object is created from the class.

```
class Student:
    def __init__(self, name):
        self.name = name

s = Student("Alex")
```

A few important notes:

- Python uses `self` to refer to the current object (like `this` in JavaScript).
- You must always include `self` as the first parameter in any method, including `__init__`.
- You don't write `new` in Python when creating an object — just call the class name like a function.

You can also include multiple arguments if needed:

```
class Student:
    def __init__(self, name, age):
        self.name = name
        self.age = age
```

2.10 Running code in Terminal

Running Python code in the terminal is surprisingly similar to running JavaScript with Node.js — the overall process is the same, only the keywords change slightly.

```
# In JavaScript
npm install express
node mycode.js

# In Python (Windows)
pip install scipy
python mycode.py

# In Python (Mac/Linux)
pip3 install scipy
python3 mycode.py
```

Just like `npm` installs packages for JavaScript, `pip` installs packages for Python. And instead of `node` to run a file, we use `python` or `python3` depending on the system.