

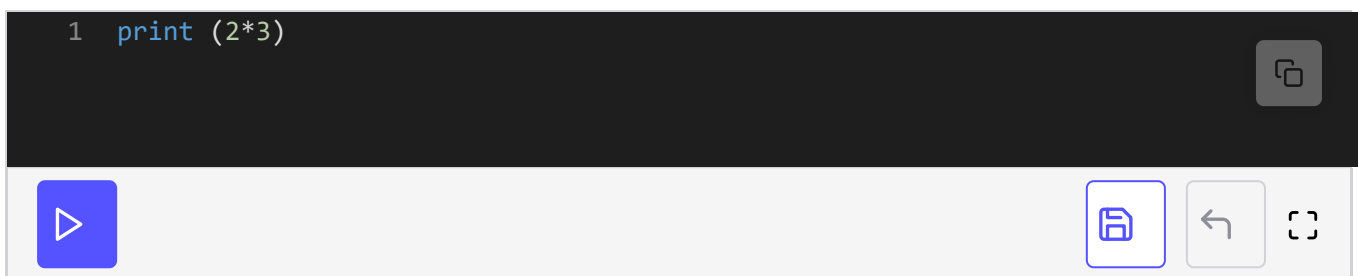
# Getting Started

A basic hands-on Python tutorial for beginners.

We will be using a computer language called Python. Python is a good language to start with because it is easy to learn. It's also easy to read and understand someone else's Python instructions. It is also very popular, and used in many different areas, including scientific research, teaching, global scale infrastructures, as well as data analytics and artificial intelligence.

There is a lot that you can learn about Python or any other computer language, but here we'll remain focused on making our own neural network, and only learn just enough Python to achieve this.

So, let's get started! Let's instruct the computer and ask it to multiply two numbers, say 2 times 3. Let's type `2*3` into the editor and click the run button that looks like an audio play button. The computer should quickly work out what you mean by this, and present the result back to you. Try it yourself:



```
1 print (2*3)
```

You can see the output “6” is correctly presented. We’ve just issued our first instruction to a computer with Python and successfully received a correct result. Our first computer program!

We really meant it when we said Python was an easy computer language. In the next cell, type the following code `print ("Hello World")` and click *Run*. The word *code* is widely used to refer to instructions written in a computer language.



```
1 # Please add your code here
2
3
```



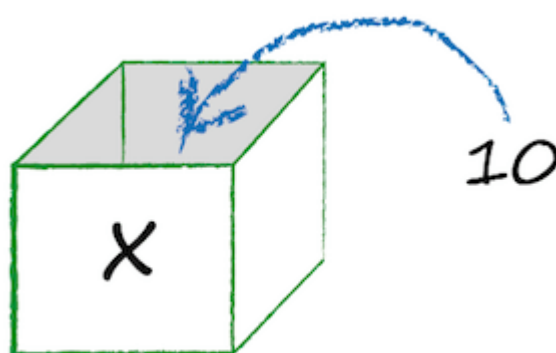
You should get a response which simply prints the phrase “*Hello World!*”

Now, let’s see what’s going on with the following code which introduces a key idea. Run it and take a look at the output.

```
1 x = 10
2 print(x)
3 print(x+5)
4
5 y = x+7
6 print(y)
7
8 print(z)
```



The first line `x = 10` looks like a mathematical statement which says  $x$  is 10. In Python this means that  $x$  is set to 10, that is, the value 10 is placed in a virtual box called  $x$ . It’s as simple as the following diagram shows.



- That 10 stays there until further notice. We shouldn’t be surprised by the `print(x)` because we used the print instruction before. It should print the value of  $x$ , which is “10”. Why doesn’t it just print the letter  $x$ ? Because Python is always eager to evaluate whatever it can, and  $x$  can be evaluated to the value 10, so it prints that. The next line `print (x+5)` evaluates  $x + 5$ , which is  $10 + 5$  or 15, so we expect it to print “15”.
- The next bit  $y = x + 7$  again shouldn’t be difficult you work out if we

follow this idea that Python evaluates whatever it can. We've told it to

assign a value to a new box labeled  $y$ , but what value? The expression is  $x + 7$ , which is  $10 + 7$ , or 17. So  $y$  holds the value 17, and the next line should print it.

- What happens with the line `print(z)` when we haven't assigned a value to  $z$  like we have with  $x$  and  $y$ ? We get an error message which is polite and tells us about the error of our ways, trying to be helpful as possible so we can fix it. I have to say, most computer languages have error messages which try to be helpful but don't always succeed.
- These boxes with labels like  $x$  and  $y$ , which hold values like 10 and 17, are called *variables*. Variables in computer languages are used to make a set of instructions generic, just like mathematicians use expressions like " $x$ " and " $y$ " to make general statements.