

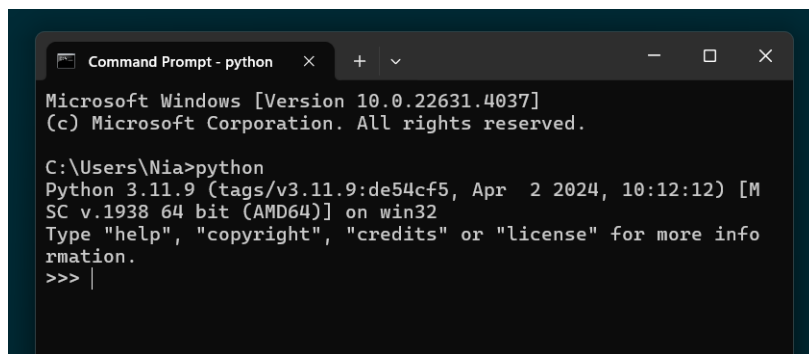
Types, Print, and If Review

1 Installation and Using Python

Installing Python

You can install Python by heading to its [website](#). Just make sure you have the appropriate version for your operating system (Windows, Mac, Etc.), and **if on Windows, check “Add to Path”** on the first screen of the installer. This allows you to run Python code on any folder in your computer!

To check if Python installed right, you can open Command Prompt on Windows by typing `cmd` in your Windows Search Bar in the bottom left corner. Then, inside the black box that opens, type `python` and press enter. You should get a version number if installation was a success! Then type `quit()` to leave this version number screen.

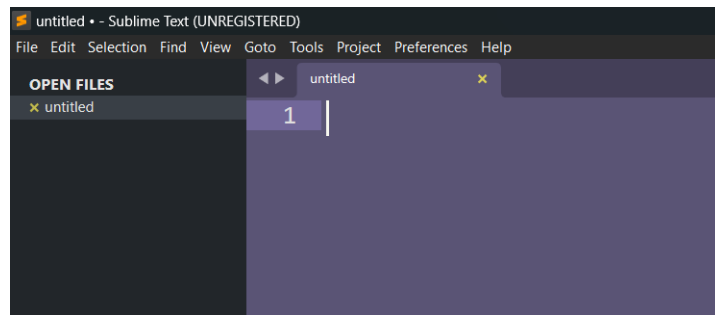
A screenshot of a Windows Command Prompt window titled "Command Prompt - python". The window shows the output of running the command 'python'. The text displayed is: "Microsoft Windows [Version 10.0.22631.4037] (c) Microsoft Corporation. All rights reserved. C:\Users\Nia>python Python 3.11.9 (tags/v3.11.9:de54cf5, Apr 2 2024, 10:12:12) [MSC v.1938 64 bit (AMD64)] on win32 Type "help", "copyright", "credits" or "license" for more information. >>> |". The prompt is currently at the '>>>' line, indicating the interactive Python shell is running.

Successful Installation!

We'll be using command prompt a lot, so you'll have plenty of time to become more comfortable with it. Think of command prompt as the ability for us programmers to get closer to the computer than the average person needs to.

Installing a Text Editor

Next, you'll need a text editor to write code in. Think of a text editor as google docs or Microsoft Word but for code. You use a text editor for writing code as they have handy features like highlighting your code in different colors to make it easier to read. For beginners, I recommend SublimeText, which you can install [here](#).



Yours will be a different color as I have a theme.

Running Python Code

Create a new Python file in Sublime Text by, via the top left menu, click **File** → **New File**. When naming your file, **you can pick any name so long as you end it with .py**. This is what tells your computer you are creating a Python file.

Next, you want to navigate via command prompt to the folder you saved your empty Python file in. Remember you can open command prompt by typing “cmd” in the Window’s search bar in the bottom left corner. In command prompt, you can enter a folder with the command `cd <folder name>`, which stands for change directory. Directory is another word for a folder on computers. If you make a mistake, you can type `cd ..` to head outwards. See the following example.

```
C:\Users\Nia>cd Documents
C:\Users\Nia\Documents>cd Other
C:\Users\Nia\Documents\Other>cd ..
C:\Users\Nia\Documents>
```

Notice how I used `cd ..` to exit my folder `Other`.

If you named your program `FirstProgram.py` and you navigated command prompt to the correct folder, then typing `python FirstProgram.py` in command prompt and pressing enter will run your program! In Sublime Text, write a simple program like this:

```
print("Hello World!")
```

And you'll find "Hello World!" on the screen in command prompt when you run your program! Try changing "Hello World" to other things in Sublime Text and see them be printed in command prompt. For example, you could replace "World" with your name.

2 Assignment Operator

Consider this Python code:

```
a = 5
a = a + 2
print(a)
```

What will show up in command prompt? a ? 5? Or maybe 7? Run it yourself.

The answer is indeed 7, as we set a to 5 and then added 2 to it. But hold on. If we think about what this code means from an algebra perspective, aren't we saying:

$$\begin{array}{rcl} a & = & a + 2 \\ -a & & -a \\ \hline 0 & = & 2 \quad (???) \end{array}$$

In programming, the equal sign $=$ is called the assignment operator. We're not saying that a is equal to $a + 2$. Rather, we're saying that the variable a should change its value to what it was before plus 2. a was not equal to $a + 2$ before, but it will be now. You can think of it like we're crossing out 5 in memory and replacing it with a 7. I like to say programming is like math that happens over time, so variables can change their values.

Fun Fact: To differentiate the assignment operator $=$ from the equal sign in

math, some programming languages use `:=` instead, which is called the Walrus operator because it looks like Walrus teeth.

3 Data Types

In our first lesson, we talked about a few data types. They were:

- `int` - For holding integers. Example: `a = 5`. if you wrote a program `print(type(a))`, you will have `int` pop up in command prompt when you run the program.
- `float` - For holding decimal numbers. Example: `a = 2.3`. It's called `float` partly because when you multiply a decimal number by 10, like $5.2 \cdot 10$, the decimal "moves over".
- `str` - For holding English words. Example: `b = "dinosaur"`. `str` is short for string, which means a string of characters in a row. Remember that your computer will have an error for `a = 5 + "7"` because `"7"` is the key 7 on your keyboard, not the number, so you're trying to add a number to text! It would be like `a = "banana" + 23`.
- `bool` - For `True` and `False`. Example: `a = True`. Short for boolean, and named after George Bool who invented using True and False for logic. Booleans are really useful for if-statements, which I'll review in the next section.

4 If Statements

What if you want code to only run some of the time? That's where if-statements come in handy! Consider this code:

```
a = True
if a:
    print("Success!")
else:
    print("Not Success.")
```

If you ran this program in command prompt, would “Success!” be printed or “Not Success”? The answer is “Success” because the top part of an if-statement happens only if the **condition**, which in this case is *a*, is true. Otherwise, the **else** section will run.

What about for this program?

```
a = 8
if a == 19:
    print("Success!")
else:
    print("Not Success.")
```

Because we use = for the assignment operator in Python, which is setting a variable to some value, to check if a variable is equal to something, we use == in programming. Here, because 8 is not equal to 19, “Not Success” will be printed.

5 Easier Homework

Create a folder on your computer called **If-Statements-Practice-1**. Using Sublime Text’s new file feature, create multiple python programs that do the following:

1. Print the numbers 1 through 5. There’s a quick way to do this you’ll learn in the future, but for now, just write 5 print statements, 1 for each number.
2. Sets *a* = "cat", and then print **True** if *a* is a string. Remember that `type(a)` is how you check the type of the variable *a*.
3. Prints the result of 2 + 6. Your answer should not be `print(8)` but instead should use 2 and 6.

6 More Challenging Homework

In the same folder, complete these problems:

1. Set x to any number you like. If x is a multiple of 2, print "**x is even**" and if x is odd, print "**x is odd**". So if you set $x = 21$, your program would print "**21 is odd**" (not "**x is odd**").

To check if a number is a multiple of 2, we can use its remainder after division. The remainder of $13 \div 2$ is 1 as $2 \cdot 6 = 12$, so we need 1 more to get all the way. In Python, we can get the remainder using the `%` sign operator. It has a special name, but for now, call it the *remainder* operator. Below are two examples to help you:

```
if(31 % 2 == 1):  
    print("True!")  
if (18 % 2 == 0):  
    print("Also True!")
```

2. Set x to any number you like. Using an if statement, an if-else statement (which in python is `elif`), and an else statement, print "**x is a multiple of 3**" if x is divisible by 3. Print "**x is 1 above**" if x is one above a multiple of 3, and then print "**x is 2 above**" if x is two above a multiple of 3. Is there any other cases?

Once again, you'll need the remainder operator. But instead of `x % 2 == 0`, you'll use a different number than 2. What number?

Good luck! We'll talk about whatever you finish next time.