

# Python – The Tutorial (PDF EDITION)

---

It is recommended to go [here](#) for the interactive tutorial version with an account to save your progress! (You need a repl.it account for the interactive version.)

## Lesson 0, INTRO TO IDEs

---

Welcome, welcome! This is a bonus lesson, Level 0! In this lesson, we're going to talk about IDEs, or Integrated Development Environments. An IDE is where you type your code, possibly install packages, run the code, and manage your files, all in one. Here's the list of Software we're going to talk about today -- Sublime Text - Repl.it - Notepad++ - TextEdit - Atom

Let's begin!

---

### SUBLIME TEXT

Features- Numerous themes

Package Control, install packages within the IDE

Multiple language Syntax highlighting

Run your code

Cons -

Can be finicky

The Git Support is a bit...weird

---

### REPL.IT (RECOMMENDED IDE)

Features -

Github support

Package Manager

Run your code

Free web hosting

Share projects easily

Save your code with an account

Friendly community

Free easy to use Database

50+ languages

Syntax highlighting for all of them

Live error Linting (Finds errors live)

Debugger

Multiplayer support

Version control

---

### Notepad++ and TextEdit

NO

DO NOT USE THESE FOR CODING. This is called a text editor, and are the worst things to use for coding because they don't support anything. at all.

---

### Atom -

Its made by Github, so deep Git support

looks pretty cool

Cons -

Cant run code in it.

---

Thats all, I hope this helped you decide which IDE is best for you!

---

## Lesson 1, Hello World!

---

A Hello World program is the first program all developers do in a language. Let's learn the Python Syntax.

The first thing to note is that there are no semicolons. The `print` Statement prints a line of code into the console/shell (Usually on Terminal, Command Prompt, Repl.it, etc its called the console, but on the Python IDLE its called the shell). Here's the line of code for Hello World --

```
print("Hello world!")
```

`print` is a default python function, with the parameter of either a string, integer, float, variable, function, math equation, etc. The 'Hello World' is our string. DO NOT ADD A SEMICOLON.

---

## Lesson 2, Data Types (Mini Lesson)

---

There are 3 types of python values, a `string()`, `int()`, and `float()` string is a string, int is an integer, and float is integers+decimals

---

## Lesson 3, Variables

---

I BELEIVE

that it is now time for you to learn variables.

Yes, the important variables.

Python is super simple.\nTo define variables, you need neither `let` nor `var`. Simply write the 'equation'.

Example --

```
test_var = 'wow'
```

Variables can equal integers, floats, strings, other integers, math equations, etc. A variable is case sensitive.

---

## Lesson 4, Conitionals

---

CONDITIONALS.

Conditionals are an important part of coding. Conditionals may include things like `if/else` statements. An example of an if statement -

```
if user_input == "hi":  
    print('big hi')
```

The syntax:\nthere are no `()` or `{}`. Define your if statement clearly, and use a colon + new line to define the action. If using a variable/int/float, type out the variable/int/float freely. If using a string, type out the word with `""` around the word. In the above example, `user_input` was a variable.

Elif

If you want to define another possibility, after the first if definition, type `elif` ---

Example --

```
if user_input == "hi":  
    print('big hi')  
elif user_input == "no":  
    print("yes")
```

Always remember to define using double equal signs `==`

You can have an unlimited number of if and elif statements.

Else

There can only be one else statement per conditional. Else is like a wildcard, it handles the remaining possibilities. Else usually goes at the bottom, and has no parameters. Example --

```
if user_input == "hi":
    print('big hi')
elif user_input == "no":
    print("yes")
else:
    print("Hello, I'm Tutorial Bot")
```

## Lesson 5, User Input

---

Ever wonder how PythonTutorialBot takes in your answers and reads them?

In this lesson, we're going to learn about user inputs. By the end of this lesson, you will be able to take user input, and use it for various things.

LET US BEGIN

Syntax: An input statement is much like a print statement, only this time, instead of print, type input at the beginning. Example -

```
input("What's your favorite color? ")
```

While this syntax is correct, currently, it does nothing. To make this work, assign it to a variable.

```
fav_color = input("What's your favorite color? ")
```

Now, we can check the variable `fav_color`'s value to do certain functions. (You can use user inputs with conditionals now!!) Treat it like a normal variable. Remember to always add hyphens, colons, spaces, etc. at the end of your question to differentiate the question and answer. The input will come in the console/shell, and you will answer by typing there.

---

## Level 6, Lists

---

Lists can store LOTS of things. They store practically everything, even other lists! (These are called 2D lists, you'll learn about them later.)

LET US BEGIN

Syntax: A list is enclosed with brackets, and its value is stored as a variable. Example -

```
my_list = ["item1", "item2", "item3"]
```

Here, we define a variable, with lots of things inside of it.

Now, if we just print this, we will get this -

```
[item1,item2,item3]
```

To get an individual value, we have to call it. To call it, we type the variable name, then the location in brackets. \nNOTE: In lists, the first value is 0, not 1. If you call by location 1, you'll get the second value. Here's how to call a value --

```
list_value_1 = my_list[0]
print(list_value_1)

#or

print(my_list[0])
```

## Lesson 7, Functions

---

FUNCTIONS.

Literally one of the most important things in python. They hold functions to perform, and they help with organization and a lot of other things.

SYNTAX: use `def function_name(params, leave empty if none): print("OOO COOL FUNCTION")`

Example --

```
def kool_function():
    print("wow cool function!")
```

We define our function, where we tell python what we want it to do.

But wait! this does nothing right now, your output fo this would be --

```
>>>
```

Yeah not kidding. To run the function, call it. After defining your code, type

```
kool_function()
```

your total code should look like this --

```
def kool_function():  
    print("wow cool function!")  
kool_function()
```

Now, if we just run this, we will get this -

```
>>> wow cool function!
```

You could define anything.

NOTE: Any variables defined inside of the function will only work in the function, and even if you defined it outside the function, it will create a local variable unless told not to. To use a variable from outside the function, type --

```
def kool_function():  
    global var1
```

## Lesson 8, EXTENSION OF LESSON 6, 2D LISTS

EXTENSION OF LEVEL 6, 2D LISTS.

Numerous times, a simple value storage isn't enough. Maybe you want to store sets of keys/values with other keys/values inside the same list? Perhaps a login system that stores username and password in a list called logins? (IT IS NOT RECOMMENDED TO USE THIS BECAUSE THIS IS VERY INSECURE.)

But anyhow, the syntax is pretty much the same as a normal list.

Example --

```
my_list = [ ["wow", "cool"], ["lol", "lmao"], ["yes", "no"] ]
```

We define our lists inside of a list.\nNow here's how to call it -

```
print(my_list[0][1])
```

This calls the first value of the first list inside of the main list. Remember that 0 means 1. (Haha yes very confusing) If you just do my\_list[0], it will print out the first list, so it will print

```
>>> [wow, cool]
```

## Lesson 9, Loops part 1

LOOPS.

What if you ever wanted to repeat something over and over? It would be quite un organized and difficult to hand copy lines of code...

This lesson focuses on loops. During the lesson, you will learn about two types of loops broken down into two levels. The first loops is a while loop, which i used less frequently. A while loop runs its contents until a value is true.

SYNTAX:

type while (your condition here): things to do here.

Example --

```
x = 0  
while x < 5:  
    x += 1  
    print(x)
```

This loop runs until x is less than five, so it will run 4 times. On the fifth time, it will be equal to 5, not less than, so it wont run.

## Lesson 10, Loops, part 2

---

### LOOPS PART 2.

This lesson focuses on for loops. The second loop is a for loop, which is used more frequently. A for loop runs its contents for a certain period of time.

#### SYNTAX:

type for var in range(20): things to do here.

Example --

```
for x in range(5):
    x += 1
    print(x)
```

This loop runs 5 times, each time adding 1 to x. `range` provides a numerical value, but there are more ways. If instead of `range(5)` u put..

- A string -
  - It would run for `length of string` number of times.
- A list -
  - it would run for the number of items in the list, BUT you cant add to x. In the case of a list, x would actually be the list item. So if u did -

```
for x in my_list:
    print(x)
```

This would print each item of a list on a sepaate line.

## Lesson 11, Dictionaries

---

### DICTIONARIES.

While lists are great for storing sets of data, Dictionaries are used for key-value pairs.

Lets start.

#### SYNTAX:

Define it asa variable. use curly braces {} to define sets. inside the set, use colons : to separate key and value.

Example --

```
my_dict = {"yes":"no", "hi":"bye"}
```

This dictioanry sotres two pairs, yes and no, and hi and bye. Heres how we call a value: We call a value by the key. Type `dict_name[key]`. this returns the value.

Example--

```
my_dict = {"yes":"no", "hi":"bye"}
print(my_dict['yes'])
```

```
#Output
>>> no
```

Now we're going to learn how to set values later in your code. Sometimes, you may want to add a pair later on in your code, or update the dict later on. Here's how you can do that --

```
my_dict["new_key"] = "new_value"
```

The above statement sets a new key `"new_key"` to the value `"new_value"` .

If the key already exists and you run that line above, it will overwrite the current pair to the new pair/update the key to the new value.

## Lesson 12, Common Modules

---

### COMMON MODULES AND THEIR FUNCTIONS

Sometimes, python doesn't always have the functions to do what you want. In this case, we use external packages, or modues.

Repl.it is great, all packages are automatically installed when u give it the command. However, locally, you need to use something called pip3 to install packages. Here's how --

Type

```
pip3 install <package name>
```

Into your computer's command prompt/terminal. In all IDEs, type -

```
import <package name>
```

This tells python to look for commands from that package.

Let's start with one of the popular modules, TIME.

to use it,

```
import time
```

It's most popular command is

```
time.sleep(num of secs)
```

This will freeze the Output for the specified number of seconds. Use it carefully, if you don't you will need to force quit your IDE.

Next, MATH

```
import math
```

Before we begin, i'll also tell you how to use python's inbuilt math (better to use both) to do some math functions.

```
5+5 #addition
5-5 #subtraction
5*5 #multiplication
5/5 #division
5//5 #floor division, rounds number after dividing.
```

```
#MATH FUNCTIONS WITH THE PACKAGE
```

```
math.sqrt(number) #Square root of number
math.factorial(x)
```

SPECIAL! LEARN HOW TO COLOR PYTHON OUTPUT -

```
from termcolor import colored
```

Now do

```
print(colored("string", "color"))
```

Cool, right?

RANDOM

Here's how to get a random number `python import random random_number = random.randint(first_number, _last_number)` The first number and last number is the range of numbers to look through. They are included in the search for random numbers.

that's all I have to give right now ;;. Click [here](#) and comment the package you would like to see for me to add it. That's all for this Tutorial, I hope liked it!