

DAY-1 INTRODUCTION AND SETUP

Welcome to your journey into the world of Python programming with Python Togo. Over the next 30 days, you'll explore one of the most popular and beginner-friendly programming languages used across industries like web development, data science, automation, and artificial intelligence. Python is a high-level, interpreted, and general-purpose programming language. Known for its simple and readable syntax, Python is widely used by both beginners and professionals. It was created by Guido van Rossum and first released in 1991 with the goal of making code more understandable and accessible. Since then, it has evolved into one of the most powerful tools in modern software development. This course is structured to help you learn Python step by step — from the very basics to real-world applications. Each day introduces a focused topic along with clear explanations, practical examples, and hands-on exercises to help you apply what you learn immediately. Whether you're starting from scratch or coming back to coding after a break, this course is designed to build your confidence and problem-solving skills with Python. You'll understand how to write clean code, think logically, and begin creating useful programs on your own. No prior experience is required — just your curiosity and commitment to learning something new every day. By the end of this challenge, you'll not only understand how Python works but also have the foundation to explore more advanced areas like automation, scripting, or data analysis.

Let's begin.

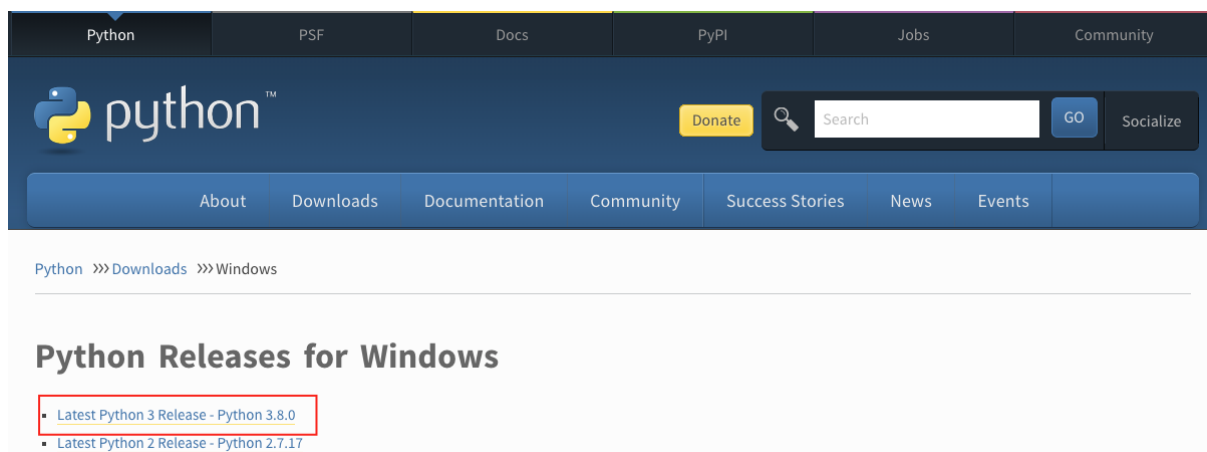
Why Python

It is a programming language which is very close to human language and because of that, it is easy to learn and use. Python is used by various industries and companies (including Google). It has been used to develop web applications, desktop applications, system administration, and machine learning libraries. Python is a highly embraced language in the data science and machine learning community. I hope this is enough to convince you to start learning Python. Python is eating the world and you are killing it before it eats you.

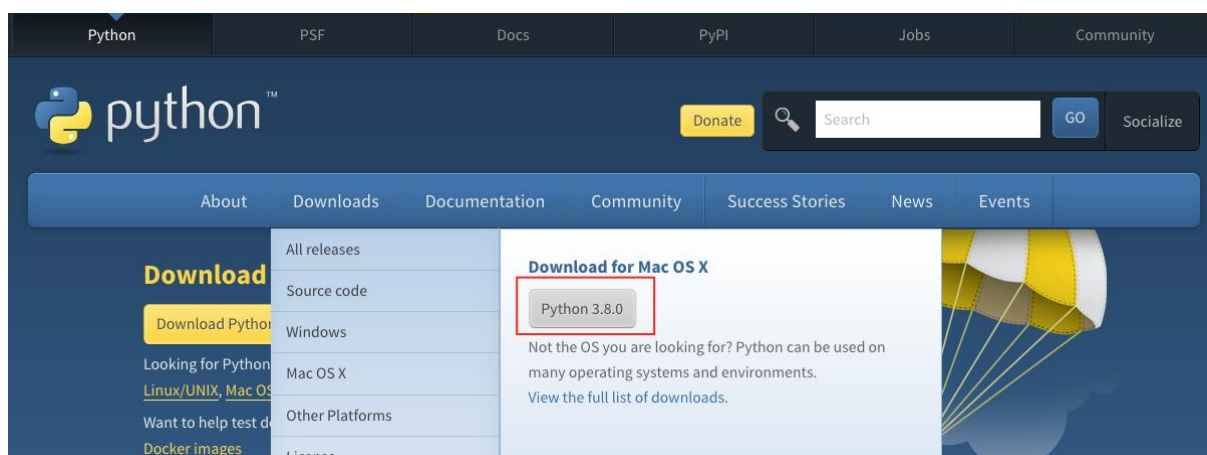
Environment Setup

Installing Python

To run a python script you need to install python. Let's [download](#) python. If you are a windows user. Click the button encircled in red.

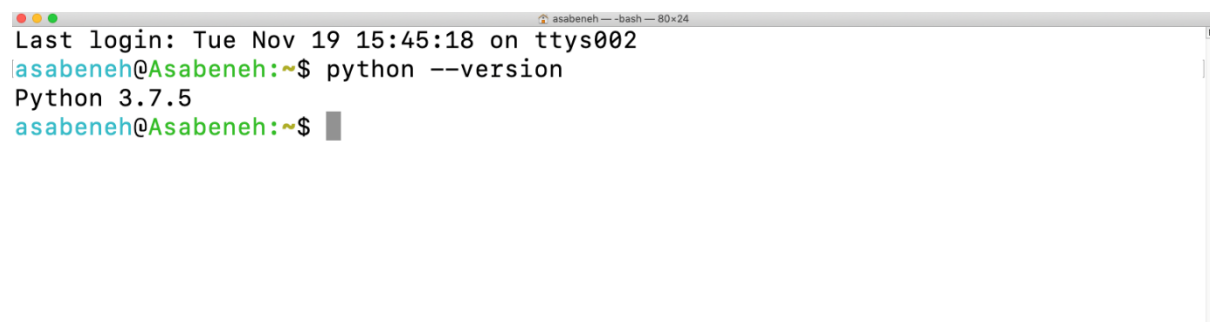


If you are a macOS user. Click the button encircled in red.



To check if python is installed write the following command on your device terminal.

```
python --version
```



```
asabeneh ~ -bash — 80x24
Last login: Tue Nov 19 15:45:18 on ttys002
asabeneh@Asabeneh:~$ python --version
Python 3.7.5
asabeneh@Asabeneh:~$
```

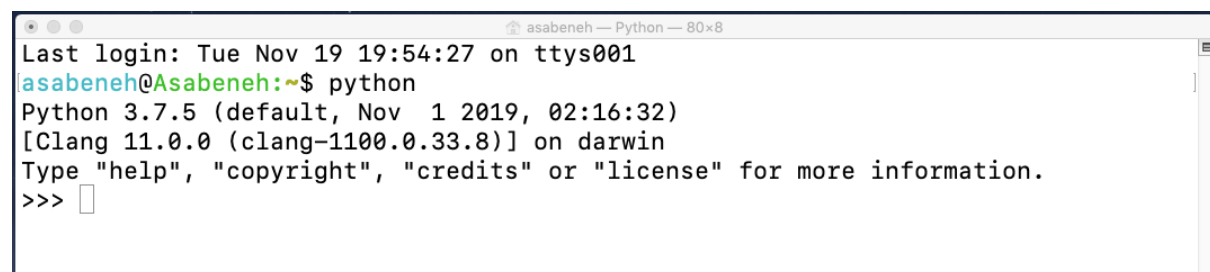
As you can see from the terminal, I am using *Python 3.7.5* version at the moment. Your version of Python might be different from mine but it should be 3.6 or above. If you manage to see the python version, well done. Python has been installed on your machine. Continue to the next section.

Python Shell

Python is an interpreted scripting language, so it does not need to be compiled. It means it executes the code line by line. Python comes with a *Python Shell (Python Interactive Shell)*. It is used to execute a single python command and get the result.

Python Shell waits for the Python code from the user. When you enter the code, it interprets the code and shows the result in the next line. Open your terminal or command prompt(cmd) and write:

```
python
```



```
asabeneh ~ Python — 80x8
Last login: Tue Nov 19 19:54:27 on ttys001
asabeneh@Asabeneh:~$ python
Python 3.7.5 (default, Nov 1 2019, 02:16:32)
[Clang 11.0.0 (clang-1100.0.33.8)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 
```

The Python interactive shell is opened and it is waiting for you to write Python code(Python script). You will write your Python script next to this symbol `>>>` and then click Enter. Let us write our very first script on the Python scripting shell.

```
asabeneh — Python — 80x10
Last login: Tue Nov 19 20:05:55 on ttys001
asabeneh@Asabeneh:~$ python
Python 3.7.5 (default, Nov 1 2019, 02:16:32)
[Clang 11.0.0 (clang-1100.0.33.8)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 2 + 3
5
>>> █
```

Well done, you wrote your first Python script on Python interactive shell. How do we close the Python interactive shell? To close the shell, next to this symbol `>>>` write **`exit()`** command and press Enter.

```
asabeneh — Python — 80x10
Last login: Tue Nov 19 20:05:55 on ttys001
asabeneh@Asabeneh:~$ python
Python 3.7.5 (default, Nov 1 2019, 02:16:32)
[Clang 11.0.0 (clang-1100.0.33.8)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 2 + 3
5
>>> exit() █
```

Now, you know how to open the Python interactive shell and how to exit from it.

Python will give you results if you write scripts that Python understands, if not it returns errors. Let's make a deliberate mistake and see what Python will return.

```
asabeneh — Python — 80x12
asabeneh@Asabeneh:~$ python
Python 3.7.5 (default, Nov 1 2019, 02:16:32)
[Clang 11.0.0 (clang-1100.0.33.8)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 2 + 3
5
>>> 2 x 3
File "<stdin>", line 1
  2 x 3
    ^
SyntaxError: invalid syntax
>>> █
```

As you can see from the returned error, Python is so clever that it knows the mistake we made and which was *Syntax Error: invalid syntax*. Using `x` as multiplication in Python is a syntax error because `(x)` is not a valid syntax in Python. Instead of `(x)` we use asterisk `(*)` for multiplication. The returned error clearly shows what to fix.

The process of identifying and removing errors from a program is called *debugging*. Let us debug it by putting `*` in place of `x`.

```
asabeneh — Python — 80x15
Last login: Tue Nov 19 20:05:55 on ttys001
asabeneh@Asabeneh:~$ python
Python 3.7.5 (default, Nov 1 2019, 02:16:32)
[Clang 11.0.0 (clang-1100.0.33.8)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> 2 + 3
5
>>> 2 x 3
      File "<stdin>", line 1
        2 x 3
          ^
SyntaxError: invalid syntax
>>> 2 * 3
6
>>> █
```

Our bug was fixed, the code ran and we got a result we were expecting. As a programmer you will see such kind of errors on daily basis. It is good to know how to debug. To be good at debugging you should understand what kind of errors you are facing. Some of the Python errors you may encounter

are *SyntaxError*, *IndexError*, *NameError*, *ModuleNotFoundError*, *KeyError*, *ImportError*, *AttributeError*, *TypeError*, *ValueError*, *ZeroDivisionError* etc. We will see more about different Python **error types** in later sections.

Let us practice more how to use Python interactive shell. Go to your terminal or command prompt and write the word **python**.

```
asabeneh — Python — 80x8
Last login: Tue Nov 19 19:54:27 on ttys001
asabeneh@Asabeneh:~$ python
Python 3.7.5 (default, Nov 1 2019, 02:16:32)
[Clang 11.0.0 (clang-1100.0.33.8)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> █
```

The Python interactive shell is opened. Let us do some basic mathematical operations (addition, subtraction, multiplication, division, modulus, exponential).

Let us do some maths first before we write any Python code:

- $2 + 3$ is 5
- $3 - 2$ is 1
- $3 * 2$ is 6
- $3 / 2$ is 1.5
- $3 ** 2$ is the same as $3 * 3$

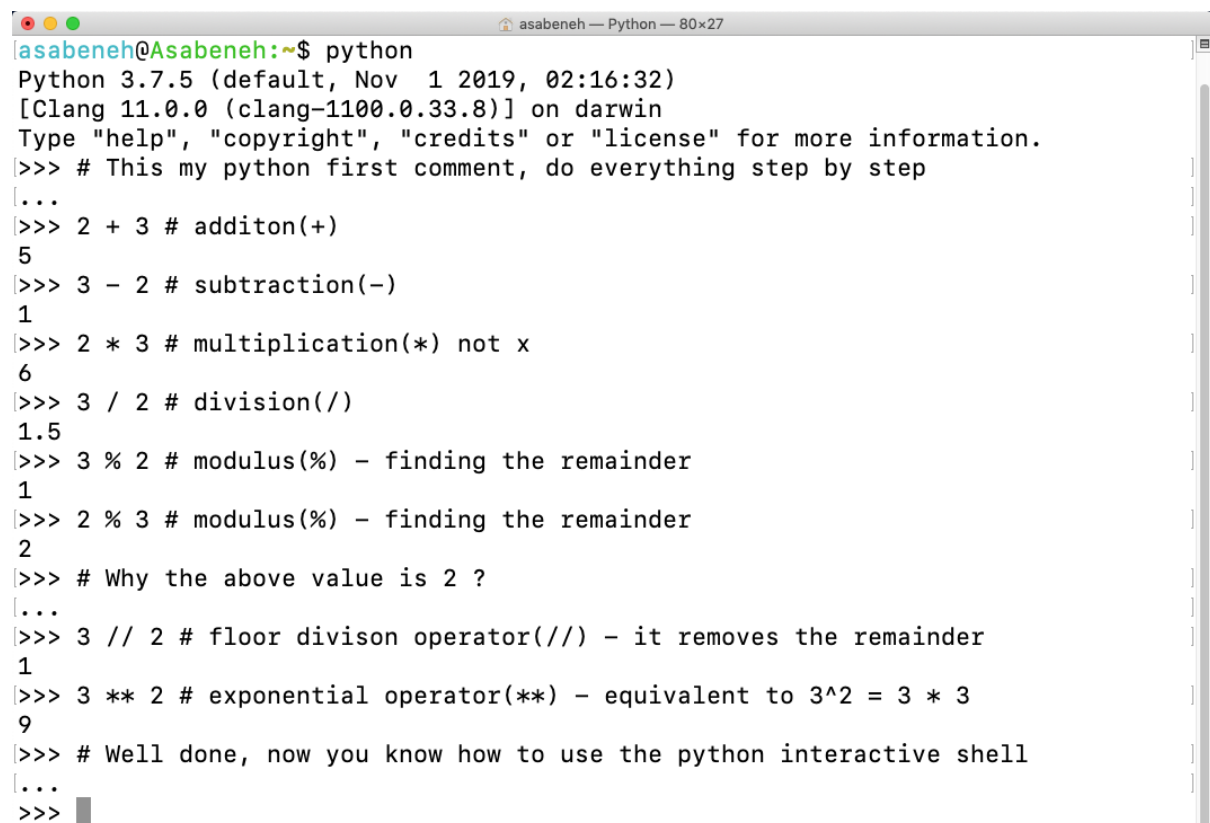
In python we have the following additional operations:

- $3 \% 2 = 1 \Rightarrow$ which means finding the remainder
- $3 // 2 = 1 \Rightarrow$ which means removing the remainder

Let us change the above mathematical expressions to Python code. The Python shell has been opened and let us write a comment at the very beginning of the shell.

A *comment* is a part of the code which is not executed by python. So we can leave some text in our code to make our code more readable. Python does not run the comment part. A comment in python starts with hash(#) symbol. This is how you write a comment in python

```
# comment starts with hash
# this is a python comment, because it starts with a (#)
symbol
```



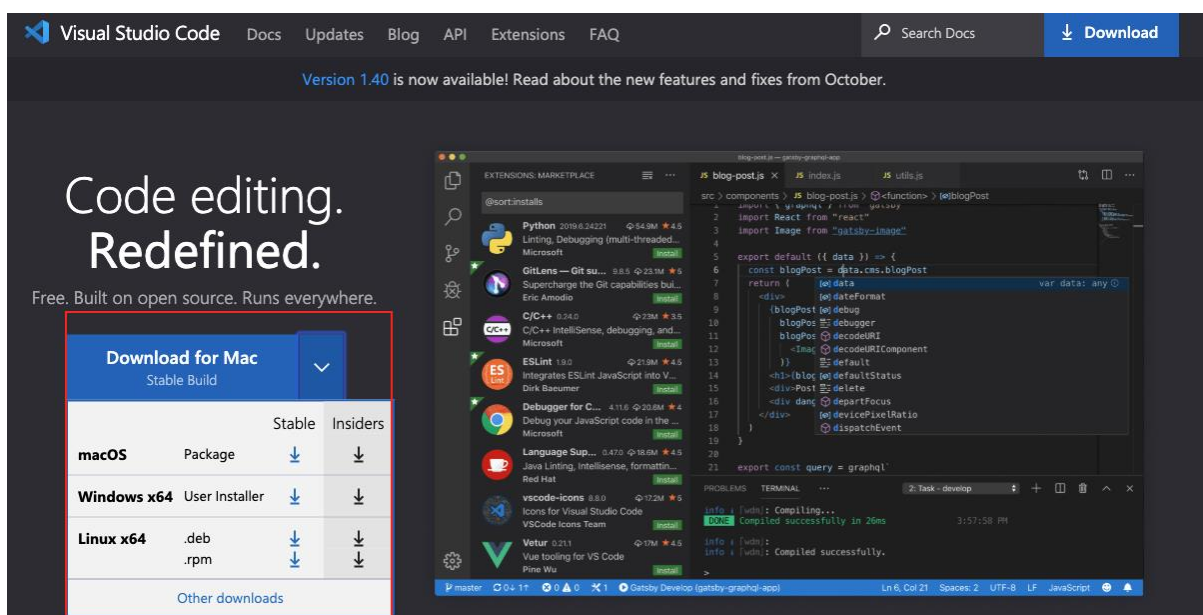
```
asabeneh@Asabeneh:~$ python
Python 3.7.5 (default, Nov 1 2019, 02:16:32)
[Clang 11.0.0 (clang-1100.0.33.8)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> # This my python first comment, do everything step by step
...
>>> 2 + 3 # additon(+)
5
>>> 3 - 2 # subtraction(-)
1
>>> 2 * 3 # multiplication(*) not x
6
>>> 3 / 2 # division(/)
1.5
>>> 3 % 2 # modulus(%) - finding the remainder
1
>>> 2 % 3 # modulus(%) - finding the remainder
2
>>> # Why the above value is 2 ?
...
>>> 3 // 2 # floor divison operator(//) - it removes the remainder
1
>>> 3 ** 2 # exponential operator(**) - equivalent to 3^2 = 3 * 3
9
>>> # Well done, now you know how to use the python interactive shell
...
>>> █
```

Before we move on to the next section, let us practice more on the Python interactive shell. Close the opened shell by writing *exit()* on the shell and open it again and let us practice how to write text on the Python shell.

```
asabeneh@Asabeneh:~$ python
Python 3.7.5 (default, Nov 1 2019, 02:16:32)
[Clang 11.0.0 (clang-1100.0.33.8)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> # Let's practice how to write text on python shell: we use single or double quote to make a string. Any text in a quote we call it string
...
>>> 'Asabeneh' # To write my name as string
'Asabeneh'
>>> "Asabeneh" # To write my name as string - now in double quote
'Asabeneh'
>>> # A string could be a single character text or a page
...
>>> 'I love teaching. And thank you so much for joining this course'
'I love teaching. And thank you so much for joining this course'
>>> # If the text is too long we use triple quote to make a string('''')
...
>>> '''We use triple quote if the text is more than one line'''
'We use triple quote if the text is more than one line'
>>> # Now you know string and how to use the python shell
...
>>> # Let's continue to the next section and install text editor code editor
...
>>> exit()
asabeneh@Asabeneh:~$
```

Installing Visual Studio Code

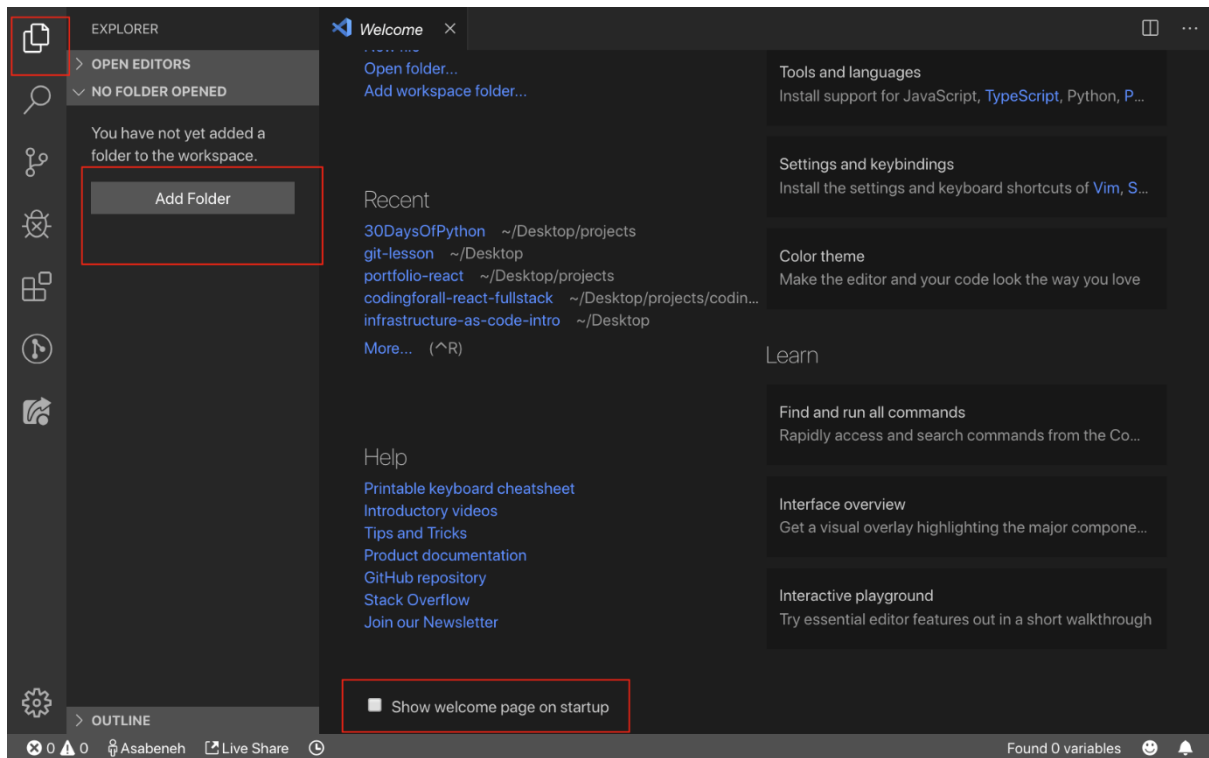
The Python interactive shell is good to try and test small script codes but it will not be for a big project. In real work environment, developers use different code editors to write codes. In this 30 days of Python programming challenge we will use Visual Studio Code. Visual Studio Code is a very popular open source text editor. I am a fan of VS Code and I would recommend to [download](#) Visual Studio Code, but if you are in favor of other editors, feel free to follow with what you have.



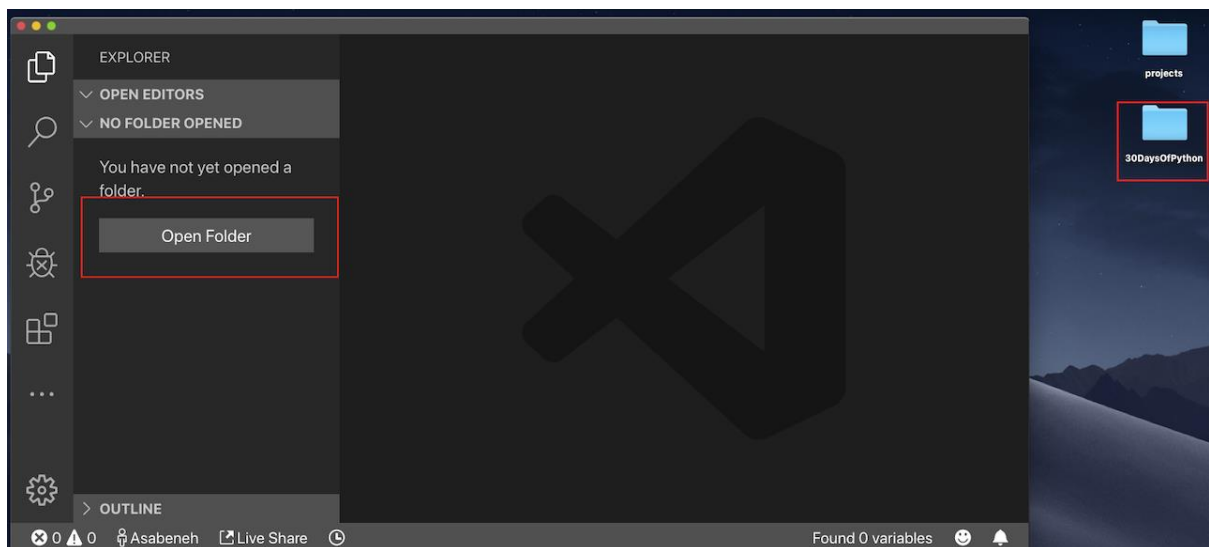
If you installed visual studio code, let us see how to use it. If you prefer a video, you can follow this Visual Studio Code for Python [Video tutorial](#)

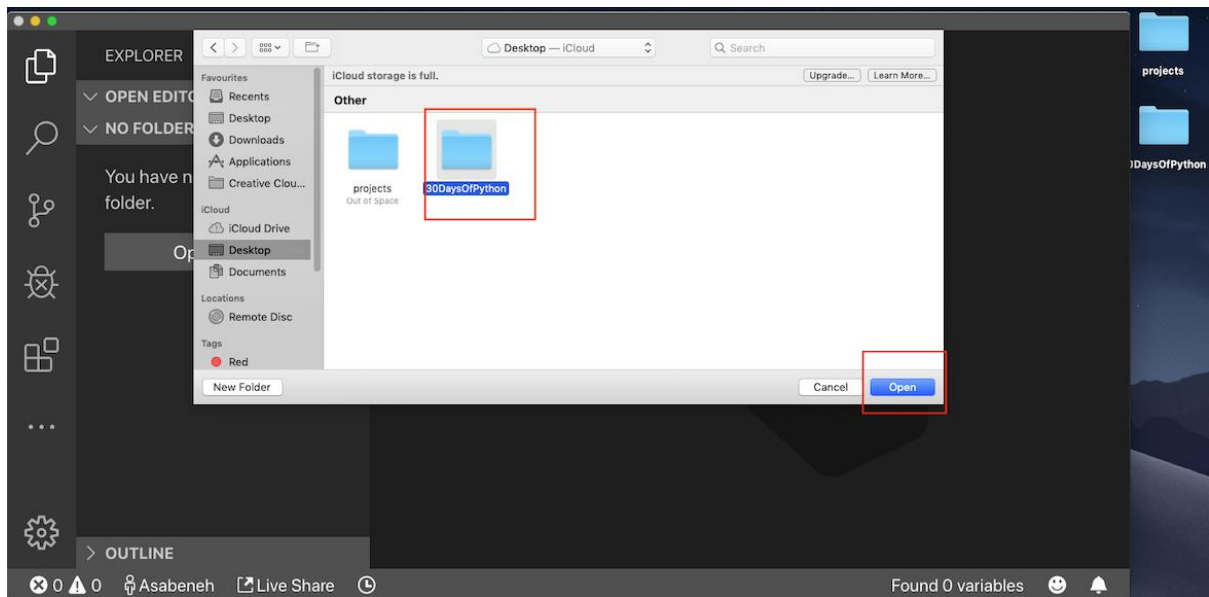
How to use visual studio code

Open the visual studio code by double clicking the visual studio icon. When you open it you will get this kind of interface. Try to interact with the labeled icons.

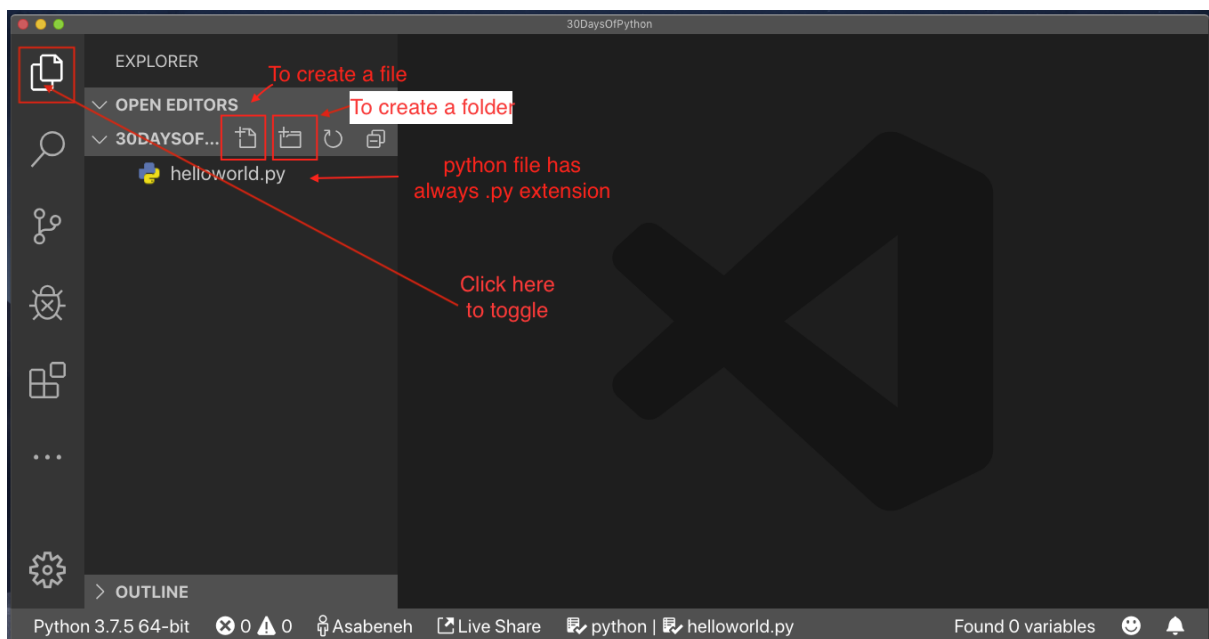


Create a folder named 30DaysOfPython on your desktop. Then open it using visual studio code.

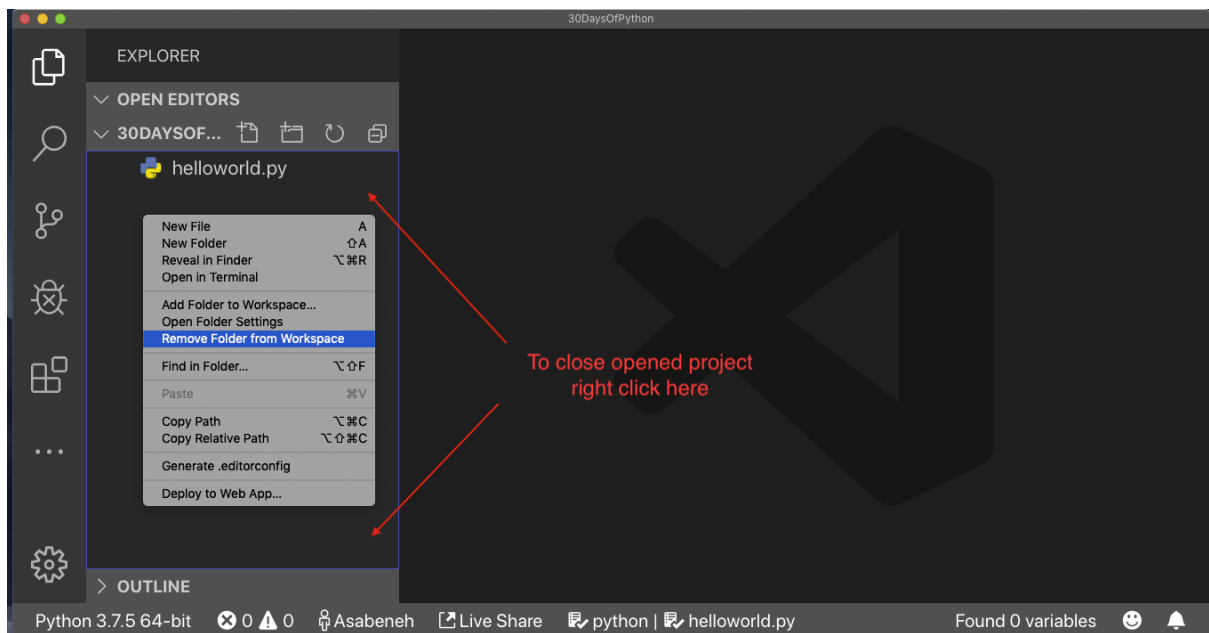




After opening it you will see shortcuts for creating files and folders inside of 30DaysOfPython project's directory. As you can see below, I have created the very first file, helloworld.py. You can do the same.



After a long day of coding, you want to close your code editor, right? This is how you will close the opened project.



Congratulations, you have finished setting up the development environment. Let us start coding.

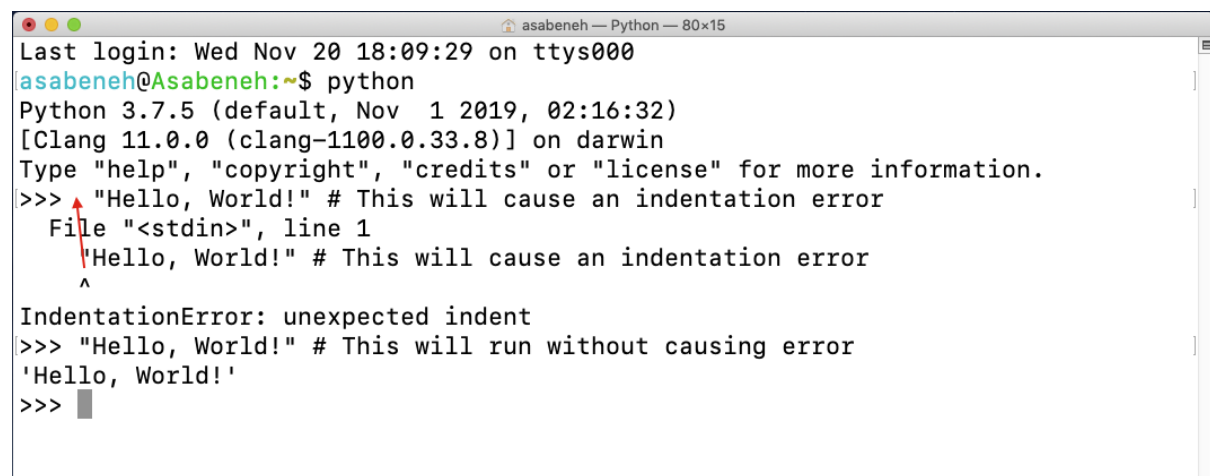
Basic Python

Python Syntax

A Python script can be written in Python interactive shell or in the code editor. A Python file has an extension .py.

Python Indentation

An indentation is a white space in a text. Indentation in many languages is used to increase code readability; however, Python uses indentation to create blocks of code. In other programming languages, curly brackets are used to create code blocks instead of indentation. One of the common bugs when writing Python code is incorrect indentation.

A screenshot of a terminal window titled 'asabeneh — Python — 80x15'. The terminal shows the following text:

```
Last login: Wed Nov 20 18:09:29 on ttys000
asabeneh@Asabeneh:~$ python
Python 3.7.5 (default, Nov 1 2019, 02:16:32)
[Clang 11.0.0 (clang-1100.0.33.8)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> "Hello, World!" # This will cause an indentation error
      File "<stdin>", line 1
        "Hello, World!" # This will cause an indentation error
        ^
IndentationError: unexpected indent
>>> "Hello, World!" # This will run without causing error
'Hello, World!'
>>> █
```

Comments

Comments play a crucial role in enhancing code readability and allowing developers to leave notes within their code. In Python, any text preceded by a hash (#) symbol is considered a comment and is not executed when the code runs.

Example: Single Line Comment

```
# This is the first comment
# This is the second comment
# Python is eating the world
```

Example: Multiline Comment

Triple quote can be used for multiline comment if it is not assigned to a variable

```
"""This is multiline comment
multiline comment takes multiple lines.
python is eating the world
"""
```

Data types

In Python there are several types of data types. Let us get started with the most common ones. Different data types will be covered in detail in other sections. For the time being, let us just go through the different data types and get familiar with them. You do not have to have a clear understanding now.

Number

- Integer: Integer(negative, zero and positive) numbers Example: ... -3, -2, -1, 0, 1, 2, 3 ...
- Float: Decimal number Example ... -3.5, -2.25, -1.0, 0.0, 1.1, 2.2, 3.5 ...
- Complex Example $1 + j$, $2 + 4j$

String

A collection of one or more characters under a single or double quote. If a string is more than one sentence then we use a triple quote.

Example:

```
'Asabeneh'
'Finland'
'Python'
'I love teaching'
'I hope you are enjoying the first day of 30DaysOfPython
Challenge'
```

Booleans

A boolean data type is either a True or False value. T and F should be always uppercase.

Example:

```
True # Is the light on? If it is on, then the value is
True
False # Is the light on? If it is off, then the value is
False
```

List

Python list is an ordered collection which allows to store different data type items. A list is similar to an array in JavaScript.

Example:

```
[0, 1, 2, 3, 4, 5] # all are the same data types - a list of
numbers
['Banana', 'Orange', 'Mango', 'Avocado'] # all the same data
types - a list of strings (fruits)
['Finland', 'Estonia', 'Sweden', 'Norway'] # all the same data
types - a list of strings (countries)
```

```
['Banana', 10, False, 9.81] # different data types in the list  
- string, integer, boolean and float
```

Dictionary

A Python dictionary object is an unordered collection of data in a key value pair format.

Example:

```
{  
'first_name':'Asabeneh',  
'last_name':'Yetayeh',  
'country':'Finland',  
'age':250,  
'is_married':True,  
'skills':['JS', 'React', 'Node', 'Python']  
}
```

Tuple

A tuple is an ordered collection of different data types like list but tuples can not be modified once they are created. They are immutable.

Example:

```
('Asabeneh', 'Pawel', 'Brook', 'Abraham', 'Lidiya') # Names  
( 'Earth', 'Jupiter', 'Neptune', 'Mars', 'Venus', 'Saturn',  
'Uranus', 'Mercury') # planets
```

Set

A set is a collection of data types similar to list and tuple. Unlike list and tuple, set is not an ordered collection of items. Like in Mathematics, set in Python stores only unique items.

In later sections, we will go in detail about each and every Python data type.

Example:

```
{2, 4, 3, 5}  
{3.14, 9.81, 2.7} # order is not important in set
```

Checking Data types

To check the data type of certain data/variable we use the **type** function. In the following terminal you will see different python data types:

```
asabeneh — Python — 80x24
Last login: Wed Nov 20 00:49:23 on ttys001
asabeneh@Asabeneh:~$ python
Python 3.7.5 (default, Nov 1 2019, 02:16:32)
[Clang 11.0.0 (clang-1100.0.33.8)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> type(10)
<class 'int'>
>>> type(3.14)
<class 'float'>
>>> type(1 + 3j)
<class 'complex'>
>>> type('Asabeneh')
<class 'str'>
>>> type([1, 2, 3])
<class 'list'>
>>> type({'name': 'Asab'})
<class 'dict'>
>>> type((2, 3, 4, 5))
<class 'tuple'>
>>> type({9.8, 3.14, 2.7})
<class 'set'>
>>> exit()
```

Python File

First open your project folder, 30DaysOfPython. If you don't have this folder, create a folder name called 30DaysOfPython. Inside this folder, create a file called helloworld.py. Now, let's do what we did on python interactive shell using visual studio code.

The Python interactive shell was printing without using **print** but on visual studio code to see our result we should use a built in function `_print()`. The `print()` built-in function takes one or more arguments as follows `print('arument1', 'argument2', 'argument3')`. See the examples below.

Example:

The file name is helloworld.py

```
# Day 1 - 30DaysOfPython Challenge

print(2 + 3)           # addition(+)
print(3 - 1)           # subtraction(-)
print(2 * 3)           # multiplication(*)
print(3 / 2)           # division(/)
print(3 ** 2)          # exponential(**)
print(3 % 2)           # modulus(%)
print(3 // 2)          # Floor division operator(//)

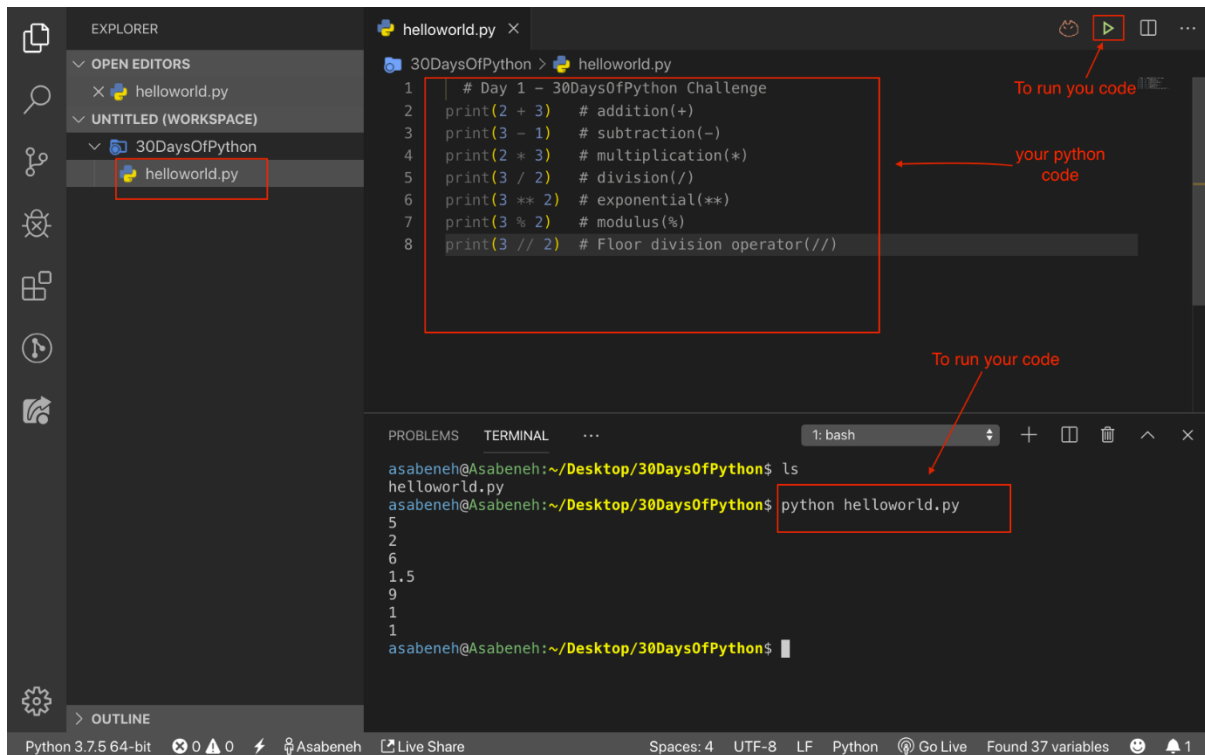
# Checking data types
```

```

print(type(10))           # Int
print(type(3.14))         # Float
print(type(1 + 3j))       # Complex number
print(type('Asabeneh'))  # String
print(type([1, 2, 3]))    # List
print(type({'name': 'Asabeneh'})) # Dictionary
print(type({9.8, 3.14, 2.7})) # Set
print(type((9.8, 3.14, 2.7))) # Tuple

```

To run the python file check the image below. You can run the python file either by running the green button on Visual Studio Code or by typing `python helloworld.py` in the terminal .



🧠 You are amazing. You have just completed day 1 challenge and you are on your way to greatness. Now do some exercises for your brain and muscles.

Exercises - Day 1

Exercise: Level 1

1. Check the python version you are using
2. Open the python interactive shell and do the following operations. The operands are 3 and 4.
 - addition(+)
 - subtraction(-)
 - multiplication(*)
 - modulus(%)
 - division(/)
 - exponential(**)
 - floor division operator(//)
3. Write strings on the python interactive shell. The strings are the following:
 - Your name
 - Your family name
 - Your country
 - I am enjoying 30 days of python
4. Check the data types of the following data:
 - 10
 - 9.8
 - 3.14
 - 4 - 4j
 - ['Asabeneh', 'Python', 'Finland']
 - Your name
 - Your family name
 - Your country

Exercise: Level 2

1. Create a folder named day_1 inside 30DaysOfPython folder. Inside day_1 folder, create a python file helloworld.py and repeat questions 1, 2, 3 and 4. Remember to use *print()* when you are working on a python file. Navigate to the directory where you have saved your file, and run it.

Exercise: Level 3

1. Write an example for different Python data types such as Number(Integer, Float, Complex), String, Boolean, List, Tuple, Set and Dictionary.
2. Find an [Euclidian distance](#) between (2, 3) and (10, 8)

 CONGRATULATIONS ! 