

How To Use Variables In Python3

Your first Python program

If you are a new to programming then today you are going to write very first program. It is traditional and almost every programmer has done it regardless of programming language.

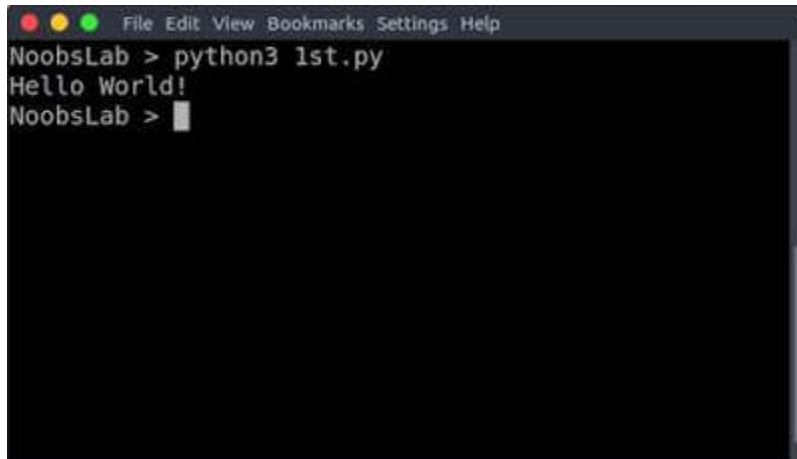
In Python it is very simple to do "Hello World!" program. You just need to write one line. There are numerous ways to run Python program, we keep it simple for now. In Linux, just type the version of Python installed on your system for example: *python3.5*.

If you don't want to install Python or don't know how to install it then you can use online Python console called "[Repl.it](https://repl.it)".

We are going to do it other way by creating a file and write our code in it. Then we go to Terminal and run file with Python3. You can see this way in the following screenshot. Once you are ready with Python console then type the following code in it and hit enter to run it.

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

You will see the output like this:

A screenshot of a terminal window with a dark background. The window has a title bar with three colored buttons (red, yellow, green) and a menu bar with the options 'File', 'Edit', 'View', 'Bookmarks', 'Settings', and 'Help'. The terminal shows the following text: 'NoobsLab > python3 1st.py', 'Hello World!', and 'NoobsLab > ' with a cursor. The text is in a light gray font.

```
NoobsLab > python3 1st.py
Hello World!
NoobsLab > 
```

Congratulations, on writing your first program in Python.

Lets see what we are doing here:

print() => it prints message to the screen, or other output device. The message can be anything string or object.

"Hello World!" => It is a message between double quotes, it is called string. We can surround string with double " " quotes or single ' ' quotes, both quotation marks functions same for example 'Hello World!'.

Variables in Python

In programming variables used to store data or we can call them containers where we put something in them in-order to access stored data later. Every variable hold certain value(s) and variables are mutable that means we can change variable's value at any time in our program.

Variables can store all sort of data like: numbers, strings, booleans and objects.

There are some rules when defining a variable in Python

programming language:

- Variable names should be descriptive, for example: *my_message*. Avoid using very short variables such as *my_m*, you will scratch your head later understanding your program.
- Be careful using lowercase letter 'l' and uppercase letter 'O' because they can be mixed with 1 and 0.
- Don't use Python reserved keywords and function names.
- Variables can only contain letters, numbers and underscores.
- Variable can't start with numbers.
- Spaces are not permitted in variables.

Lets see how to create variables:

We will modify our first 'Hello World' program to show you, how variables work!

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

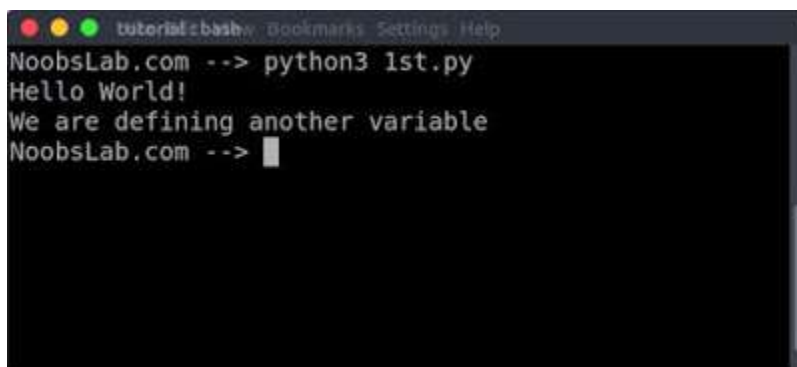
First line of the program, the string '*Hello World!*' is stored in variable called '*message*'. We can have give any name to the variable.

Then we are using print function to output our message on the screen. Also, we can defined variable '*message*' as

many times as we want in our program.

We can reassign a new value to '*message*' in the same program and there won't be any issue. Lets see an example:

```
message = "Hello World!"  
  
print(message)  
  
message = 'We are defining another variable'  
  
print(message)
```



Another example with numbers:

```
number = 100  
print(number)
```

Assign variable to another variable

You can also assign a variable which has a value to new variable. See following example for more clarification:

```
number = 100  
print(number)
```

```
new_number_variable = number
print(new_number_variable)
```

In this example, we assigned value of variable '*number*' to new variable called '*new_number_variable*'. Now we can use '*new_number_variable*' to print the same value.

Compact assignments

Python allows us to make multiple assignment in just one line, it is useful when you want to make your program compact but make less readable/complex:

```
varX, varY, varZ = 'Hello', 50, '1 Penny'
print(varX)
print(varY)
print(varZ)
```

Multiple assignments

We can assign same value to multiple variables. In the following example, all variables hold the same value, you can print each variable to check its value:

```
my_number = number = last_number = first_number =
200
print(my_number)
print(number)
print(first_number)
print(last_number)
```

Strings

String is a data type in programming language, strings in Python are very simple at initial glance but we can use them in many ways.

A string is a sequence of letters, numbers or symbols enclosed in single or double quotation marks. String are immutable data type that means we can't change it once defined. Strings does not require to be defined in advance.

String defined using double quotation marks:

```
greeting = "Hello There!"
```

String defined using single quotes:

```
hint = 'Using single quotes'
```

Using apostrophe in a string. Well, there are two ways of doing it:

```
saying = "If you can't help yourself then nobody will."
```

The other way of using apostrophe with escape character, if you still want to use single quotes:

```
saying = 'If you can\'t help yourself then nobody will.'
```

Combining and concatenating strings using addition operator, in simple words you can join multiple strings:

```
example_concatenation_string = "First part" + '  
and ' + "last part"  
  
print("Lap" + 'top')  
print("Another: " + example_concatenation_string)
```

Multiplying a string, you will find it useful later:

```
example_multiply = "Hey There!"*5
```

Know the length of the string using **len** function:

```
print(len("Hey There!"))  
length_of_string = "This is a string."  
print(len(length_of_string))
```

Strings can be written on multiple lines:

```
quote = """A person who never made a  
mistake never tried  
anything new."""  
print(quote)
```

Find out if the sub-string is present in a string:

```
quote = "A person who never made a mistake never  
tried anything new."  
  
print("mistake" in quote)  
  
print("learning" in "You are learning Python 3.")
```

Special characters in strings:

We can do formatting in strings using special characters, for instance we need line break, tabs or other formatting. These formatting can be done in a string using special character called escape.

Line break in a string:

```
quote = "Albert Einstien said:\nA person who never  
made a mistake never tried anything new."  
  
print(quote)
```

Tab in a string:

```
print("Languages:\n\tC++\n\tPython\n\tJava\n\tC")
```

Printing a backslash character:


```
print("Languages:\\C++\\Python\\Java\\C")
```

Making string omit the recognition of escape character:

```
print(r"Omitting these escape characters \t and \n.")
```

String formatting

Convert numerical values to string using **str** function:

```
print(str(1))  
print(str(33.33))  
print(str(987+11j))
```

Use numbers in a string:

```
print("He is " + str(23) + " years old.")
```

Strings case formatting:

The string case formatting is very useful, for instance you ask user to enter the username and it has to be unique for each user. In such case, you can collect the input from user and store it in lowercase then compare it with all the usernames in your list and if it is already taken then you can notify the user, if not you allow user to create the username.

The first one is title case formatting, it is useful for

scenarios like: person or place name and so on.

```
name = 'albert einstein'  
print(name.title())
```

To convert string to lower case:

```
name = 'ALBERT EINSTEIN'  
print(name.lower())
```

Use this method to change case to upper:

```
name = 'albert einstein'  
print(name.upper())
```

We will see string formatting in more detail in upcoming articles.

Numbers

Numbers are important data type and used in every program, for example: score in games, represent data, store information in web applications and so on. Numbers without decimal considered as integers in Python. Python also supports the order of operations, let us use multiple operations in one expression.

Use of add (+), subtract (-), multiply (*) and divide (/) sign

in the Python:

```
print(10+2)
print(12-2)
print(6*2)
print(24/2)
```

Exponents use two multiplication operators in Python:

```
print(2**2)
print(3**9)
```

Order of operation in a single expression:

```
print(5-1 * 2)
print((5-1) * 2)
```

Floats, you can use floats without worrying about how they will behave:

```
print(0.1 + 0.8)
print(1.1 * 2.99)
```

Example of a square root in Python:

```
print(100 ** 0.5)
```