

Variables / Data Types

Variables

```
x = str(3)    # x will be '3'
y = int(3)    # y will be 3
z = float(3)  # z will be 3.0
```

- A variable name must start with a letter or the underscore character
- A variable name cannot start with a number
- A variable name can only contain alpha-numeric characters and underscores (A-z, 0-9, and _)
- Variable names are case-sensitive (age, Age and AGE are three different variables)

Assign Multiple Values:

Python allows you to assign values to multiple variables in one line:

```
x, y, z = "Orange", "Banana", "Cherry"
```

And you can assign the *same* value to multiple variables in one line:

```
x = y = z = "Orange"
```

The Python `print()` function is often used to output variables.

```
x = "Python"
y = "is"
z = "awesome"
print(x)
print(x, y, z)
print(x + y + z)

x = 5
y = 6
print(x + y)
```

Output Variables:

The best way to output multiple variables in the `print()` function is to separate them with commas, which even support **different data types**:

```
x = 5
y = "John"
print(x, y)
```

Global Variables:

Variables that are created outside of a function (as in all of the examples above) are known as global variables.

Global variables can be used by everyone, both inside of functions and outside.

```
x = "awesome" # This is a global variable!
def myfunc():
```

```
print("Python is " + x)
myfunc()
```

If you create a variable with the same name inside a function, this variable will be local, and can only be used inside the function. The global variable with the same name will remain as it was, global and with the original value!

Normally, when you create a variable inside a function, that variable is local, and can only be used inside that function.

To create a global variable inside a function, you can use the `global` keyword.

```
def myfunc():
    global x
    x = "fantastic"
myfunc()
print("Python is " + x)
```

Also, use the `global` keyword if you want to change a global variable inside a function:

```
x = "awesome"
def myfunc():
    global x
    x = "fantastic"
myfunc()
print("Python is " + x)
```

Data Types

In programming, data type is an important concept. Variables can store data of different types, and different types can do different things.

Python has the following data types built-in by default, in these categories:

Text Type: `str`

Numeric Types: `int`, `float`, `complex`

Sequence Types: `list`, `tuple`, `range`

Mapping Type: `dict`

Set Types: `set`, `frozenset`

Boolean Type: `bool`

Binary Types: `bytes`, `bytearray`, `memoryview`

None Type: `NoneType`

You can get the data type of any object by using the `type()` function:

```
x = 5
print(type(x))
```

In Python, the data type is set when you assign a value to a variable:

Example	Data Type
<code>x = "Hello World"</code>	<code>str</code>
<code>x = 20</code>	<code>int</code>
<code>x = 20.5</code>	<code>float</code>
<code>x = 1j</code>	<code>complex</code>
<code>x = ["apple", "banana", "cherry"]</code>	<code>list</code>
<code>x = ("apple", "banana", "cherry")</code>	<code>tuple</code>
<code>x = range(6)</code>	<code>range</code>
<code>x = {"name" : "John", "age" : 36}</code>	<code>dict</code>

If you want to specify the data type, you can use the following constructor functions:

Example	Data Type
<code>x = str("Hello World")</code>	str
<code>x = int(20)</code>	int
<code>x = float(20.5)</code>	float
<code>x = complex(1j)</code>	complex
<code>x = list(("apple", "banana", "cherry"))</code>	list
<code>x = tuple(("apple", "banana", "cherry"))</code>	tuple
<code>x = range(6)</code>	range
<code>x = dict(name="John", age=36)</code>	dict

To verify the type of any object in Python, use the `type()` function! (`print (type(x))`)

Format Strings:

Метод `format()` принимает переданные аргументы, форматирует их и помещает в строку там, где находятся заполнители `{}`:

```
age = 36
txt = "My name is John, and I am {}"
print(txt.format(age))
```

The `format()` method takes unlimited number of arguments, and are placed into the respective placeholders:

```
quantity = 3
itemno = 567
price = 49.95
myorder = "I want {} pieces of item {} for {} dollars."
print(myorder.format(quantity, itemno, price))

# or use index numbers {0} to be sure the arguments are placed in the correct placeholders:

quantity = 3
itemno = 567
price = 49.95
myorder = "I want to pay {2} dollars for {0} pieces of item {1}."
print(myorder.format(quantity, itemno, price))
```

Escape Characters:

Code	Result
\'	Single Quote
\\	Backslash
\n	New Line
\r	Carriage Return
\t	Tab
\b	Backspace
\f	Form Feed
\ooo	Octal value
\xhh	Hex value