Variables

Variables

Variables are containers for storing data values.

Creating Variables

Python has no command for declaring a variable.

A variable is created the moment you first assign a value to it.

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

```
    5
    John
```

Variables do not need to be declared with any particular type, and can even change type after they have been set.

```
x = 4       # x is of type int
x = "Sally" # x is now of type str
print(x)
```

```
    Sally
```

Casting

If you want to specify the data type of a variable, this can be done with casting.

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

Get the Type

You can get the data type of a variable with the type() function.

```

```

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

```
<class 'int'>
<class 'str'>
```

String variables can be declared either by using single or double quotes.

```
x = "John"
# is the same as
x = 'John'
```

Case-Sensitive

Variable names are case-sensitive.

```
# This will create two variables:

a = 4
A = "Sally"
#A will not overwrite a
print(a)
print(A)
```

```
4
Sally
```

Variable Names

A variable can have a short name (like x and y) or a more descriptive name (age, carname, total_volume). Rules for Python variables.

- 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).

```
# Legal variable names:

myvar = "John"
my_var = "John"
_my_var = "John"
myVar = "John"
MYVAR = "John"
myvar2 = "John"
```

```
# Illegal variable names:

2myvar = "John"
my-var = "John"
my var = "John"
```

Remember that variable names are case-sensitive.

Many Values to Multiple Variables

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

```
x, y, z = "Orange", "Banana", "Cherry"
print(x)
print(y)
print(z)
```

```
Orange
Banana
Cherry
```

Note: Make sure the number of variables matches the number of values, or else you will get an error.

One Value to Multiple Variables

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

```
x = y = z = "Orange"
print(x)
print(y)
print(z)
```

```
Orange
Orange
Orange
```

Unpack a Collection

If you have a collection of values in a list, tuple etc. Python allows you to extract the values into variables. This is called unpacking.

```
# Unpack a list:

```

```
fruits = ["apple", "banana", "cherry"]
x, y, z = fruits
print(x)
print(y)
print(z)
```

```
apple
banana
cherry
```

Output Variables

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

```
x = "Python is awesome"
print(x)
```

```
Python is awesome
```

In the print() function, you output multiple variables, separated by a comma.

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

```
Python is awesome
```

You can also use the + operator to output multiple variables.

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

```
Python is awesome
```

Notice the space character after "Python " and "is ", without them the result would be "Pythonisawesome".

For numbers, the + character works as a mathematical operator.

```
x = 5
y = 10
print(x + y)
```

```
15
```

In the print() function, when you try to combine a string and a number with the + operator, Python will give you an error.

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

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)
```

```
5 John
```

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.

```
# Create a variable outside of a function, and use it inside the function

x = "awesome"

def myfunc():
  print("Python is " + x)

myfunc()
```

```
Python is awesome
```

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.

```
# Create a variable inside a function, with the same name as the global variabl

x = "awesome"

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

myfunc()

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

```
Python is fantastic
Python is awesome
```