Video 1: Variables and Data Types

Table of Contents

Introduction

The 1st thing to know is that all Python files end with the extension `.py`. Then you have to understand that every programming language must have the ability to **Accept**, **Store** and **Name** Data.

You have to be able to receive data from the keyboard, or from other parts of your program and assign a name to that data. This data is either a single value or multiple values that are assigned a name. Data that is assigned a name and that contains data is called a Variable. Python has many different ways to store lists of data which I'll cover later.

It is convenient to assign names to data. If I want to store my age in Python I'd type `my_age = 43`. If I wanted to store my name I'd type `my_name = "Derek"`.

Rules for Naming Variables

Your variables can start with a letter or `_` (Underscore) After the 1st letter you can use numbers such as `num_1` You can't put spaces in variable names `my_age` is ok, but my age is not

> **Tip**
>
> It is considered good practice to separate words with underscores (`my_age` vs. `myAge`)

> **Keywords that you can't use for variable names**
>
> `and`, `del`, `from`, `not`, `while`, `as`, `elif`, `global`, `or`, `with`, `assert`, `else`, `if`, `pass`, `yield`, `break`, `except`, `import`, `print`, `class`, `exec`, `in`, `raise`, `continue`, `finally`, `is`, `return`, `def`, `for`, `lambda`, `try`

Try running this code where you assign your name to a variable and then print a message.

```
my_name = "Derek"
print("Hello", my_name)
```

`"Hello"` is known as a string. The `print()` function prints out the your screen the values between its parentheses. If you have multiple values separate them with commas.

Data is stored in essentially boxes in your computers memory. The size of the box you assign is referred to as a data type. If you want to store values with decimal places you store that data in a float data type. If you want to store a series of characters, numbers, etc. you store in a string data type.

Strings

A String is a data type that starts and ends with a `"`, `'`, or `'''` and contains letters, numbers and other characters. If you find that you want to use a double quote inside of a String proceed it with a backslash like this.

```
print("\"We never really grow up, we only learn how to act in public\" - Bryan White")
```

Escape Sequences

`\"` is one of many Escape Sequences. Here are other common Escape Sequences:

- Newline: `\n`
- Backslash: `\\`

- Single Quote: `\'`
- Backspace: `\b`
- Tab: `\t`

I'll do more with Strings later in the tutorial.

Number Data Types

There are 3 main number types in Python. Integers, floats and complex numbers. I'll cover complex numbers later.

Integers are values that don't have decimal values. `3`, `8`, `100000` are integers. Floats contain decimal values. Pi for example is a `float`.

There is no maximum value for an integer, as long as you have enough memory. You can however get a practical maximum size with this. Note that you'll have to import the `sys` module for this code to work. A module provides prewritten code you can use in your program.

```
import sys
print(sys.maxsize)
```

You can get the maximum size for a `float` like this

```
import sys
print(sys.float_info.max)
```

Please note however that errors can occur when using `float`s. This is true with all programming languages. When you create a variable a specific amount of space is set aside. If you create a value larger then that space allows errors can creep in. For example

```
f1 = 1.1111111111111111
f2 = 1.1111111111111111
f3 = f1 + f2
print(f3)
```

As you can see `float`s are accurate to 15 digits. Later I'll introduce data types with more accuracy.

Here is an example of a complex number. I'll cover them in more detail later.

```
cn1 = 5 + 6j
```

Booleans

A boolean data type can have either a value of `True` or `False`. You'll see how extremely valuable they are later.

```
can_vote = True
```

Dynamic Typing

Python is dynamically typed. What that means is a variables data type is determined by the value you assign to it. This is different from other languages. A variables value can also be changed even if that may sometimes not make sense. For example

```
my_age = 43
my_age = "Dog"
```

Many errors can occur if you don't make sure you are using the correct values. Some times you will find the need to convert from one type to another.

Casting

Casting allows you to convert from one type to another. Here is how you cast to the different types. I'll use the `type()` function to display the new data type for each variable.

```
print("Cast ", type(int(5.4)))   # float to int
print("Cast 2 ", type(str(5.4))) # float to string
print("Cast 3 ", type(chr(97)))  # unicode character to string
print("Cast 4 ", type(ord('a'))) # character to unicode
print("Cast 5 ", type(float(2))) # integer to float
```

Commenting

You may have noticed the `#` symbol used in the code above. `#` is used when you want to comment about what your code is doing. Everything you type after a `#` is ignored. It is very important to comment your code because what you understand about your code today you may forget 3 months from now.

You can also create multi-line comments like this

```
'''
I'm a multi-line
comment '''
```

Common Variable Errors

Variable names are case sensitive. For example `Age` is not the same as `age`.

```
age = 2
Age = 3
```

Make sure you are casting to the correct data type when working with variables. Also make sure that you surround calculations with parentheses when they produce a single value.

```
num_1 = "1"
num_2 = "2"
print("1 + 2 =", (int(num_1) + int(num_2)))
```

That's all for now. In the next video we'll learn about accepting user input and performing math calculations. Please take this quiz to reenforce what you've learned.