# Data Types Introduction

*by Sophia*

WHAT'S COVERED

In this lesson, you will learn about values, variables and the basic data types used in Python. Specifically, this lesson covers:

## 1. Values and Variables

So far, we've seen some examples of variables and values but haven't really explored what they are. A **value** is one of the basic things a program works with, like a letter or number. We've used a few examples so far where we've hardcoded certain values into our program, like 2, or "Hello World". These examples belong to different types: 2 is an integer, and "Hello World" is a string, so-called because it contains a "string" of letters. You can quickly identify strings, because they are enclosed in quotation marks. Remember that you can use either single or double quotes, as long as you are consistent.

If you're not sure what type a value has, the interpreter can tell you by putting the variable in a function named `type()`. As you'd expect, it will tell you what the type of a variable is:

EXAMPLE

```
print(type("Hello World"))
```

In the output, "Hello World" is identified as a class string

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

We've mentioned variables a few times so far, but in essence, they are containers we use to store data values. We need these variables so the programs we write can change as the program executes. Unlike many other languages, you do not need to define the data type when you assign a value to your variable. This is unique to Python, but it also forces you to think about the data types and the variables if you want to make use of them later on, because the results may not be what you intended them to be. In Python, the variable does not need to be declared or defined in advance. With Python, we use an assignment operator to assign a value to a variable. We can simply do that using the = **operator** (equal sign).

EXAMPLE

```
myVar = 1
```

What this will do is set the variable `myVar` to a value of 1. Once this is done, we can then use `myVar` in a **statement** (a unit of code that the Python interpreter can execute) or an **expression** (a combination of values, variables, and operators) where the value that's stored in `myVar` would be output, such as the following:

EXAMPLE

```
myVar = 1
print(myVar)
```

Here is the output.

```
1
```

Later on, if we wanted to change the value of `myVar` and use it again, the new value would be changed instead.

EXAMPLE

```
myVar = 1
print(myVar)
myVar = 2
print(myVar)
```

The variable `myVar` has changed.

```
1
2
```

KEY CONCEPT

There are some rules to these variable names. These include:

- Must start with a letter or an underscore character.
- Cannot start with a number.
- Can only contain letters, numbers, and the underscore character.
- Variable names are case sensitive.

Since they are case sensitive, all of these would reference different variables due to the case sensitivity.

EXAMPLE

```
myVar="1"
MYVAR="2"
myvar="3"
MyVar="4"
```

These are also legal variable names as they follow the variable naming rules.

⮫ EXAMPLE

```
_MyVar="5"
myVar2="6"
```

Let’s see what this looks like if we output them all.

⮫ EXAMPLE

```
myVar="1"
MYVAR="2"
myvar="3"
MyVar="4"
_MyVar="5"
myVar2="6"
print(myVar, MYVAR, myvar, MyVar, _MyVar, myVar2)
```

Looks like they all worked.

```
1 2 3 4 5 6
```

It’s important to note that even though we can create different variables in the same program with the same name but different cases, it’s generally a bad practice. It would be confusing to you and anyone else that’s reviewing the code.

To you, or to someone else especially, these examples could cause confusion if they are all used in the same program.

⮫ EXAMPLE

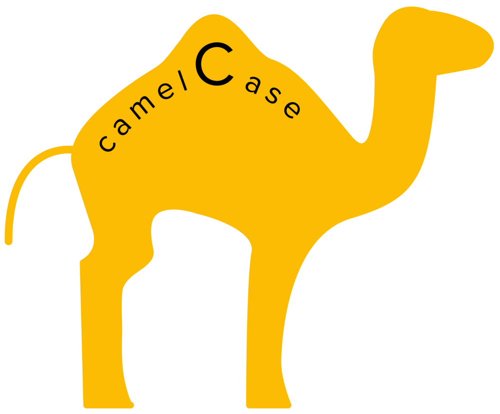
```
myVar
mYVar
MYVAR
```

These, on the other hand, are illegal variable names:

⮫ EXAMPLE
2myVar – illegal due to it starting with a number.
my-var – illegal due to it having a "-". Only letters, numbers and underscores are permitted.
My var – illegal due to it having a space between My and var. Only letters, numbers, and underscores are permitted.

It's always important to name our variables with names that are descriptive enough to indicate what they are being used for. There are also situations where you may have variables that use multiple words as part of the naming. There are certain techniques that can make it easier to follow.

**Camel Case**:



**Camel case** is one of the most widely used practices of combining phrases without spaces (remember a Python variable cannot contain spaces). We've used camel case in our examples so far, where each word except the first word starts with a capital letter.

EXAMPLE
Examples of camel case:

```
myVar
myVarForLaterUse
```

Even though this type of case is seen extensively in programming, you have likely seen this in the non-programming world as well. Think of iPhone or eBay. That's right—they use camel case too.

**Pascal Case**:

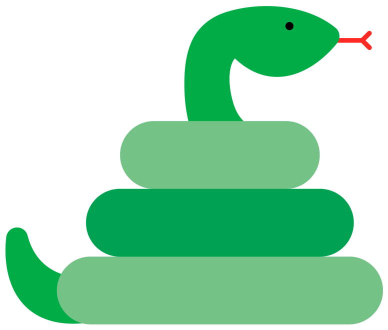
There's also **Pascal case**, which is similar to camel case, except the first word also starts with a capital letter.

EXAMPLE

Examples of Pascal case:

```
MyVar
MyVarForLaterUse
```

**Snake Case**:



One last option is **snake case**, where all words are lowercase but separated by an underscore.

EXAMPLE

Examples of snake case:

```
my_var
my_var_for_later_use
```

It doesn't matter which approach you use, as long as you're consistent.

TERMS TO KNOW

**Value**

One of the basic units of data, like a number or string, that a program manipulates.

=

The `=` operator (equal sign) is an assignment operator that is used to assign values to variables.

**Statement**

A unit of code that the Python interpreter can execute.

**Expression**

A combination of values, variables, and operators.

**Camel Case**
Combining words where each word except the first word will start with a capital letter.

**Pascal Case**
Combining words where each word starts with a capital letter.

**Snake Case**
Combining words where each word is separated by an underscore and all of the words are in lowercase.

---

# 2. Python Basic Data Types

In Python, we have several categories of data types. Within each category, we have some built-in data types that are available by default. Basic data types include:

- Text
- Numeric
- Boolean

**Text Type Category**
Within the text type category, we have the **str** (string) data type. The string data type consists of characters that are enclosed in either single or double quotes.

EXAMPLE
An example of a string would be:

```
myVar="Hello World"
```

**Numeric Type Category**

Within the numeric type, we have three main built-in data types. These are `int`, `float`, and complex.

**int** refers to an integer or a whole number. It can either be a positive or a negative number, but it does not have any decimals.

EXAMPLE
An example of an integer would be:

```
myInt = 1
myOtherInt = -200
```

**float** refers to a floating-point number. It is a number that can be positive or negative and uses decimal values.

EXAMPLE
An example of a float would be:

```
myFloat = 3.141517
MyOtherFloat = -20.1
```

A **complex** number is the sum of a real number and an imaginary number. This is one that we won't get into. They are written with the character j, which is the imaginary part:

EXAMPLE
An example of a complex would be:

```
myComplex = 20+4j
myOtherComplex = 6j
```

**Boolean Type Category**

The **boolean** type category just has the `bool` data type. This type can only have one of two values—True or False. These values are also considered numeric values, where True is equal to 1 and False is equal to 0.

Beyond these common basic types, there are other advanced, built-in data types that we'll cover later in the course. These include:

- Sequence Data Type
  - List
  - Tuple
  - Range
- Mapping Data Type
  - Dict
- Set Type
  - Set

These data types are more complex and each have specific uses.

TERMS TO KNOW

**Str**
A text data type consisting of characters that are enclosed in either single or double quotes.

**Int**
A numeric data type consisting of an integer or a whole number. It can either be a positive or a negative number but it does not have any decimals.

**Float**
A numeric data type referring to a floating point number. It is a number that can be positive or negative and uses decimal values.

**Complex**
A numeric data type consisting of a number that is the sum of a real number and an imaginary number.

**Bool**

A boolean data type consisting of a value of either True or False.

## SUMMARY

In this lesson, we learned more about **values and variables**, including how a value is one of the basic things a program works with, like a letter or number. We discussed that the variables hold values and can change values within a program. We learned that variables need to be named appropriately based on some “rules” and that their names should be meaningful. We also demonstrated the basics of writing variables in different case structures. Finally, we learned about **Python basic data types** including text, numeric, and boolean.

Best of luck in your learning!

## TERMS TO KNOW

=

The = operator (equal sign) is an assignment operator that is used to assign values to variables.

**Bool**

A boolean data type consisting of a value of either True or False.

**Camel Case**

Combining words where each word except the first word will start with a capital letter.

**Complex**

A numeric data type consisting of a number that is the sum of a real number and an imaginary number.

**Expression**

A combination of values, variables, and operators.

**Float**

A numeric data type referring to a floating point number. It is a number that can be positive or negative and uses decimal values.

**Int**

A numeric data type consisting of an integer or a whole number. It can either be a positive or a negative number but it does not have any  decimals.

**Pascal Case**

Combining words where each word starts with a capital letter.

| | |
|---|---|
| **Snake Case** | Combining words where each word is separated by an underscore and all of the words are in lowercase. |
| **Statement** | A unit of code that the Python interpreter can execute. |
| **Str** | A text data type consisting of characters that are enclosed in either single or double quotes. |
| **Value** | One of the basic units of data, like a number or string, that a program manipulates. |