**Types**

A type is how Python represents different types of data. In this video, we will discuss some widely used types in Python. You can have different types in Python. They can be integers like 11, real numbers like 21.213, they can even be words. Integers, real numbers, and words can be expressed as different data types. The following chart summarizes three data types for the last examples. The first column indicates the expression. The second column indicates the data type. We can see the actual data type in Python by using the type command. We can have int, which stands for an integer and float that stands for float, essentially a real number. The type string is a sequence of characters. Here are some integers. Integers can be negative or positive. It should be noted that there is a finite range of integers but it is quite large. Floats are real numbers. They include the integers but also numbers in between the integers. Consider the numbers between 0 and 1. We can select numbers in between them. These numbers are floats. Similarly, consider the numbers between 0.5 and 0.6. We can select numbers in between them. These are floats as well. We can continue the process zooming in for different numbers. Of course there is a limit but it is quite small. You can change the type of the expression in Python, this is called typecasting. You can convert an int to a float. For example, you can convert or cast the integer 2 to a float 2.0. Nothing really changes, if you cast a float to an integer, you must be careful. For example, if you cast the float 1.1 to 1, you will lose some information. If a string contains an integer value, you can convert it to int. If we convert a string that contains a non-integer value, we get an error. Check out more examples in the lab. You can convert an int to a string or a float to a string. Boolean is another important type in Python. A Boolean can take on two values. The first value is True, just remember we use an uppercase T. Boolean values can also be False with an uppercase F. Using the type command on a Boolean value, we obtain the term bool. This is short for Boolean, if we cast a Boolean True to an integer or float, we will get a 1. If we cast a Boolean False to an integer or float, we get a 0. If you cast a 1 to a Boolean, you get a True. Similarly, if you cast a 0 to a Boolean, you get a False.

**Expressions and Variables**

Expressions describe a type of operation the computers perform. Expressions are operations the python performs. For example, basic arithmetic operations like adding multiple numbers. The result in this case is 160. We call the numbers operands, and the math symbols in this case, addition, are called operators. We can perform operations such as traction using the subtraction sign. In this case, the result is a negative number. We can perform multiplication operations using the asterisk. The result is 25. In this case, the operands are given by negative and asterisk. We can also perform division with the forward slash- 25 / 5 is 5.0; 25 / 6 is approximately 4.167. In Python 3, the version we will be using in this course, both will result in a float. We can use the double slash for integer division, where the result is rounded. Be aware, in some cases the results are not the same as regular division. Python follows mathematical conventions when performing mathematical expressions. The following operations are in a different order. In both cases, Python performs multiplication, then addition, to obtain the final result. There are a lot more operations you can do with Python, check the labs for more examples. We will also be covering more complex operations throughout he course. The expressions in the parentheses are performed first. We then multiply the result by 60. The result is 1,920. Now, let's look at variables. We can use variables to store values. In this case, we assign a value of 1 to the variable my\_variable using the assignment operator, i.e, the equal sign. We can then use the value somewhere else in the code by typing the exact name of the variable. We will use a colon to denote the value of the variable. We can assign a new value to my\_variable using the assignment operator. We assign a value of 10. The variable now has a value of 10. The old value of the variable is not important. We can store the results of expressions. For example, we add several values and assign the result to x. X now stores the result. We can also perform operations on x and save the result to a new variable-y. Y now has a value of 2.666. We can also perform operations on x and assign the value x. The variable x now has a value: 2.666. As before, the old value of x is not important. We can use the type command in variables as well. It's good practice to use meaningful variable names; so, you don't have to keep track of what the variable is doing. Let say, we would like to convert the number of minutes in the highlighted examples to number of hours in the following music data-set. We call the variable, that contains the total number of minutes, total\_min. It's common to use the underscore to represent the start of a new word. You could also use a capital letter. We call the variable that contains the total number of hours, total\_hour. We can obtain the total number of hours by dividing total\_min by 60. The result is approximately 2.367 hours. If we modify the value of the first variable, the value of the variable will change. The final result values change accordingly, but we do not have to modify the rest of the code.

**String Operations**

In Python, a string is a sequence of characters. A string is contained within two quotes. You could also use single quotes. A string can be spaces or digits. A string can also be special characters. We can bind or assign a string to another variable. It is helpful to think of a string as an ordered sequence. Each element in the sequence can be accessed using an index represented by the array of numbers. The first index can be accessed as follows: We can access index six. Moreover, we can access the 13th index. We can also use negative indexing with strings. The last element is given by the index negative one. The first element can be obtained by index negative 15 and so on. We can bind a string to another variable. It is helpful to think of string as a list or tuple. We can treat the string as a sequence and perform sequence operations. We can also input a stride value as follows: The two indicates we'd select every second variable. We can also incorporate slicing. In this case, we return every second value up to index four. We can use the len command to obtain the length of the string. As there are 15 elements, the result is 15. We can concatenate or combine strings. We use the addition symbols. The result is a new string that is a combination of both. We can replicate values of a string. We simply multiply the string by the number of times we would like to replicate it- in this case, three. The result is a new string. The new string consists of three copies of the original string. This means you cannot change the value of the string, but you can create a new string. For example, you can create a new string by setting it to the original variable and concatenate it with a new string. The result is a new string that changes from Michael Jackson to Michael Jackson is the best. Strings are immutable. Back slashes represent the beginning of escape sequences. Escape sequences represent strings that may be difficult to input. For example, backslashes "n" represent a new line. The output is given by a new line after the backslashes "n" is encountered. Similarly, backslash "t" represents a tab. The output is given by a tab where the backslash, "t" is. If you want to place a backslash in your string, use a double backslash. The result is a backslash after the escape sequence. We can also place an "r" in front of the string. Now, let's take a look at string methods. Strings are sequences and as such, have apply methods that work on lists and tuples. Strings also have a second set of methods that just work on strings. When we apply a method to the string A, we get a new string B that is different from A. Let's do some examples. Let's try with the method "Upper". This method converts lowercase characters to uppercase characters. In this example, we set the variable A to the following value. We apply the method "Upper", and set it equal to B. The value for B is similar to A, but all the characters are uppercase. The method replaces a segment of the string- i.e. a substring with a new string. We input the part of the string we would like to change. The second argument is what we would like to exchange the segment with. The result is a new string with a segment changed. The method find, find substrings. The argument is the substring you would like to find. The output is the first index of the sequence. We can find the substring Jack. If the substring is not in the string, the output is negative one.