Data Types

Monday, December 13, 2021 2:50 PM

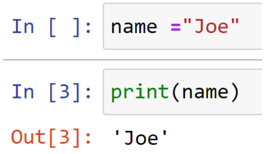
BASIC DATA TYPES

String	"Joe", 'Joe'
Int	1,2,3,4,5
float	1.0, 20.5
bool	True, False

These are the 4 main data types in Python.
There is no command in Python for declaring variables other than using defining a name for the variable, the assignment operator = followed by the value which can be any data type .
There is another data type called complex numbers.
(the Introduction section explores more about the Python operators)

VARIABLES

This is a simple code example that shows a basic variable:



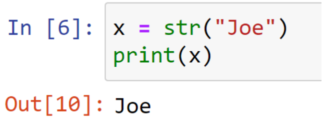
```
In [ ]: name ="Joe"

In [3]: print(name)
Out[3]:  'Joe'
```

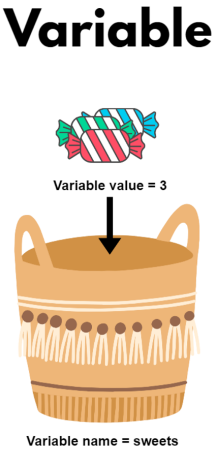
Here we are saying that the variable called name is 33 because we have used something called the assignment = operator. Note that this is different to the equal to == operator. This variable contains an integer because it is a whole number.

If you would like to specify the datatype of a variable then this can be done by casting, the variable is assigned as a string using the assignment operator =. The value is then put inside the parenthesis.

This is a casting example:



```
In [6]: x = str("Joe")
        print(x)
Out[10]: Joe
```

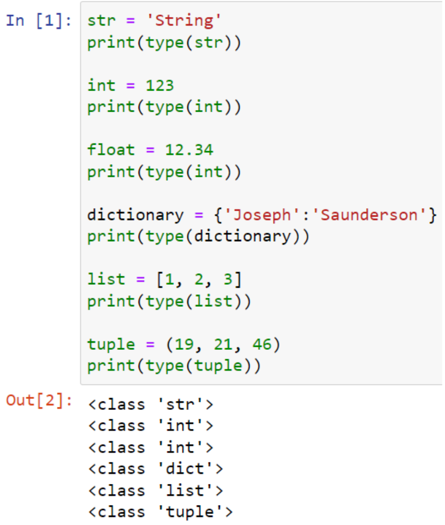


A visual representation of a variable, think of a variable as a container with a value stored inside it. In this case, the basket is a variable called sweets which holds a value equal to 3 which are represented as sweets.

CHECKING DATA TYPES

To check data types in python of a variable, use the built in `type()` method that return the class type of an argument object passed as a parameter.

This is combined with `print()` method, see below



```
In [1]: str = 'String'
        print(type(str))

        int = 123
        print(type(int))

        float = 12.34
        print(type(int))

        dictionary = {'Joseph':'Saunderson'}
        print(type(dictionary))

        list = [1, 2, 3]
        print(type(list))

        tuple = (19, 21, 46)
        print(type(tuple))
```

```
Out[2]: <class 'str'>
        <class 'int'>
        <class 'int'>
        <class 'dict'>
        <class 'list'>
        <class 'tuple'>
```