**Python Data Types**

After learning about variable initialization and assignment, you should be aware that data types are serious business. They can determine the success or failure of your project. Therefore, you should know them extremely well. This document should serve as a quick reference guide for the data types we will be using most often in this class. Research each of the terms below and write their definitions in the boxes below

|  |
| --- |
| **str :**  **Represents text strings in Python are identified as a contiguous set of characters represented in the quotation marks. Python allows for either pairs of single or double quotes. Subsets of strings can be taken using the slice operator ([ ] and [:] ) with indexes starting at 0 in the beginning of the string and working their way from -1 at the end.**  **The plus (+) sign is the string concatenation operator and the asterisk (\*) is the repetition operator. For example −**  **#!/usr/bin/python  str = 'Hello World!'  print str # Prints complete string print str[0] # Prints first character of the string print str[2:5] # Prints characters starting from 3rd to 5th print str[2:] # Prints string starting from 3rd character print str \* 2 # Prints string two times print str + "TEST" # Prints concatenated string**  **This will produce the following result −**  **Hello World! H llo llo World! Hello World!Hello World! Hello World!TEST** |
| **Integer:**  **Signed intergers, with no decimal, that can be positive or negative** |
| **float:**  **The datatype of decimal numbers** |
| **list:**  **A list of multiple data types enclosed in square brackets**  **Lists are the most versatile of Python's compound data types. A list contains items separated by commas and enclosed within square brackets ([]). To some extent, lists are similar to arrays in C. One difference between them is that all the items belonging to a list can be of different data type.**  **The values stored in a list can be accessed using the slice operator ([ ] and [:]) with indexes starting at 0 in the beginning of the list and working their way to end -1. The plus (+) sign is the list concatenation operator, and the asterisk (\*) is the repetition operator. For example −**  **#!/usr/bin/python  list = [ 'abcd', 786 , 2.23, 'john', 70.2 ] tinylist = [123, 'john']  print list # Prints complete list print list[0] # Prints first element of the list print list[1:3] # Prints elements starting from 2nd till 3rd  print list[2:] # Prints elements starting from 3rd element print tinylist \* 2 # Prints list two times print list + tinylist # Prints concatenated lists**  **This produce the following result −**  **['abcd', 786, 2.23, 'john', 70.200000000000003] abcd [786, 2.23] [2.23, 'john', 70.200000000000003] [123, 'john', 123, 'john'] ['abcd', 786, 2.23, 'john', 70.200000000000003, 123, 'john']** |
| **tuple: Like lists, but cannot be updated, and are in parenthesis**  **A tuple is another sequence data type that is similar to the list. A tuple consists of a number of values separated by commas. Unlike lists, however, tuples are enclosed within parentheses.**  **The main differences between lists and tuples are: Lists are enclosed in brackets ( [ ] ) and their elements and size can be changed, while tuples are enclosed in parentheses ( ( ) ) and cannot be updated. Tuples can be thought of as read-only lists. For example −**  **#!/usr/bin/python  tuple = ( 'abcd', 786 , 2.23, 'john', 70.2 ) tinytuple = (123, 'john')  print tuple # Prints complete list print tuple[0] # Prints first element of the list print tuple[1:3] # Prints elements starting from 2nd till 3rd  print tuple[2:] # Prints elements starting from 3rd element print tinytuple \* 2 # Prints list two times print tuple + tinytuple # Prints concatenated lists**  **This produce the following result −**  **('abcd', 786, 2.23, 'john', 70.200000000000003) abcd (786, 2.23) (2.23, 'john', 70.200000000000003) (123, 'john', 123, 'john') ('abcd', 786, 2.23, 'john', 70.200000000000003, 123, 'john')**  **The following code is invalid with tuple, because we attempted to update a tuple, which is not allowed. Similar case is possible with lists −**  **#!/usr/bin/python  tuple = ( 'abcd', 786 , 2.23, 'john', 70.2 ) list = [ 'abcd', 786 , 2.23, 'john', 70.2 ] tuple[2] = 1000 # Invalid syntax with tuple list[2] = 1000 # Valid syntax with list** |