PYTHON

**Data Types:**

* String – str
* Numbers – int ,float, complex
* Array – list, tuple
* set
* dictionary

**NOTE:** type() function returns the type of object.

**String:**

A string is a collection of one or more characters put in a single quote, double-quote or triple quote. In python there is no character data type, a character is a string of length one. It is represented by str class.

**Example:**

var = 'This is a string'

var1 = "This is also a string"

print(var, var1)

print(type(var), type(var1))

**OUTPUT:**

This is a string This is also a string

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

**Numbers:**

**Integers**– This value is represented by int class. It contains positive or negative whole numbers (without fraction or decimal). In Python there is no limit to how long an integer value can be.

**Float**– This value is represented by float class. It is a real number with floating point representation. It is specified by a decimal point. Optionally, the character e or E followed by a positive or negative integer may be appended to specify scientific notation.

**Complex Numbers** – Complex number is represented by complex class. It is specified as *(real part) + (imaginary part)j*. For example – 2+3j

**Example:**

var1 = 1

var2 = 3.5

var3 = 5+7j

print(type(var1), type(var2), type(var3))

**Output:**

<class 'int'> <class 'float'> <class 'complex'>

**List:**

* Lists are just like the arrays, declared in other languages.
* A single list may contain Data Types like Integers, Strings, as well as Objects.
* Lists are mutable, and hence, they can be altered even after their creation.
* List in Python are ordered and have a definite count.
* The elements in a list are indexed according to a definite sequence and the indexing of a list is done with 0 being the first index.

**Example:**

mylist = [1,'Hello',3.4,[1,2,3]]

print(mylist)

print(type(mylist))

**Output:**

[1, 'Hello', 3.4, [1, 2, 3]]

<class 'list'>

**Tuple:**

* Tuple are similar to lists, but they are immutable data type.
* Tuples in python are ordered.
* Trying to add, update or delete an element in tuple produces error.
* Negative indexing is possible in tuples as possible in the lists data type(-1 is the index of the last element, -2 represents the last before element and so on).
* A tuple can have any number of items and they may be different data types as well.
* Tuples in python are faster than lists.

**Example:**

mytuple = ("Bonjour", 2.2, [1, 2, 3])

print(mytuple)

print(type(mytuple))

print(len(mytuple))

**Output:**

("Bonjour", 2.2, [1, 2, 3])

<class 'tuple'>

3

**Note:** Tuples are represented in parenthesis.

**Set:**

* A set is a collection of unordered and unindexed elements.
* Sets are mutable and has no duplicate elements
* A set is represented by curly brackets- {}

**Example:**

myset = {'a', 'b', 'c', 'd'}

print(myset)

print(type(myset))

myset.add('d')

print(myset)

**Output:**

{'b', 'c', d', 'a'}

<class 'set'>

{'a', 'b', 'c', 'd'}

**Note:**

* While printing the set, the elements can be printed in any order as the set data type is unordered regardless of how they are created.
* In the above example, the element 'd' will not be added to the set, as it is already present in the set(set contains only distinct values).
* Indexing, slicing cannot be done in set.

## Dictionary:

Dictionary in Python is an unordered collection of data values, used to store data values like a map, which unlike other Data Types that hold only single value as an element, Dictionary holds **key - value** pair. Key value is provided in the dictionary to make it more optimized.

**Note –**Keys in a dictionary doesn’t allows Polymorphism.

**Example:**

myDict = {1: [1, 2, 3, 4], 'Name': 5} #Dictionary with the use of Mixed Keys

print(myDict)

print(type(myDict))

**Output:**

{1: [1, 2, 3, 4], 'Name': 5}

<class 'dict'>

## Adding elements to a Dictionary

# Creating an empty Dictionary

myDict={}

print(myDict)

**Output**: {}

# Adding elements one at a time

myDict[0] ='hello’

myDict[2] ='world'

myDict[3] =1

print(myDict)

**Output**:

{0: 'hello', 2: 'world', 3: 1}

# Adding set of values

# to a single Key

myDict['Value\_set'] =2, 3, 4

print(myDict)

**Output**:

{0: 'hello', 2: 'world', 3: 1, 'Value\_set': (2, 3, 4)}

# Updating existing Key's Value

myDict[2] ='Welcome'

print(myDict)

**Output**:

{0: 'hello', 2: 'Welcome', 3: 1, 'Value\_set': (2, 3, 4)}

# Adding Nested Key value to Dictionary

myDict[5] ={'Nested':{'1': 'one', '2': 'two'}}

print(myDict)

**Output**:

{0: 'hello', 2: 'Welcome', 3: 1, 5: {'Nested': {'1': 'one', '2': 'two'}}, 'Value\_set': (2, 3, 4)}