**9.3. Data Structures List**

* 1. Numbers
  2. String
  3. **List**
  4. Tuple
  5. Dictionary
  6. Set

REQ : 1. CRUD :

2. State (variable)

3. Behavior (Operators, DM, Loops)

Every value in Python has a datatype. Since everything is an object in Python programming, data types are actually classes and variables are instance (object) of these classes.

Type represents the kind of value and determines how the value can be used. All data values in Python are encapsulated in relevant object classes. Everything in Python is an object and every object has an identity, a type, and a value.

Like another object-oriented language such as Java or C++, there are several data types which are built into Python. Extension modules which are written in C, Java, or other languages can define additional types.To determine a variable's type in Python you can use the **type()** function.

**List memory creation:**

list1 = [4, 3, 6, 2, 1, 8, 4, 3] *#indexing*

*# 0 1 2 3 4 5 6 7*

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |

323213213

*# 0 1 2 3 4 5 6*

list2 = [3, **True**, 4.5, **'HelloWorld'**, **'Madhu'**, 123, [10,20,30]]

# matrix of matrices 01234567789 01234 0 1 2

0 1 2 3

0 1 2 3 4 5 6 7 8 9

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | 1 | 1 | 0 | 1 | 0 | 1 | 1 | 1 | 1 | |  |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | | H | E | L | L | O | W | O | R | L | D | |  |  |  |  |  |  |  |  |  |  | | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | |

0 1 2 3

**Sequences :**

**There are 6 sequence types:**

1. Strings
2. **Lists**
3. Tuples
4. Bytearrays
5. buffers and
6. range **objects**

There are certain things you can do with **all sequence types**.

These operations include

* **indexing**
* **slicing**
* adding
* multiplying and
* **checking for membership**.

Python has built-in functions for :**finding the length of a sequence** and

**for finding its max and min elements.**

**5.3 Lists:**

The list is a most versatile datatype available in Python which can be written as a list of comma-separated values (items) between square brackets.

\*\***items in a list need not be of the same type.**

list1 = [1997, 'chemistry', 10.5, 'physics',[10,20,30], (),{},{} ]

emp\_ids = [3, 5, [10,20,[3,4] ], True, 50.5 ]

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 3 | 5 | |  |  |  | | --- | --- | --- | | 10 | 20 |  | | 0 | 1 | 2 | | True | 50.5 |

0 1 2 3 4

12345

list3 = ["a", "b", "c", "d"]

|  |  |  |  |
| --- | --- | --- | --- |
| 1997 | 'chemistry' | 2000 | “Physics” |

0 1 2 3

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | | 0 | 1 | 0 | 1 | 0 | 1 | 1 | 0 | | |  |  |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | --- | --- | | C | H | E | M | I | S | T | R | Y |   0 1 2 3 4 5 6 7 8 |  |  |

0 1 2 3

List1[1][3]

|  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| |  |  |  |  |  |  |  | | --- | --- | --- | --- | --- | --- | --- | | 10001 | 00111 | 100001 | 100001 | 100001 | 011100 | 100011100 | |  |  |  |

123456

List1 list1[0] list1[1] list1[2] list1[3]

Step 1 : Prepare indexing for list at outer level

Step 2 : Go to each index, and prepare inner indexing for each index.

Step 3 :

Similar to string indices,

list indices start at 0, and lists can be **sliced, concatenated** and so on.

**List operations :**

1. **Creating list**: **“C” CREATE**

list1 = ['physics', 'chemistry', 1997, 2000, [1,2,4]]

1. **Accessing** values in Lists **“R” READ**

To access values in lists, use the square brackets for slicing along with the index or indices to obtain value available at that index

list1 =['physics','chemistry',1997,2000];

list2 =[1,2,3,4,5,6,7];

print"list1[0]: ", list1[0]

print"list2[1:5]: ", list2[1:5]

list1[0]: physics

list2[1:5]: [2, 3, 4, 5]

1. **Updating** values in Lists: **“U” UPDATE**

You can update single or multiple elements of lists by giving the slice on the left-hand side of the assignment operator,

list=['physics','chemistry',1997,2000]

print"Value available at index 2 : "

print list[2]

list[2]=2001

print"New value available at index 2 : "

print list[2]

Value available at index 2 :

1997

New value available at index 2 :

2001

1. **Delete** List Elements in Lists:

To remove a list element, you can use either the **del** statement if you know exactly which element(s) you are deleting or the **remove()** method if you do not know.

For example −

list1 =['physics','chemistry',1997,2000];

print list1

del list1[2]

print"After deleting value at index 2 : "

print list1

['physics', 'chemistry', 1997, 2000]

After deleting value at index 2 :

['physics', 'chemistry', 2000]

**Basic List Operations:**

Lists respond to the + and \* operators much like strings; they mean concatenation and repetition here too, except that the result is a new list, not a string.

In fact, lists respond to all of the general sequence operations we used on strings in the prior chapter.

|  |  |  |
| --- | --- | --- |
| **Python Expression** | **Results** | **Description** |
| **len**([1, 2, 3]) | 3 | Length |
| [1, 2, 3] **+** [4, 5, 6] | [1, 2, 3, 4, 5, 6] | Concatenation |
| ['Hi!'] **\*** 4 | ['Hi!', 'Hi!', 'Hi!', 'Hi!'] | Repetition |
| 3 **in** [1, 2, 3] | True | Membership |
| for x **in** [1, 2, 3]: print x, | 1 2 3 | Iteration |

**Indexing, Slicing, and Matrixes:**

Because lists are sequences, indexing and slicing work the same way for lists as they do for strings.

Assuming following input −

L = ['spam', 'Spam', 'SPAM!']

|  |  |  |
| --- | --- | --- |
| **Python Expression** | **Results** | **Description** |
| L[2] | 'SPAM!' | Offsets start at zero |
| L[-2] | 'Spam' | Negative: count from the right |
| L[1:] | ['Spam', 'SPAM!'] | Slicing fetches sections |

**Built-in List Functions & Methods**

Python includes the following list functions −

|  |  |
| --- | --- |
| **Sr.No.** | **Function with Description** |
| 1 | [**cmp(list1, list2)**](https://www.tutorialspoint.com/python/list_cmp.htm) Compares elements of both lists. |
| 2 | [**len(list)**](https://www.tutorialspoint.com/python/list_len.htm)Gives the total length of the list. |
| 3 | [**max(list)**](https://www.tutorialspoint.com/python/list_max.htm)Returns item from the list with max value. |
| 4 | [**min(list)**](https://www.tutorialspoint.com/python/list_min.htm)Returns item from the list with min value. |
| 5 | [**list(seq)**](https://www.tutorialspoint.com/python/list_list.htm)Converts a sequence into list. |

**Python includes following list methods**

|  |  |
| --- | --- |
| **Sr.No.** | **Methods with Description** |
| 1 | [**list.append(obj)**](https://www.tutorialspoint.com/python/list_append.htm) Appends object obj to list **APRIECIRS** |
| 2 | [**list.pop(obj=list[-1])**](https://www.tutorialspoint.com/python/list_pop.htm) Removes and **returns** last object or obj from list |
| 3 | [**list.remove(obj)**](https://www.tutorialspoint.com/python/list_remove.htm) Removes object obj from list |
| 4 | [**list.insert(index, obj)**](https://www.tutorialspoint.com/python/list_insert.htm) Inserts object obj into list at offset index |
| 5 | [**list.extend(seq)**](https://www.tutorialspoint.com/python/list_extend.htm) Appends the contents of seq to list |
| 6 | [**list.count(obj)**](https://www.tutorialspoint.com/python/list_count.htm)**Returns** count of how many times obj occurs in list |
| 7 | [**list.index(obj)**](https://www.tutorialspoint.com/python/list_index.htm) **Returns** the lowest index in list that obj appears |
| 8 | [**list.reverse()**](https://www.tutorialspoint.com/python/list_reverse.htm) Reverses objects of list in place |
| 9 | [**list.sort(obj)**](https://www.tutorialspoint.com/python/list_count.htm)Sorts the elements |

**10 copy**

**Shallow copy vs Deep copy**

([]))

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| H | E | L | L | O | W | O | R | L | D |
| 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| 0 | 0 | 1 | 0 | 1 | 0 | 0 |  | 0 | 0 | 1 | 0 | 1 | 0 | 0 |  | 0 | 0 | 1 | 0 | 1 | 0 | 0 |  | 0 | 0 | 1 | 0 | 1 | 0 | 0 |  |

**123478248**

Message1 = “HELLOWORLD”

Message1 = message1+”*Welcome to Python”*

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Helloworldwelcome to python  0123456789 | | | | | | | | | | | | |
| 0101001 | ---- | ----- |  |  |  |  |  |  |  |  | 01000101001 |

5498614811