ISIM 2021

International Symposium on
IoT and ML for
Ecosystem Restoration &
Multihazard Resilience
05th to 09th of June 2021

Introduction to Python Programming

Lakshmi S. Gopal

Amrita Center for Wireless Networks & Applications

Amrita School of Engineering

Amrita Vishwa Vidyapeetham

AMRITA VISHWA VIDYAPEETHAM
DEEMED TO BE UNIVERSITY

Contents

- Data Structures - Lists
- Accessing values in a list
- Updating values in a list
- Adding new elements to a list
- Deleting list elements
- Basic list operations

Data Structures - Python Lists

- Data structures are like shelves/cupboards that can store bundles of data
- Each of this ‘cupboard’, ‘shelf’ etc form what is analogous to a data structure.
- Some common data structures in Python :
  - Lists
  - Tuples
  - Set
  - Dictionary

What is a List?

- A list is a collection of some items
- The items need not be of the same data type
- A list always uses square brackets to enclose its contents
- A list represents a sequence of contents.

```
['hai', '12', 12, 12.3]
```

What is a List?

- Lists are ordered.
- Lists can contain any arbitrary objects.
- List elements can be accessed by index.
- Lists are mutable.
- Lists are dynamic.

Creating Lists

- Creating an empty list
  - emp_list = []
- Creating a list of integers
  - integer_list = [1, 2, 3]
- Creating a list with different data types elements
  - diff_list = [1, “Python”, 3.14]
- Creating a nested list
  - nested_list = [“Python", [3.14,2.23], [‘P’,’Y’,’T’,’H’,’O’,’N’]]

Accessing values in a list

We can access the elements of a list using the following methods:

1. List index
2. Negative indexing
3. Slicing

Accessing values in a list

1. Using list index

- we use the index operator []
- index of a list begins at 0
  - so, if a list contains 10 elements, the indices will be 0 to 9
  - if we try accessing an element out of this range, IndexError is raised
- similar method can be used to access nested lists as well

Accessing values in a list

1. Using list index - Examples

```
my_list=['hai','12',12,12.3]
print(my_list[1])
```

```
12
```

```
my_list=['hai','12',[12,12.3]]
print(my_list[2][1])
```

```
12.3
```

Accessing values in a list

2. Using negative indexing

- Python allows negative indexing for its sequences.
- The index of -1 refers to the last item, -2 to the second last item and so on.

```
my_list=[2,4,6,8,10,12]
print(my_list[-1])
print(my_list[-2])
print(my_list[-3])
```

```
12
10
8
```

```
my_list=[2,4,6,8,10,12,[1,2,3]]
print(my_list[-1][-1])
```

```
3
```

Accessing values in a list

3. Slicing

- We can access a range of items in a list by using the slicing operator :(colon).

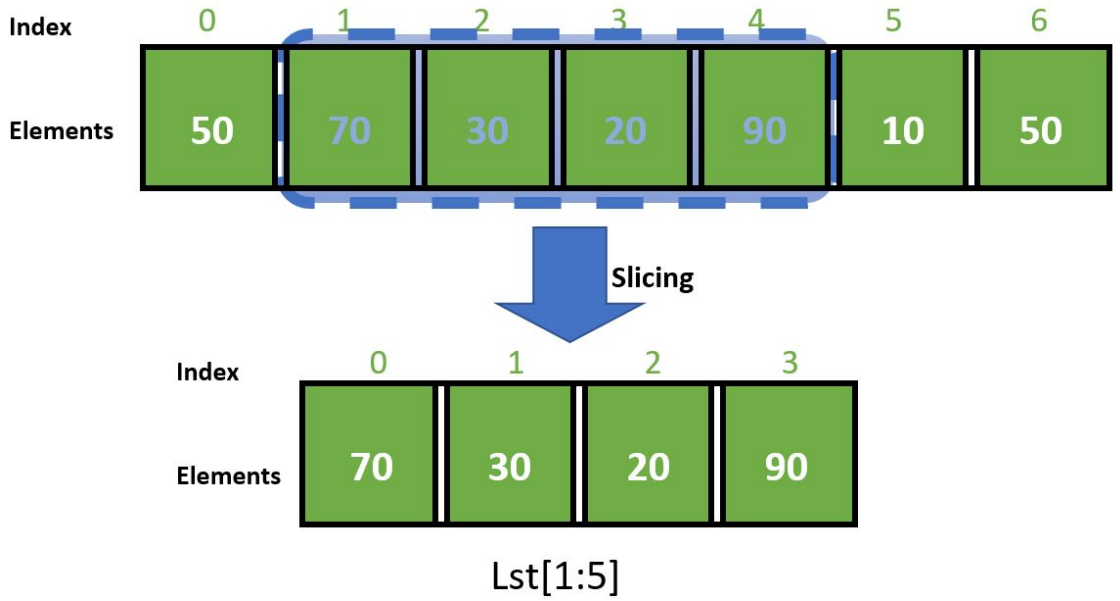

Image source : https://www.geeksforgeeks.org/python-list-slicing/

Updating values in a list

- values in a list can be modified
- lists are mutable
- using list index or negative index

```
my_list=[2,4,6,8,10,12]
my_list[1]=10
print(my_list)
```

```
[2, 10, 6, 8, 10, 12]
```

```
my_list=[2,4,6,8,10,12]
my_list[-3]=10
print(my_list)
```

```
[2, 4, 6, 10, 10, 12]
```

Adding new elements

- using in-built functions in Python
  - append()
    - adds an element to the end of a list
  - extend()
    - add multiple elements to a list
  - insert()
    - insert an element in a specific position in the list
    - need 2 arguments - the element and the position

Deleting list elements

- using in built functions :
  - pop()
    - takes up an optional argument - position of the item to be deleted
    - by default it removes the last element
  - remove()
    - takes one argument - element to be deleted
  - clear()
    - used to empty a list

Basic List operations

- finding length of a list
  - using built-in function len()
- concatenation
  - concatenating lists using '+' operator
- Repetition
  - repeat elements using '*' operator
- iterating through a list
  - using loops
- Sorting a list
  - using built-in function sort( )
- searching an element
  - using built-in function index(element,start,end)

Thank you !