List

Introduction

- Contains multiple values that are logically related
- List is a type of mutable sequence in Python
- Each element of a list is assigned a number – index / position
- Can do indexing, slicing, adding, multiplying, and checking for membership
- Built-in functions are available for finding length of a sequence, for finding its largest and smallest elements, etc

What is a List?

- Most **versatile data type** in Python
- **Comma-separated items** can be collected in **square brackets**

- Good thing is..
  - THE ITEMS IN THE **LIST NEED NOT BE OF SAME TYPE**

Creating a list

- Creating an EMPTY list

  listname = []

Example:

```
L1 = []
MyList = []
Books = []
```

- Creating a list with items

listname = [item1, item2, ….]

Click to add text

Example:

```
Temp = [100, 99.8, 103, 102]
S = ['17MIS0001', 'Arun', 99.9]
 L2 = [1, 2, 3, 4, 5, 6, 7]
Course = ['Python', 'C', 'C++', 'Java']
```

Accessing Values

- Using index or indices

```
    >>>L1 = [1, 2, 3, 4, 5, 6]
    >>>print (L1[3])      #indexing
    >>>4
    >>>print (L1[2:5])    #slicing
    >>>[3, 4, 5]
>>> x=list(“123456")
>>> x
        [‘1', ‘2', ‘3', ‘4‘, ’5’, ’6’]
```

Updating Elements

- **Update** an element in list using index

```
>>>L1 = [1, 2, 3, 4, 5, 6]
>>>L1[2] = 111
>>>L1
    [1, 2, 111, 4, 5, 6]
```

Deleting Elements

- **Delete** an element in list using index

```
>>>L1 = [1, 2, 3, 4, 5, 6]

>>>del(L1[4])

>>>L1
    [1, 2, 3, 4, 6]
```

Basic Operations in List

- >>> **len([1, 2, 3])** # Length

  3

- >>> X1=[1, 2, 3] **+** [4, 5, 6] # Concatenation

  [1, 2, 3, 4, 5, 6]

- >>> x2=['Ni!'] * 4 # Repetition
- >>>x2
- **['Ni!', 'Ni!', 'Ni!', 'Ni!']**

Basic Operations in List

- >>> [1, 2] + **list("34")**
  # Same as [1, 2] + **["3", "4"]**
  [1, 2, '3', '4‘]
- >>> [1, 2] + [**"34“]**
  [1, 2, ‘34‘]

```
>>> mylist=[1, 2] + list("34")
>>>print(mylist)
   [1, 2, '3', '4‘]
>>> x=list("ramu")
>>> x
          ['r', 'a', 'm', 'u']
```

List Iteration

- `>>> 3 in [1, 2, 3]` # Membership

  ```
  True
  ```

- `>>> 4 in [1, 2, 3]` # Membership

  ```
  False
  ```

- `>>> for x in [1, 2, 3]:`

  ```
          print(x, end=' ')
  1 2 3
  ```

```
list1=[1,2,3]
for x in list1:
   print(x,end=' ')

1 2 3
```

List Comprehensions

```
>>> res = [c * 4 for c in 'SPAM']
# List comprehensions
>>> res
['SSSS', 'PPPP', 'AAAA', 'MMMM']
>>>x=['SSSS', 'PPPP', 'AAAA', 'MMMM']
```

- expression is functionally **equivalent** to a **for loop** that builds up a list of results manually
- list comprehensions are **simpler** to code and likely **faster to run** today:

List Comprehensions

```
# List comprehension equivalent ...
>>> res = []
>>> for c in 'SPAM':
        res.append(c * 4)
>>> res
['SSSS', 'PPPP', 'AAAA', 'MMMM']
>>>a=[10,20,30]
>>>a.appen(5)
>>>a
[10,20,30,5]
```

```
x=[10]
for c in 'spam':
    x.append(c*4)
print(x)

[10,'SSSS', 'PPPP',
'AAAA', 'MMMM']
```

Indexing, Slicing

```
>>> L = ['spam', 'Spam', 'SPAM!‘,’SpaM’]

>>> L[2]              # Offsets start at zero

'SPAM!'

>>> L[–2]             # Negative: count from the right

'Spam'

>>> L[1:]              # Slicing fetches sections

['Spam', 'SPAM!‘,’SpaM’]
```

Insertion, Deletion and Replacement

```
>>> L = [11, 22, 33]
>>>L[1:2]
22
>>> L[1:2] = [4, 5]        # Replacement/insertion
>>> L
[11, 4, 5, 33]
>>> L[1:1] = [6, 7]    # Insertion (replace nothing)
>>> L
[11, 6, 7, 4, 5, 3]
>>> L[1:2] = []                # Deletion (insert nothing)
>>> L
[1, 7, 4, 5, 3]
```

Insertion, Deletion and Replacement

```
>>> L = [1]
# Insert all at :0, an empty slice at front
>>> L[:0] = [2, 3, 4]
>>> L
[2, 3, 4, 1]
# Insert all at len(L):, an empty slice at end
>>> L[len(L):] = [5, 6, 7]
 >>> L
[2, 3, 4, 1, 5, 6, 7]
```

List method calls

```
# Append method call: add item at end
>>> L = ['eat', 'more', 'SPAM!']
>>> L.append('please')
>>> L
['eat', 'more', 'SPAM!', 'please']
>>> L.sort()                    # Sort list items ('S' < 'e')
>>> L
['SPAM!', 'eat', 'more', 'please']
```

More on Sorting Lists

```
>>> L = ['abc', 'ABD', 'aBe']
>>> L.sort()                    # Sort with mixed case
>>> L
['ABD', 'aBe', 'abc']
>>> L = ['abc', 'ABD', 'aBe']
>>> L.sort(key=str.lower)       # Normalize to lowercase
>>> L
['abc', 'ABD', 'aBe']
```