6 Lists

Jin Wang,

MSIS, Rutgers

Table of contents

1. A list is a sequence

Define a list

```
>>> list_1 = [10, 20, 30, 40]
>>> list_2 = ['Rutgers', 'Princeton', 'NYU']
```

`len()` function

```
>>> list_1 = [10, 20, 30, 40]
>>> len(list_1)
4
```

List slices

```
>>> l = ['a', 'b', 'c', 'd', 'e', 'f']
>>> l[1:3]
['b', 'c']
>>> l[:4]
['a', 'b', 'c', 'd']
>>> l[3:]
['d', 'e', 'f']
>>> l[:]
['a', 'b', 'c', 'd', 'e', 'f']
```

2. Lists are mutable

```
>>> l = ['a', 'b', 'c', 'd', 'e', 'f']
>>> l[0] = 'w'
>>> l
['w', 'b', 'c', 'd', 'e', 'f']
```

3. List methods

`.append()`

- `append()` method adds a new element to the end of a list.

```
>>> l = ['a', 'b', 'c']
>>> l.append('d')
>>> l
['a', 'b', 'c', 'd']
```

`.extend()`

- `.extend()` takes a list as an argument and appends all of the elements:

```
>>> l1 = ['a', 'b', 'c']
>>> l2 = ['d', 'e']
>>> l1.extend(l2)
>>> l1
['a', 'b', 'c', 'd', 'e']
```

`.sort()`

- `.sort()` sorts the list ascending by default.
- There is an optional parameter, `reverse`, in `sort` method. Default is `reverse=False`. `reverse=True` will sort the list descending.
- For numbers:

```
>>> l = [4, 10, 5, 2]
>>> l.sort()
>>> l
[2, 4, 5, 10]
```

- or

```
>>> l = [4, 10, 5, 2]
>>> l.sort(reverse=True)
>>> l
[10, 5, 4, 2]
```

4. Void and fruitful method/function

fruitful function/method

- A function/method that returns values

void function/method

- A function that doesn't return values.

5. Traversing a list

- The most common way to traverse the elements of a list is with a for loop. The syntax is the same as for strings:

```
universities = ['Rutgers', 'NYU', 'Princeton']
for i in universities:
        print(i)
```

- We can also traverse a list through indices.

```
universities = ['Rutgers', 'NYU', 'Princeton']
for i in range(len(universities)):
        print(universities[i])
```

6. List operations

`+` operator

- The + operator concatenates lists:

```
>>> a = [1, 2, 3]
>>> b = [4, 5, 6]
>>> c = a + b
>>> c
[1, 2, 3, 4, 5, 6]
```

`*` operator

- The * operator repeats a list a given number of times:

```
>>> [0] * 4
[0, 0, 0, 0]
>>> [1, 2, 3] * 3
[1, 2, 3, 1, 2, 3, 1, 2, 3]
```

`in` operator

- The `in` operator helps find out whether an element is in a list.

```
>>> universities = ['Rutgers', 'NYU', 'Princeton']
>>> 'Rutgers' in universities
True
>>> 'Harvard' in universities
False
```

7. Deleting elements

Method 1: `.pop()`

- If you know the index of the element you want, you can use pop:

```
>>> t = ['a', 'b', 'c']
>>> x = t.pop(1)
>>> t
['a', 'c']
>>> x
'b'
```

- `pop` modifies the list and returns the element that was removed.

Method 2: `del`

- If you don't need the removed value, you can use the `del` operator:

```
>>> t = ['a', 'b', 'c']
>>> del t[1]
>>> t
['a', 'c']
```

- To remove more than one element, you can use del with a slice index:

```
>>> t = ['a', 'b', 'c', 'd', 'e', 'f']
>>> del t[1:5]
>>> t
['a', 'f']
```

Method 3: `.remove()`

- If you know the element you want to remove (but not the index), you can use remove:

```
>>> t = ['a', 'b', 'c']
>>> t.remove('b')
>>> t
['a', 'c']
```

- The return value from remove is None.

8. List operation 1: Map

Format of `map()`

```
map(function, iterable)
```

Lambda function

9. List operation 2: Filter

10. List comprehension

11. Practice questions

1. Find even numbers, using loop and `append()` method.
    - Method 1:
        - Define a list containing a list of numbers
        - Use `for` loop to traverse the elements in the list. If the element is an even number, put it into a new list using `append` method.
        - For example, given a list `[2, 3, 10, 17, 20]`, the result is [2, 10, 20].
    - Method 2:
        - Use comprehension to finish the question.

2. Find prime numbers
    - Create a function which takes an integer, return `True` if the number is a prime number and return `False` otherwise.
    - Define a list containing a list of numbers
    - Use `filter` function to obtain the prime numbers in the list and print them.
        - For example, given a list `[2, 3, 10, 17, 20]`, the prime numbers are 2, 3 and 17.

3. Capitalize all words in a list, using `map` function.
    - Create a function which takes a string and returns capitalized string.
    - Define a list containing several words/strings.
    - Use `map` function to capitalize all words in the list.
    - For example, given a list `['Rbs', 'Rutgers']`, the result is `['RBS', 'RUTGERS']`.
    - Use comprehension to finish the question.

4. Write a function called `nested_sum` that takes a list of lists of integers and adds up the elements from all of the nested lists. For example:

```
>>> t = [[1, 2], [3], [4, 5, 6]]
>>> print(nested_sum(t))
21
```