Lists Advanced

SoftUni Team

Technical Trainers

Software University

SoftUni

Software University

https://softuni.bg

Table of Contents

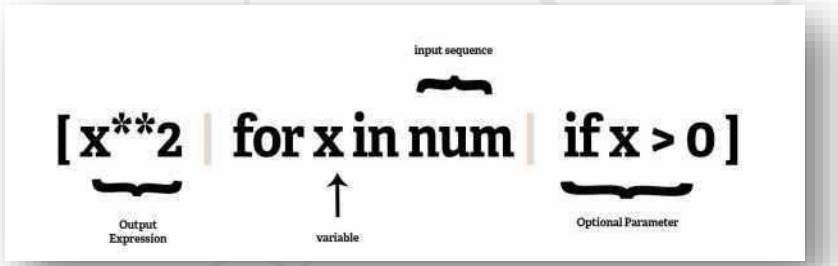
List Comprehensions

What is Comprehension?

- Comprehensions provide us with a **short** way to **construct** new **sequences**
- They allow **sequences** to be built from other sequences
- They require less **memory**
- They have shorter **syntax** and better **performance**

Structure

- A list comprehension consists of the following parts:
  - an **input sequence**
  - a **variable** representing members of the input sequence
  - an **optional predicate** expression
  - an **output expression** producing elements in the output list



List Comprehensions

- Creating a list using the **range** function

Output Expression

```
x = [num for num in range(5)]
# [0, 1, 2, 3, 4]
```

Variable

- Getting the square values of numbers in a list

```
nums = [1, 2, 3, 4]
squares = [x**2 for x in nums]
# [1, 4, 9, 16]
```

Input Sequence

List Comprehensions

Software University

- Using `if` statement in a list comprehension

```
nums = [1, 2, 3, 4, 5, 6]
evens = [num for num in nums if num % 2 == 0]
# [2, 4, 6]
```

Optional Parameter

- Using `if-else` statements in a list comprehension

```
nums = [1, 2, 3, 4, 5, 6]
filtered = [True if x % 2 == 0 else False for x in nums]
# [False, True, False, True, False, True]
```

List Methods

Adding Elements

- Using the `append()` method

```
my_list = [1, 2, 3]
my_list.append(4) # [1, 2, 3, 4]
```

Add single element at the end

- Using the `extend()` method

```
my_list = [1, 2, 3]
my_list.extend([4, 5]) # [1, 2, 3, 4, 5]
```

Add multiple elements at the end

- Using the `insert()` method

```
my_list = [1, 2, 3]
my_list.insert(1, 4) # [1, 4, 2, 3]
```

Add single element at a specific index

Removing Elements

- Using the `clear()` method

```
my_list = [1, 2, 3]
my_list.clear() # []
```

Removes all elements

- Using the `pop()` method

```
my_list = [1, 2, 3]
number = my_list.pop(0) # [2, 3]; number -> 1
```

Removes element by index and returns it

- Using the `remove()` method

```
my_list = [1, 2, 3]
my_list.remove(1) # [2, 3]
```

Removes by value (first occurrence)

More Useful Methods

- Using the `count()` method

```
my_list = [1, 2, 3, 2, 2]
my_list.count(2) # 3
```

Finds all occurrences in a list

- Using the `index()` method

```
my_list = [1, 2, 3, 2, 2]
last = my_list.index(2) # 1
```

Finds the index of the first occurrence

- Using the `reverse()` method

```
my_list = [1, 2, 3]
my_list.reverse() # [3, 2, 1]
```

Reverses the elements

Advanced Functions

Using Lambda Operators

sorted() Function

- Sorts the elements of a list in **ascending** order

```
numbers_list = [6, 2, 1, 4, 3, 5]
sorted_numbers = sorted(numbers_list)
# [1, 2, 3, 4, 5, 6]
```

- Sorts the elements of a list in **descending** order

```
numbers_list = [6, 2, 1, 4, 3, 5]
sorted_numbers = sorted(numbers_list, key=lambda x: -x)
# [6, 5, 4, 3, 2, 1]
```

map() Function

- Use it to convert a list of **strings** to a list of **integers**

```
strings_list = ["1", "2", "3", "4"]
numbers_list = list(map(int, strings_list)) # [1, 2, 3, 4]
```

Returns int(x) for each element x in the list

- It **applies a function** to **every item** of an iterable

```
numbers_list = [1, 2, 3, 4]
doubled_list = list(map(lambda x: x*2, numbers_list))
# [2, 4, 6, 8]
```

- It returns an **iterator object**, so you need to convert it **into a list**

filter() Function

- Use it to filter elements that fulfill a given condition

```
numbers_list = [1, 2, 3, 4, 5, 6]
even_numbers = list(filter(lambda x: x % 2 == 0, numbers_list))
# [2, 4, 6]
```

Filter all the even numbers

- The lambda should return either **True** or **False**
- It returns an **iterator object**, so you need to convert it **into a list**

Additional List Manipulations

Swapping List Elements

- You can use the following syntax to swap two or more list elements

```
nums = [1, 2, 3]
nums[0], nums[1], nums[2] = nums[2], nums[0], nums[1]
# 1 swaps with 3
# 2 swaps with 1
# 3 swaps with 2
```

- The **first** element on the **left** swaps with the **first** on the **right**, etc.

Concatenating Lists

- You can use the "+" operator to join two lists

```
nums_list_1 = [1, 2, 3]
nums_list_2 = [4, 5, 6]
final_list = nums_list_1 + nums_list_2
print(final_list) # [1, 2, 3, 4, 5, 6]
```

- Always the second list is added at the end of the first

The set() Function

- You can use the `set()` function to extract only the unique elements from a list

```
numbers = [1, 2, 2, 3, 1, 4, 5, 4]
unique_numbers = list(set(numbers)) # [1, 2, 3, 4, 5]
```

- The `set()` function returns a **set** with the unique values
- You will learn more about **sets** in the advanced python module

The reduce() Function

- The `reduce()` function in Python implements a mathematical technique commonly known as **folding** or **reduction**
  - **Applies** a function (or callable) to the **first two items** in an iterable, generating a **partial result**
  - **Uses** that **partial result**, together with the **third item** in the iterable, to generate another **partial result**
  - **Repeats** the process until the iterable is exhausted, ultimately returning a **single cumulative value**

The reduce() Function

- This function is defined in the "functools" module

```
from functools import reduce

list = [1, 3, 5, 6, 2]

sum = reduce(lambda a, b: a + b, list)  # 17

max = reduce(lambda a, b: a if a > b else b, list)  # 6
```

Using reduce to compute the sum of the list

Using reduce to compute the maximum element from the list

Summary

- We learned:
  - Some additional methods that can be used with lists
  - Some basic lambda functionality
  - How to swap list elements

Trainings @ Software University (SoftUni)

- Software University – High-Quality Education, Profession and Job for Software Developers
  - softuni.bg, softuni.org
- Software University Foundation
  - softuni.foundation
- Software University @ Facebook
  - facebook.com/SoftwareUniversity

License

- This course (slides, examples, demos, exercises, homework, documents, videos and other assets) is **copyrighted content**
- Unauthorized copy, reproduction or use is illegal
- © SoftUni – https://softuni.org
- © Software University – https://softuni.bg