

# Lists Advanced



**SoftUni Team**  
Technical Trainers



**SoftUni**



**Software University**  
<https://softuni.bg>

# Table of Contents

1. List Comprehensions
2. List Methods
3. Advanced Functions
4. Additional List Manipulations
5. The **set()** function



sli.do

**#fund-python**



**[x for x in y]**

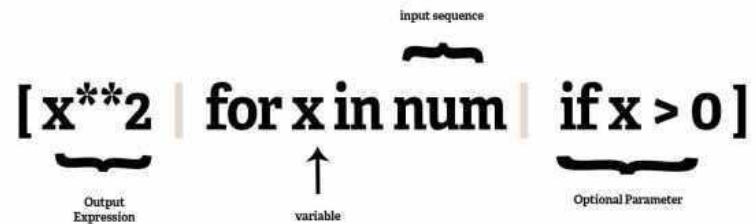
**List Comprehensions**

# 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**



- 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

  
The diagram illustrates the structure of a list comprehension: `[ x**2 | for x in num | if x > 0 ]`. It uses brackets and vertical bars to separate the components. Labels with arrows point to specific parts: 'Output Expression' points to `x**2`, 'variable' points to `x`, 'Input sequence' points to `num`, and 'Optional Parameter' points to `if x > 0`.

# 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 (2)

- 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** statement 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]
```





# Problem: No Vowels

- Write a program that receives a **text** and **removes** all the **vowels** from it
- Print the **new text string** after removing the vowels
- The **vowels** that should be considered are '**a**', '**o**', '**u**', '**e**', '**i**'

ILovePython



LvPythn

# Solution: No Vowels

```
text = input()
vowels = ['a', 'u', 'e', 'i', 'o', 'A', 'U', 'E', 'I', 'O']
no_vowels = ''.join([x for x in text if x not in vowels])
print(no_vowels)
```





**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

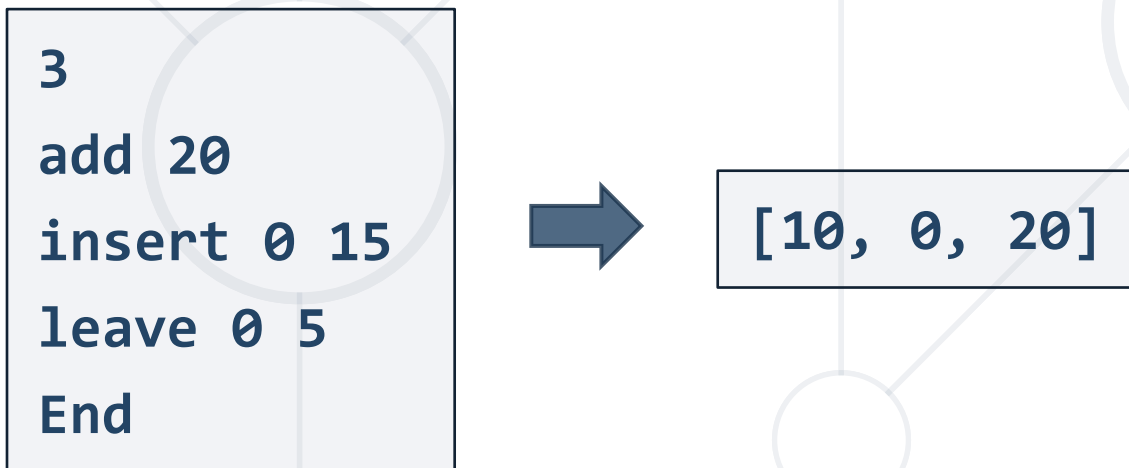
- Using the **remove()** method

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

Removes by value (first occurrence)

# Problem: Trains

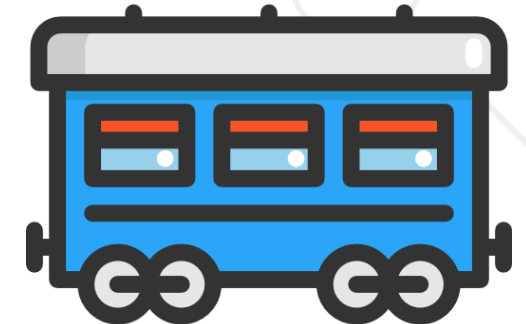
- You will receive how many wagons the train has
- Until you receive "**End**", you will get some of the commands:
  - **add {people}** -> adds the people in the last wagon
  - **insert {index} {people}** -> adds the people at the given wagon
  - **leave {index} {people}** -> removes the people from the wagon



# Solution: Trains

```
train_length = int(input())
train = [0] * train_length
command = input()
while command != "End":
    tokens = command.split(" ")
    key_word = tokens[0]
    if key_word == "add":
        # Implement
        # Add the other cases
    command = input()
print(train)
```

Generate list with  
same values



# Problem: Todo List

- You will be receiving to-do notes until you get the command **"End"**
- The notes will be in the format: **"{priority}-{note}"**
- Return the list of to-do notes sorted by priority (**ascending**)
- Hint: use the **pop()** and the **insert()** methods

2-Walk the dog  
1-Drink coffee  
6-Dinner  
5-Work  
End



['Drink coffee', 'Walk the dog', 'Work', 'Dinner']



```
notes = [0] * 10
```

Creating list with 10 zeros

```
while True:
```

```
    command = input()
```

```
    if command == "End":
```

```
        break
```

```
    tokens = command.split("-")
```

```
    priority = int(tokens[0]) - 1
```

```
    note = tokens[1]
```

```
    notes.pop(priority)
```

```
    notes.insert(priority, note)
```

```
# Add only the elements that are not 0
```

- 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

# Problem: Palindrome Strings

- You will receive words separated by a **single space** and a **palindrome**
- Print a list containing **all** the **palindromes**
- Print the number of **occurrences** of the **palindrome** in the format: "**Found palindrome {number} times**"

```
wow father mom wow shirt stats  
wow
```



```
['wow', 'mom', 'wow', 'stats']  
Found palindrome 2 times
```

# Solution: Palindrome Strings

```
strings = input().split(" ")
searched_palindrome = input()
palindromes = []
for word in strings:
    if word == "".join(reversed(word)):
        palindromes.append(word)
print(f"{palindromes}")
print(f"Found palindrome
{palindromes.count(searched_palindrome)} times")
```

Reversed returns iterator object, so we join it to a string



# **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]
```

# Problem: Sorting Names

- Write a program that reads a single **string** with **names** separated by comma and space ", "
- Sort the names by their **length** in **descending order**
  - If 2 or more names have the **same length**, sort them in **ascending order** (alphabetically)
- Print the resulting list

Ali, Marry, Kim, Teddy, Monika, John



["Monika", "Marry", "Teddy", "John", "Ali", "Kim"]

# map() Function

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

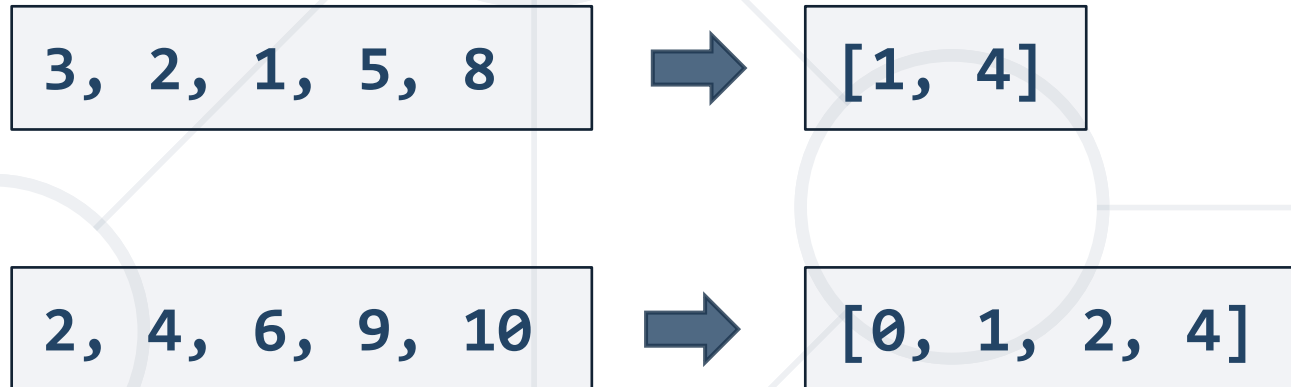
```
numbers_list = [1, 2, 3, 4, 5, 6]
even_numbers = list(filter(lambda x: x % 2 == 0, numbers_list))
# [2, 4, 6]
```

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**

# Problem: Even Numbers

- Write a program that reads a single **string** with **numbers** separated by comma and space ", "
- Print the **indices** of all **even numbers**



# Solution: Even Numbers

```
# Convert the list of strings into a list of numbers
number_list = list(map(int, input().split(", ")))
# Find all the even numbers' indices
found_indices_or_no = map(
    lambda x: x if number_list[x] % 2 == 0 else 'no',
    range(len(number_list)))
# Filter only the indices
even_indices = list(filter(lambda a: a != 'no', found_indices_or_no))
print(even_indices)
```

# Problem: The Office

- Read the problem description [here](#)

1 2 3 4 2 1  
3



Score 2/6. Employees are not happy!

2 3 2 1 3 3  
4



Score: 3/6. Employees are happy!

```
employees = input().split(" ")
happiness_factor = int(input())
employees = # Use map to multiply each element with the factor
filtered = # Use filter to get all the numbers >= than the average
if len(filtered) >= len(employees) / 2:
    print(f"Score: {len(filtered)}/{len(employees)}. Employees are happy!")
else:
    print(f"Score: {len(filtered)}/{len(employees)}. Employees are not happy!")
```



# **Additional List Manipulations**

- 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.

- 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



- 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

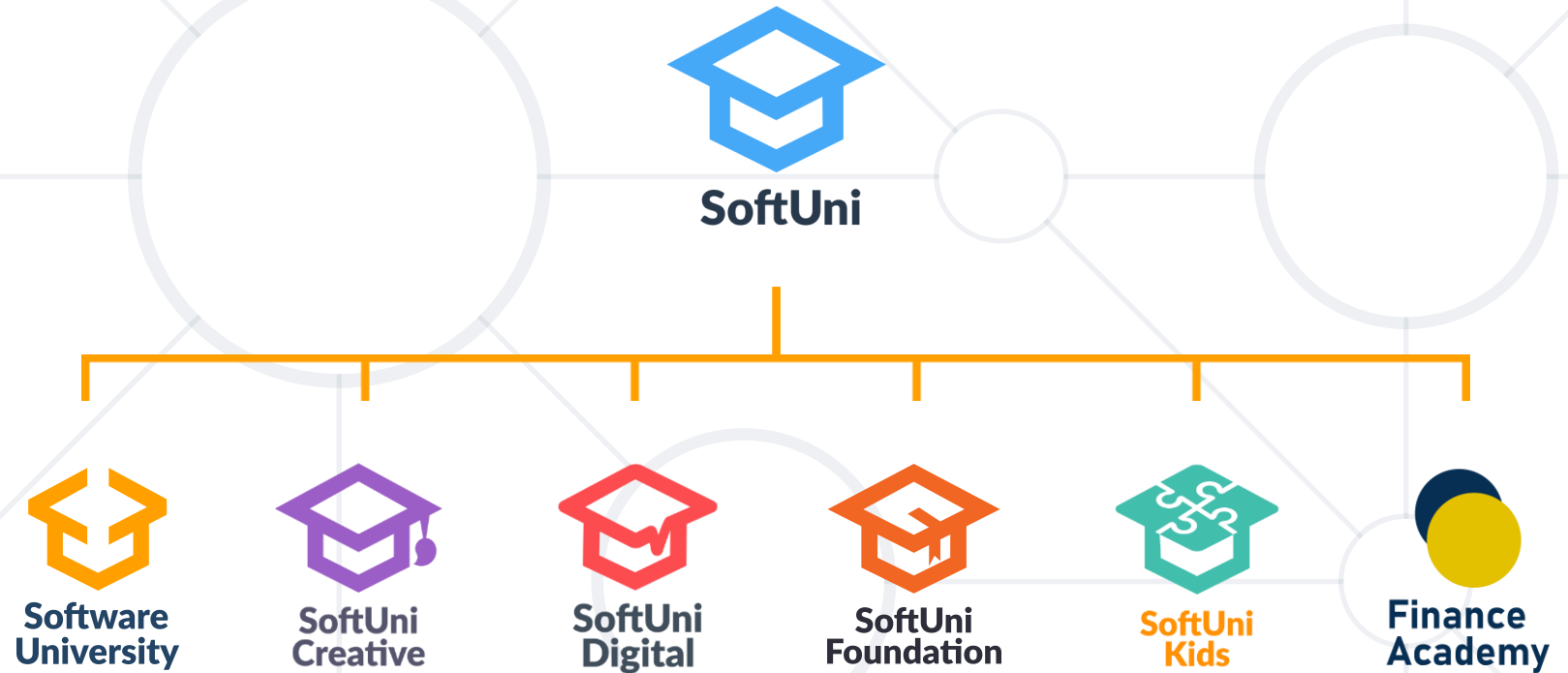


**Live Exercises**

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



# Questions?



# SoftUni Diamond Partners

**SUPER  
HOSTING  
.BG**



**Coca-Cola HBC  
Bulgaria**



**POKERSTARS**  
POKER | CASINO | SPORTS  
a Flutter International brand

**INDEAVR**  
Serving the high achievers



**AMBITIONED**

 **DRAFT  
KINGS**



**SOFTWARE  
GROUP**

createX



**Postbank**

Решения за твоето утре

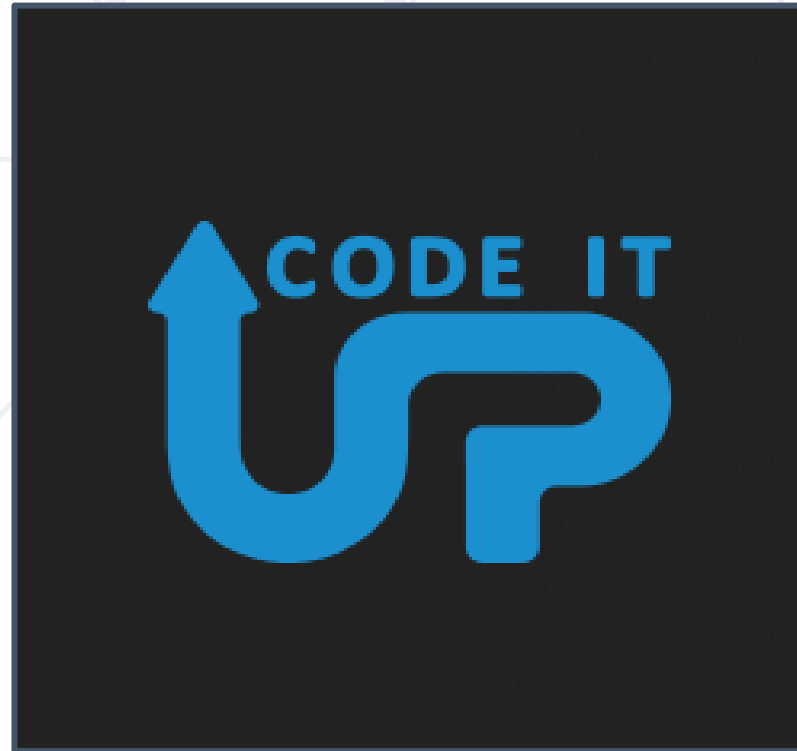


**BOSCH**

**DXC**  
TECHNOLOGY



**SmartIT**



- Software University – High-Quality Education, Profession and Job for Software Developers

- [softuni.bg](http://softuni.bg), [softuni.org](http://softuni.org)

- Software University Foundation

- [softuni.foundation](http://softuni.foundation)

- Software University @ Facebook

- [facebook.com/SoftwareUniversity](https://facebook.com/SoftwareUniversity)

- Software University Forums

- [forum.softuni.bg](http://forum.softuni.bg)



- 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>

