

# Lists Basics



**SoftUni Team**  
Technical Trainers



**SoftUni**



**Software University**

<https://softuni.bg>

# Table of Contents

1. List Definition and Usage
2. Storing Data
3. Creating Lists
4. Accessing Elements
5. List Manipulations
6. Looping through Lists
7. Searching in Lists




[sli.do](https://sli.do)

**#fund-python**



# List Definition and Usage

# Definition

- 
- A **list** is a **collection** which is **index supported** and **changeable (mutable)**
  - It allows duplicate members
  - In Python lists are written with square brackets

```
list_example = ["apple", "banana", "cherry"]
```

List Element

# Usage in Programming

- Lists are very useful for storing **multiple elements**
- They can **expand** and **shrink**
- In Python a single list can store **elements** with **different data types**
- Lists are the **basis** for the other abstract data types like **queues**, **stacks** and their variations





**Storing Data**

# Data in Python Lists

- In Python, a **list** can store data of any data type like:
  - integers
  - floats
  - strings
  - objects
  - other lists
  - mixed data





```
todo_list = ["Do the dishes", "Clean my room"]
```

List of strings

```
favourite_numbers = [7, 21, 65]
```

List of integers

```
random_list = [7, "Peter", 9.99]
```

List of mixed data



# Creating Lists

- Lists in Python can be created by just placing the **sequence** inside the **square brackets**

```
my_list = [1, 2, 3]
```

- Or using the **list** function

```
empty_list = list()
```

- A list may contain duplicate values

```
my_list = [1, 2, 3, 2, 3, 3]
```

# Splitting Strings to List

- You can use the **split** function to split a string and create a list

```
some_text = "a b c d"  
my_list = some_text.split(" ")  
print(my_list) # ['a', 'b', 'c', 'd']
```

separator

- You can split by different separator

```
some_text = "a, b, c, d"  
my_list = some_text.split(", ")  
print(my_list) # ['a', 'b', 'c', 'd']
```

# Joining Lists into a String

- You can create a string from a list using **string.join()**

```
my_list = ["a", "b", "c"]  
print("-".join(my_list)) # a-b-c
```

String separator

- The result of the join function is always a **string**
- **Note**: In python, you can only join a **list of strings**

```
print(" ".join([1, 2, 3])) # error
```

This will not work



**[index]**

**Accessing Elements**

- Use **square brackets** to get an element by an index
- Indices describe the **position** of an element
- We always **start** counting indices from **0**

```
list_of_numbers = [1, 5, 7]
print(list_of_numbers[0]) # 1
print(list_of_numbers[1]) # 5
print(list_of_numbers[2]) # 7
```

# Using the "-" Sign

- In Python you can use the **negative sign** to access an element
- The negative sign will start counting **from** the **end** of the list

```
my_pets = ["cat", "dog", "parrot"]  
print(my_pets[-1]) # parrot  
print(my_pets[-2]) # dog  
print(my_pets[-3]) # cat
```



# Problem: Strange Zoo

- You are at the zoo and the meerkats look strange:
  - You will receive **3 strings**: (tail, body, head)
  - Re-arrange the elements in an array, so that the animal looks normal again: (head, body, tail)

```
my tail  
my body seems on place  
my head is on the wrong end!
```



```
['my head is on the wrong end!',  
'my body seems on place',  
'my tail']
```

# Solution: Strange Zoo

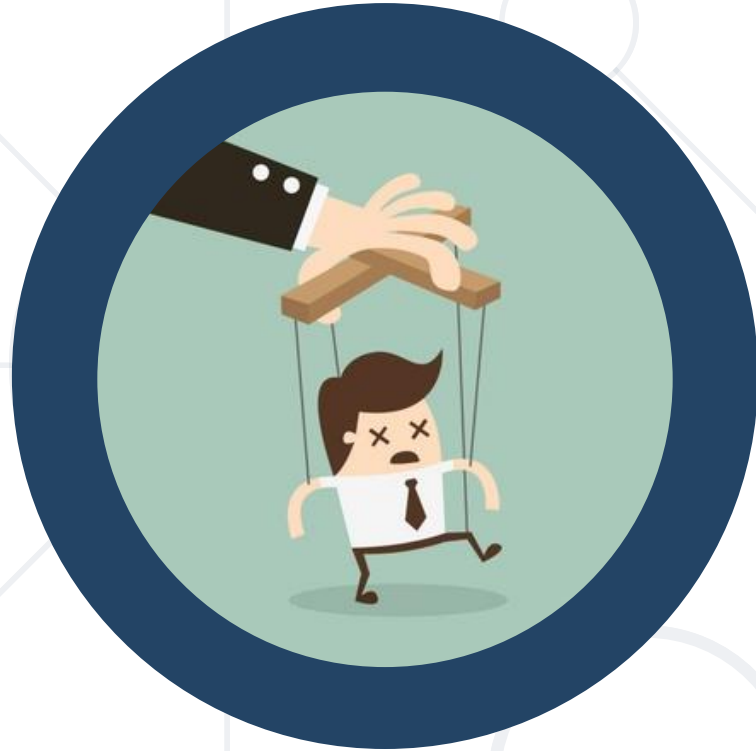
## ■ First variant

```
tail = input()
body = input()
head = input()
meerkat = [head, body, tail]
print(meerkat)
```

## ■ Second variant

```
tail = input()
body = input()
head = input()
meerkat = [tail, body, head]
meerkat[0], meerkat[2] =
meerkat[2], meerkat[0]
print(meerkat)
```

Pythonic way for swapping elements



# **Lists Manipulation**

- Use the **append** function to add a new element

```
empty_list = []  
empty_list.append(2)  
empty_list.append(3)  
print(empty_list)  
# [2, 3]
```

- Use the **remove** function to remove a particular element

```
list_of_numbers = [1, 2, 3, 4, 5]
list_of_numbers.remove(3)
list_of_numbers.remove(1)
print(list_of_numbers)
# [2, 4, 5]
```

- Create a program that reads a single number **n**
  - On the next **n** lines you will receive **names** of courses. You should create a **list of them and print it**

2

PB Python  
PF Python



['PB Python', 'PF Python']

```
n = int(input())
courses = []

for n in range(n):
    current_course = input()
    courses.append(current_course)

print(courses)
```





# Looping Through Lists



- There are **two ways** you can loop through a list using **for** loops

- Iterating over the elements

```
my_list = ["dog", "cat", "fish"]  
for element in my_list:  
    print(element, end=" ") # dog cat fish
```

- Using generated list with **range**

```
for index in range(len(my_list)):  
    print(my_list[index], end=" ") # dog cat fish
```

- You can also use **while** loops to iterate through a list
  - In the first example, we iterate until we reach the end of the list
  - In the second example, we iterate until there are no more elements in the list

```
my_list = ["dog", "cat", "fish"]
i = 0
while i < len(my_list) :
    print(my_list[i], end=" ")
    i += 1
```

```
my_list = ["dog", "cat", "fish"]
while my_list:
    print(my_list[0], end=" ")
    current_element = my_list[0]
    my_list.remove(current_element)
```

# Problem: List Statistics

- You will be given a number **n**
- On the next **n** lines you will receive **integers**
- Create and print **two lists**:
  - One with all the **positives (including 0)** numbers
  - One with all the **negatives** numbers
- Finally print the following:  
"Count of positives: {count\_positives}  
Sum of negatives: {sum\_of\_negatives}"

# Solution: List Statistics

```
n = int(input())

positives = []
negatives = []

for n in range(n):
    # Read the number and add it to the corresponding list

# Print the positive numbers list
# Print the negative numbers list
# Print the statistics
```



**Searching for Elements**

- Use the keyword "**in**" to check if an element is in a list

```
my_list = [1, 2, 3, 4]
if 3 in my_list:
    print("The number 3 is in the list")
```

- Usually the "**in**" keyword is used with **if-else** statements

- The "**not in**" keywords are used to check if an element is **NOT** in a list

```
my_list = [1, 2, 3, 4]
if 5 not in my_list:
    print("The number 5 is not in the list")
```

- The "**not in**" keywords are also mainly used with **if-else** statements

# Problem: Search

- You will receive a number **n** and a **word**
- On the next **n** lines you will be given some **strings**
- **Add** them in a list and **print** them
- Then **filter** out only the strings that **include** the given **word** and **print** the list again



```
n = int(input())
word = input()

strings = []
for i in range(n):
    current_string = input()
    strings.append(current_string)
print(strings)

for i in range(len(strings) - 1, -1, -1):
    if word not in strings[i]:
        strings.remove(strings[i])
print(strings)
```

# Problem: Numbers Filter

- You will receive a single number **n**
- On the next **n** lines you will receive **integers**
- After that you will be given one of the following **commands**:  
even, odd, negative, positive
- Filter all the numbers that **fit** in the category (**0** counts as a **positive and even**)
- Print the result

# Solution: Numbers Filter

```
n = int(input())
numbers = []
filtered = []

for i in range(n):
    current_number = int(input())
    numbers.append(current_number)

command = input()
if command == "even":
    # Add the even numbers to filtered
# Implement the other cases
```

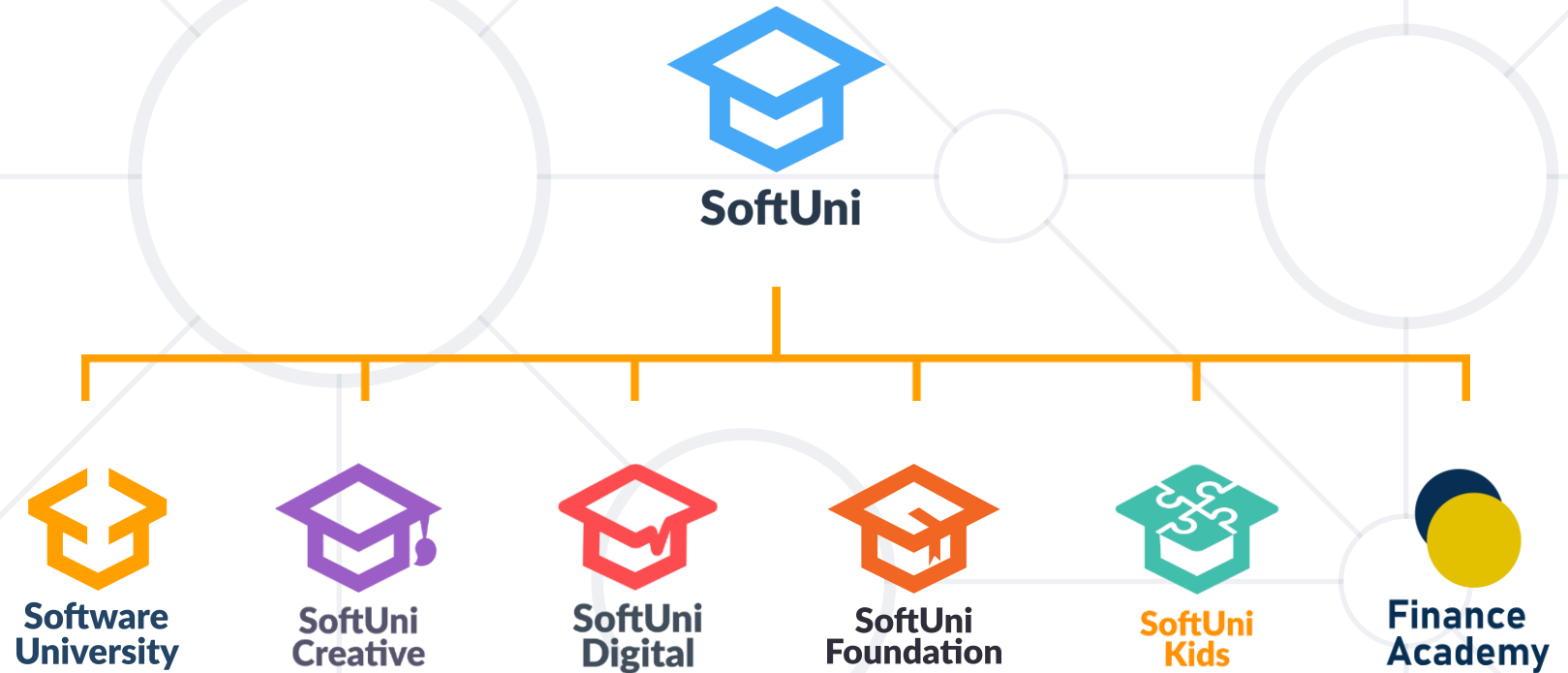


**Live Exercises**

- We learned:
  - What lists are in python
  - How to create lists
  - How to add and remove elements from lists
  - How to loop through lists and access its 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

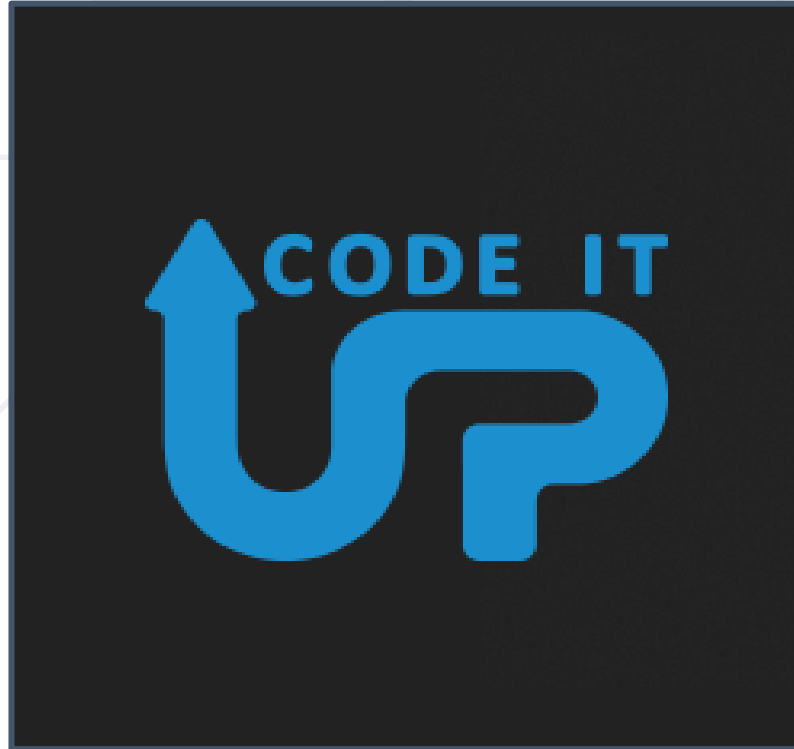


**BOSCH**

**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://about.softuni.bg>
- © Software University – <https://softuni.bg>

