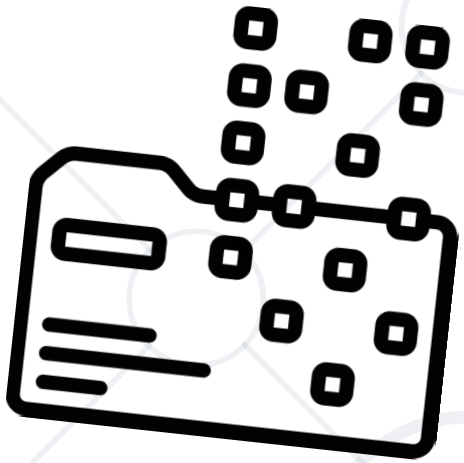


# Tuples and Sets



SoftUni Team  
Technical Trainers



**SoftUni**



Software University

<https://softuni.bg>

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

**#python-advanced**

## 1. Tuples

- Definition
- Usage
- Methods

## 2. Sets

- Definition
- Operators
- Methods





# Tuples

Read-Only Collection

# Definition

- **Tuples** are part of the standard language
- Tuples are **immutable\* objects**
  - \*but the **objects**, inside the tuples, **are mutable**
- Tuples are sequences, just like lists
- Tuples **cannot be changed** unlike lists
- Tuples use **parentheses**, whereas lists use square brackets



# Creating a Tuple

- To create a tuple, place values within brackets

```
t = (1, 2, 3)  
print(t[0]) # 1
```

- You can also use commas

```
t = 1, 2, 3  
print(t) # (1, 2, 3)
```

- Creating tuple with a single element

```
t = (1, )
```

- There are only **two** available tuple methods:
  - **count** – returns the number of times a specified value occurs

```
numbers = (1, 2, 1, 3, 1)  
numbers.count(1) # 3
```

- **index** - returns the index of a particular element

```
names = ("Peter", "George", "George")  
names.index("George") # 1
```

- **Tuple unpacking** allows to extract tuple elements and assign them to elements

```
data = (1, 2, 3)
x, y, z = data
print(x) # 1
print(y) # 2
print(z) # 3
```

The number of elements on the left = length of the tuple



# Problem: Count Same Values

- You will be given numbers separated by a space
- Count the **occurrences** of each value and print it
- Try using the dictionary method **.items()** to iterate over each of them

-2.5 4 3 -2.5 -5.5 4 3 3 -2.5 3



-2.5 - 3 times  
4.0 - 2 times  
3.0 - 4 times  
-5.5 - 1 times

# Solution: Count Same Values

```
numbers = tuple(map(float, input().split()))

nums_and_occurrences = {}
for num in numbers:
    if num not in nums_and_occurrences:
        nums_and_occurrences[num] = 0
    nums_and_occurrences[num] += 1

[print(f"{key} - {value} times") for key, value in
nums_and_occurrences.items()]
```

# Problem: Students' Grades

- You will receive a number (count of input lines: n)
- On the next n-lines you will be given "**{name} {grade}**"
- For each student print all his/her **grades** and finally his/her **average grade**, formatted to the second decimal point

```
4
Vladimir 4.50
Petko 3.00
Vladimir 5.00
Petko 3.66
```



```
Vladimir -> 4.50 5.00 (avg: 4.75)
Petko -> 3.00 3.66 (avg: 3.33)
```

# Solution: Students' Grades

```
count = int(input())
students = {}
for _ in range(count):
    line = tuple(input().split())
    student, grade = line
    if student not in students:
        students[student] = []
    students[student].append(float(grade))
```

*# Print the result*



# Sets

Unique Sequence

# Definition

- Set is an **unordered collection** of items
- Every element of a set is **unique**
- Sets are **mutable**, so we can add or remove items from it
- Sets can be used to perform mathematical **set operations** (union, intersection, symmetric difference, etc.)



```
a = set([1, 2, 3, 4])
b = set([3, 4, 5, 6])
a | b # Union -> {1, 2, 3, 4, 5, 6}
a & b # Intersection -> {3, 4}
a < b # Subset -> False
a > b # Superset -> False
a - b # Difference -> {1, 2}
a ^ b # Symmetric Difference -> {1, 2, 5, 6}
```

You can also use methods instead of symbols


- Each operator is associated to a symbol and a method name

```
a = set([1, 2, 3, 4])  
b = set([3, 4, 5, 6])  
a.union(b)           # Equivalent to a | b  
a.intersection(b)    # Equivalent to a & b  
a.issubset(b)         # Equivalent to a <= b  
a.issuperset(b)       # Equivalent to a >= b  
a.difference(b)       # Equivalent to a - b  
a.symmetric_difference(b) # Equivalent to a ^ b
```



# Set Comprehension

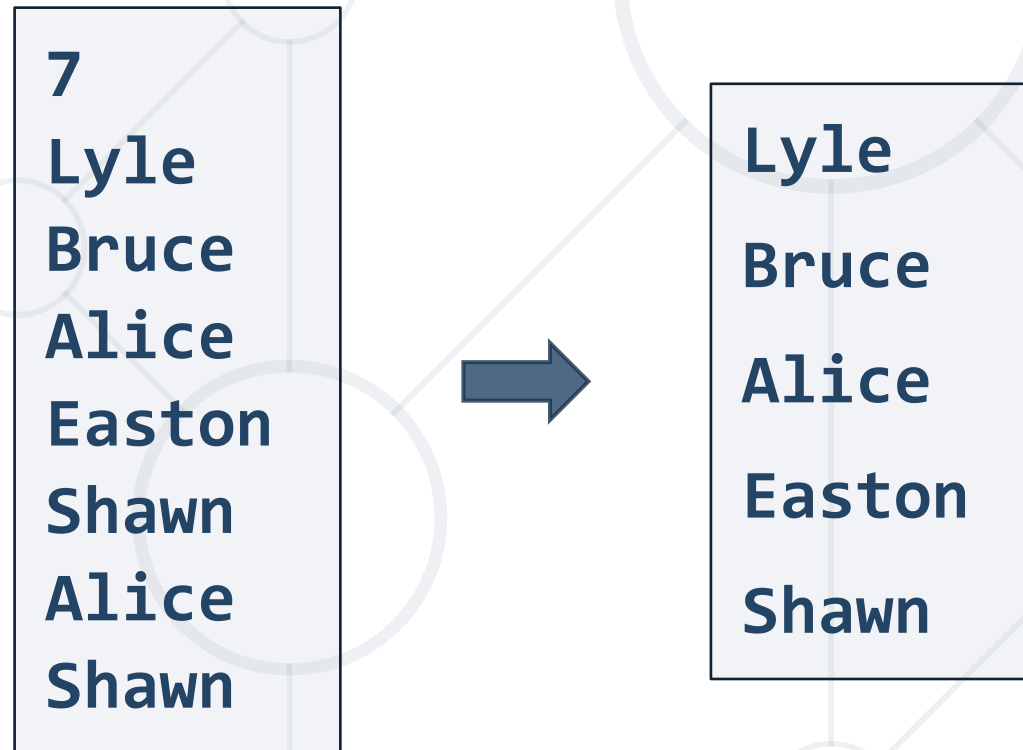
- Set comprehensions are pretty similar to list comprehensions
- The only difference is that set comprehensions use curly brackets `{ }`



```
nums = [1, 2, 3, 4, 4, 5, 6, 2, 1]
unique = {num for num in nums}
# {1, 2, 3, 4, 5, 6}
```

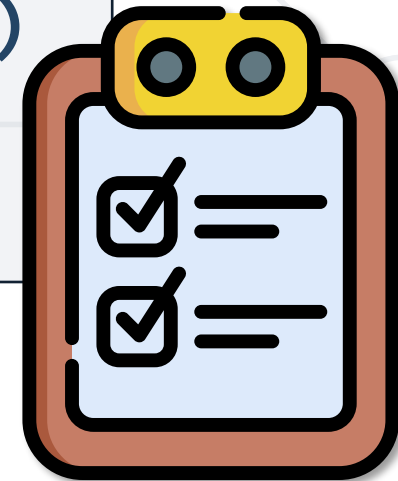
# Problem: Record Unique Names

- You will be given a list and you should print unique items
  - The order does **not** matter



# Solution: Record Unique Names

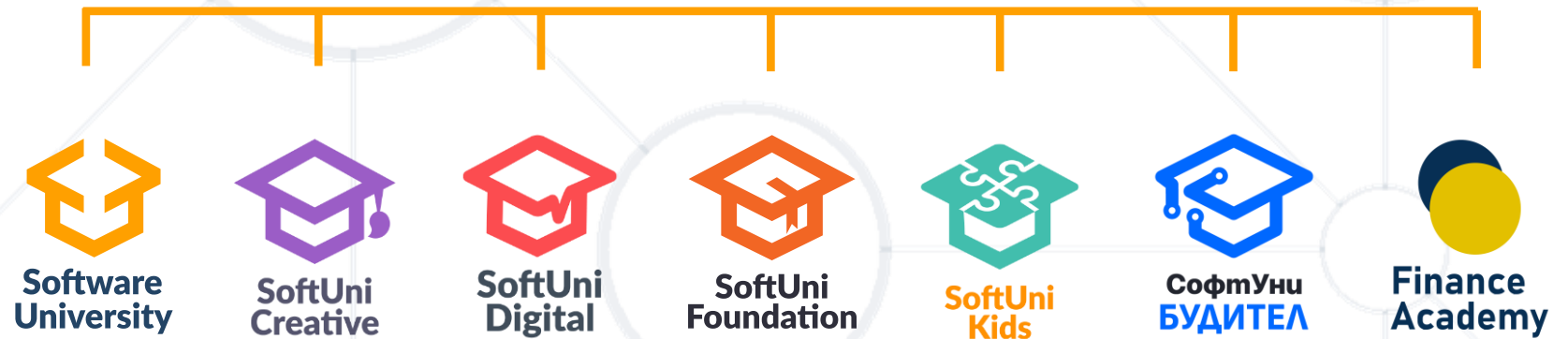
```
n = int(input())
unique_names = set()
for i in range(n):
    unique_names.add(input())
for person in unique_names:
    print(person)
```



- Tuples are **immutable**
- Tuples can hold **nonunique** elements
- Tuples are **ordered** collections
- Sets are **mutable**
- Sets hold **unique** elements
- Sets are **unordered** collections



# Questions?



# SoftUni Diamond Partners



- 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)



- 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>

