# 3 Data Structures - Part 2

Data Science Practicum 2021/22, Lesson 3

Marko Tkalčič

Univerza na Primorskem

## Table of Contents

## Data Collections

- List: ordered and mutable
- Tuple: ordered and immutable
- Set: unordered and immutable and unindexed
- Dictionary: unordered and mutable and indexed

## Table of Contents

## Tuple

- Data collection that is ordered and immutable
- Can have duplicates
- Declared with parenthesis ()

```
my_tuple1 = (1,2,3,4,5)
```

## Tuple

- Data collection that is ordered and immutable
- Can have duplicates
- Declared with parenthesis ()

```
my_tuple1 = (1,2,3,4,5)
```

- working with tuples is similar to working with lists
    - create
    - slice
    - print

## Tuple

- adding elements

```
my_tuple1 = (1,3,5,11)
```

```
#method 1
my_tuple1 = my_tuple1 + (77,)
print(my_tuple1)
```

```
#method 2
my_list = list(my_tuple1)
my_list.append(77)
my_tuple1 = tuple(my_tuple1)
print(my_tuple1)
```

## Tuples counting

```
my_tuple1 = (1,3,5,11,5,4,5)
my_tuple1.count(5)
```

## Table of Contents

## Exercise - Tuple

- Data collection that is ordered and immutable
- Can have duplicates
- Represented with parenthesis ()

## Exercise - Tuple

- Data collection that is ordered and immutable
- Can have duplicates
- Represented with parenthesis ()

- create a tuple w int elements from 1 to 10

## Exercise - Tuple

- Data collection that is ordered and immutable
- Can have duplicates
- Represented with parenthesis ()

- create a tuple w int elements from 1 to 10

```
t1 = (1,2,3,4,5,6,7,8,9,10)
print(t1)
```

## Exercise - Tuple

- print the last three elements
- print the third element

## Exercise - Tuple

- print the last three elements
- print the third element

```
print(t1[-3:])
print(t1[2])
```

## Table of Contents

## Sets

- Data collection that is **unordered, mutable,** and **unindexed**
- No duplicates
- Represented with with curly bracket {}
- Examples:

```
my_set1 = {3.14, 33, 5, 3}
my_set2 = {3.14,”Mon”,3,”Sun”}
my_set3 = {“Mon”,”Tue”,”Wed”,”Thu”,”Fri”,”Sat”,”Sun”}
```

## Sets

- adding

```
my_set2.add(“Off”)
```

- removing

```
my_set2.remove(“Off”)
```

- intersection of sets

```
my_set3 = my_set1 & my_set2
```

- union of two sets

```
union_set = my_set1 | my_set2
```

## Table of Contents

## Exercise - Set

- create a set with five names of people

## Exercise - Set

- create a set with five names of people

```
names = {"Jim", "Anna", "Robert", "Alice", "John"}
```

## Exercise

- using the `input()` function add a user-generated name to the set

## Exercise

- using the `input()` function add a user-generated name to the set

```
new_name = input()
names.add(new_name)
print(names)
```

```
Marko
{'Alice', 'Robert', 'Anna', 'Jim', 'Marko', 'John'}
```

- why the new name is not appended (at the end)?

## Exercise

- using the `input()` function remove a name from the set

## Exercise

- using the input() function remove a name from the set

```
to_remove = input()
names.remove(to_remove)
print(names)
```

- in sets we don’t work with indeces
- what happens if the name is not in the set?

## Exercise

- using the `input()` function check if a name is the set

## Exercise

- using the input() function check if a name is the set

```
name = input()
if name in names:
    print("yes")
else:
    print("no")
```

## Table of Contents

## Dictionary

- Data collection that is unordered, mutable, and indexed
    - *As of Python version 3.7, dictionaries are ordered. In Python 3.6 and earlier, dictionaries are unordered.*
- Can have duplicates as values, not as keys
- Represented with curly brackets and specifying the **key-value** pairs `{key:value}`

Example:

```
my_dict =
{“key1”:“Mon”,“key2”:”Tue”,“key3”:”Wed”,
 “key4”:”Thu”,“key5”:”Fri”,“key6”:”Sat”,
 “key7”:”Sun”}
```

## Dictionary operations

- access an element a dictionary

```
my_dict[“key3”]
```

- get keys of a dictionary

```
my_dict.keys()
```

- check if a key exists in a dictionary:

```
“key1” in my_dict
```

- add to a dictionary

```
my_dict.update({“key8”:”Off”})
```

- delete from a dictionary

```
del(my_dict[“key8”])
```

## Table of Contents

## Exercise

- create the dictionary

```
my_dict = {"key1":"Mon","key2":"Tue","key3":"Wed","key4":"Thu","key5":"Fri","key6":"Sat","key7":"Sun"}
```

## Exercise

- get the value of the key key4

## Exercise

- get the value of the key key4

```
my_dict["key4"]
```

## Exercise

- check if the key `key8` exists in the dictionary

## Exercise

- check if the key key8 exists in the dictionary

```
if "key8" in my_dict:
    print("yes")
else:
    print("no")
```

## Exercise

- add a new key-value pair to the dictionary

## Exercise

- add a new key-value pair to the dictionary

```
my_dict.update({"filename":"test.txt"})
print(my_dict)
```

## Exercise

- add a new key-value pair to the dictionary

```
my_dict.update({"filename":"test.txt"})
print(my_dict)
```

- remove the key-value pair where the key is key4

## Exercise

- add a new key-value pair to the dictionary

```
my_dict.update({"filename":"test.txt"})
print(my_dict)
```

- remove the key-value pair where the key is key4

```
del(my_dict["key4"])
print(my_dict)
```

## Table of Contents

## Collections summary

| Collection | Creation | Duplicates | Ordered | Mutable | Scenario |
|---|---|---|---|---|---|
| List | [] | Yes | Yes | Yes (add, remove elements) | when you want to store similar elements `groceries=['bread','butter','cheese']` |
| Tuple | () | Yes | Yes | No (the tuple items can not be deleted by using the del keyword, you can delete the whole tuple) | Use a tuple when you know what information goes in the container that it is. For example, when you want to store a person's credentials for your website. `person=('ABC','admin','12345')` |
| Set | {} | No | Yes | No | Is the element X in the collection? |
| Dictionary | {:} | Keys no, values yes | Yes/No | duplicates not allowed, key immutable, value mutable | address-book search by key |

## Collections summary

| Collection | Creation | Duplicates | Ordered | Mutable | Scenario |
|---|---|---|---|---|---|
| List | [] | Yes | Yes | Yes (add, remove elements) | when you want to store similar elements `groceries=['bread','butter','cheese']` |
| Tuple | () | Yes | Yes | No (the tuple items can not be deleted by using the del keyword, you can delete the whole tuple) | Use a tuple when you know what information goes in the container that it is. For example, when you want to store a person's credentials for your website. `person=('ABC','admin','12345')` |
| Set | {} | No | Yes | No | Is the element X in the collection? |
| Dictionary | {:} | Keys no, values yes | Yes/No | duplicates not allowed, key immutable, value mutable | address-book search by key |

**Tuples have structure, lists have order.**

## References

Part of the material has been taken from the following sources. The usage of the referenced copyrighted work is in line with fair use since it is for nonprofit educational purposes.

- https://stackoverflow.com/questions/6200910/relationship-between-scipy-and-numpy
- https://www.geeksforgeeks.org/difference-between-list-vs-set-vs-tuple-in-python/
- https://medium.com/@paulrohan/python-list-vs-tuple-vs-dictionary-4a48655c7934
- https://stackoverflow.com/questions/626759/whats-the-difference-between-lists-and-tuples/626871#626871