Contents

[Table of Figures 1](#_Toc115950589)

[Lists 2](#_Toc115950590)

[Other Methods 4](#_Toc115950591)

[Code 4](#_Toc115950592)

[Tuples 7](#_Toc115950593)

[Dictionaries 8](#_Toc115950594)

[my\_dictionary.keys() 8](#_Toc115950595)

[my\_dictionary.values() 9](#_Toc115950596)

[my\_dictionary.items() 9](#_Toc115950597)

[Code 10](#_Toc115950598)

[Sets 10](#_Toc115950599)

[References 12](#_Toc115950600)

# Table of Figures

[Figure 1- python\_Lists1 – Produce lists 2](#_Toc115946869)

[Figure 2 - python\_Lists2 – Concattenate Lists 2](#_Toc115946870)

[Figure 3 - python\_Lists3 – Concatenate Lists (nested) 3](#_Toc115946871)

[Figure 4 - python\_Lists4 – Example of mutable list 3](#_Toc115946872)

[Figure 5 - python\_Lists5 –Appending Lists 3](#_Toc115946873)

[Figure 6 - Experimenting with other methods - python\_Lists\_other\_methods.py 4](#_Toc115946874)

[Figure 7 - python\_Tuple1.py 7](#_Toc115946875)

[Figure 8 - Value error changing an index (tuple) 7](#_Toc115946876)

[Figure 9 - Results of python\_dict1.py 8](#_Toc115946877)

# 

# Lists

An organised series of objects is a list. In contrast to arrays, list components can be of any type, including texts and numbers. The distinction between lists and arrays is that sequences come in various sizes and types (Minecakir 2020). Lists are mutable and can change, unlike strings. Square brackets define lists.

Graphical user interface, text, application

Description automatically generated

Figure 1- python\_Lists1 – Produce lists

A screenshot of a computer

Description automatically generated

Figure 2 - python\_Lists2 – Concattenate Lists

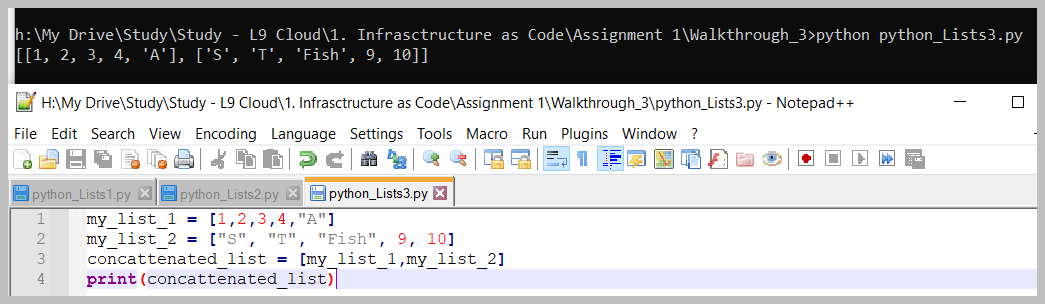


Figure 3 - python\_Lists3 – Concatenate Lists (nested)

A screenshot of a computer

Description automatically generated

Figure 4 - python\_Lists4 – Example of mutable list

Graphical user interface, text, application

Description automatically generated

Figure 5 - python\_Lists5 –Appending Lists

## Other Methods

Below it is shown in figure 6 that I have experimented with other methods such as append, remove, reverse and clear.

Graphical user interface, text, application

Description automatically generated

Figure 6 - Experimenting with other methods - python\_Lists\_other\_methods.py

## Code

my\_list = [1,2,3,4,"A"]

a = len(my\_list)

print(a)

slice\_1=my\_list[1:3:1]

print(slice\_1)

my\_character = my\_list[-1]

print(my\_character)

my\_list2 = [1,2,3,4,"A"]

b = len(my\_list2)

print(b)

slice\_2=my\_list2[1:2:2]

print(slice\_2)

my\_character2 = my\_list2[-1]

print(my\_character2)

\_\_\_\_\_

my\_list\_1 = [1,2,3,4,"A"]

my\_list\_2 = ["S", "T", "Fish", 9, 10]

concattenated\_list = my\_list\_1 + my\_list\_2

print(concattenated\_list)

\_\_\_\_\_

my\_list\_1 = [1,2,3,4,"A"]

my\_list\_2 = ["S", "T", "Fish", 9, 10]

concattenated\_list = [my\_list\_1,my\_list\_2]

print(concattenated\_list)

\_\_\_\_\_

my\_list\_1 = ["S","T","Fish",9,10]

print(my\_list\_1)

my\_list\_1[2] = "Chips"

print(my\_list\_1)

my\_list\_1[3] = "9.5"

print(my\_list\_1)

\_\_\_\_\_

my\_list = ["One","Two","Three"]

print(my\_list)

my\_list.append("Four")

print(my\_list)

\_\_\_\_\_

my\_list = ["One","Two","Three"]

print(my\_list)

my\_list.append("Four")

print(my\_list)

#Remove

my\_list.remove ("Two")

print(my\_list)

#Append

my\_list.append ("Nine Hundred and thirty six")

print(my\_list)

#Reverse

my\_list.reverse()

print(my\_list)

#Clear

my\_list.clear()

print(my\_list)

# Tuples

Tuples are like lists but immutable and defined by regular brackets. Figure 7 below shows how Pythom differentiates them. Tuples are perfect for ensuring the integrity of data.

Graphical user interface, text, application

Description automatically generated

Figure 7 - python\_Tuple1.py

If it attempts to change the value of the tuiple using its index, it will display a value error. Figure 8 below displays this error.

Graphical user interface, text, application

Description automatically generated

Figure 8 - Value error changing an index (tuple)

# Dictionaries

Dictionaries are unordered structures for storing objects. A dictionary, another composite data type offered by Python, is a collection of items comparable to a list. The following traits apply to both dictionaries and lists:

* Both can change.
* Both have movement. They can expand and contract as necessary.
* Both are nestable. One list may include another list. Another dictionary may be found inside one. A list can also be found in a dictionary, and vice versa.

Lists and dictionaries differ principally in the way that elements are accessed:

* Indexing allows access to list elements based on their location inside the list.
* Keys are used to access dictionary components (Sturtz n.d).

Adding a dictionary requires the use of curly brackets. Adding extra entries or editing entries requires the use of square brackets. A directory can hold any object.

Graphical user interface, text, application

Description automatically generated

Figure 9 - Results of python\_dict1.py

### my\_dictionary.keys()

The use of the my\_dictionary.keys() code will extract just the keys and list them in the output.

Graphical user interface, text, application

Description automatically generated

Figure 10 - Extracting just the keys

### my\_dictionary.values()

The use of the my\_dictionary.values() code will extract just the keys and list them in the output.

Graphical user interface, text, application

Description automatically generated

Figure 11 - Extracting the values

### my\_dictionary.items()

In Python Dictionary, the items () method returns the list with all dictionary keys with values.

Graphical user interface, application

Description automatically generated

Figure 12 - Returning all items using items

## Code

my\_dictionary = {"Fname":"John", "SName":"ORaw","Occupation":"Rocket Scientist"}

print(my\_dictionary)

print("Works as a " + my\_dictionary ["Occupation"])

**#Adding a new Entry to the dictionary**

my\_dictionary["Salary"] = "Not Enough!"

print(my\_dictionary)

**#editing an entry**

my\_dictionary["Occupation"] = "Professor"

print(my\_dictionary)

**#print just the keys - Extracting the Keys**

print(my\_dictionary.keys())

**#Extracting the values**

print(my\_dictionary.values())

**#Returning all values**

print(my\_dictionary.items())

# Sets

Some of the features of sets in Python are that duplicates are not allowed, items in a set are unordered, and the value of items in a set can not be modified once the set is created (Abba 2022).

Graphical user interface, text, application

Description automatically generated

Figure 13 - Sets

## Code

my\_set=set()

print(type(my\_set))

print(my\_set)

my\_set.add(1)

my\_set.add(2)

my\_set.add(3)

my\_set.add(2)

print(my\_set)

# References

Abba, I.V. (2022) Python Set – How to Create Sets in Python [online], *freeCodeCamp.org*, available: https://www.freecodecamp.org/news/python-set-how-to-create-sets-in-python/ [accessed 6 Oct 2022].

Minecakir (2020) List Methods in Python [online], *Medium*, available: https://python.plainenglish.io/list-methods-in-python-236c4fa285a0 [accessed 4 Oct 2022].

Sturtz, J. (n.d) Dictionaries in Python – Real Python [online], available: https://realpython.com/python-dicts/ [accessed 5 Oct 2022].