# Python Comprehensions

Review the topic of comprehensions and slicing in Python with this e-book. Check out the table of contents to navigate to each topic.

### **Contents**

- 1: List comprehensions
- 2: Set and dictionary comprehensions
- 4: List slicing

# List comprehension

Used to create a list without explicitly appending objects or using a separate loop.

```
new_collection = [element for element in collection]
```

The element variable takes the value of each element inside the collection.

The value is then appended to the <a href="new\_collection">new\_collection</a> until the whole <a href="collection">collection</a> is iterated over.

# List comprehension with conditionals

Conditionals can be added to list comprehensions to make them more versatile.

```
Creating a new list only with even numbers

numbers = [i for i in range(5) if i % 2 == 0]

print(numbers)
```

### Set comprehension

Uses the same syntax as a list comprehension.

The only difference is that curly brackets are used instead of square brackets.

```
collection = [1, 2, 2, 3, 4, 3, 5]
numbers = {i for i in collection}
print(numbers)
```

# Dictionary comprehension

Used to create a dictionary using pairs of values from other collections.

```
dictionary = {
    collection_1[i] : collection_2[i]
    for i in collection
}
Key of a new dictionary
Value of a new dictionary
```

#### **Extra Resources**

- 1. List comprehensions
- 2. Set comprehensions

# Example

```
friends = ["Rolf", "Dane", "Josh"]
numbers = [1,2,3]
dictionary = {
    friends[i] : numbers[i]
    for i in range(3)
}
print(dictionary)
```

### Output

```
{
    "Rolf": 1,
    "Dane": 2,
    "Josh": 3
}
```

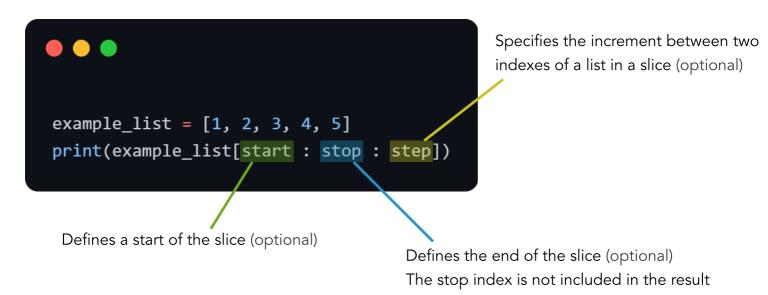
#### **Extra Resources**

1. Dictionary comprehensions

## List slicing

Used to extract a part of the list.

### Example



We are going to analyze different outcomes of the example code in the table below it

Slice used in example	Result	Explanation
example_list[0:4]	1, 2, 3, 4	Start at index 0, end at stop-1 with step=1
example_list[0:4:2]	1, 3	From index 0 to 3, skipping indexes 1 and 3
example_list[:4:2]	1, 3	Default start=0. End at stop-1 with step=2
example_list[0::]	1, 2, 3, 4, 5	Start=0. Default stop=len(list) with step=1
example_list[-4:-1]	2, 3, 4	Go from 4 to 1, counting from end of list
example_list[4:1:-1]	5,4,3	From 4 to 1+1, negative step goes backwards
example_list[::]	1, 2, 3, 4, 5	All defaults: start=0, stop=len(list), step=1

Negative indexes count from the end of the list, starting at -1 and ending at -len(list).

They are calculated with: <a href="index - len(list">index - len(list)</a>.