Workshop: Section 4 - Introduction to Python: Part 2

Philipp Chapkovski
University of Bonn
chapkovski@uni-bonn.de

September 11th - 12th, 2023

Lists and Dictionaries

- **Lists**: Ordered collections of elements

```
my_list = [1, 2, 3, "apple"]
print(my_list[0])  # Output: 1
```

- **Dictionaries**: Key-value pairs

```
my_dict = {"key": "value", "name": "John", "age": 30}
print(my_dict["name"])  # Output: John
```

Operations with Lists (Expanded)

- Appending: `my_list.append(4)`
- Removing: `my_list.remove("apple")`
- Sorting: `my_list.sort()`
- Index: `index = my_list.index(4)`
- Slicing: `first_two = my_list[:2]`, `last_two = my_list[-2:]`
- Making copies: `copy_list = my_list.copy()`

Operations with Dictionaries

- **Adding or Updating Elements**:

```
my_dict["new_key"] = "new_value"
```

- **Deleting Elements**:

```
del my_dict["key"]
```

- **Getting Elements**:

```
value = my_dict.get("key", "default_value")
```

- **Keys, Items, and Values**:

```
keys = my_dict.keys()
items = my_dict.items()
values = my_dict.values()
```

List Comprehensions

- **Basic Syntax**: `[expression for item in iterable]`

```
squares = [x*x for x in range(5)]
```

- **With Conditionals**: `[expression for item in iterable if condition]`

```
even_squares = [x*x for x in range(5) if x % 2 == 0]
```

Functions in Python (Part 1)

Function Definition

- Defining a function using `def` keyword

```
def greet(name):
    return f"Hello, {name}!"
```

Calling Functions

- How to call a function and store its return value

```
message = greet("John")
```

- Nested function calls

```
result = add(multiply(2, 3), 4)   # add(6, 4)
```

Functions in Python (Part 2)

Function Parameters

- Positional and keyword arguments

```
def add(a, b):
    return a + b
result = add(b=2, a=3)  # Keyword arguments
```

- Setting default parameter values

```
def add(a, b=0):
    return a + b
```

- Using `return` to send back a result

```
def multiply(a, b):
    return a * b
```

Imports in Python

Importing Modules: - *How to import standard Python modules*

```
import math
```

Importing Functions: - *Import specific functions from a module*

```
from math import sqrt
```

Aliasing Modules: - *Giving a module a short alias*

```
import numpy as np
```

Importing from Custom Modules: - *Importing user-defined modules*

```
from my_module import my_function
```

Mutability and Immutability: Part 1

Mutable Types

```
my_list = [1, 2, 3]
my_list[0] = 4   # Allowed
```

Special Focus: Lists

```
original_list = [1, 2, 3]
copy_list = original_list
copy_list[0] = 4   # original_list is also changed
```

Mutability and Immutability: Part 2

Immutable Types

```
my_tuple = (1, 2, 3)
my_tuple[0] = 4  # Raises an error
```

When to Use

- Use mutable types for data that needs to change
- Use immutable types for data that should remain constant

Try-Except Structure: Part 1

Basic Example

```
try:
    x = 1 / 0
except ZeroDivisionError:
    x = 0
```

Multiple Exceptions

```
try:
    # some code
except (ZeroDivisionError, ValueError):
    # handle exception
```

Try-Except Structure: Part 2

General Capture

```
try:
    # some code
except Exception as e:
    print("An error occurred:", e)
```

Finally Block

```
try:
    # some code
except ZeroDivisionError:
    # handle exception
finally:
    # this code runs no matter what
```

Exercise 1: Operations with Lists

Create a list of integers and use list comprehension to create a new list that contains only even numbers.

```
# Sample List: [1, 2, 3, 4, 5]
# Expected Output: [2, 4]
```

Exercise 1 - Answer

```
even_list = [x for x in [1, 2, 3, 4, 5] if x % 2 == 0]
```

Exercise 2: Operations with Dictionaries

Create a dictionary and change the value of an existing key.

```
# Sample Dictionary: {'name': 'John', 'age': 30}
# Change 'name' to 'Jane'
```

Exercise 2 - Answer

```
my_dict = {'name': 'John', 'age': 30}
my_dict['name'] = 'Jane'
```

Exercise 3: Simple Function

Write a function that takes a list as an argument and returns the sum of all elements in the list.

```
# Sample List: [1, 2, 3]
# Expected Output: 6
```

Exercise 3 - Answer

```
def sum_list(my_list):
    return sum(my_list)
```