Python for Data Analysis

Built-in Data Structures, Functions, and Class (Python Basics / Week 3)

Atefeh Khazaei
atefeh.khazaei@port.ac.uk

What we will learn this week?

- ❑ Data Structures and Sequences
- ❑ Functions
- ❑ Class in Python

Data Structures and Sequences

- Python’s data structures are simple but powerful.
- Mastering their use is a critical part of becoming a proficient Python programmer.

Data Structures and Sequences (cont.)
Tuple

- A tuple is a fixed-length, immutable sequence of Python objects.
- The easiest way to create one is with a comma-separated sequence of values:

```
tup = 3, 4, 5
print(tup)
```

```
(3, 4, 5)
```

- When you’re defining tuples in more complicated expressions, it’s often necessary to enclose the values in parentheses,
    - Creating a tuple of tuples:

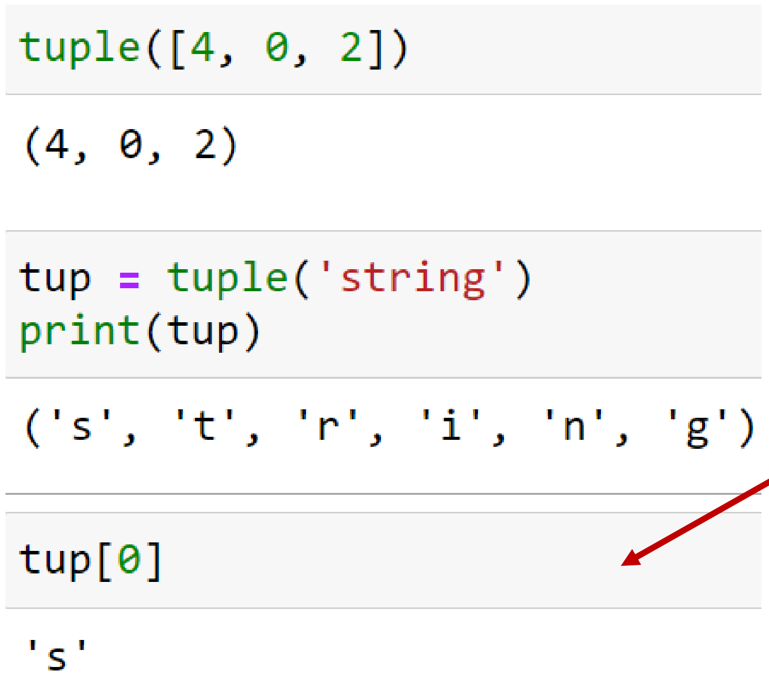
```
nested_tup = (4, 5, 6), (7, 8)
print(nested_tup)
```

```
((4, 5, 6), (7, 8))
```

Data Structures and Sequences (cont.)

Tuple

- We can convert any sequence or iterator to a tuple by invoking tuple:



```
tuple([4, 0, 2])
```

```
(4, 0, 2)
```

```
tup = tuple('string')
print(tup)
```

```
('s', 't', 'r', 'i', 'n', 'g')
```

```
tup[0]
```

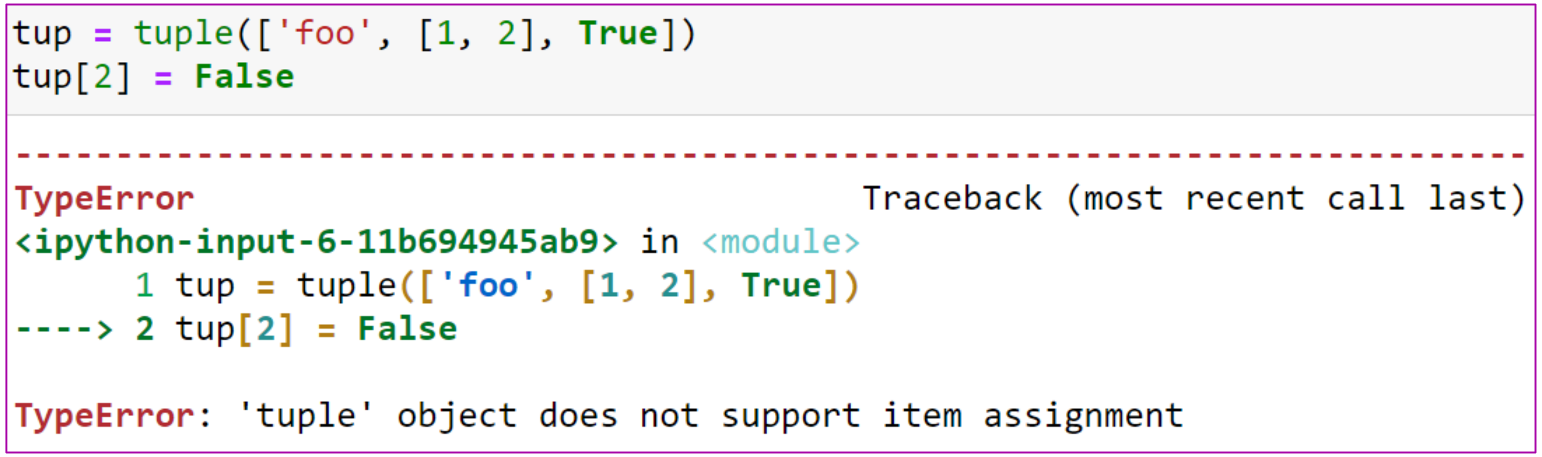
```
's'
```

- Elements can be accessed with square brackets [] as with most other sequence types.
- As in C, C++, Java, and many other languages, sequences are 0-indexed in Python.

Data Structures and Sequences (cont.)

Tuple

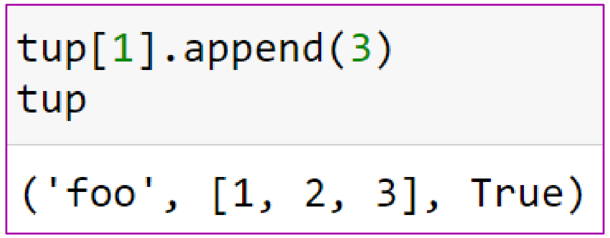
- While the objects stored in a tuple may be mutable themselves, once the tuple is created it’s not possible to modify which object is stored in each slot.



```
tup = tuple(['foo', [1, 2], True])
tup[2] = False
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-6-11b694945ab9> in <module>
      1 tup = tuple(['foo', [1, 2], True])
----> 2 tup[2] = False

TypeError: 'tuple' object does not support item assignment
```

- If an object inside a tuple is mutable, such as a list, you can modify it in-place.



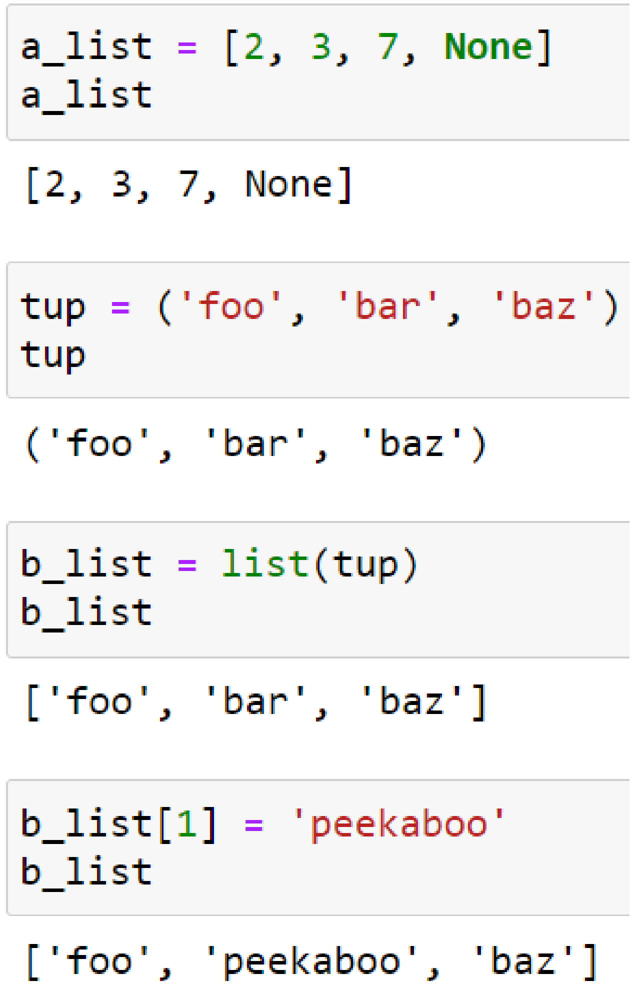
```
tup[1].append(3)
tup
```

```
('foo', [1, 2, 3], True)
```

Data Structures and Sequences (cont.)

List

- In contrast with tuples, lists are variable-length and their contents can be modified in-place.
- We can define them using square brackets [] or using the list type function.



```
a_list = [2, 3, 7, None]
a_list
```

```
[2, 3, 7, None]
```

```
tup = ('foo', 'bar', 'baz')
tup
```

```
('foo', 'bar', 'baz')
```

```
b_list = list(tup)
b_list
```

```
['foo', 'bar', 'baz']
```

```
b_list[1] = 'peekaboo'
b_list
```

```
['foo', 'peekaboo', 'baz']
```

Data Structures and Sequences (cont.)

List

- Lists and tuples are semantically similar (though tuples cannot be modified) and can be used interchangeably in many functions.

- You will see more functions in the practical sheet:
  - *insert()*
  - *extend()*
  - *append()*
  - *remove()*
  - *sort()*

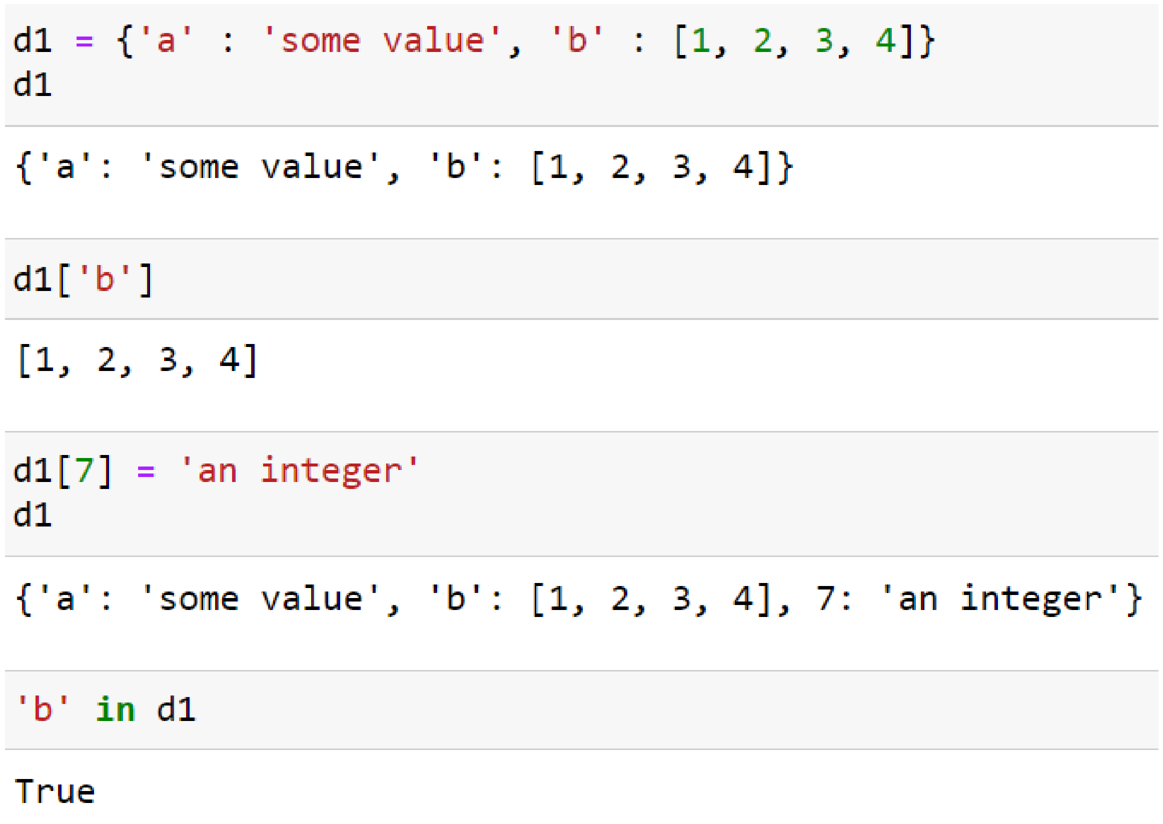
Data Structures and Sequences (cont.)
Dictionary (Dict)

- **dict** is likely the most important built-in Python data structure.
- A more common name for it is **hash map** or **associative array**.
- It is a flexibly sized collection of key-value pairs, where **key** and **value** are Python objects.

Data Structures and Sequences (cont.)

Dictionary (Dict)

- ❑ One approach for creating one Dict is to use curly braces {} and colons to separate keys and values:



```
d1 = {'a' : 'some value', 'b' : [1, 2, 3, 4]}
d1
```

```
{'a': 'some value', 'b': [1, 2, 3, 4]}
```

```
d1['b']
```

```
[1, 2, 3, 4]
```

```
d1[7] = 'an integer'
d1
```

```
{'a': 'some value', 'b': [1, 2, 3, 4], 7: 'an integer'}
```

```
'b' in d1
```

```
True
```

Functions

- ❑ Functions are the primary and most important method of code organization and reuse in Python.
- ❑ If you anticipate needing to repeat the same or very similar code more than once, it may be worth writing a reusable function.
- ❑ Functions can also help make your code more readable by giving a name to a group of Python statements.

Functions (cont.)

- Each function can have **positional arguments** and **keyword arguments**.
- Keyword arguments are most commonly used to specify default values or optional arguments.

Keyword

```python
def my_function(x, y, z=1.5):
    if z > 1:
        return z * (x + y)
    else:
        return z / (x + y)
```

x, y: positional arguments
z: Keyword arguments

- Keyword
- Multiple return statements
- Without encountering a return statement,
  None is returned automatically

Call the function:

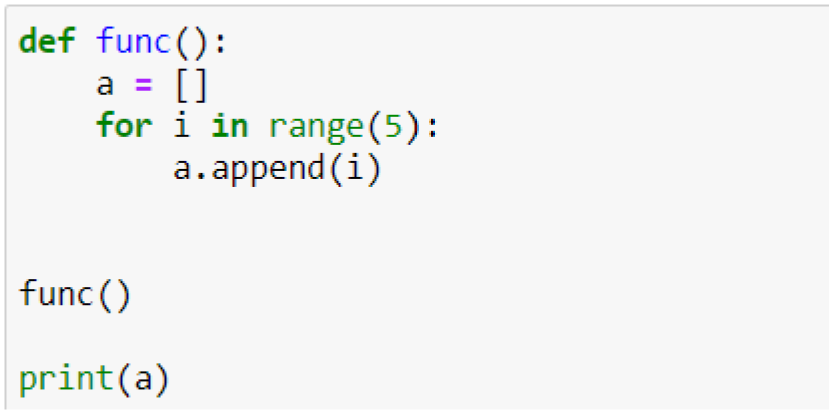
```python
my_function(5, 6, z=0.7)
my_function(3.14, 7, 3.5)
my_function(10, 20)
```

Functions (cont.)
Namespaces, Scope, and Local Functions

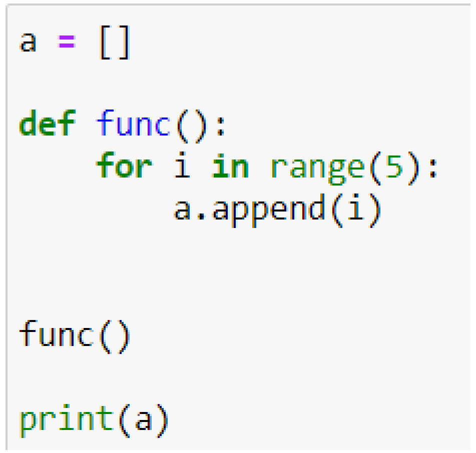
- Functions can access variables in two different scopes: **global** and **local**.
- An alternative and more descriptive name describing a variable scope in Python is a **namespace**.
- Any variables that are assigned within a function by default are assigned to the local namespace.
  - The local namespace is created when the function is called and immediately populated by the function’s arguments.
  - After the function is finished, the local namespace is destroyed.

Functions (cont.)

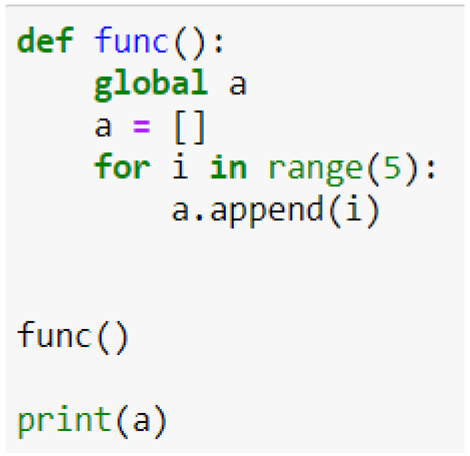
Namespaces, Scope, and Local Functions



```python
def func():
    a = []
    for i in range(5):
        a.append(i)


func()

print(a)
```



```python
a = []

def func():
    for i in range(5):
        a.append(i)


func()

print(a)
```

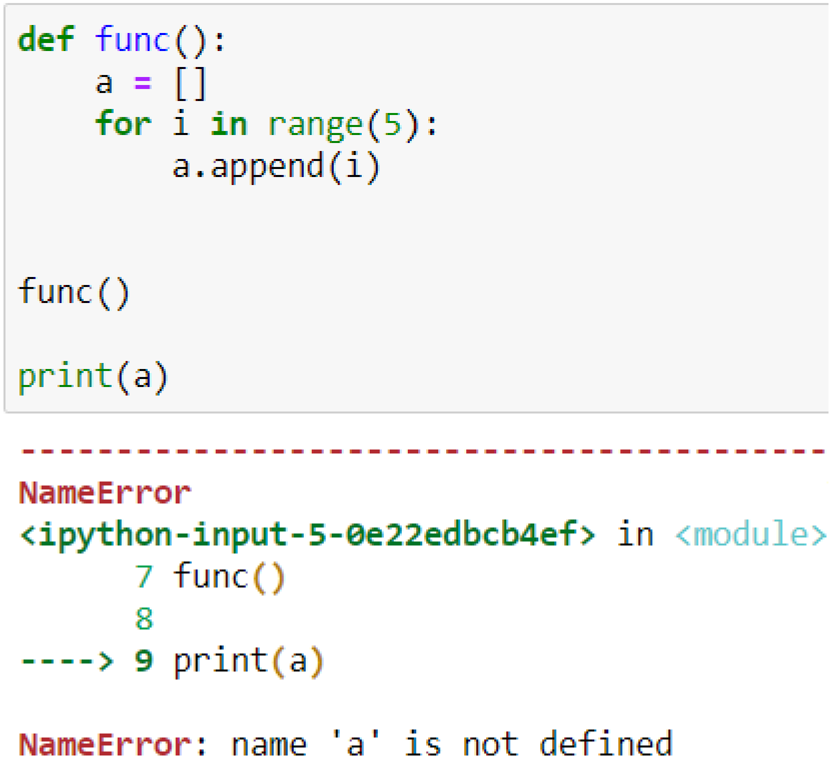


```python
def func():
    global a
    a = []
    for i in range(5):
        a.append(i)


func()

print(a)
```

Outputs?

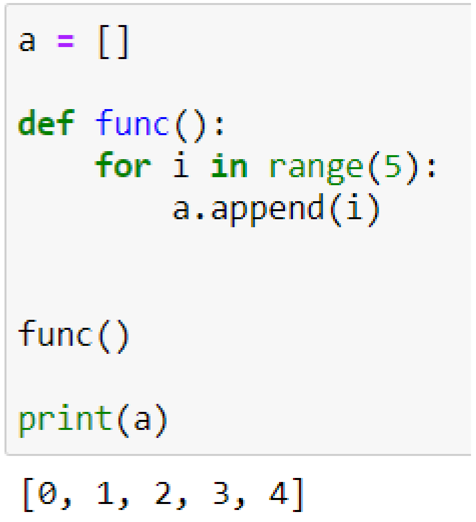
Functions (cont.)

Namespaces, Scope, and Local Functions



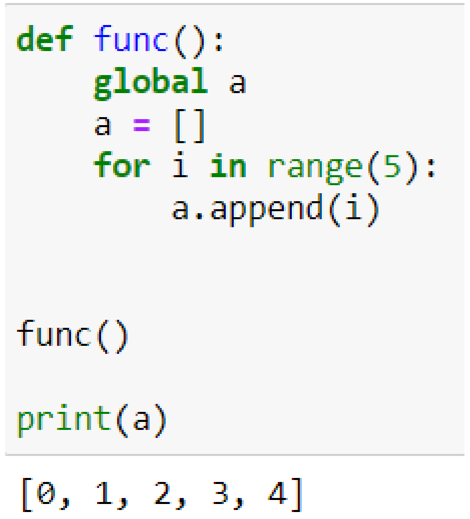
```python
def func():
    a = []
    for i in range(5):
        a.append(i)


func()

print(a)
```

```
---------------------------------------------
NameError
<ipython-input-5-0e22edbcb4ef> in <module>
      7 func()
      8
----> 9 print(a)

NameError: name 'a' is not defined
```



```python
a = []

def func():
    for i in range(5):
        a.append(i)


func()

print(a)
```

```
[0, 1, 2, 3, 4]
```



```python
def func():
    global a
    a = []
    for i in range(5):
        a.append(i)


func()

print(a)
```

```
[0, 1, 2, 3, 4]
```

Functions (cont.)
Returning Multiple Values

- In comparison with Java and C++, Python can return multiple values from a function with simple syntax.
- What's happening here is that the function is actually just returning one object, namely a tuple, which is then being unpacked into the result variables.

```python
def f():
    a = 5
    b = 6
    c = 7
    return a, b, c
```

```python
a, b, c = f()
print(a)
print(b)
print(c)
```

```
5
6
7
```

```python
return_value = f()
print(return_value)
```

```
(5, 6, 7)
```

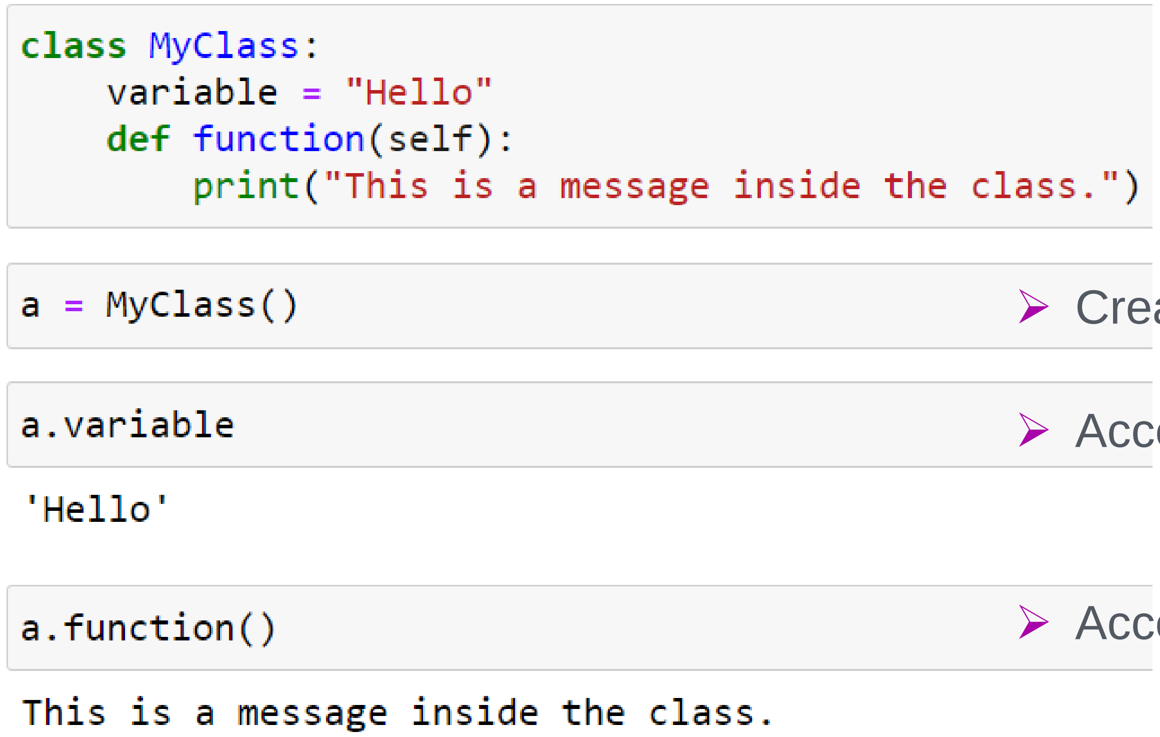
Class in Python

How create a class?

- Python is an object-oriented programming language.
- Almost everything in Python is an object, with its properties and methods.
- So we can think that a class is like an object constructor.
- Use ‘class’ keyword to create class and the name of class follows

Class in Python (cont.)

How create a class?



```
class MyClass:
    variable = "Hello"
    def function(self):
        print("This is a message inside the class.")
```

```
a = MyClass()
```

- Create an object of *MyClass* called *a*

```
a.variable
```

```
'Hello'
```

- Access the attribute of the object (named *variable*)

```
a.function()
```

```
This is a message inside the class.
```

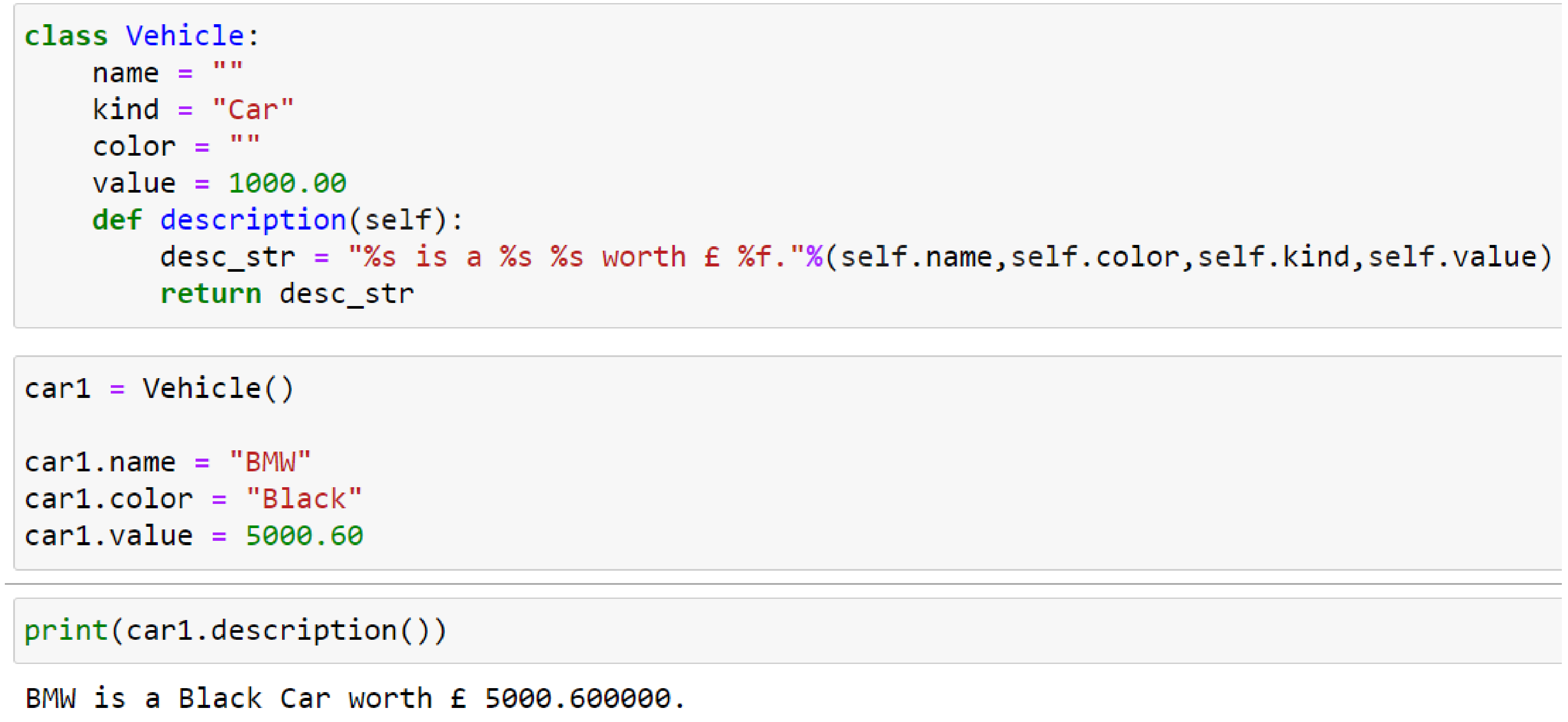
- Access the method of the object (named *function*)

Class in Python (cont.)
Attributes and Methods in class

- ❑ Variables of a class are called as **attributes** of the class
  - ❑ Attributes can be any type of data
- ❑ **Methods** are created in the same way with function creation,
  - ❑ When a function is defined in the class we need to use ‘self’ keyword as the first parameter of the function.

Class in Python (cont.)
Attributes and Methods in class

- An example:



```python
class Vehicle:
    name = ""
    kind = "Car"
    color = ""
    value = 1000.00
    def description(self):
        desc_str = "%s is a %s %s worth £ %f."%(self.name,self.color,self.kind,self.value)
        return desc_str
```

```python
car1 = Vehicle()

car1.name = "BMW"
car1.color = "Black"
car1.value = 5000.60
```

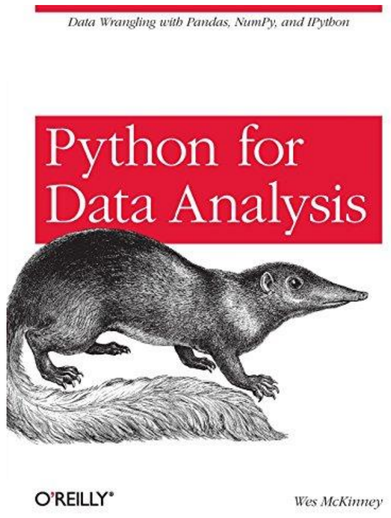
```python
print(car1.description())
```

```
BMW is a Black Car worth £ 5000.600000.
```

Class in Python (cont.)
“self” key word in a class

- **self** represents the instance of the class.
- By using the “**self**” keyword, we can access the attributes and methods of the class in python.

References & More Resources



- References:
  - McKinney, Wes. *Python for data analysis: Data wrangling with Pandas, NumPy, and IPython*. O'Reilly Media, Inc., 2012.

- More Resources:
  - Python Data Analysis on Linkedin Learning: https://www.linkedin.com/learning/python-data-analysis-2
  - Learning Python on Linkedin Learning https://www.linkedin.com/learning/learning-python
    - To use Linkedin Learning, you can log in with your university account: https://myport.port.ac.uk/study-skills/linkedin-learning

COURSE
Python Data Analysis
By: Michele Vallisneri

COURSE
Learning Python
By: Joe Marini

Practical Session

- ❑ Please download Week03_sequences.ipynb and Week03_dicts.ipynb files; then complete their comments.
- ❑ Please read the practical sheet (Week03_Practicals.pdf) and do the exercise.