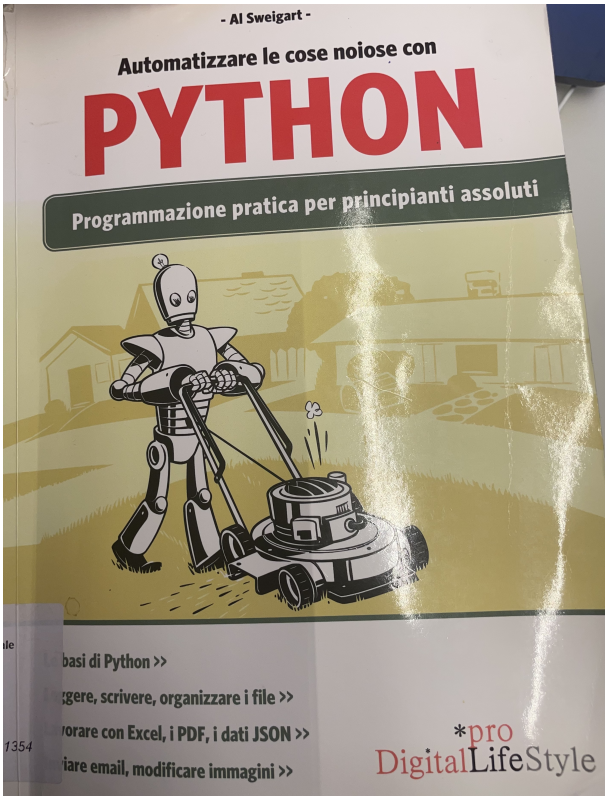
Python

Prepared by Simone Capodivento



March 6, 2025

Contents

1 Functions

```
def hello():
    print('Hello')
    print('Hello!!!')
    print('Hello to you.')

hello()
hello()
hello()  # Function calls
```

```
def hello(name):
    print('Hello ' + name)  # name is a parameter

hello('Alice')
hello('Bob')
```

1.1 Statement return

```
import random

def getAnswer(answerNumber):
    if answerNumber == 1:
        return 'It is safe'
    elif answerNumber == 2:
        return 'Exactly'
    elif answerNumber == 3:
        return 'Yes'
    elif answerNumber == 4:
        return 'Uncertain answer, try again'
```

```
    elif answerNumber == 5:
        return 'Ask again later'
    elif answerNumber == 6:
        return 'Focus and ask again'
    elif answerNumber == 7:
        return 'My answer is no'
    elif answerNumber == 8:
        return 'The outlook is not very good'
    elif answerNumber == 9:
        return 'Very doubtful'

r = random.randint(1, 9)  # Generate a random number between 1 and 9
fortune = getAnswer(r)  # Call the function with the generated number
print(fortune)  # Print the fortune message
```

Python Code Examples

Basic Print Statements

```
print('Hello')
print('World')
```

Print with End Parameter

```
print('Hello', end="")
print('World')
```

Print with Multiple Arguments

```
>>> print('cats', 'dogs', 'mice')
    cats dogs mice
>>> print('cats', 'dogs', 'mice', sep=',')
    cats,dogs,mice
```

Scope and Variable Visibility

Local Variables

Local variables cannot access variables from other local scopes. A new local scope is created every time a function is called.

Global Variables

Global variables can only be accessed within the global scope.

Lists

A list is a value that contains one or more values in a determined sequence. The elements inside the list are called items, and they are comma-separated.

```
['cat', 'bat', 'mouse', 'elephant']
```

The positions of these elements are called indexes.

Examples

```
spam = ['cat', 'bat', 'mouse', 'elephant']
spam[0]
'cat'

spam[1]
'bat'

spam[2]
'mouse'

'hello' + spam[0]
'hellocat'

'hello ' + spam[0]
'hello cat'

spam = [['cat', 'bat'], [10, 20, 30, 40, 50]]
spam[0]
['cat', 'bat']
```

Negative Indexes

```
spam = ['cat', 'bat', 'mouse', 'lion']
spam[-1]
'lion'
```

Sublists and Slices

```
spam[1:4]
```

Working with Lists

Flexibility of Manipulation

```
catNames = []
while True:
    print('Enter the cat name ' + str(len(catNames) + 1) +
          ' (or press enter to finish):')
    name = input()
    if name == '':
        break
    catNames = catNames + [name]  # list concatenation
    print('The cat names are:')
    for name in catNames:
        print(' ' + name)
```

For Loops with Lists

```
for i in range(len(supplies)):
    print('Index ' + str(i) + ' in supplies is: ' + str(supplies[i]))

# Example output:
```

```
index 0 in supplies is: pens
index 1 in supplies is: staplers
index 2 in supplies is: flamethrower
index 3 in supplies is: binders
```

in and not in Operators

The in and not in operators are used to check whether a value exists in a list.

```
supplies = ['pens', 'staplers', 'flamethrower', 'binders']

'pens' in supplies
# True

'notebook' in supplies
# False

'staplers' not in supplies
# False
```

2 In and not in Operators

```
'howdy' in ['hello', 'hi', 'howdy', 'heyas']
# True

'howdy' not in spam
# False

'cat' not in spam
# True
```

```
myPets = ['Zophie', 'Pooka', 'Fat-tail']
print('Enter the name of an animal:')
name = input()
if name not in myPets:
    print('I don\'t have a pet named ' + name)
else:
    print(name + ' is one of my pets.')
```

3 += Operator

```
spam = 'Hello'
spam += ' world!'

spam
# 'Hello world!'
```

```
spam = ['hello', 'hi', 'howdy', 'heyas']
spam.index('hello')
# 0
```

4 append() and insert() to Add Values to Lists

```
spam = ['cat', 'dog', 'bat']
spam.append('moose')
```

5 Remove Values from Lists with `remove()`

```
spam = ['cat', 'dog', 'bat']
spam.remove('bat')
```

6 `sort()`

```
spam.sort(reverse=True)

spam
# ['bat', 'cat', 'dog']
```

7 Using `str.lower`

```
spam = ['a', 'z', 'A', 'Z']
spam.sort(key=str.lower)

spam
# ['a', 'A', 'z', 'Z']
```

8 Tuple

```
eggs = ('hello', 42, 0.5)

eggs[0]
# 'hello'

eggs[1:3]
# (42, 0.5)

len(eggs)
# 3
```

9 Conversion Type Functions: `list()` and `tuple()`

```
tuple = ['cat', 'dog', 5]

tuple = ('cat', 'dog', 5)
```

10 Reference Assignment

```
spam = 42
cheese = spam
spam = 100

spam
# 100
```

```
spam = [0, 1, 2, 3, 4, 5]

cheese = spam

cheese[1] = 'Hello!'

spam
# [0, 'Hello!', 2, 3, 4, 5]

cheese
# [0, 'Hello!', 2, 3, 4, 5]
```

11 Passing References

```
def eggs(someParameter):
    someParameter.append('Hello')

spam = [1, 2, 3]
eggs(spam)

print(spam)
# [1, 2, 3, 'Hello']
```

12 `copy()` and `deepcopy()`

```
import copy
spam = ['A', 'B', 'C', 'D']
cheese = copy.copy(spam)
cheese[1] = 42

spam
# ['A', 'B', 'C', 'D']

cheese
# ['A', 42, 'C', 'D']
```

If the list you need to copy contains lists, use `copy.deepcopy()` instead of `copy.copy()`. Type of dictionaries The index of dictionaries is called **keys**, and they consist of key-value pairs.

```
myCat = {'size': 'fat', 'color': 'gray', 'temperament': 'cheerful'}
```

Dictionaries and lists

```
pets = {'cat': ['Zophie', 'Pooka'], 'dog': ['Rex', 'Buddy']}
print(pets['cat'])
# ['Zophie', 'Pooka']
```

Python Dictionaries and Lists

Dictionaries store key-value pairs and do not maintain a specific order. Below are essential methods for working with them.

Accessing Keys, Values, and Items

```
spam = {'color': 'red', 'age': 42}

for v in spam.values():
    print(v)

for k, v in spam.items():
    print(f'Key: {k} Value: {v}')
# Output:
# Key: color Value: red
# Key: age Value: 42
```

Retrieving Values with `get()`

`get()` returns the value of a key or a default if the key is missing.

```
spam = {'name': 'Pooka', 'age': 5}
color = spam.get('color', 'unknown')
print(color)  # Output: unknown
```

Setting Default Values with `setdefault()`

`setdefault()` assigns a value to a key only if it is absent.

```
spam = {'name': 'Pooka', 'age': 5}
spam.setdefault('color', 'black')
print(spam)  # Output: {'name': 'Pooka', 'age': 5, 'color': 'black'}
```