TECHNISCHE UNIVERSITÄT DRESDEN

DRESDEN concept

Maximilian Moeller
Faculty of Computer Science

An introduction to Python 3

RoboLab Autumn Course // 01 Nov 2019

Overview

Overview

- developed by Guido van Rossum
- high-level programming language
- designed for easy readability and simplicity

Installation

Installation

Python

Windows:

https://www.python.org/downloads/

Ubuntu:

$ sudo apt install python3

MacOS:

$ brew install python3

IDE

Pycharm:

https://www.jetbrains.com/pycharm/

-> professional

Visual Studio Code:

https://code.visualstudio.com/

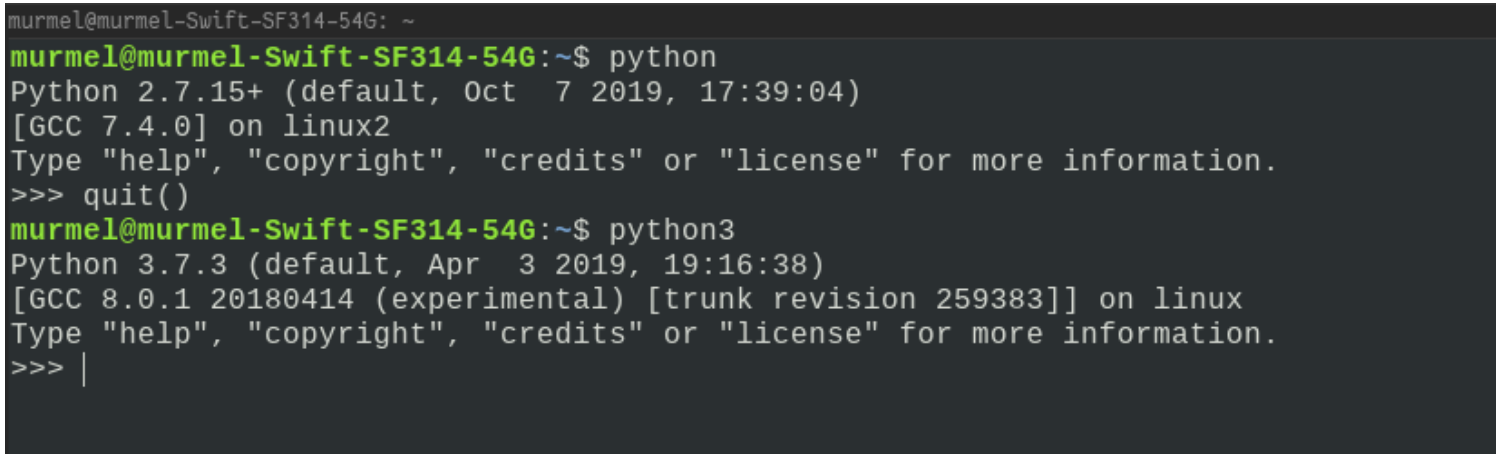
First Steps

- verify installation
- we are using Python 3.7.3

- documentation:
  - https://docs.python.org/3.7/

- read the „Zen of Python“:
  >>> import this

- install PEP 8 and get used to it



Zen of Python

Beautiful is better than ugly.
Explicit is better than implicit.
Simple is better than complex.
Complex is better than complicated.
Flat is better than nested.
Sparse is better than dense.
Readability counts.
...

Basics

Basics

```python
# string
my_string = "abc" + 'def'.upper()
# my_string = "abcDEF"

# boolean
my_bool = (True or False) and not (False and True)
# my_bool = True

# integer
my_int = (2019 % 11) ** (2019 // 1024 + 1)
# my int = 36

# float
my_float = 0.648 * 1.132

my_list = [1, "foo"]
my_list.append([2,3])
my_list[0] = "ROBO...bsss"
# my_list = ["ROBO...bsss", "foo", [2,3]]
```

```python
print(my_list[2][1])
# 3

# tuple
my_tuple = (1,2,'hello_world')

# dictionary
my_dict = {
            "Turing" : "Alan",
            "Lovelace" : "Ada"
}
my_dict["Hopper"] = "Grace"
# my dict = {'Turing': 'Alan', 'Lovelace': 'Ada', 'Hopper': 'Grace'}

```

Basic Concepts

- interpreted
- inferred dynamic typing
- indentation determines structure
- no semicolons, parentheses or curly brack

```
1   # this is a comment
2   tutor_name = "Max"
3   tutor_age = 19
4   student_knows_how_to_program = True
5   student_has_a_question = False
6
7   def ask_question(question):
8       if question == None or question == "":
9           pass
10      else:
11          print(question)
12
13  if student_knows_how_to_program:
14      print("all easy")
15  elif student_has_a_question:
16      student_question = "How do I ..."
17      print(tutor_name + ask_question(student_question))
```

Object-oriented Example

- pay attention with these standart parameters:
  - lists
  - dicts
  - sets

- usefull resources:
  - https://docs.python.org/3/library/index.html
  - https://www.w3schools.com/python/python_intro.asp

```python
class Table:

    def __init__(self, height, width, depth, material="wood"):
        self.height = height
        self.width = width
        self.depth = depth
        self.material = material
        self.items = {}

    def get_dimensions(self):
        return self.height, self.width, self.depth

    @staticmethod
    def leg_count():
        return 4

    def get_area(self):
        return self.width * self.depth

    # CAUTION!!!
    def lay_down(self, items = []):
        self.items.update(items)
```

Python in RoboLab

- only import from standart library and ev3dev.ev3 (assignment 5)
- carefully read the documentation
- then read it again
- “First, solve the problem.
  Then, write the code.”

```
1   # bitwise operators: <<, >>, ~, &, |, ^
2   foo = 5 # 0b101
3   bar = 6 # 0b110
4   res = (foo & bar) ^ bar # res = 2
5
6   # lists of lists
7   my_matrix = [[1,2,3],
8                [4,5,6],
9                [7,8,9]]
10
11  [1] in list(map(lambda x: [x], [1,2,3]))
12  # true
13
14  """
15          further hints:
16          - Enums
17          - zip()
18  """
```

Unit Tests

Unit Tests

- you have to write your own tests
- write useful tests
- edge cases

```
1  class Inventory:
2      def __init__(self, items):
3          self.items = items
4
5      def add_item(self, item):
6          self.items.append(item)
7      def delete_item(self, item):
8          self.items.remove(item)
```

Unit Tests

```
1  from Inventory import Inventory
2  import unittest
3
4  class MyInventoryTest(unittest.TestCase):
5      def setUp(self):
6          self.empty = Inventory([])
7          self.inventory = Inventory(['apple', 'milk', 'cheese', 'apple'])
8
9      def test_empty_inventory(self):
10         self.assertEqual(self.empty.items, [])
11
12     def test_add(self):
13         self.inventory.add_item('banana')
14         self.assertEqual(self.inventory.items, ['apple', 'milk', 'cheese', 'apple', 'banana'], 'Adding Banana failed')
15
16     def test_remove_single_item(self):
17         self.inventory.delete_item('milk')
18         self.assertEqual(self.inventory.items, ['apple', 'cheese', 'apple'], 'Drinking milk failed')
19
20     def test_remove_double_item(self):
21         self.inventory.delete_item('apple')
22         self.assertEqual(self.inventory.items, ['milk', 'cheese', 'apple'], 'Where is my second apple?')
```

Dos and Don'ts

Dos and Don'ts

- be consistent in formatting
- useful names
- comment your code (especially **why** you did it that way)
- no import loops!

```
1   import turtle
2
3   def MyVeryOwnFunction1(x,y,z):
4       for x in y:
5           return x in z
6
7   def this_is_a_function_with_a_very_short_and_useful_name(a,b,c,d=[]):
8       d = d.append(a)
9       c /= (b*c / c **(c|b))
10      return a
11
```

Questions?

Sources

- https://www.w3schools.com/python/python_intro.asp
- https://www.python.org/dev/peps/pep-0008/
- https://www.python.org/
- https://de.wikipedia.org/wiki/Python_(Programmiersprache)
- Structure based on „An introduction to Python 3” held by Georg Lauterbach