Python Seminar 2018

Ana Bulović
Jorin Diemer
Jens Hahn

Plan for today

1. Homework
2. Functions
3. Small assignment
4. Built-ins

```python
def rock_paper_scissors(user_input, computer_input):
    if user_input is 'rock' and computer_input is 'paper':
        return 'computer wins'
    elif user_input is 'rock' and computer_input is 'scissors':
        return 'you win'
    elif user_input is 'rock' and computer_input is 'rock':
        return 'tied game'
    elif user_input is 'paper' and computer_input is 'paper':
        return 'tied game'
    elif user_input is 'paper' and computer_input is 'scissors':
        return 'computer wins'
    elif user_input is 'paper' and computer_input is 'rock':
        return 'you win'
    elif user_input is 'scissors' and computer_input is 'paper':
        return 'you win'
    elif user_input is 'scissors' and computer_input is 'scissors':
        return 'tied game'
    elif user_input is 'scissors' and computer_input is 'rock':
        return 'computer wins'


def rock_paper_scissors_2(user_input, computer_input):
    rules = {'rock': {'paper': 'computer wins', 'scissors': 'you win', 'rock': 'tied game'},
             'paper': {'rock': 'you win', 'scissors': 'computer wins', 'paper': 'tied game'},
             'scissors': {'rock': 'computer wins', 'paper': 'you win', 'scissors': 'tied game'}}

    return rules[user_input][computer_input]
```

Functions

- reusable, separated block of code
- Specialized for certain task
- You don‘t have to write the same things again and again and again
- Not executed until you call it

Definition

- Definition of function with `def` statement
- Specify a name
- Optional argument in brackets
- Indent by one tab (most times 4 spaces)

```
def function_name(argument):
      pass or return
```

Example

- You have to write many emails for several people
- Everytime you have to introduce this person
- Define standard introduction

```
def introduce(name):
      print('Hello, my name is ' + str(name))
```

Call a function

```
Introduce('Ana')
```

- This would print:

  Hello, my name is Ana

```
def introduce(name):
      print('Hello, my name is ' + str(name))
```

Return

```
hello = Introduce('Ana')
```

- A reference is created which points to the Return value of the function
- You can use your introduction sentence outside of the function

```
def introduce(name):
    return 'Hello, my name is ' + str(name)
```

Scope

- (local) Variables you define inside a function are not known anywhere else
- Global variables are known within the function
- locals are used before globals

```
glo = ‘I am global‘

def introduce(name):
        print(glo)
        s = ‘Hello, my name is ‘
        print(s + str(name))

print(s)
```

Default values

- Default values are used when you don‘t specify the input
- Define default values with =
- If you have several arguments, default values are placed behind the others

```
def introduce(age, name=‘Ana‘):
      print(‘My age is ‘ + str(age))
      return ‘Hello, my name is ‘ + str(name)
```

Doc strings

- String which describes the function
- String is displayed when you call the **help()** function

```
def introduce(age, name='Ana'):
    """This function is doing this and
     that"""
     print('My age is ' + str(age))
     return 'Hello, my name is ' + str(name)
```

Exercise

- Write a function which returns the absolute value of a number
- Define a function which return the maximal value in a list of intergers
- Write these functions in assignment_3.py
- Name the functions **maximum** and **absolute**

Built ins

- Predefined functions known by the python interpreter
- Always available
- 68 functions, most of them need specific datatype as input
- Conversion of data types (casting)

```
a = str(3)
```

Built ins

- General:
  - help(), id(), range(), type(), input()
- Sequence:
  - enumerate(), len()
- Mathematical:
  - max(), min(), sum(), abs()
- Casting:
  - dict(), float(), int(), list(), set(), tuple(), str()

HOMEWORK

Write functions doing the following job:

- Average of a list
- Median of a list