Setting up for the course Python 3.4+

- Installing:
  - Windows
    - Download and execute .exe file
    - https://www.python.org/downloads/
  - Linux
    - sudo apt-get install python3
  - Mac
    - Download and install
    - https://www.python.org/downloads/
- Clone / download
  - https://github.com/spartam/basic_python_course

Python

OR AT LEAST YOUR INTRODUCTION TO

Who Am I ?

- Master student Computer Science: WISE
- Active member of InfoGroep
- Python self taught
- Three years experience

Who are you?

- Field of study
- Programming experience

What are we going to see?

- Installing and importing modules
- Functions and Control flow
- Data Structures
- Slicing
- Classes
- Modules
- Decorators

Installing and importing modules

- pip (python installs packages)
  - Package manager for python modules
- pip install <package name>
- Virtual environment is recommended

```
1  import os
2  from os import getcwd
3
4  print(os.getcwd())
5  print(getcwd())
6  print(os.listdir())
7  print(listdir())
```

```
[redacted]Infogroep\lesje\2016-2017
[redacted]Infogroep\lesje\2016-2017
['import.py', 'Python.pptx', '~$Python.pptx']
Traceback (most recent call last):
  File "[redacted]\Infogroep\lesje\2016-2017\import.py", line
7, in <module>
    print(listdir())
NameError: name 'listdir' is not defined
```

Functions and Control flow

```python
from datetime import date

def python():
    print("is cool")

def return_today():
    return date.today()

def for_loop(x):
    '''example of a for loop displaying the use of range and the if control flow
    Also displays the function with an argument and string formatting'''
    for el in range(x): # creates a range from 0 to x - 1
        if el < 2 :
            print("I'm number %s. We continue" % el)
        elif el < x - 1 :
            print("I'm number %s. We're reaching the end" % el)
        else:
            print("I'm number %s and here we stop" % el)

def while_loop(count=2):
    '''example of a while loop structure. It also displays the use of a keyword
    argument and string formatting'''
    end = count * -1
    while count >= end:
        print('while %s > %s' % (count, end))
        count -= 1
```

```python
python()
print('--')
for_loop(5)
print('--')
while_loop()
print('--')
while_loop(count=3)
print('--')
return_today()
print('--')
print(return_today())
```

```
is cool
--
I'm number 0. We continue
I'm number 1. We continue
I'm number 2. We're reaching the end
I'm number 3. We're reaching the end
I'm number 4 and here we stop
--
while 2 > -2
while 1 > -2
while 0 > -2
while -1 > -2
while -2 > -2
--
while 3 > -3
while 2 > -3
while 1 > -3
while 0 > -3
while -1 > -3
while -2 > -3
while -3 > -3
--
--
2016-12-07
```

Functions and Control flow

- Exercises
  - Create factorial
    - Hint: range(y, x)

```
1  for i in range(4, 7):
2      print(i)
```

```
4
5
6
```

  - Create a function triplets
    - Given a number X loops over every number from 0 to X
    - Prints the number if it’s dividable by 3
    - Use the function “divideable_by” from the file “auxilary_functions”
      - DO NOT COPY THE CODE! Use import!

Data Structures

- Dictionaries
- Lists
- Sets
- Tuples

Data Structures

- Dictionaries
  - Key – value pairs
  - Dict.update(dict)
  - Dict[key] returns or sets value
  - Dictionary comprehension
  - Dict.keys()

```
1  data = {'name' : 'Jan', 'age' : 21}
2  data['new'] = 'new_value'
3
4  print(data)
5  print(data['name'])
```

```
{'age': 21, 'new': 'new_value', 'name': 'Jan'}
Jan
```

Data Structures

- List
  - Append
  - List[index] returns or sets value
  - List comprehension
  - Can be used as stack
    - List.push
    - List.pop
  - Slicing
  - in operator

```
 9   data = [1, 2, 3, 4, 5]
10   data.append(6)
11
12   print(data)
13   print(data[3])
```

```
[1, 2, 3, 4, 5, 6]
4
```

Data Structures

- Sets
  - Only unique values
  - Set.add
  - No indexing
    - Pop
    - Iterate over every element
  - Mathematical operations
    - A – B : Every element in A which is not in B
    - A | B : Every element in either A or B
    - A & B : Every element in both A and B
    - A ^ B : Every element in A or B but not both
  - in operator

```
17  s = set((1, 2, 3, 1))
18
19  print(s)
```

```
{1, 2, 3}
```

Data Structures

- Tuples
  - Immutable
  - in operator

Data Structures

- Comprehension

```python
from auxilary_functions import even

data = [el for el in range(21) if even(el)]
print(data)
```

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20]
```

```python
data = {el : el * 2 for el in range(11)}
print(data)
```

```
{0: 0, 1: 2, 2: 4, 3: 6, 4: 8, 5: 10, 6: 12, 7: 14, 8: 16, 9: 18, 10: 20}
```

Data Structures

- Comprehension exercise
  - Create a list containing all countries where English is an official language
  - Create a set containing all continents

```
1  data = {
2          'Belgium'        : {
3                                   'languages' : ['NL', 'DE', 'FR'],
4                                   'continent' : 'Europe'
5                             },
6          'France'         : {
7                                   'languages' : ['FR'],
8                                   'continent' : 'Europe'
9                             },
10         'New Zealand'    : {
11                                  'languages' : ['EN', 'MI'],
12                                  'continent' : 'Oceania'
13                            }
14         }
15
16
17 print([key for key in data if data[key]['continent'] == 'Europe'])
```

```
['Belgium', 'France']
```

Slicing

- Works for
  - Lists
  - Strings
    - No assignment

```
1   string = 'hello world'
2
3   print(string[:5])
4   print(string[6:])
```

```
hello
world
```

```
1   data = [1, 2, 5, 6]
2   data[2:2] = [3, 4]
3   print(data)
```

```
[1, 2, 3, 4, 5, 6]
```

```
1   data = [1, 2, 5, 6]
2   data[2:] = [3, 4]
3   print(data)
```

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

```
1   data = [1, 2, 5, 6]
2   data[2] = [3, 4]
3   print(data)
```

```
[1, 2, [3, 4], 6]
```

Classes

- Exercise
  - Extend the class with __eq__

```python

class coordinate:
    x = 0
    y = 0

    def __init__(self, **kwargs):
        keys = kwargs.keys()
        if 'x' in keys:
            self.x = kwargs['x']
        if 'y' in keys:
            self.y = kwargs['y']

    def __str__(self):
        return 'x : %s\ty: %s' %(self.x, self.y)

if __name__ == '__main__':
    P = coordinate()
    print(P)
    P = coordinate(x=2)
    print(P)
    P = coordinate(y=3)
    print(P)
    P = coordinate(x=5, y=7)
    print(P)
```

```
x : 0   y: 0
x : 2   y: 0
x : 0   y: 3
x : 5   y: 7
```

Modules

- To import from a directory
  - __init__.py
  - From <directory> import *
    - __all__ = [list of files in directory]

Decorators

```python
functions = set()

def decorator(command):
    def new_func(self, *args, **kwargs):
        print('executed %s with:\n\targuments %s\n\tkeywordarguments %s' % (command.__name__, args, kwargs))
        command(self, *args, **kwargs)
    functions.add((command.__name__, command))
    return new_func



@decorator
def add(x, y):
    return x + y

add(5, 3)
print(functions)
```

```
executed add with:
    arguments (3,)
    keywordarguments {}
{('add', <function add at 0x004A2300>)}
```