Introduction

Python is a popular open-source scripting language focusing on simplicity and clarity. It also supports object-oriented programming. It is the purpose of this introduction to provide you a starting point for learning Python and a reference.

Python is available in two releases, version 2 and version 3, both of which are under active development but not completely compatible with each other. Python 2 code can use extensions which make it compatible with Python 3 code. Both Python versions are available in the CIP pool. Calling `python` on the command line in the CIP pool will start Python 2. To start Python 3 call `python3`.

During these exercises we will use Python 3, but it is important to be aware of the different versions,

These boxes contain advanced knowledge, which is not necessary to know for these exercises but which we find very useful. Feel free to read these and follow the documentation links if you are interested!

Python's popularity is also due to the possiblity to generate Python language interfaces for code written in other languages such as C++. Frameworks like OpenCV make use of this to provide their functionality easily accessible through Python. Furthermore, the Python runtime can be embedded into other applications which then become "scriptable" through the Python language.

Installation under Linux

The easiest way to install Python under Linux is through the system's package manager. Most standard packes (e.g. NumPy and matplotlib) are directly available in Linux distributions. Distributions usually offer both version 2 and 3 releases of Python with the default version depending on the distribution.

For Ubuntu 14.04 LTS, install the **python3**, **python3-numpy** and **python3-matplotlib**. A list of the available Python packages for Ubuntu 14.04 LTS can be found here. You may need to manually install some packages required for the exercises.

Installation at CIP

Python 3 is already available at the CIP. However you need to install some additional packages by executing:

pip3 install --user cvxopt

pip3 install --user py-cpuinfo

pip3 install --user sklearn (0.17.1)

pip3 install --user pillow

Installation under Windows

There are several ways of installing Python on Windows. It can be downloaded from the official website or installed through a Python distribution. Similar to Linux, several projects provide Python distributions which bundle Python along popular and often used packages. For this class, we recommend Anaconda which also provides extensive documentation.

Download Anaconda from the official website and run the installer. After installation run the Anaconda Prompt. This opens a command line window where Anaconda's tools and Python are available. Anaconda's management tool is called *conda* which allows to install, remove and update packages. Also, packages can be installed by using *pip install* if they are not available via conda.

For this exercise we will need the *cvxopt* package. All other packages should be included in Anaconda. To install *cvxopt* execute *pip install cvxopt* in a command prompt.

More information on how to manage packages using conda can be found in the official documentation.

Anaconda supports the installation of different Python versions in parallel in so called *environments*. This makes it possible to install and use Python 2 and 3 in parallel. More information about Anaconda environments can be found here.

Editors and IDEs

There are several ways to write Python code. Python files using the file ending *.py* can be directly written using a standard text editor. IDEs (integrated development environment) are also available for Python. One popular Python IDE is PyCharm. Another way to write Python code is through iPython Notebooks.

If you are writing your Python code in a text editor and are unsure on how to run it from the command line, take a look at this page.

For Windows, we recommend using Eclipse with Pydev.

Running Python Scripts from the Command Line

You can execute your Python scripts on the command line by calling python with the path to the script file as parameter, by typing `python3 script.py`. This will execute *script.py*. If you have multiple Python versions (i.e. versions 2 and 3) installed on Linux, you can control which version is used by typing either `python2 script.py` or `python3 script.py`.

You can make Python scripts directly executable under Bash by adding the line `#!/bin/python` at the beginning of your script. This has to be exactly the first line in order to work. It will be disregared by as comment by Python because of the `#`, but tells bash to use Python to execute it. You have to mark your script as executable by using *chmod*. Type `chmod +x script.py` to mark the script file as executable. You can now directly run the script by typing `./script.py` into your command line. The line at the top of your python file is called a shebang and works for other languages too.

"Hello, World!"

```
1 import absolute_import, division, print_function
2
3 print('Hello, World!')
```

Import Statements

In order to add existing functionality from other modules to a Python program, import statements are used. Using the import command means that Python starts to look for that module and if found, will make it available to the current program. If the module is not found, an error occurs. The example below shows three different imports.

```
1 import cv2
2 import numpy as np
3 import matplotlib.pyplot as plt
```

The first line imports OpenCV. Functions from the imported module use the module name as a prefix. For instance, `cv2.imread()` would load an image file. Because module names can be quite long, there is the possiblity of defining a shorthand for the module name. In line 2 for example, numpy is imported and `np` will be used for accessing functions instead of `numpy`. The same applies to matplotlib in line 3. For standard modules like numpy and matplotlib, we alsways use the same shorthands, i.e. we always use np for numpy. This makes code more easily understandable.

We could write line 3 from the example above differently. `from matplotlib import pyplot as plt` would do exactly the same. It is just a different way of writing it.

Controlling the Program Flow

Instead of using braces {, } as in other languages, code blocks in Python are defined by indentations.

In Python, the standard conditional expressions are available: `==`, `!=`, `>`, `<`, `<=`, `>=`. An if-statement looks like this

```
if x == 5:
  print('x equals 5')
```

Note that the indentation is important. It is also common to forget the colon in the first line. There is also an if-else-statement

```
if y == 7:
  print('y equals 7')
else:
  print('y does not equal 7')
```

A for-loop looks like this

```
for i in range(0, 10):
  print(i)
```

More about loops and branching can be found in tutorials or the documentation.

Further Tutorials and Reading

There are many tutorials and introductions available for Python, Numpy and OpenCV. Stefan Steidl also compiled a several iPython Notebooks which you can run or view yourself. You find these notebooks on the slides website for the lecture *Introduction to Pattern Recognition*. The following list contains links to more tutorials:

- Python 2.7
- NumPy
- Matplotlib
- OpenCV