Foundation of programming (CK0030)

Technical topics

Francesco Corona

CK0030
FdP

FdP - When and where

Analogical when and where

- TUE: 08:00-10:00 [was: 07:00-10:00]
- THU: 08:00-10:00

- TUE: Bloco and Sala TBA
- THU: Bloco and Sala TBA

Online, where things happen

- Website: **Foundations of programming** (click me)
- SIGAA: **SIGAA** (click me and find me, if you can)

Remark

The course website is the main communication channel, not SIGAA

FdP - Material

A course on computer programming using Python and math examples

Lectures and material are mostly based on the following textbook:

- *A primer on scientific programming with Python*: 4th and/or 5th Edition, by Hans Petter Langtangen (**Book website**)

Remark

Course slides and notebooks will suffice

FdP - Material (cont.)

Other primary references are the **official Python documentation** page (with tutorials, library and language references) and the collection of **Python books**[1], among which the following ones relate to this course

- *Python essential reference*, by D. Beazley
- *Learning Python and Programming Python*, by M. Lutz
- *Computing with Python - An introduction to Python for science and engineering*, by C. Fuerher, J. E. Solem and O. Verdier
- ...

[1] Tambem tem livros sobre Python em Português.

FdP - Content

We have chosen to use the Python programming language because it combines expressive power with clean, simple, and compact syntax

- Python is easy to learn and well suited for an introduction to coding
- Python is similar to MATLAB, a language for math computing
- It is easy to combine Python with compiled languages (Fortran, C, and C++, which are widely used for scientific computations)

FdP - Content (cont.)

- Intro to variables, objects, modules, and text formatting
- Programming with WHILE- and FOR-loops, and lists
- Functions and IF-ELSE tests

- Data reading and writing
- Error handling
- Making modules

- Arrays and array computing
- Plotting curves and surfaces

Remark

These are the core blocks of the course, blocks are strongly related

FdP - Content (cont.)

Example formulas are introduced and primitively computed in the early lectures, then they are used to produce tables of numbers

- Formulas are encapsulated in more sophisticated functions

Function inputs are user-provided and fetched from command line

- Validity checks of the input are performed

The result of computing formulas are shown as graphs

FdP - Content (cont.)

Remark

After the blocks, you should have enough knowledge of programming

- You will be able to solve mathematical problems
- In a so-called ‘MATLAB-style’ way of coding

Class programming, including user-defined types for math computations

- Object-oriented programming, with class hierarchies and inheritance

In the end of the course, if and only if time allows

FdP - Evaluation

We use problem sets covered by books, papers and webpages

- We expect you **not to copy**, refer to, or look at the solutions
- We expect you to **want to learn** and not google for answers[2]

Remark

The purpose of problem sets is to **help you think** about the material

- Not give us the right answers, 'cause we know them

[2] If you happen to use other material (we know you will), such material must always be acknowledged with a **citation** on the submitted solution. To avoid making a laughing stock of yourself, it is important that you check the **correctness** of the copied solution.

FdP - Evaluation (cont.)

Each of you must hand in his/her own answers

- Homeworks must be done **individually**

Also, each of you must write his/her own code

Remark

It is acceptable, however, for you to **collaborate** in figuring out answers

- We assume that you take the responsibility to make sure you personally understand the solution to any collaborative work[3]

As part of the evaluation, we will request you to **defend your homework**

[3] Though, you must always indicate on each homework with whom you collaborated.

FdP - Evaluation (cont.)

To typeset assignments, you are encouraged to use the template LaTeX[4]

- Download me **here**
- Check me out **here**

[4] LaTeX? Yes, LaTeX!

FdP - Evaluation (cont.)

Assignments must be returned **before deadline** via SIGAA

- You'll get notified of the opening of a new task

Delayed submissions are emailed to the teaching assistant

Remark

Delays will be penalised

- $[00h, 24h)$, -20% of the grade;
- $[24h, 48h)$, -40% of grade;
- $\ldots$

Accessing Python
Technical topics

Accessing Python

A Python system for scientific computing used to be difficult to install

- This problem is more or less solved today

There are several options for getting easy access to Python
and the most important packages for scientific computations

- The biggest issue is to make a choice

Accessing Python (cont.)

Strictly required software

- **Python**, version 2.7.x
- **NumPy**, array computing
- **Matplotlib**, plotting

Desired software packages

- **IPython**, iterative computing
- **SciTools**, add-ons to numpy
- **SymPy**, symbolic mathematics
- **SciPy**, advanced mathematics
- ...

If you get interested

- **pytest** or **nose**, code testing
- ...

Accessing Python (cont.)

These software and software packages need to run on either

- UFC computers (ask admins about it)
- Your computers (install that stuff)
- A web service (OK, to start only)

GNU/Linux, Mac OSX and Windows offer various possibilities

- You can install each individual package (very system dependent)
- You can install a pre-built environment (**Anaconda**, **Canopy**, ...)

Accessing Python (cont.)

Remark

- You have a Windows computer, get rid of Windows and install a Debian distribution of GNU/Linux
- You have a Windows computer and you really really like it, split the drive and install Debian
- You have a Windows computer and you really really really like it, ask the TA (and a doctor)

- You have a Mac OSX computer, get rid of Mac OSX and install a Debian distribution of GNU/Linux
- You have a Mac OSX computer and you really really like it, split the drive and install Debian
- You have a Mac OSX computer and you really really really like it, keep it and install Anaconda

- You have a GNU/Linux computer, make it Debian, apt-get install the stuff and get rolling

Typing code

Technical topics

Typing code

Code consist of plain text, a program to store text in a file is needed

- For writing code you need special programs, called **editors**, they preserve exactly the characters you type
- Word-type programs aim at producing sort-of-nice-looking reports by formatting the text, not code

A text editor

Typing code

A text editor

Some of the most widely used editors for writing programs are **Atom**, **Sublime Text**, **Emacs**, and **Vim** are available on all major platforms

Some simpler alternatives are

- GNU/Linux: **Gedit**/**Pluma**
- Mac OSX: **TextWrangler**
- Windows: **Notepad++**

Remark

Python comes with **Idle**, it is its own editor used to write programs

A text editor

Spyder is a graphical application for developing and running Python programs, and it is available on all major platforms

- Spyder comes with Anaconda and some other pre-built environments for (scientific) computing using Python

Spyder window contains a plain **text editor**, a **shell** to run programs, a **file browser** and a display for **documentation**

Executing code

Technical topics

Executing code

Remark

To run a Python program, you need a **terminal window**

- This is the window where you issue commands (Unix commands in GNU/Linux and Mac OSX, and DOS commands in Windows)

In a terminal window, one first moves to the right folder, there one executes code (`prog.py`) by typing `python prog.py arg1 arg2`

Whatever the program prints can be seen in the terminal window

The (interactive) shell

Executing code

The (interactive) shell

The second simplest way of executing a Python program is **IPython**

- You start IPython either by the command `ipython` in a terminal window, or by double-clicking the IPython icon (on Windows)
- Run a program (`prog.py`) by typing `run prog.py arg1 arg2`

Remark

Executing Python code in IPython works the same on all platforms

Notebooks
Executing code

Notebooks

A **IPython notebook** is an interactive tool for developing Python code

- You can either run it locally on your computer or in a web service

Remark

The interface to a notebook is a **web browser**, you write the code and see all the results in the browser window

Web services
Executing code

Web services

You can **avoid installing Python** on your machine by using a web service that allows you to write/run Python code

There are two excellent web services with notebooks:

- **SageMathCloud** at `https://cloud.sagemath.com`
- **Wakari** at `https://www.wakari.io/wakari`

Remark

At both sites you must **create an account** before you can write notebooks in a browser and download them to your computer