Fall term 2017/2018

MATH 5027: SCIENTIFIC PYTHON

Course coordinator: Professor Roberta Sinatra, sinatrar@ceu.edu
Teaching Assistant: Johannes Wachs, PhD student in Network Science, wachs_johannes@phd.ceu.edu - please send him all inquiries about schedule, classroom and technical requests
No. of Credits: 3
Prerequisites: Basic programming skills in any programming language (e.g. familiarity with logical statements, for loops, with different variables), Basic statistics.
IMPORTANT: This course can accommodate a maximum of 30 students. Priority is given to Mathematics students (Master and PhD) and Network Science PhD students. All other students are selected based on the entry test score. Students that take the course for grade have priority over auditors. **All students, both registered and in the waiting list must take the entry test on the first day**.

Course schedule: this course will take place twice a week during the first half of the term, until October 24th. Please find the detailed schedule on the course webpage.
If you want to continue using Python, check "CNSC6012: Data and Network Visualization" – the course follows a similar schedule, but in the second half of the term.
Course Level: Master and PhD
Office: Roberta Sinatra N11 609, Johannes Wachs N11 611
Office hours: Roberta Sinatra: by appointment; Johannes Wachs: Tue and Thu 1.30-2.30pm.

Brief introduction to the course:
This course will provide a comprehensive, fast-paced introduction to Scientific Python. The course will run with theoretical classes, hands-on sessions and tutorials. As this is currently the only Python course offered at CEU, *it tries to fit the needs of students at different programming levels. If you are already a good programmer, this course will be probably too easy for you. If you have never programmed in your life, this course might be very fast-paced, so consider complementing it with tutorials from other sources, like those of code-academy*. However, we will do our best to adapt classes and exercises based on the student feedback during the tutorials and hands-on sessions.
We expect you to come to lectures and labs, ask questions when you get stuck, do teamwork (yes, even if you are the best in the class and able to complete tasks on your own!) and develop a project taking advantage of tutorials. The course will have an intensive schedule, taking place mostly during the first six weeks of the term.

The goals of the course:
The overarching goal is to equip students with enough programming experience to start working in any area of computation and data-intensive research. This course will lay a foundation from which new tools and techniques can be explored.

The learning outcomes of the course:
By the end of the course, students will have experience with techniques which are vital to effective scientific research, including:
- The basic syntax and use of Python as a scientific tool, including writing and executing scripts to automate common tasks, using the IPython interpreter for interactive exploration of data and code, and using the Jupyter notebook to share and collaborate.
- Loading data from a variety of common formats
- Manipulating data efficiently with Numpy
- Basic web scraping
- Use of web APIs
- Use of special python packages, like networkx
- Performing basic data mining and machine learning analysis with Scipy and Scikit-learn

Course Organization:
Lectures: 8 classes of 120min (theory and hands-on sessions), 4 classes of 150min+15 min break (hands-on and tutorials). Use of a computer will be required during most of the lectures. Students can use their own laptops or the facilities provided by CEU. Instructions for installation of Python and the required packages will be provided during the first class.
Project coaching and tutorials: 2 classes of 120 min.

Topics
Lectures:
- Introduction to Python Programming: why Python? Basics of Python and IPython
- Data structures: lists, tuples, dictionaries
- Functions and Packages
- Numerical computing (Numpy: how to handle data efficiently and statistics)
- Scientific computing (Scipy: special functions, integration, linear algebra. Basics of Scikit-learn)
- Control flow and Pandas

Hands-on:
- Use of a web-API
- Handling and visualizing datasets

- Analysis of one dataset (with elements of multivariate analysis)
- Complex networks analysis

Tutorials will be designed during the course, based on the students programming background and their feedback during the first lectures.

Tentative calendar

First week

1st class: Entry test. Introduction to Python and Jupyter notebook. Fast-paced introduction to the use of a shell, finding a folder in a computer

2nd class: running a script, type of variables, strings, lists

Second week

3rd class: More on lists, control flow. Hands-on session: web scraping.

4th class: Dictionaries and file parsing. Examples with existing files.

Third week

5th class: Functions and packages

6th class: Hands-on session on web-API

Fourth week

7th class: Characteristics of Numpy. Data analysis with Numpy

8th class: Scipy, Scikit-learn and basic machine learning applications

Fifth week

9th class: Hands-on session on networks with Networkx

10th class: Pandas, data-frames and applications

Sixth week

11th class: Basic visualization with Matplotlib

12th class: Hands-on session on analysis of a dataset

Mid-November: 1 open class on final projects

Early December: Project presentations

Course Requirements/Assessment

Students are expected to attend lectures and hands-on sessions, to hand in one assignment during the course and to develop a project, alone or in pairs, during the entire term.

Grading:

- Attendance of the classes, hands-on sessions and teamwork: 30% of the final grade
- Assignment(s): 30% of the final grade
- Final project: 40% of the final grade

Final Project

For the final project, students will have to apply and show proficiency with the tools studied during the course. Possible projects will include, but will not be limited to: the analysis of a dataset, implementation and application of an algorithm, development of an interactive tool. A number of options for projects will be suggested in class. The students will also be free to design their own project within the guidelines that will be provided in the lecture.

Suggested reading

- Online resources and documentation provided during classes
- Bill Mark Lutz, Learning Python, O'Reilly (2013) – Also available for free online
- Bill Lubanovic, Introducing Python, O'Reilly (2014)
- Wes McKinney, Python for Data Analysis, O'Reilly (2013)

Further information, such as the course website, assessment deadlines, office hours, contact details etc. will be given during the course.

The instructor reserves the right to modify this syllabus as deemed necessary any time during the term. Any modifications to the syllabus will be discussed with students during a class period. Students are responsible for information given in class.

Cheating

In short: don't do it! You may work with friends to help guide problem solving or consult stack overflow (or similar) to work out a solution, but copying—from friends, previous students, or the Internet—is strictly prohibited. NEVER copy blindly blocks of code – we can tell immediately.

If caught cheating, you will fail this course. Ask questions in recitation and at office hours. If you're really stuck and can't get help, write as much code as you can and write comments within your code explaining where you're stuck.