**Subject name**:

Scientific Python for Engineers

**Code**:

---

**ECTS Credits:**

3 ECTS

**Language:**

English

**Responsible Unit:**

240 – ETSEIB – Escuela Técnica Superior de Ingeniería Industrial de Barcelona

**Department:**

Automatic Control – ESAII

**Beginning of the subject**:

2013

**Degrees**:

Master's degree in Automatic Control and Robotics

**Subject’s Responsible**:

Dr. Manel Velasco / Dr. Alexandre Perera

|  |
| --- |
| **Requirements** |

Previous capabilities:

The student should have basic skills in mathematics, linear algebra, elementary calculus, and algorithmics.

Requirements:

An empty USB-pen (8Gb).

|  |
| --- |
| **Faculty** |

Responsible Lecturer:

Dr. Manel Velasco / Dr. Alexandre Perera

Lecturers:

Dr. Manel Velasco / Dr. Alexandre Perera

Students’ attention timetable:

To be defined.

|  |
| --- |
| **Methodology** |

**Teaching methodology:**

The teaching methodology will combine lectures together with supervised learning based on problems and autonomous learning. The classes will be organized in tutorial sessions with practical problems/challenges intermingled. The master classes will be focused in the explanation of fundamental concepts by the lecturers, promoting the active participation of students. All sessions will be developed in a computer room.

**Qualification:**

The acquired competences and capabilities will be assessed on the basis of the solutions to the proposed challenges, which will have to be delivered on class.

|  |
| --- |
| **Objectives** |

The objective of this course is to introduce scientific python fundamentals.

**Learning Outcomes**

* Proficiency on basic python.
* Skills on Numeric Python (NumPy).
* Knowledge of Scientific Python (SciPy) and usage of basic Machine learning Scientific Kits.
* Abilities in the use of Computer Algebraic Systems under Python.

**Mandatory Contents**

* Introduction to *Python*.
* Introduction to *NumPy*.
* *SciPy* Fundamentals.
* Introduction to *sklearn*.
* Introduction *Sympy* and control library.

|  |
| --- |
| **Skills** |

**Specific:**

**General:**

**Basic:**

**Transversal:**

|  |
| --- |
| **Contents** |

|  |  |
| --- | --- |
| **1. Introduction to Python** | **Dedication**: 25 h  Theory 5 h  Practices/problems: 4 h  Autonomous learning: 16 h |
| **Description:**   1. Introduction    1. Why python?    2. Python History    3. Installing Python    4. Python resources 2. Working with Python    1. Workflow    2. ipython vs. CLI    3. Text Editors    4. IDEs    5. Notebook 3. Getting started with Python    1. Introduction    2. Getting Help    3. Basic types    4. Mutable and in-mutable    5. Assignment operator    6. Controlling execution flow    7. Exception handling 4. Functions and Object Oriented Programming    1. Defining Functions    2. Decorators    3. Writing Scripts and New Modules    4. Input and Output    5. Standard Library    6. Object-oriented programming    7. Magic Functions 5. Iterators and Generators    1. Iterators    2. Generators 6. Creating Graphic Interfaces (optional) 7. Debugging code    1. Avoiding bugs    2. Debugging workflow    3. Python's debugger    4. Debugging segfaults using gdb   **Specific objectives**:  **Linked activities:** Lectures. | |

|  |  |
| --- | --- |
| **2. Introduction to Numpy** | **Dedication**: 19.5 h  Theory 4 h  Practices/problems: 3 h  **Autonomous learning:** 12.5 h |
| **Description:**   1. Introduction to NumPy    1. Overview    2. Arrays    3. Operations on arrays    4. Advanced arrays (ndarrays)    5. Notes on Performance (\%timeit in ipython) 2. Matplotlib    1. Introduction    2. Figures and Subplots    3. Axes and Further Control of Figures    4. Other Plot Types    5. Animations 3. Plotting with Mayavi    1. Mlab: the scripting interface    2. Interactive work 4. Advanced Numpy    1. Life of ndarray    2. Universal functions    3. Interoperability features    4. Array siblings: chararray, maskedarray, matrix    5. Summary    6. Contributing to Numpy/Scipy   **Linked activities**: Lectures and problems/practices sessions. | |

|  |  |
| --- | --- |
| **3. Scientific Python with Scipy and Sympy. Code profiling.** | **Dedication**: 19.5 h  Theory 4 h  Practices/problems: 3 h  **Autonomous learning:** 12.5 h |
| **Description:**   1. Scipy    1. Introduction    2. Input/Output    3. Statistics    4. Linear Algebra    5. Fast Fourier Transforms    6. Optimization    7. Interpolation    8. Numerical Integration    9. Signal Processing    10. Image Processing    11. Special Functions 2. Sparse Matrices in SciPy    1. Introduction    2. Storage Schemes    3. Linear System Solvers    4. Others 3. Optimizing code    1. Optimization workflow    2. Profiling your code    3. Speeding your code 4. Sympy    1. First Steps with SymPy    2. Algebraic manipulations    3. Calculus    4. Equation solving    5. Linear Algebra   **Linked activities:** Lectures and problems/practices sessions. | |

|  |  |
| --- | --- |
| **4. Case Studies on Python applications** | **Dedication**: 11 h  Theory 1 h  Practices/problems: 3 h  **Autonomous learning:** 7 h |
| **Description:**   1. Python scikits    1. Introduction    2. scikit-timeseries    3. scikit-audiolab 2. scikit-learn    1. Datasets    2. Sample generators    3. Unsupervised Learning       1. Clustering       2. Gaussian Mixture Models       3. Novelty/Outliers Detection    4. Supervised Learning       1. Linear and Quadratic Discriminant Analysis       2. Nearest Neighbors       3. Support Vector Machines       4. Partial Least Squeares    5. Feature Selection 3. Practical Introduction to Scikit-learn    1. Solving an eigenfaces problem       1. Goals       2. Data description       3. Initial Classes       4. Importing data    2. Unsupervised analysis       1. Descriptive Statistics       2. Principal Component Analysis       3. Clustering    3. Supervised Analysis       1. k-Nearest Neighbors       2. Support Vector Classification       3. Cross validation 4. Modelling of dynamical systems    1. Control of dynamical systems    2. Guided example with quadrotors   **Linked activities:** Lectures and problems/practices sessions. | |
|  | |

|  |
| --- |
| **References** |

**Basic:**

* *Python Scientific Lecture Notes*, open source project for creating python related teaching material, available at <http://scipy-lectures.github.com/>
* Mark Pilgrim, *Dive Into Python*, <http://www.diveintopython.net/> (FDL licensed book)

**Complementary**:

* Allen B. Downey, Jeffrey Elkne and Chris Meyers, *How to Think Like a Computer Scientist* , Open Book valiable at <http://openbookproject.net/thinkcs/>
* Cody Jackson, *Learning to Program Using python*, published at https://www.createspace.com/3611970
* Hans Petter Langtangen, *A Primer on Scientific Programming with Python*, Springer 2009
* Sandro Tosi, *Matplotlib for Python Developers*, Lightning Source, 2009
* Ivan Idris, *NumPy 1. 5 Beginner's Guide*, Packt Publishing Ltd, 2011