

PYT: Introduction to Python

Descriptive details concerning the subject:

Name of subject: Advanced Programming

Code: PYT

Type of Subject: Optional

ECTS: 5

Total Hours: 125

Scheduling:

Course: 1st Course

Period: 2nd Trimestre

Coordinator: Jordi Villà-Freixa

Department: Experimental and Health Sciences

Teaching Staff:

Language: English

Lecturer: Jordi Villà i Freixa

Department: Experimental and Health Sciences

Language: English

Lecturer: Miguel Hernández

Department: Experimental and Health Sciences

Presentation

Attaining expertise in the python language is the core aim of the subject. This includes comprehensive knowledge of python syntax and idiom along with inbuilt functions and types as well as a familiarity with the various parts of its standard library. Another objective is developing an understanding of advanced code design principals from object oriented techniques like encapsulation and polymorphism to proper documentation and effective variable naming.

The subject is concerned with providing students with the tools to create, understand and maintain complex software in a productive manner. It moves beyond simple programming constructs such as loops and conditionals to introduce and explain higher level coding paradigms and efficient coding practices which enable students to view and manage the long term development of a program. It provides students with the ability to develop not only effective programmatic solutions but also to produce code which is clear and understandable – qualities which have a dramatic impact on productivity.

This subject is essential for any student wishing to write or deploy their own software and critical for one wishing to develop software in a group environment. It also provides an invaluable advantage when tackling the programming problems that will be presented in other subjects.

The subject is completely incremental. Each lecture and exercise follows on builds on the last. It is strongly advised that students keep up to date with class material.

Prerequisites

It is highly recommended that students have at least the competencies provided by the “Introduction to Algorithmics” and “Introduction to Perl”. A solid foundation in fundamental programming concepts e.g. basic types (int, float, string), variables and control constructs (if, for, while) is essential, however previous knowledge of python is not required. A solid understanding of unix filesystem basics is also recommended.

General Competencies

Instrumentals

1. Expertise in procedural Python.
2. Proficient reading and interpretation of python documentation.
3. Ability to produce well written and documented code
4. Understanding object oriented concepts and terminology
5. Ability to factor a problem into objects.
6. Knowledge of how object oriented programming is enabled in python.
7. Ability to apply object oriented concepts to problems using python.
8. Knowledge of different development methodologies.

Interpersonal

1. Ability to solve individually a given problem

Systemic

1. Analysis and syntesis abilities
2. Ability to search for and understand programming documentation

Specific

1. To understand python terminology
2. To know the python built in types
3. To know the python builtin functions
4. To know understand how to interact with the file system
5. To be able to write a script
6. To know how to document functions and modules
7. To have an overview of the python standard libraries
8. To be able to read python documentation
9. To understand and use naming conventions
10. To understand concepts of class, object and method
11. To understand and use inheritance.
12. To understand and use encapsulation.
13. To understand and use polymorphsim.
14. To be familiar with python developement tools.
15. To be familiar with unit tests and test first development

Learning Aims

To have expertise at programming procedural python. To understand object oriented concepts and their implementation in python. To be able to use these concepts to code solutions to problems using python.

General Assessment Criteria

The evaluation will consist on three blocks:

1. continual assesment. Several exercises will be provided over the course of the subject. The exercises will mostly involve the incremental development of a python program. After each exercise is complete the teacher will provide solutions which can be used as the basis for the next exercise by students who failed to complete the previous one.
2. final project. The students will be assigned a tutor who will help them in building a small programming project related with some of the areas of expertise of the master. The student is encouraged to work on the project from the beginning of the course, progressively introducing the new concepts acquired in the classes and with the exercises.
3. Examination. A final examination will be taken to assess the level of knowledge achieved during the course.

Competencies Evaluation	Attainment Indicator	Assessment Procedure	Scheduling
Expertise in Python at the procedural level.	Ability to implement solutions to exercises.	Exercises	Progressive
Proficient reading and interpretation of python documentation.	Implicit in implementing effective solutions to exercises.	Exercises	Progressive
Ability to produce well written and documented code	Code produced for exercises.	Exercises	Progressive
Understanding object oriented concepts and terminology	Ability to implement object oriented solutions to the exercises.	Exercises	Progressive
Ability to factor a problem into objects.	Ability to implement object oriented solutions to the exercises.	Exercises	Progressive
Knowledge of how object oriented programming is enabled in python.	Ability to implement object oriented solutions to the exercises.	Exercises	Progressive
Ability to apply object oriented concepts to problems using python.	Ability to implement object oriented solutions to the exercises.	Exercises	Progressive

Knowledge of different development methodologies	Specific Exercises	Exercises	Progressive
Ability to solve individually a given problem	Implicit in developing a solution to an exercise.	Exercises	Progressive
Analysis and synthesis abilities	Implicit in developing a solution to an exercise.	Exercises	Progressive
Ability to search for and understand programming documentation	Implicit in developing a solution to an exercise.	Exercises	Progressive
To understand python terminology	Implicit in understanding an exercise.	Exercises	Progressive
To know the python built in types	Ability to implement solutions to exercises.	Exercises	Progressive
To know the python builtin functions	Ability to implement solutions to exercises.	Exercises	Progressive
To know understand how to interact with the file system	Ability to implement solutions to exercises.	Exercises	Progressive
To be able to write a script	Ability to implement solutions to exercises.	Exercises	Progressive
To know how to document functions and modules	Exercise code	Exercises	Progressive
To have an overview of the python standard libraries	Ability to implement solutions to exercises.	Exercises	Progressive
To be able to read python library documentation	Implicit in developing a solution to an exercise.	Exercises	Progressive
To understand and use naming conventions	Exercise code	Exercises	Progressive
To understand concepts of class, object and method.	Implicit in understanding exercise focused on object oriented programming.	Exercises	Progressive
To understand and use encapsulation.	Ability to implement object oriented solutions to the	Exercises	Progressive

	exercises.		
To understand and use inheritance.	Ability to implement object oriented solutions to the exercises.	Exercises	Progressive
To understand and use polymorphism.	Ability to implement object oriented solutions to the exercises.	Exercises	Progressive
To be familiar with python development tools.	Implicit in developing a exercise solutions.	Exercises	Progressive
To be familiar with unit tests and test first development	Ability to complete specific exercise.	Exercises	Progressive
To understand and be able to use exceptions	Ability to complete specific exercises.	Exercises	Progressive

Contents

The topics covered by block 2 and 5 are introduced gradually throughout the course in parallel with the other blocks.

Block 1 – Introduction to Python

Concepts	Procedures	Attitudes
Modules. Variable binding. Object properties (<code>__doc__</code> , <code>__name__</code>). Documentation strings. Naming conventions. Built in types – int, float, list, dictionaries, strings and files. String formatting. Python control structures. Defining functions. Introduction to the built in functions. Concept of a method. IDLE.	A number of simple exercises which involve writing modules containing functions using python built in types and investigating their properties. All functions and modules must be properly documented. Learn to use python versions of control structures. Learn how to tie the simple functions together in a executable script. Demonstrate the special features of IDLE which reward proper documentation.	Achieve a basic understand of python modules. To be able to understand how python handles variable assignment. To understand the concept of object properties. To become familiar with the python built in data types and control structures. To understand and use the python documentation system and its advantages. To be introduced to consistent and clear naming conventions. To learn how to write a python script, To

Block 2 – Python Libraries

Concepts

Importing modules. Important and useful python standard modules covering string services, mathematics, data types, file and directory access, general operating system modules and data persistence. Reading python documentation. Third party libraries – database access, numeric, scipy, biopython.

Procedures

To expand the code exercise by exercise by implementing functionality requiring use of external libraries. Libraries are introduced step by step starting with simple and/or important modules (sys, os, random) and advancing to more complex and specialised one by the end of the course, (pickle, popen, urllib)

Attitudes

To become familiar with the various parts of the python standard library and what they can do. To understand how to interact with the file system. To understand how to get system information. To understand how to perform basic mathematical operations. To learn how to store data. To learn how to find and read module documentation. To have an overview of what third party modules are available.

Block 3 – Introduction to Object Oriented Programming

Concepts

Object, class, instance, instantiate, instance variable, initialisation, member, subclass, superclass, method.

Procedures

Simple exercises which introduce core concepts of “class”, “object” and “inheritance”. Connecting previous work with examining object properties and using built in objects (list, dict) with the technical terminology of object oriented programming.

Attitudes

To learn what a class is and its use. To learn how inheritance works and how its used. To begin to use object oriented terminology when talking about previously introduced types e.g. lists and dictionaries. To be able to create classes and use them.

Block 4 – Object Oriented Design

Concepts

Encapsulation, interfaces, persistence and the key concept of polymorphism.

Procedures

Refactoring previously written code to be object oriented with an emphasis on encapsulation. Extending this code with new behaviour that uses these new concepts – object persistence

Attitudes

To understand the concept of encapsulation. To understand the concept of a class interface. To become familiar with the concept of programming to an interface. To be able to factor a

using pickle, interactive command sessions using the cmd() modul – and which also illustrate how object orientated programming simplifies many problems. To move completely to using object oriented terminology so it becomes second nature.

problem into objects. To understand the concept of polymorphsim and to have an idea of its power. To become more familiar with thinking about problems in a object oriented fashion.

Block 5 – Advanced Development Techniques

Concepts

Unit Tests, Plugins, exceptions, python debugger, profiling.
High level module organisation.

Procedures

Introduction to the use to the python debugger for analysing problems. Using the profiler to find code bottleneck while emphasising the dangers of premature optimisation.
Updating excercise code to use exceptions for error handling.
Writing unit tests for a previously implemeneted class.
Adopting a “test first” development strategy for one exercise. Explanation of how modules can be arranged above file level.
Demonstrations of how code can be extended through plugins.

Attitudes

To understand when and how to use the python debugger and profiler. To understand and be able to implement an exception based error handling strategy. To be able to write unit tests and understand their importance. To be exposed to a “test first” development strategy and to understand its benifits.
To have an overview about how a plugin architecture can be implemented. To understand how to use modules to organise large code bases.

Further reading:

<http://www.greenteapress.com/thinkpython/>