# Syllabus

## Introduction to Python

Andres Mendez-Vazquez

October 17, 2016

## Contents

## Preliminaries:

**Course Objectives:**

To teach the basic on Python in order to start introducing the people in the basic concepts of Data Sciences.

**Prerequisites:**

1. Knowledge in Java, C++, Data Structures (At least basics in Big O notation).
2. Machine with
   (a) Python 2.7
   (b) Spyder
   (c) Jupyter Notebook

# 1 Introduction

## 1.1 History of Python

1. Python was conceived in the late 1980s.
2. By Guido Van Rossum
   (a) Benevolent dictator for life!!!
3. Python reached version 1.0 in January 1994.
4. The major new features included in this release were the functional programming tools:
   (a) lambda - the basic lambda calculus implementation.
   (b) map - it applies a given function to each element of a list.
   (c) filter - it filters out items based on a test function.
   (d) reduce - it is a really useful function for performing some computation on a list and returning the result.
5. Standard Implementations
   (a) CPython
   (b) Jython (1997)
   (c) IronPython (2006) - .NET
   (d) PyPy (2007) written in RPython
6. Exotic Implementation
   (a) Stackless Python (2000) does not depend on the C call stack.

## 1.2 Interpreted or Compiled

1. Putting Stuff in perspective
   (a) Compilation is not restricted to ahead-of-time compilation to native machine code.
2. Practically Python is compiled to bytecode

## 1.3 The Memory Management System

Memory management in Python involves a private heap containing all Python objects and data structures.

The Python memory manager has different components:

1. At the lowest level, a raw memory allocator (Interacting with the OS) ensures that there is enough room in the private heap for storing all Python-related data.
2. On top of the raw memory allocator
   (a) Several object-specific allocators operate on the same heap.
   (b) They implement distinct memory management policies adapted to every object type.

Here, the quote "What does it mean for everything to be an object?" comes to mind. It is means that

1. Types have member functions,
2. Functions have attributes,
3. Modules can be passed as arguments,
4. etc.

Thus, you have a hierarchy for the variable names (Fig. 1) once everything is an object.

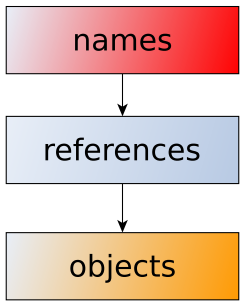

Figure 1: The Reference Name Hierarchy

### 1.4 Garbage Collector (GC) in Python

Here, we would talk about

1. Generic Algorithms for Garbage Collection.
2. CPython Implementation
3. Cycles, Finalizers and Uncollectables
4. Architecture of the GC-as-a-Service library

### 1.5 The Interpreter

Now, we are ready to look at the interpreter...

- This is a classic simply type # python

Here, we have that we can talk of

1. Assign a variable to a value.
2. Another variable to a variable.
3. The use of “del” with the variables - it allows reference count to decrease
   (a) Related to the Garbage Collector
   (b) Some Basic Formating

## 2 The Basic Structures

### 2.1 Variables

There is a list of things that need to be covered:

1. Assigning Values to Variables.
2. Multiple Assignment
3. Standard Data Types
   (a) Numbers
   (b) String
   (c) List
   (d) Tuple
   (e) Dictionary

## 2.2 Python Basic Operators

Here, we need to review:

1. Arithmetic Operators
2. Comparison (Relational) Operators
3. Assignment Operators
4. Logical Operators
5. Bitwise Operators
6. Membership Operators
7. Identity Operators

## 2.3 Control Flow Structures

1. The If Statement
2. Loop Instructions
   (a) For Instruction
   (b) While Instruction
   (c) Techniques
3. Else in loops
4. The range instruction
5. break and continue Statements
6. pass Statements

## 2.4 Defining Functions

Here, we classically define the syntax for functions using:

1. Fibonacci function

Then, we look at

1. Default Argument Values
2. Keyword Arguments
3. Arbitrary Argument Lists

In addition, we look at small expressions using lambda expressions.

# 3 Basic Data Structures in Python

## 3.1 Lists

Review the most basic functions for a list:

1. append(x)
2. extend(x)
3. remove(x)
4. pop()
5. index(x)
6. count(x)
7. sort()
8. reverse()

Then given all this operations it is possible to use a list as

1. Stacks
2. Queues

Here, you need to be careful because the emulation vs native implementation.

## 3.2 Tuples

Here, we will review the Tuple.

## 3.3 Sets

They are an unordered collection with no duplicate elements

## 3.4 Dictionaries

Another useful data type built into Python is the dictionary. It can be seen as an unordered set of key:value pairs. Here, we will see how to substitute the case statement by a dictionary

## 3.5 Hash Tables

Here, we will review some of the basics in

1. Hash Functions
2. Hash Tables

## 3.6 Trees

As you can imagine trees are an evolution from the chain lists. Thus, we will review how to :

1. Build a tree.
2. Traverse a tree
3. Expression trees

## 3.7 Graphs

Look at the basics on how building graphs by using lists and dictionaries. Then, look at

1. Breath First Search
2. Depth First Search

## 3.8 Divide and Conquer

Here, review the basics of divide and conquer using

1. Merge Sort
2. Quick Sort

# 4 Modules in Python

A module is a file containing Python definitions and statements.

- The file name is the module name with the suffix .py appended.
- The module's name is available as a value of the global variable ___name___.

## 4.1 Importing functions in Python

Here, we talk about how import modules already in the package list and user modules. Here, we need to talk about

- Execute modules as scripts
- PYTHONPATH
- Compilation of Python Programs

## 4.2 Standard Modules

Here, we need to review some of the useful functions from sys module.

### 4.3 The dir() function

The built-in function dir() is used to find out which names a module defines.

### 4.4 Packages

Packages are a way of structuring Python's module namespace. We need to teach how to structure the namespace

## 5 IO Operations in Python

Give the different ways to present the outputs.

### 5.1 Formating the Output

The classic way to present the output is by

- Expression statements
- The print statement

Together with the string format allows really fancy outputs.

### 5.2 Reading and Writing Files

Here, we will see how to open files to read them and do something with the input. In our specific case, convert structures int json.

## 6 Errors

Here,we need to review:

1. Error Syntaxes
2. Exceptions
3. Handing Exceptions

## 7 Regular Expressions

Here, we will review the module "re" for

- Simple Patterns
- Matching Characters
- Compiling Regular Expressions - Quite interesting.

Thus, we need to explore:

1. match function
2. search function
3. search and replace
4. Modifiers

# 8 Functional Programming - Lambda Calculus - in Python

Review the difference between:

1. Procedural
2. Declarative
3. Object-Oriented
4. Functional

## 8.1 Provability

Say something about the formal provability on functional languages.

## 8.2 Iterators

One of the key stones for the functional programming.

## 8.3 Generator

Common operations on an iterator's output are:

1. Performing some operation for every element.
2. Selecting a subset of elements that meet some condition.

A way to support those operations we have list comprehensions and generator expressions.

## 8.4 Buil-in Functions

Review the

1. map function
2. filter function

### 8.5 Lambda Expressions

One way to write small functions is to use the lambda statement. These are small functions taking a series of parameters and a way to combine these parameters.

# 9 Classes and Objects in Python

Though some people consider OOP to be a modern programming paradigm, the roots go back to 1960s with Simula67. The four major principles of object orientation are:

1. Encapsulation
2. Data Abstraction
3. Polymorphism
4. Inheritance

## 9.1 Python Scopes and Namespaces

It is necessary to review the following concepts

- A namespace is a mapping from names to objects.
  - There is absolutely no relation between names in different namespaces.
  - The namespace built-in names is created when the Python interpreter starts up, and is never deleted.
  - The local namespace for a function is created when the function is called, and deleted when the function returns or raises an exception.
- A scope is a textual region of a Python program where a namespace is directly accessible. You have four levels:
  - The innermost scope

## 9.2 Class Syntaxis

It is quite simple:

```
class ClassName:
    <statement-1>
    <statement-2>
     .
     .
     .
     .
    <statement-N>
```

## 9.3 Class Objects

Class objects support two kinds of operations: attribute references and instantiation.

1. Attribute references use the standard syntax used for all attribute references in Python: obj.name.
2. An instance is an object in memory. Basically you create object and instantiate them when you are using them.

## 9.4 Method Objects

It refers to the method after the instantiation. We will look at the syntaxes and other issues.

## 9.5 Class and Instance Variables

Generally speaking, instance variables are for data unique to each instance and class variables are for attributes and methods shared by all instances of the class.

## 9.6 Inheritance

First look at the class syntaxes:

```
class DerivedClassName(BaseClassName):
```

Here, we need to talk about :

1. Instantiation
2. Overriding
3. Checking inheritance
4. The limited multiple inheritance