EPAM Python Software Engineer Training

Lesson 2: Introduction

Contents

1 Course

1.1 What is python

Python is one of the mainstream programming languages nowadays, placed as the 4th most popular language after Java, C#, and PHP. It can be used for a wide range of solutions including (most common):

- all sorts of web applications;
- as a high-level language for back-end solutions;
- all sorts of emulators;
- scintific applications;
- (seldom) desktop applications;

The main advantage of python is its readability that none of other languages provides. Still it is quite efficient for certain application domains and has a large active community. It is typically pre-installed on most modern Linux flavors. A (FreeBSD + C + Python) or (Linux + Python) is a wide-spread combo in many complex high-load high-availability solutions.

Please, visit an Official Site for more information.

1.2 Python flavors

An official python interpretter is *CPython*; at the same time it is the most wide-spread flavor due to its long history and stability.

There are also a lot of alternative python interpretters which are either specialized for specific application domains or show better performance:

- *IronPython*: a MicroSoft flavor for .Net;
- *Jython*: an Oracle flavor for JVM;
- *PyPy*: a python interpretter written in python; often it is faster than *CPython* but is quite young, hence, considered unstable;

- *stackless*: a python interpretter with cooperative multi-threading and minimal stack; often is better scalable than *CPython* with native thread support;

It is so easy to customize python interpretter that there are a lot of proprietary modification of it around.

1.3 C support

Python is well-known for its support for C/C++ libraries. In general any shared library can be imported as a regular python module if that library provides a specific "pythonish" interface. Many builtin python modules are in fact shared libraries. This property is typically leveraged to offload performance critical algorithms to C (e.g. massive string operations).

There are three ways to include a non-pythonish shared library into python:

- through a *ctypes* module:

  Just dynamically load all necessary types and functions as appropriate python objects. It is the fastest way to test some library.

- through a *C* wrapper library:

  Write a pythonish wrapper for non-pythonish shared library. Usually this solution shows the best performance, but is hard to maintain.

- through a *Cython* wrapper module:

  Cython is a programming language - a mix of *C* and *Python*. Also, *Cython* is a compiler producing a pythonish *C* module off a python-like code; later that *C* module is compiled into a pythonish shared library. This is the most elegant yet efficient method. However, not all python flavors are supported (e.g. *PyPy* is not).

1.4 Useful tools

Every python developer must know how to install python and a few useful tools:

- *pip*: a modern application to install packages from a python package index (PyPI); and official PyPI is *http://pypi.python.org/*. On Linux it is typically installed as a *python-pip* Linux package.
- *setuptools*, *distutils*, *distribute*: popular python packaging libraries; *distribute* is considered a road-map for the nearest future. This and below are installed as a *pip* package. See lesson X.
- *docutils*: a standard python documentation package (see Lesson 2).
- *virtualenv*: a common package to create virtual development environments.

1.5 Style

Please follow PEP8 and a Google Python Style Guide. The latter one take precedence. Main rules are: always use 4 spaces for indentation, single quotes for string literals, 80 characters as line limit.

Any IDE can be used; I am using *Vim*. Anyway, please, make sure that your IDE never puts a <*TAB*> character and never leaves a trailing white space.

2 Tasks

2.1 Installation

Install *python*, *pip*, and *docutils* on your computer.

2.2 Workspace

For each lesson create a folder named `lesson#` under which for each task create a file named `task#.ext` where `#` is a lesson/task number and `ext` is dependent onto a task (`py` in most cases).