Modular Programming

Lect. PhD. Arthur Molnar

Babes-Bolyai University

arthur@cs.ubbcluj.ro

Overview

1 Modular Programming

- Introduction
- Python Modules
- Python Packages
- Modular programming in Lab3-4
- The Eclipse IDE
- How to approach Assignment 3-4

Modules

Modular programming - a software design technique that increases the extent to which software is composed of independent, interchangeable components called **modules**, each of which accomplishes one aspect within the program and contains everything necessary to accomplish this.

Modules

Modules are:

- Independent
- Interchangeable

Modules

Discussion

Why is modular programming needed? Advantages and drawbacks...

Modules

- Allows grouping related functionalities
- Allows easier delivery and deployment of related functionalities
- Helps with solving naming conflicts

Modules in Python

A Python module[1] - a file containing Python statements and definitions (executable statements).

- **Name**: The file name is the module name with the suffix " .py" appended
- **Docstring**: triple-quoted module doc string that defines the contents of the module file. Provide summary of the module and a description about the module's contents, purpose and usage.
- **Executable statements**: function definitions, module variables, initialization code

[1] https://docs.python.org/3/tutorial/modules.html

Importing modules

In order to use a module it must be imported first. The import statement:

1. Searches the global namespace for the module. If the module exists, it is already imported and nothing more needs to be done.
2. Searches for the module.
3. Variables and functions defined in the module are inserted into a new symbol table (a new namespace). Only the module name is added to the current symbol table

Module search path

Where does the **'import spam'** statement search for module *spam.py*?

- Built-in modules with the given name
- Directories in the *sys.path* variable:
    - Directory containing the input script
    - Directories specified by environment variable **PHYTONPATH**
    - Directories specified by the environment variable **PYTHONHOME**, an installation-dependent default path

If the module name can't be found anywhere, an **ImportError** exception is raised.

Demo

Modules

ex11_modules.zip

Demo

Environment Variables

This website has more info on accessing and changing environment variables in the Windows OS - *www.computerhope.com/issues/ch000549.htm*

Learning more about modules

- **dir(module_name)** can be used to examine the module's symbol tables.
- **help(module_name)** can be used to get help on the module, its data types and functions.
- **pydoc** - A module that allows you to save extracted documentation to HTML format. Best used in command line at the operating system prompt.

Packages

- Packages[2] are a way of structuring Python's module namespace by using "dotted module names"
- **A.B** denotes submodule **B** found in package **A**.
- The same rules apply for importing packages as with modules
- On the drive, directory hierarchies represent packages, so **B.py** will be found in a directory called **A**
- Each package directory contains an *__init__.py* file, telling Python to interpret it as a collection of modules
- *__init__.py* can be empty, or include package initialization code.

[2] https://docs.python.org/3/tutorial/modules.html#packages

Required modules for Lab3-4

Create modules for:

- **User interface** - Functions related to user interaction. Contains input and data validation, print operations. This is the only module where input/print operations are present.
- **Functions** - Contains functions required to implement program features
- **Application coordinator** - Initialize the UI and start the application.

The Eclipse IDE

- Eclipse is an advanced IDE that is free (as in "free beer"[3]), configurable and easy to use
- Provides lots of plugins to allow development in many languages, including Java, C/C++, Python...
- What you should be familiar with
  - Setting it up for Python development
  - Working with projects, navigating and editing source files and program resources (e.g. text files)
  - Running project configurations, debugging, running tests

[3] http://www.gnu.org/philosophy/free-sw.html

Eclipse + PyDev

- By default, Eclipse can be used to develop Java software
- To develop in Python, you need the PyDev[4] plugin
- PyDev links Eclipse to the installed Python interpreted and libraries, gives you wizards for project creation, syntax highlighting, code completion and Python-specific features

[4] http://www.pydev.org/

Installing Eclipse + PyDev

1. Install Java 8 JDK[5]
2. Install Eclipse IDE for Java Developers[6]
3. Start Eclipse for the first time, see it works
4. Install PyDev plugin[7]
5. Configure Eclipse with the installed version of Python
6. Start coding!

[5] http://www.oracle.com/technetwork/java/javase/downloads/jdk8-downloads-2133151.html

[6] http://www.eclipse.org/downloads/eclipse-packages/

[7] Section "Installing with the update site" - http:www.pydev.org/manual_101_install.html

Demo

Code review

The code in the following archive is a modular implementation of the calculator program for rational numbers: **ex12_calculatorModular.zip**

How to approach Assignment 3-4

- You must solve all requirements for Assignment 3-4 in 2 successive iterations
- You have 1 week time for every iteration, with first one due in week 4
- Each iteration consists of 3 features groups (there are 6 in total for every problem statement)
- When an iteration is complete, certain functionalities must be presented to your client (that's us :-))
- This allows checking that the project is turning out in accordance with client expectation