Programming and Problem Solving through Python Language
O Level / A Level

Chapter -3 : Introduction to Python Language

Python is a high-level general purpose programming language:

- Because code is automatically compiled to byte code and executed, Python is suitable for use as a scripting language, Web application implementation language, etc.
- Because Python can be extended in C and C++, Python can provide the speed needed for even compute intensive tasks.
- Because of its strong structuring constructs (nested code blocks, functions, classes, modules, and packages) and its consistent use of objects and object-oriented programming, Python enables us to write clear, logical applications for small and large tasks.

Varieties of Python:

- CPython -- Standard Python 2.x implemented in C.
- Jython -- Python for the Java environment -- http://www.jython.org/
- PyPy -- Python with a JIT compiler and stackless mode -- http://pypy.org/
- Stackless -- Python with enhanced thread support and microthreads etc. -- http://www.stackless.com/
- IronPython -- Python for .NET and the CLR -- http://ironpython.net/
- Python 3 -- The new, new Python. This is intended as a replacement for Python 2.x. -- http://www.python.org/doc/.

Names and tokens

- Allowed characters: a-z A-Z 0-9 underscore, and must begin with a letter or underscore.
- Names and identifiers are case sensitive.
- Identifiers can be of unlimited length.
- Special names, customizing, etc. -- Usually begin and end in double underscores.
- Special name classes -- Single and double underscores.
    - Single leading single underscore -- Suggests a "private" method or variable name. Not imported by "from module import *".
    - Single trailing underscore -- Use it to avoid conflicts with Python keywords.
    - Double leading underscores -- Used in a class definition to cause name mangling (weak hiding). But, not often used.
- Naming conventions -- Not rigid, but:
    - Modules and packages -- all lower case.
    - Globals and constants -- Upper case.
    - Classes -- Bumpy caps with initial upper.
    - Methods and functions -- All lower case with words separated by underscores.
    - Local variables -- Lower case (with underscore between words) or bumpy caps with initial lower or your choice.
    - Good advice -- Follow the conventions used in the code on which you are working.
- Names/variables in Python do not have a type. Values have types.

Comments

- Everything after "#" on a line is ignored.
- No block comments, but doc strings are a comment in quotes at the beginning of a module, class, method or function.
- Editors with support for Python often provide the ability to comment out selected blocks of code, usually with "##".
- A doc string is written as a quoted string that is at the top of a module or the first lines after the header line of a function or class. We can use triple-quoting to create doc strings that span multiple lines.

Blocks and indentation

- Python represents block structure and nested block structure with indentation, not with begin and end brackets.
- The empty block -- Use the pass no-op statement.
- Benefits of the use of indentation to indicate structure:
    - Reduces the need for a coding standard. Only need to specify that indentation is 4 spaces and no hard tabs.
    - Reduces inconsistency. Code from different sources follow the same indentation style. It has to.
    - Reduces work. Only need to get the indentation correct, not both indentation and brackets.
    - Reduces clutter. Eliminates all the curly brackets.
    - If it looks correct, it is correct. Indentation cannot fool the reader.

Lines

- Statement separator is a semi-colon, but is only needed when there is more than one statement on a line. And, writing more than one statement on the same line is considered bad form.
- Continuation lines -- A back-slash as last character of the line makes the following line a continuation of the current line. But, note that an opening context" (parenthesis, square bracket, or curly bracket) makes the back-slash unnecessary.

Program structure

- Execution -- def, class, etc are executable statements that add something to the current name-space. Modules can be both executable and import-able.
- Statements, data structures, functions, classes, modules, packages.
- Functions
- Classes
- Modules correspond to files with a "*.py" extension. Packages correspond to a directory (or folder) in the file system; a package contains a file named "__init__.py". Both modules and packages can be imported
- Packages -- A directory containing a file named "__init__.py". Can provide additional initialization when the package or a module in it is loaded (imported).