Chapter 1 - Introduction to Python

Python is a very **High Level, object-oriented, dynamic1, strong typing2, interpreted** & **interactive** programming language.

1. Dynamic Programming language: It at runtime executes many common programming tasks which static programming languages perform during compilation such as

- Computation of code at runtime and late binding
- alteration of Objects at runtime
- Assembling of code at runtime based on the class of instances

2. Strongly Typed: Typing errors are prevented at runtime using the implicit type conversion, also it don't have static type checking, i.e. compiler don't check or enforce type constraint rules. The term **duck typing** is now used to describe the dynamic typing paradigm. Duck typing is an application of the duck test in type safety. It requires that type checking be done at runtime only, and is implemented by use of dynamic typing or by reflection.

It is also an open source language (with license compatible with the *General Public License (GPL)*, but less restrictive, allowing Python to be even incorporated into proprietary products). Its specification is maintained by the Python Software Foundation (PSF).

Key Features

- It uses an elegant syntax, making the code easier to read
- Easy-to-use language that makes it simple to get code working. This makes it ideal for prototype development and other ad-hoc programming tasks, without compromising maintainability
- Default installation contains large standard library which supports most of the common programming tasks, such as connecting to web servers, searching text with regular expressions, reading and modifying files etc
- Python's interactive mode, makes it easy to validate snippets of code.

- Bundled development environment called IDLE.
- Is easily extended by adding new modules implemented in a compiled language such as C or C++.
- Can also be embedded into an application to provide a programmable interface.
- Runs anywhere, including Mac OS X, Windows, Linux, and Unix.
- Is free software in two senses. It doesn't cost anything to download or use Python, or to include it in your application. It can be freely modified and re-distributed, because while the language is copyrighted it's available under an open source license.

Some programming-language features of Python are:

- Many basic data types:
- numbers (floating point, complex, and unlimited-length long integers)
- strings (both ASCII and Unicode)
- Collections (lists, dictionaries)
- Python supports object-oriented programming with classes and multiple inheritance.
- Code can be grouped into modules and packages.
- The language supports raising and catching exceptions, resulting in cleaner error handling.
- Data types are strongly and dynamically typed. Mixing incompatible types (e.g. attempting to add a string and a number) causes an exception to be raised, so errors are caught sooner.
- Python contains advanced programming features such as generators and list comprehensions.
- Python's automatic memory management frees you from having to manually allocate and free memory in your code.

It is possible to integrate Python with other languages such as C and Fortran. In general terms, it has many similarities with other dynamic languages such as Perl and Ruby.

History

The language was created in 1990 by **Guido van Rossum**, at National Research Institute for Mathematics and Computer Science in the Netherlands (CWI) and had originally focused on users as physicists and engineers. Python was designed from another existing language at the time, called ABC. *Guido van Rossum*

Versions

The official implementation of Python is maintained by the PSF and written in C, and therefore is also known as CPython. The latest stable version is available for download at:

http://www.python.org/download/

For Windows platforms, simply run the installer. For other platforms, such as Linux, Python is usually already part of the system, but in some cases it may be necessary to compile and install the interpreter from the source files.

There are also implementations of Python for. NET (IronPython), JVM (Jython) and Python (PyPy).

Running programs

Example of Python program:

```
# the character "#" indicate that rest of the line is a comment
# and will be ignored by the interpreter

# A list of musical instruments
instruments = ['Drums', 'Flute', 'Harmonium', "Guitar"]

# for each instrument in the list of instruments
for instrument in instruments:
    print (instrument)
```

In above example, `instruments` is a list containing the items “Drums”, “Flute”, “Harmonium” and “Guitar” and as the `for` loop is executed `instrument` corresponds to, an item from items on the list, one at a time.

Executing the code

The source files are usually identified by the extension “.py” and can be run directly by the interpreter:

```
python apl.py
```

Thus `apl.py` will run. On Windows, the file extensions “.py”, “. pyw”, “. pyc” and “. pyo” are associated with Python automatically during installation, so just click a the file to run it. The “. pyw” files run with an alternate version of the interpreter that does not open the console window.

Dynamic Typing

Python uses dynamic typing, which means that the type of a variable is inferred by the interpreter at runtime (this is known as *Duck Typing*). By the time a variable is created by attribution the interpreter defines the type of a variable, along with the operations that can be applied.

Typing of Python is strong, ie, the interpreter checks whether the transactions are valid and does automatic coercions between incompatible types. In Python, coercions are performed automatically only between types that are clearly related, as integer and long integer. To perform the operation between non-compatible types, you must explicitly convert the type of the variable or variables before the operation.

Compilation and interpretation

The source code is translated by Python to bytecode, which is a binary format with instructions for the interpreter. The bytecode is cross platform and can be distributed and run without the original source.

By default, the parser compiles the code and stores the bytecode on disk, so the next time you run it, there is no need to recompile the program, reducing the load time of execution. If the source files are changed, the interpreter will be responsible for regenerating the bytecode automatically, even using the *interactive shell*. When a program or a module is invoked, the interpreter performs the analysis of the code, converts to symbols, compiles (if there is no updated bytecode on disk) and runs it in the Python virtual machine.

The bytecode is stored in files with the extension ". pyc" (normal bytecode) or ". pyo" (optimized bytecode). The bytecode can also be packaged along with an executable interpreter, to facilitate the distribution of the application, eliminating the need to install Python on each computer.

Interactive Mode

The Python interpreter can be used interactively, where lines of code are typed into a *prompt* (command line) *shell* similar to the operating system.

```
python
```

It is ready to receive commands after the appearance of the signal `>>>` on the screen:

```
Python 2.6.4 (r264:75706, Nov  3 2009, 13:20:47)    [GCC 4.4.1] on
linux2   Type "help", "copyright", "credits" or "license" for more
```

`information. >>>`

On Windows, the interactive mode is also available via the icon "Python (command line)".

The interactive mode is a distinguishing feature of the language, as it is possible to test and modify code snippets before inclusion in programs, to extract and convert data or even analyze the state of the objects in memory, among other possibilities.

Besides the traditional interactive mode of Python, there are other programs that act as alternatives to more sophisticated interfaces (such as PyCrust): [PyCrust] (files/pycrust.png)

Common IDE & Tools

There are many development tools for Python, such as IDEs, editors and shells (that take advantage of the interactive capabilities of Python).

Integrated Development Environments (IDEs) are software packages that integrate various development tools in an environment consistent with the goal of increasing developer productivity. Generally, IDEs include such features as syntax highlighting (colorized source code according to the syntax of the language), source browsers, integrated shell and *code completion* (the editor presents possible ways to complete the text it can identify while typing). Among Python IDEs, there are most popular ones:

- PyScripter
- Atom
- SPE (Stani's Python Editor)
- Eric
- PyDev (plug-in for Eclipse IDE)
- vim
- Sublime Text

  [PyScripter] (files/pyscripter.png)

Entire list

(from https://wiki.python.org/moin/IntegratedDevelopmentEnvironments?action=show&redirect=IDE)

Name	**Platform**	Updated	**Notes**
Thonny	Windows, Linux, Mac OS X, more	2016	For teaching/learning programming. Focused on program runtime visualization. Provides stepping both in statements and expression s, no-hassle variables view, separate mode for explaining references etc.

Nam e	**Pla tform**	Upd ated	**Not es**
Komod o	Windo ws/Li nux/M ac OS X	2012	Multi -lang uage IDE with suppo rt for Pytho n 2.x and Pytho n 3. Avail able as Komod o IDE (comm ercia l).

Nam e	**Pla tform**	Upd ated	**Not es**
LiCli pse	Linux /Mac OS X/Win dows	2015	Comme rcial Eclip se-ba sed IDE which provi des a stand alone bundl ing PyDev , Works pace Mecha nic, Eclip se Color Theme , Start Explo rer and AnyEd it, along with light weigt h suppo rt for other langu ages, and other usabi lity enhan cemen ts (such as multi -care t-edi tion) .

Nam e	**Pla tform**	Upd ated	**Not es**
NetBe ans	Linux , Mac, Solar is, Windo ws	2009	Pytho n/Jyt hon suppo rt in NetBe ans – Open sourc e, allow s Pytho n and Jytho n Editi ng, code- compl etion , debug ger, refac torin g, templ ates, synta x analy sis, etc.; see also http: //wik i.net beans .org/ Pytho n. UP- DAT E: Netbe ans 7.0 relea sed witho ut Pytho n suppo rt. Check http: //wik i.net beans .org/ Pytho n70Ro

Name	Platform	Updated	Notes
PyCharm	Linux/Mac OS X/Windows	2014	Free open-source IDE with a smart Python editor providing quick code navigation, code completion, refactoring, unit testing and debugger. Has a commercial Professional edition that fully supports Web development with Django, Flask, Mako and Web2Py and allows to develop remotely. Free PyCharm professional

Name	**Platform**	Updated	**Notes**
Python for VS Code	Linux/Mac OS X/Windows	2016	Free open-source extension for Visual Studio Code. Supports syntax highlighting, debugging, code completion, code navigation, refactoring, with support for Django, multi threaded, local and remote debugging.

Nam e	**Pla tform**	Upd ated	**Not es**
KDeve lop	Linux /Mac OS X/(Wi ndows)	2014	Free open-sourc e IDE with a focus on stati c analy sis-b ased code compl etion , navig ation and highl ighti ng. Also featu res a VI emula tion mode.

Nam e	**Pla tform**	Upd ated	**Not es**
PyDev	Eclip se	2015	Free, open-sourc e plugi n for Eclip se – Allow s Pytho n, Jytho n, and IronP ython editi ng, code-compl etion , debug ger, refac torin g, quick navig ation , templ ates, code analy sis, unitt est integ ratio n, Djang o integ ratio n, etc.

Name	**Platform**	Updated	**Notes**
Wing IDE	Windows, Linux, Mac OS X	2016	Commercial Python IDE with advanced debugger, editor with vi, emacs, visual studio and other key bindings, auto-completion, auto-editing, snippets, goto-definition, find uses, refactoring, unit testing, source browser, and much more. There are several product levels, including free and paid versions with a

Name	**Platform**	Updated	**Notes**
PyScripter	Windows	2012	MIT licensed IDE written in Delphi with debugger, integrated unit testing, source browser, code navigation and syntax coloring/auto-completing editor.

Name	Platform	Updated	Notes
Pyshield	Windows, Linux	2010	Commercial IDE tool used to edit, debug Python script, publish encrypted scripts, build a standalone executable file, manage more files by project view, and make installation in various forms (.msi, .tar.gz, .rpm, .zip, .tar.bz2). It includes an editor simulating Emacs python-mode, a GUI debugger simulating GDB, a

Nam e	**Pla tform**	Upd ated	**Not es**
Spyde r	Windo ws/Li nux/M ac OS X	2012	Free open- sourc e scien tific Pytho n devel opmen t envir onmen t provi ding MATLA B-lik e featu res: conso le with varia ble brows er, sys.p ath brows er, envir onmen t varia bles brows er, integ rated plott ing featu res, autoc omple tion and toolt ips -edito r with synta x highl ighti ng, class /func tion

Name	Platform	Updated	Notes
IDLE	Windows/Linux/Mac OS X/All Tk Platforms	2009	Multi-window colorized source browser, autoindent, autocompletion, tooltips, code context panel, search in files, class and path browsers, debugger, executes code in clean separate subprocess with one keystroke. 100% pure Python, part of Python 2.x and 3.x distributions.

Nam e	**Pla tform**	Upd ated	**Not es**
IdleX	Windo ws/Li nux/M ac OS X/All Tk Platf orms	2012	IdleX is a colle ction of over twent y exten sions and plugi ns that provi de addit ional funct ional ity to IDLE, a Pytho n IDE provi ded in the stand ard libra ry. It trans forms IDLE into a more usefu l tool for acade mic resea rch and devel opmen t as well as explo rator y progr ammin g.

Nam e	**Pla tform**	Upd ated	**Not es**
µ.dev	Windo ws (need s to be compi led manua lly for other platf orms)	2010	An open- sourc e IDE, creat ed using Lazar us. It's only for Pytho n. inclu de synta x highl ighti ng, proje ct manag er, and uses pdb for debug ging.

Nam e	**Pla tform**	Upd ated	**Not es**
Pyzo (form erly IEP)	Windo ws/Li nux/M ac OS X	2016	Open- sourc e Pytho n IDE focus ed on inter activ ity and intro spect ion, which makes it very suita ble for scien tific compu ting. Its pract ical desig n is aimed at simpl icity and effic iency . Pyzo consi sts of two main compo nents , the edito r and the shell , and uses a set of plugg able tools to help the progr

Nam e	**Pla tform**	Upd ated	**Not es**
Pytho nTool kit (PTK)	Windo ws/Li nux/M ac OS X	2011	An inter activ e envir onmen t for pytho n built aroun d a matla b style conso le windo w and edito r. It was desig ned to provi de a pytho n based envir onmen t simil iar to Matla b for scien tists and engin eers howev er it can also be used as a gener al purpo se inter activ e pytho n envir onmen t espec ially for inter activ e GUI progr ammin

Name	**Platform**	Updated	**Notes**
PyStudio	Windows/Linux/Mac OS X	2012	Open-source plugin that adds syntax checking, integrated debugger and module search to Editra, a general purpose developer's text editor that supports python syntax highlighting, auto-indent, auto-completion, class browser, and can run scripts from inside the editor.

Nam e	**Pla tform**	Upd ated	**Not es**
Pytho n Tools for Visua l Studi o	Windo ws	2013	Open-sourc e plugi n for Visua l Studi o 2010, 2012 and 2013. Suppo rts synta x highl ighti ng, debug ging and rich intel lisen se, refac torin g, objec t brows er, MPI clust er debug ging, Djang o intel lisen se and debug ging, devel opmen t REPL windo w and a debug ging REPL windo w. Suppo rts mixed -mode Pytho n/C/C ++

Name	Platform	Updated	Notes
Exedore	Mac OS X	2013	Commercial with feature-limited free trial. A Mac-native, single-window IDE inspired by Xcode. Features integrated debugger, tabs, code completion with tab triggers, syntax highlighting themes, search and replace with regex, integrated REPL sessions, goto definition, file browser, integ

Nam e	Pla tform	Upd ated	Not es

There are also text editors specialized in programming code, which have features like syntax colorization, export to other formats and convert text encoding.

These editors support multiple programming languages, Python among them:

- SciTE
- Notepad++

Shell is the name given to interactive environments for executing commands that can be used to test small pieces of code and for activities like data crunching (extraction of information of interest in masses of data and subsequent translation to other formats).

Beyond the standard Python **Shell**, there are others available:

- PyCrust
- IPython
- Reinteract
- bpython
- PyroShell

Packers are utilities that are used to build executables that comprise the bytecode, the interpreter and other dependencies, allowing the application to run on machines without Python installed, which facilitates program distribution.

Among packers for Python, are available:

- py2exe (Windows only)
- cx_Freeze (portable)

Frameworks are collections of software components (libraries, utilities and others) that have been designed to be used by other systems.

Some of the most known *frameworks* availble are:

- Web: Django, TurboGears, Zope and web2py.
- Graphic interface: wxPython, PyGTK and PyQt.
- Scientific processing: NumPy and SciPy.
- Image processing: PIL.
- 2D: Matplotlib and SVGFig.
- 3D: Visual Python, PyOpenGL and Python Ogre.
- Object-relational mapping: SQLAlchemy e SQLObject.

Culture

The name Python was taken by Guido van Rossum from british TV program *Monty Python's Flying Circus*, and there are many references to the show in its documentation. For instance, Python's oficial package repository was called Cheese Shop, the name of one of the frames of the program. Currently, the repository name is Python Package Index (PYPI).

The goals of the project are summarized by Tim Peters in a text called *Zen of Python*, which can be found in Python itself using the command:

```
import this
```

The Zen of Python

- Beautiful is better than ugly.
- Explicit is better than implicit.
- Simple is better than complex.
- Complex is better than complicated.
- Flat is better than nested.
- Sparse is better than dense.
- Readability counts.
- Special cases aren"t special enough to break the rules.
- Although practicality beats purity.
- Errors should never pass silently.
- Unless explicitly silenced.
- In the face of ambiguity, refuse the temptation to guess.
- There should be one – and preferably only one – obvious way to do it.
- Although that way may not be obvious at first unless you're Dutch.
- Now is better than never.
- Although never is often better than *right* now.
- If the implementation is hard to explain, it's a bad idea.
- If the implementation is easy to explain, it may be a good idea.
- Namespaces are one honking great idea – let's do more of those!