Introduction to Python

What is Python ?

Python is a high-level general-purpose dynamic multi-paradigm programming language that emphasises on a clean and simple syntax that makes it largely orthogonal, yet programs written using this language are highly readable and maintainable.

Features of Python

Now, let me elaborate on the features of this programming language that allows you to understand most of the jargons quoted in the above definition.

High level programming language

Python programming cater to more general audience than computer science engineers. You do not need to understand computer architecture, memory model, CPU register-width, platform endian-ness and all those “computer engineering” centric jargons to write programs in Python. More over, the language does not overwhelm you with “pointers” like in C, or complex OO-jargons (abstract classes, interfaces, generics and so on) from languages like Java. This clearly means that python allows you to focus more on the problem domain while writing your program than the programming language and platform

General purpose programming language

Python can be used to write programs that deal with scientific computing, data processing, network and process automation, web development, GUI development, game development and so on. It has all the features that a programmer needs to become productivity in developing applications for a specific domain. Though Python aims to be simple in syntax, it supports rich set of standard built-in and library functions that allow developing programs employing complex algorithms.

Dynamic programming language

Python is designed to be a fully dynamic programming language. Python programs are essentially interpreted and this allows them to be more flexible at run-time. Developing applications that use complex design patterns and algorithms is easier in python when compared to many other static programming languages.

A dynamic programming language could imply two paradigms:

Dynamic typing

A dynamic typing in a programming language allows programmers to use variables and functions within their programs without declaration. This emphasises on programmer productivity. In Python, variables can be used directly after first defining them. Remember that definition is different than declaration - we will discuss more about that in the later chapter. One of the main advantages of dynamic typing is that the language decides the storage size and type of the variable, so that the programmer can focus more on functionality than language intricacies.

> Dynamic typing however expects programmers to be self-disciplined. Typos in variable names for instance, could lead to duplicate definitions and logical errors in programs that might become a bit difficult to detect without proper testing.

Runtime dynamic programming

This is a feature in most modern high level programming languages that allow part or all of definitions (variable definition, function definition, class definition, and so on) at run time. In Python, all definitions happen at the run time allowing programs to define functions, classes, variables - all in run time and dispose them off, when not needed. This makes programs more flexible by design and more memory efficient.

> Note: Runtime dynamic programming implies a interpreter-based language design which might not perform as fast as a fully compiled programming language in many cases. In general, it might be a difficult to develop hard real-time applications using interpreter based dynamic programming language. Do not expect programs written in python to be as fast as the ones written in a language like C in a true one-to-one performance comparison. Having said that, in most real-world scenarios, python programs are quite fast that you'll hardly notice any difference.

Python is a **fully** dynamic programming language. But Python is also a compiled-interpreted language. When a python program is run using the python interpreter (also known as the "Python runtime"), it first compiles the python program to an intermediate byte-code. Syntax checking, parsing and variables scope are decided during compilation of the python program. After compilation, the python interpreter interprets the generated byte-code to native machine code for execution.

> Note: How Python programs are compiled and interpreted large depends on the python interpreter's implementation. There are variants of python interpreters that allow Just-In-Time compilation thus providing high performance comparable to fully compiled programming languages. I'll discuss more on that in the later chapters.

Multi-paradigm programming language

Python allows you to write programs that support procedural, OOP, AOP, functional and concurrent programming paradigms. Simply said - Python is fully *buzz-word* compliant. However, Python does not force any of the paradigms on the programmer. Rather, it allows programmers to better choose/decide the right programming paradigm for their given task. It aims to be more practical than being rigid in design.

Highly orthogonal programming language

Python is based on a very simple and succinct design and that includes a fully consistent syntax coupled with orthogonal language design. In Python, the core syntax is un-ambiguous and non-redundant. Another fact is that, in Python, all identifiers are variables, and all variables are references to objects. This implies that functions, classes, library modules and file handles are all objects which can be referred using a variable. I'll discuss more about this in the salient features of the language.

Fully introspective programming language

Python allows detecting the type of an object, finding out the objects attributes and behaviour, memory usage, reference counts and all other features fully in runtime. This is part of being a dynamic programming language that allows designing applications that use runtime polymorphism more effectively.

Exception-based error-handling

In Python, all runtime errors can be handled using a well designed exception handling construct. This allows development of robust applications that are resilient to runtime exceptions. Having a dedicated exception handling constructs allows developing programs that are more readable and maintainable.

Highly extensible programming language.

Python programs can benefit from C, C++, Java and .NET library frameworks as they can be extended via extension library modules. This makes Python - a more complete programming language. Any features lacking in Python can be extended from other mainstream languages (C/C++/Java/.NET)

Very clear and readable syntax

Python's biggest advantage is it's very clear and readable syntax. As a language, Python provides a very minimal learning curve to master the core syntax. Most of Python's complexity in learning curve lie in its language idioms (i.e., using the best logic to solve a particular problem) and its builtin and library functions.

Portable across many platforms

Python programs can run on various OS platforms that include all variants and clones of UNIX (that include Linux, BSD family, Solaris, Mac OSX, HP/UX and AIX), Microsoft Windows, Ameoba OS, AmigaOS, Android platform and iOS platform. Python can also run on top of Java Virtual Machine (JVM) using Jython and .NET platform using IronPython. Today, python programs run on PCs, tablets, smartphones, television set-top boxes and space-craft orbit control stations.

Rich set of libraries (Batteries included)

One of the motivating reasons for programmers to learn and use python these days is its rich set and support for libraries. This essentially means that you do not have to reinvent the wheel to get your job done - it's very likely solved by so many others that there would be a ready-made library module that would get the job done for you with much ease.

Highly active community

Today, Python enjoys a very active and vibrant community of developers who aid in promoting, knowledge-sharing and code contribution. Many companies prefer to use python today for its active community support.

History of Python

[* image of Guido Van Rossum *] Python was created by Guido Van Rossum - a dutch computer programmer in 1989 as a hobby project. Eventually, the language syntax and succinctness was liked by many amongst UNIX hacker community that more people contributed further towards its development. The language was named python initially as a joke as Guido Van Rossum was a fan of a television series called *Monty Python's Flying Circus*. [* image of Monty Python's Flying Circus *]

In 1999, Guido Van Rossum proposed Python as a *Computer Programming Language for Everybody* to DARPA for funding it's development. Since the late 90's python programs made it's way into enterprises and research and development as a replacement for programs written using languages like Perl, UNIX Shell, Tcl and also C++/Java owing to its more practical and elegant syntax.

Today, Python is used in almost every single large organisation owing to its rich feature-set, active community support, free as is in freedom (Open Source) and very minimal learning curve.

Many organisations have reported drastic improvement in programmer productivity while using Python for development.

Python implementations

Python was initially created by Guido Van Rossum using the C Programming language. This variant of python is still being maintained by Guido and his team. Sometimes, this is called as CPython. You can find more details and also download CPython from its official website: http://www.python.org/

However, there are many alternative variants of python being developed and maintained by others. Some of them are as below:

Jython

This project was started by Jim Hugunin as JPython sometime in 1997. Eventually, this project was taken over by Barry Warsaw in 1999, Samuele Pedroni in the year 2000, Brian Zimmer in 2005 and finally being maintained by Ted Leung and Frank Wierzbicki since 2008 - both hired by Sun Microsystems during that time. This project was renamed as Jython and employed the Java6 Scripting extensions to become a first-class scripting language on JVM alongside with Groovy.

Jython enjoys the benefit of Java's robustness, platform independence, scalability, performance (JIT), threading support and garbage collector ergonomics - while maintaining the simplistic python syntax and features. Jython also provides the benefit of instantiating and working with Java classes using pythonic idioms and programs.

Some large scale applications written in Python (that are heavily multithreaded and perform lots of runtime object instantiations) have proven to scale better when run in Jython than in CPython.

Jython can be downloaded from its official website: http://www.jython.org/

IronPython

Since 2003, Jim Hugunin started implementing Python on the .NET platform. He was eventually employed at Microsoft to sustain development of this project. This project came to be known as the IronPython project and is actively being maintained by Microsoft as a free community developed programming language that allows writing python programs that run on the .NET platform.

IronPython can be downloaded from its official website: http://ironpython.codeplex.com/

PyPy

PyPy is a fast implementation of Python that initially started off as a project for implementing Python using Python language itself (hence, the name). PyPy focusses more on speed and efficiency of execution and also memory management using better garbage collection strategies. PyPy achieves speed by employing JIT and stackless mode for concurrency. As of date, PyPy is still in its experimental stage. However, owing to its rapid development and strong community support, it's already stable enough that some websites have used pypy for powering their server-side python code.

PyPy can be downloaded from this official website: http://www.pypy.org/

Python 2 vs Python 3

There are two major versions of Python today. One of them is simply called Python 2 and the

current stable version of it is Python 2.7.6 (at the time of writing). This version is fully backward compatible to Python 2.0. This means that programs written for Python 2.0, Python 2.2, Python 2.4 series will continue for run on Python 2.7.6 without any compatibility issues.

Python 3.0 was released in December 2008, with the current stable release being Python 3.3.3. Python 3 drifts away from maintaining full backward compatibility with Python 2.0. The rationale behind dropping backward compatibility was to remove redundant functionality, evolve into cleaner language syntax and consistency. Even after 5 years since it's release, most developers are in the dilemma of whether to use Python 2 or use Python 3 for their development.

As much as possible, it is recommended to use Python 3 for your development as it is more consistent and better in design than Python 2. However, not all third-party libraries are currently ported to work on Python 3.

Thus, if your project does depend on third-party libraries that might not be available on Python 3, then stick to Python 2. But do work on a migration plan for Python 3 as that's what the community will maintain in the long run.

Most of the concepts covered in the rest of the book will be based on Python 3 with notes that indicate differences between a feature in Python 2 and Python3 when introduced for the first time.

Python licensing

Python is a free/open source software governed by Python Software Foundation License (PSFL) which is a BSD-style license and fully compatible with GNU General Public License (GPL). PSFL is an approved license by both Free Software Foundation (FSF) as well as Open Source Initiative (OSI).

Companies/organisations and individuals can use Python to develop applications without having to pay any license fee to its developers and maintainers. Developers can also choose to distribute programs developed using Python as closed source product.

Installing Python

Installation of Python 3 on Linux

Most mainstream Linux based OS distributions currently have Python 2.7 series installed by default. You can install Python 3 on most of these distributions alongside with Python 2 by following a simple distribution-specific package installation procedure. Note that installing Python 3 using the standard distribution package managers will not override your current Python 2 setup.

On all RPM based Linux distributions (RHEL, CentOS, Fedora), you can install Python 3 by using yum as below:

```
$ sudo yum install python3
```

On all debian based Linux distributions (Debian, Mint, Ubuntu), you can install Python 3 by using apt-get as below:

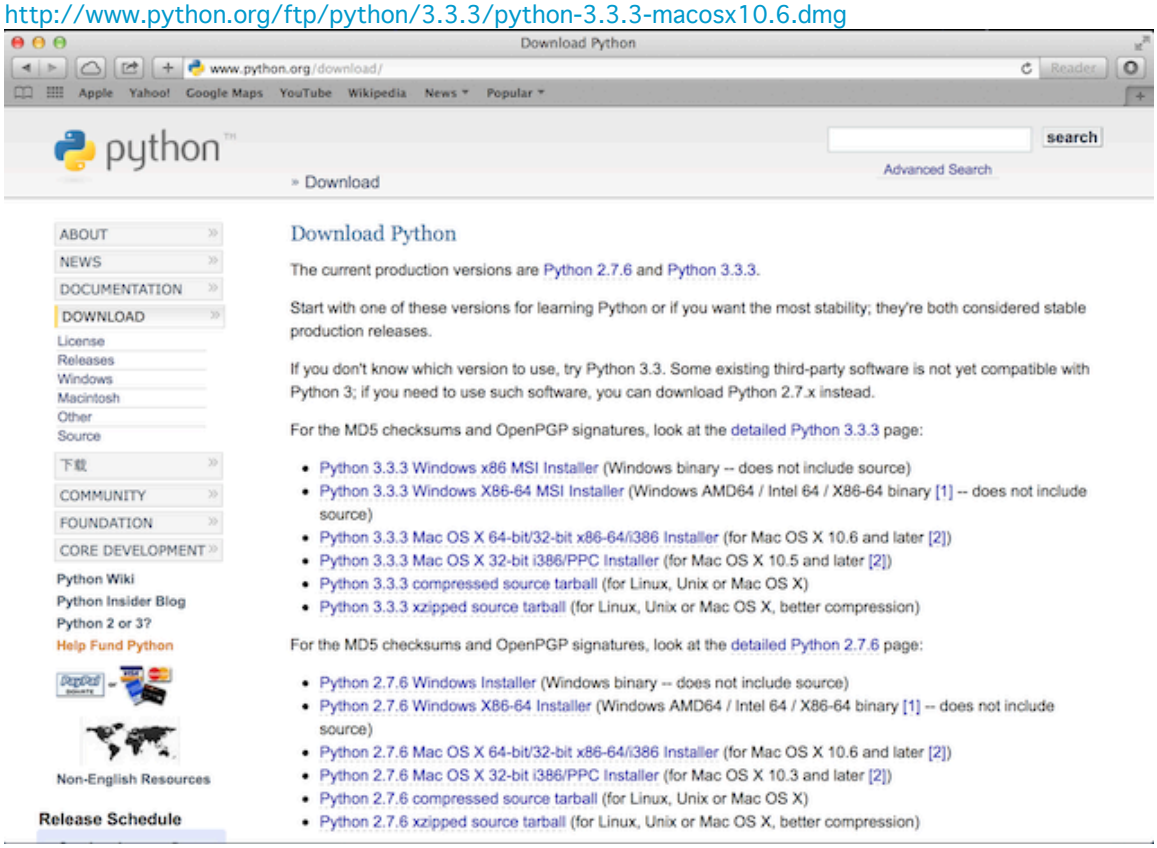
```
$ sudo apt-get install python3
```

For other Linux distributions, find the distribution's package management tools and use them to install the same. For instance, SLES and OpenSUSE use zypper, Gentoo Linux uses emerge, ARCHLinux uses pacman and Slackware Linux uses slapt-get.

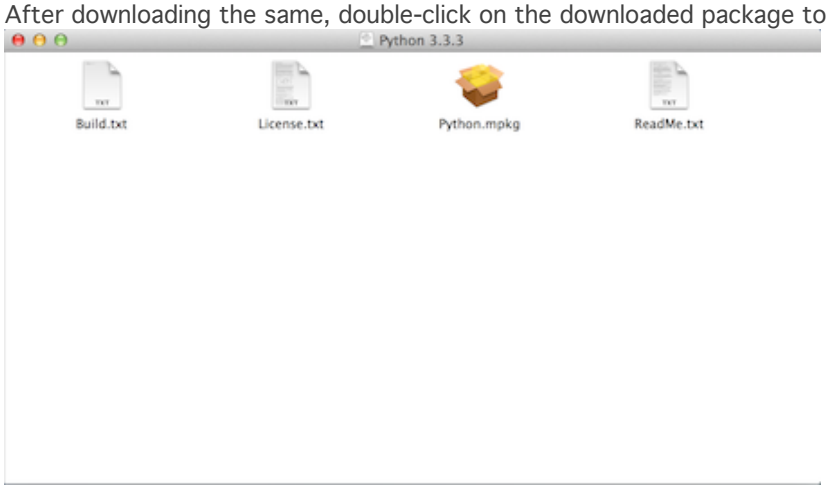
For those who wish to build python from the source code, you can download the latest tarball of python from the following URL: http://www.python.org/ftp/python/3.3.3/Python-3.3.3.tgz

Installation of Python 3 on Mac OSX

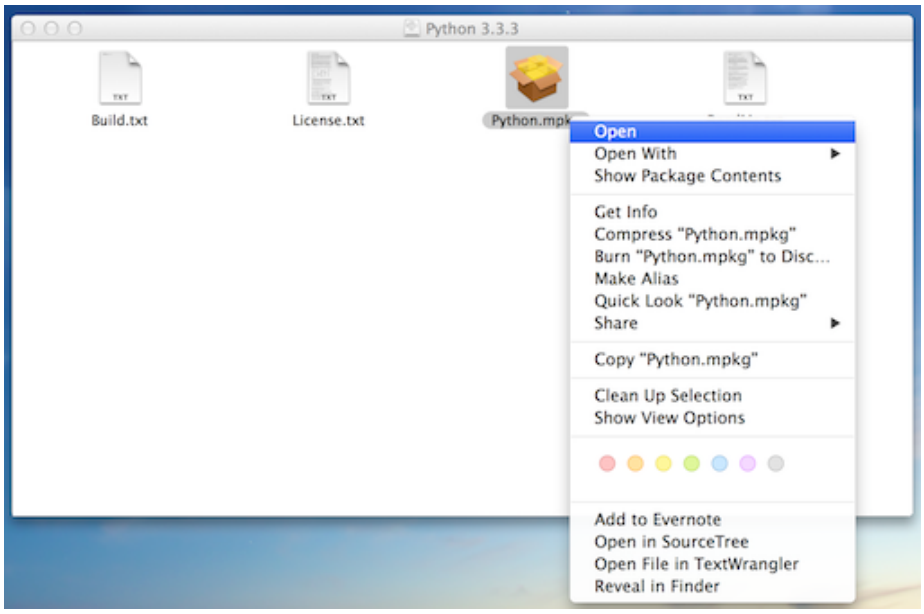
Mac OSX comes pre-installed with Python 2.7 series. To install Python3 alongside with the default Python2, you can download the latest version from the following URL: http://www.python.org/ftp/python/3.3.3/python-3.3.3-macosx10.6.dmg



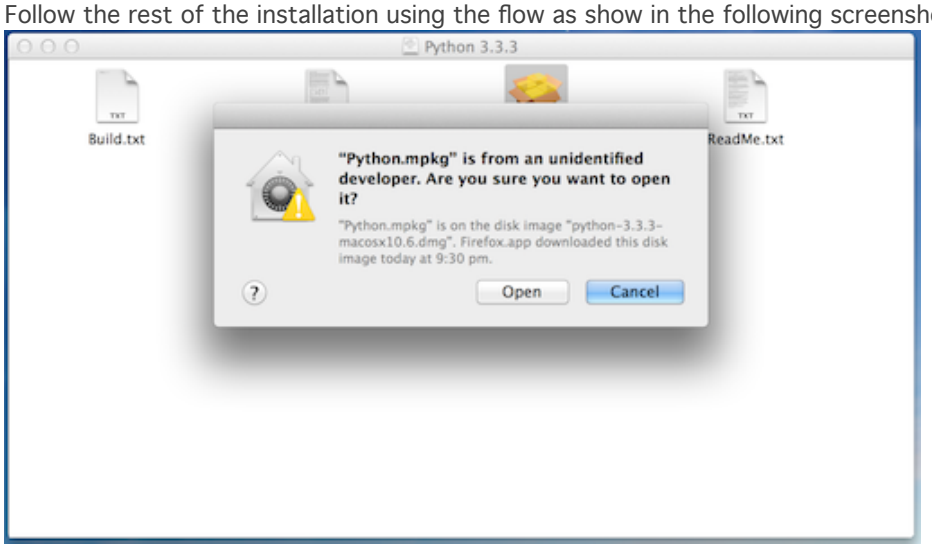
After downloading the same, double-click on the downloaded package to open the same.



Upon opening the package, right-click on the icon Python.mpkg and select **Open** menu item from the pop-up menu.



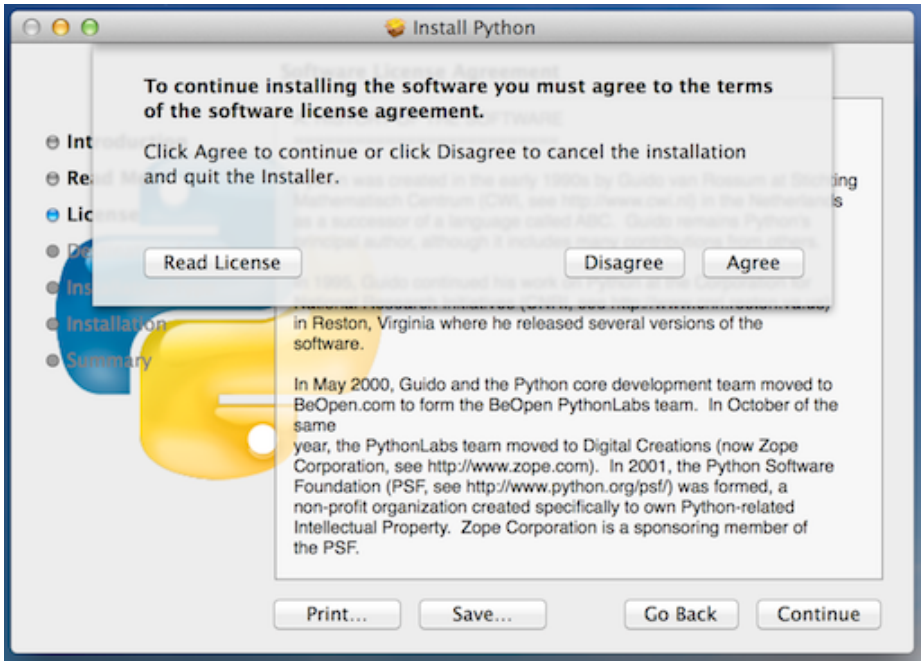
Follow the rest of the installation using the flow as show in the following screenshots:

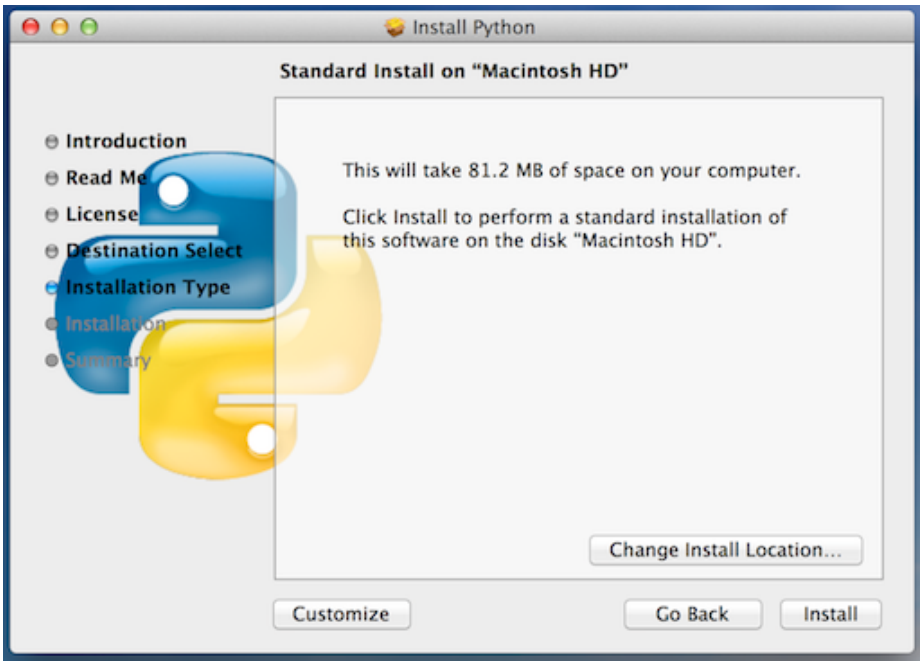


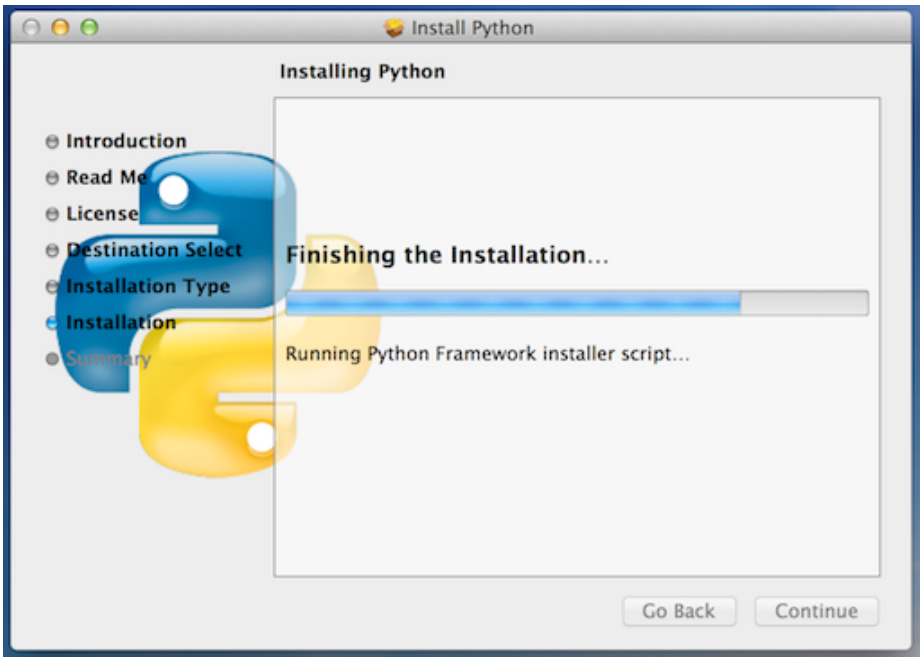

Install Python
Welcome to the Python Installer
Introduction
Read Me
License
Destination Select
Installation Type
Installation
Summary
This package will install Python 3.3.3 for Mac OS X 10.6 or later.
Python for Mac OS X consists of the Python programming language interpreter, plus a set of programs to allow easy access to it for Mac OS X users including an integrated development environment IDLE and a set of pre-built extension modules that open up specific Macintosh technologies to Python programs.
See the ReadMe file and the Python documentation for more information.
NOTE: This package will not update your shell profile by default. Double-click Update Shell Profile at any time to make 3.3.3 the default Python 3 version. This version can co-exist with other installed versions of Python 3 and Python 2.
IMPORTANT: IDLE and other programs using the tkinter graphical user interface toolkit require specific versions of the Tcl/Tk platform independent windowing toolkit. Visit http://www.python.org/download/mac/tcltk/ for current information on supported and recommended versions of Tcl/Tk for this version of Python and Mac OS X.
Go Back
Continue

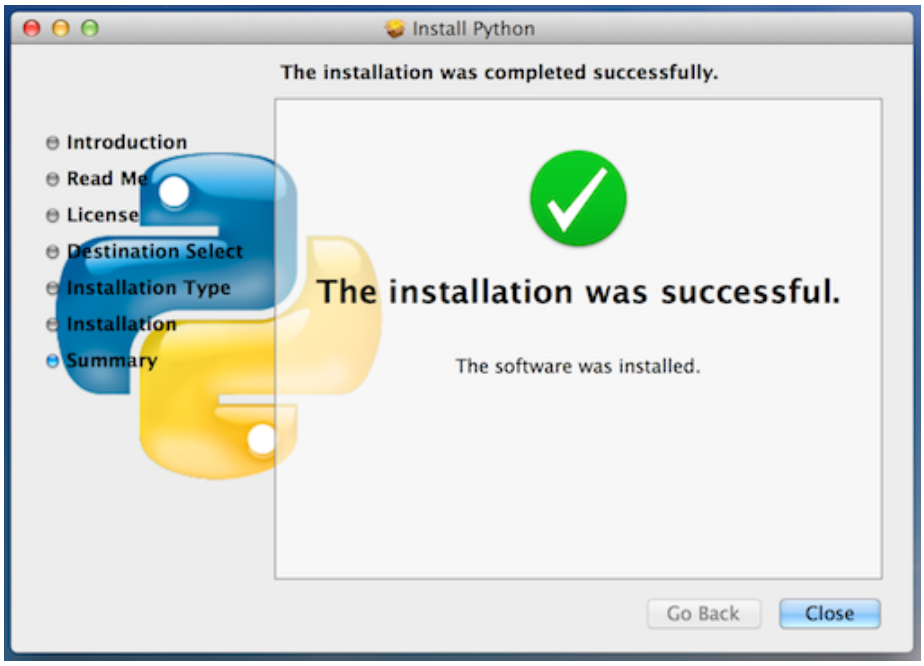

Install Python
Important Information
Introduction
Read Me
License
Destination Select
Installation Type
Installation
Summary
This package will install Python 3.3.3 for Mac OS X 10.6 or later for the following architecture(s):
i386, x86_64.
**** IMPORTANT ****
Installing on OS X 10.8 (Mountain Lion) or later systems
If you are attempting to install on an OS X 10.8+ system, you may see a message that Python can't be installed because it is from an unidentified developer. This is because this Python installer package is not yet compatible with the Gatekeeper security feature introduced in OS X 10.8. To allow Python to be installed, you can override the Gatekeeper policy for this install. In the Finder, instead of double-clicking, control-click or right click the "Python" installer package icon. Then select "Open using ... Installer" from the contextual menu that appears.
**** IMPORTANT ****
Update your version of Tcl/Tk to use IDLE or other Tk applications
Print...
Save...
Go Back
Continue


Install Python
Software License Agreement
Introduction
Read Me
License
Destination Select
Installation Type
Installation
Summary
A. HISTORY OF THE SOFTWARE
==========
Python was created in the early 1990s by Guido van Rossum at Stichting Mathematisch Centrum (CWI, see http://www.cwi.nl) in the Netherlands as a successor of a language called ABC. Guido remains Python's principal author, although it includes many contributions from others.
In 1995, Guido continued his work on Python at the Corporation for National Research Initiatives (CNRI, see http://www.cnri.reston.va.us) in Reston, Virginia where he released several versions of the software.
In May 2000, Guido and the Python core development team moved to BeOpen.com to form the BeOpen PythonLabs team. In October of the same
year, the PythonLabs team moved to Digital Creations (now Zope Corporation, see http://www.zope.com). In 2001, the Python Software Foundation (PSF, see http://www.python.org/psf/) was formed, a non-profit organization created specifically to own Python-related Intellectual Property. Zope Corporation is a sponsoring member of the PSF.
Print...
Save...
Go Back
Continue

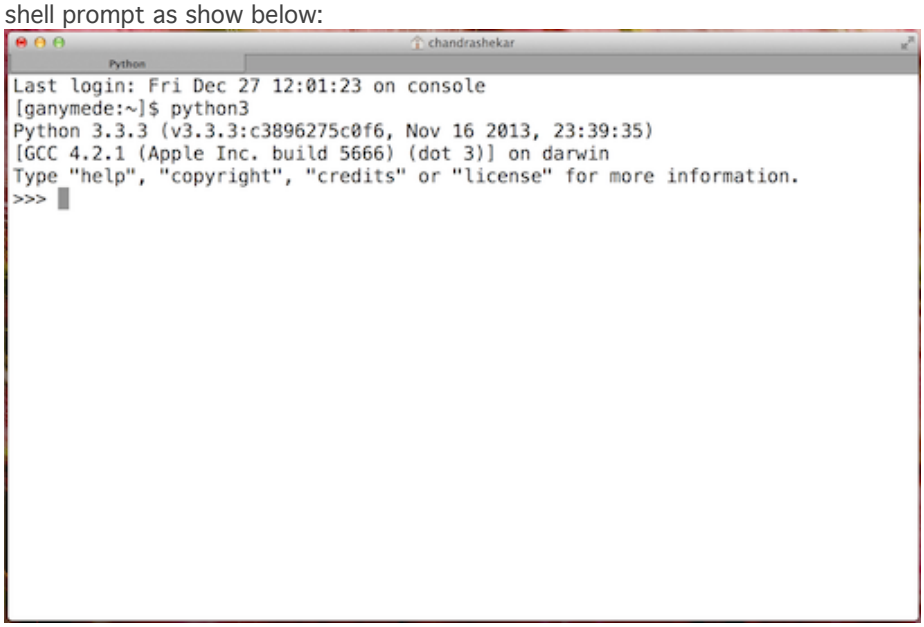

Install Python
To continue installing the software you must agree to the terms of the software license agreement.
Click Agree to continue or click Disagree to cancel the installation and quit the Installer.
Read License
Disagree
Agree
Installation
Summary
in Reston, Virginia where he released several versions of the software.
In May 2000, Guido and the Python core development team moved to BeOpen.com to form the BeOpen PythonLabs team. In October of the same
year, the PythonLabs team moved to Digital Creations (now Zope Corporation, see http://www.zope.com). In 2001, the Python Software Foundation (PSF, see http://www.python.org/psf/) was formed, a non-profit organization created specifically to own Python-related Intellectual Property. Zope Corporation is a sponsoring member of the PSF.
Print...
Save...
Go Back
Continue


Install Python
Standard Install on "Macintosh HD"
Introduction
Read Me
License
Destination Select
Installation Type
Installation
Summary
This will take 81.2 MB of space on your computer.
Click Install to perform a standard installation of this software on the disk "Macintosh HD".
Change Install Location...
Customize
Go Back
Install


Install Python
Installing Python
Introduction
Read Me
License
Destination Select
Installation Type
Installation
Summary
Finishing the Installation...
Running Python Framework installer script...
Go Back
Continue



Now, to test the currently installed python, open up a terminal and try typing 'python3' at the shell prompt as show below:



If you are greeted with the python prompt (the `>>>` sign), then python installation seemed to have worked fine. Also confirm the Python version (3.3.3) in the greeting message before the `>>>` sign.

To exit from the python prompt (`>>>`), you can type `exit()` and press the **return** key on the keyboard. You can also press and hold **control** key and then press and release the **d** on the keyboard (also known as the Ctrl-d sequence).

Installation of Python 3 on Windows platform

```
[TODO]
```