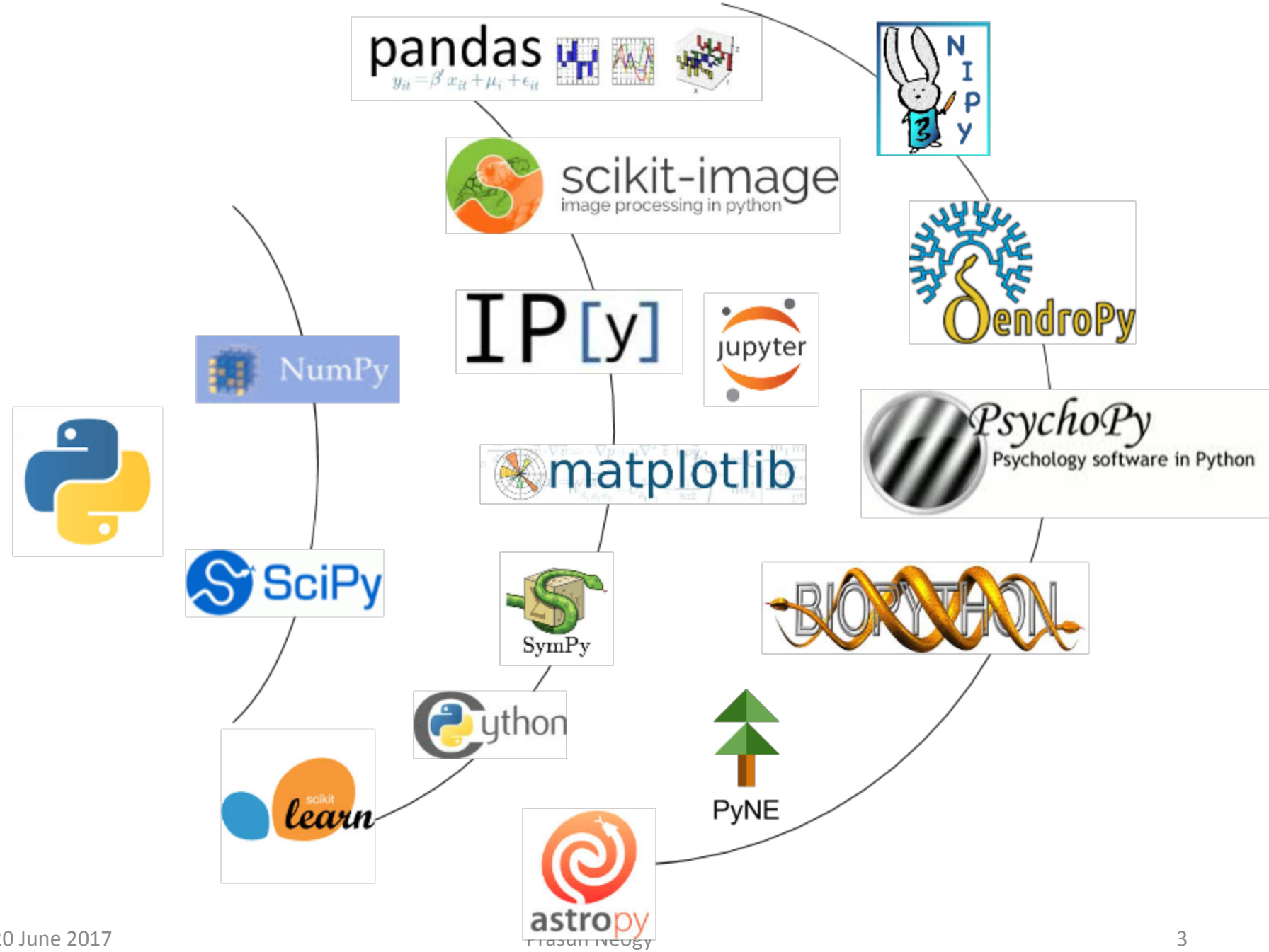
Python – S1

Contents


pandas
NIPY
scikit-image
image processing in python
IP[y]
jupyter
NumPy
SendroPy
PsychoPy
Psychology software in Python
matplotlib
SciPy
SymPy
BIOPYTHON
Cython
scikit learn
PyNE
astropy

Introduction to Python

Introduction to Python

- A basic Programming philosophy:

"Make it run, make it right, make it fast"

− Kent Beck

Introduction to Python

- Top Ten programming Languages [as per 2-17 rankings]

1 JavaScript
2 Java
3 Python
4 PHP
5 C#
5 C++
7 CSS
7 Ruby
9 C
10 Objective-C

[http://redmonk.com/sogrady/2017/03/17/language-rankings-1-17/]

Introduction to Python

According to Wikipedia, **computer programming** is:

"...a process that leads from an original formulation of a computing problem to executable computer programs. Programming involves activities such as analysis, developing understanding, generating algorithms, verification of requirements of algorithms including their correctness and resources consumption, and implementation (commonly referred to as coding) of algorithms in a target programming language".

In a nutshell, coding is telling a computer to do something using a language it understands.

Introduction to Python

- Python is a general-purpose, high-level, and interpreted dynamic language.

- It is commonly defined as an *object-oriented scripting language*—a definition that blends support for OOP with an overall orientation toward scripting roles.

- It is commonly used both for standalone programs and for scripting applications in a wide variety of domains, and is generally considered to be one of the most widely used programming languages in the world.

- Perhaps one of its drawback is -- its *execution speed* may not always be as fast as that of fully compiled and lower-level languages such as C and C++.

Introduction to Python

- Its design philosophy emphasizes code readability, and its syntax allows programmers to express concepts in fewer lines of code than possible in languages such as C++ or Java.

- The standard implementations of Python today compile (i.e., translate) source code statements to an intermediate format known as *byte code* and then interpret the byte code.

- Byte code provides portability, as it is a platform-independent format. However, because Python is not normally compiled all the way down to binary machine code, some programs will run more slowly in Python than in a fully compiled language like C.

Introduction to Python

- Python supports multiple programming paradigms, including object-oriented, procedural and functional programming styles.

- It features a dynamic type system and automatic memory management and has a large and comprehensive standard library.

- Python interpreters are available for many operating systems, allowing Python code to run on a wide variety of systems.

- Python's design and philosophy have influenced several other programming languages, including : Cobra, Go, Groovy, Ruby, Swift, etc.

- Python has an internal garbage collector, like Java, for freeing up memory when it is not needed. The main emphasis of Python lies in its readable code and potential for high productivity;

A Brief history of Python

- Python was conceived in the late 1980s, and its implementation began in December 1989 by Guido van Rossum at Centrum Wiskunde & Informatica (CWI) in the Netherlands as a successor to the ABC Language, capable of exception handling and interfacing with the operating system – Amoeba.

- Van Rossum is Python's principal author, and his continuing central role in deciding the direction of Python is reflected in the title given to him by the Python community, as 'benevolent dictator for life'(BDFL).

- Python 2.0 was released on 16 October 2000 and had many major new features, including a cycle-detecting garbage collector and support for Unicode.

- With this release the development process was changed and became more transparent and community-backed.

A Brief history of Python

- Python 3.0 (which early in its development was commonly referred to as Python 3000 or py3k), a major, backwards-incompatible release, was released on 3 December 2008 after a long period of testing.

- Many of its major features have been backported to the backwards-compatible Python 2.6.x and 2.7.x version series.

- One of the principle strengths of this new language (Python) was how easy it was to extend, and its support for multiple platforms - a vital innovation in the days of the first personal computers.

- Capable of communicating with libraries and differing file formats, Python quickly took off.

Who uses Python today ?

Besides individual users, Python is used by :

- *Google* makes extensive use of Python in its web search systems.
- The popular *YouTube* video sharing service is largely written in Python.
- The *Dropbox* storage service codes both its server and desktop client software primarily in Python.
- The *Raspberry Pi* single-board computer promotes Python as its educational language.
- The widespread *BitTorrent* peer-to-peer file sharing system began its life as a Python program.
- *Pixar* use Python in the production of animated movies.
- Sites like : Quora, Youtube Reddit, etc are written in Python.
- The *NSA* uses Python for cryptography and intelligence analysis.
- Walt Disney Feature Animation, RedHat, Nokia, IBM, Netflix, Intel, Cisco, Qualcomm, JPMorgan Chase, etc.

- And so on…..

Latest news on Python !!

This news article is from : thenewstack.io on 15-June-2017

“Each day, over 95 million photos and videos are shared on Instagram. The unstoppable photo-centric social media platform has over 600 million registered users — 400 million of whom are active every day. Talk about operating at scale: Instagram kills it at levels most companies can barely even dream about.

Even more impressive, though, is the fact that Instagram serves this incredible amount of traffic, reliably and steadily so, by running Python (with a little help from Django) under the hood. Yes, that Python — the easy to learn, jack-of-all-trades general purpose programming language. The one everybody in the industry dismisses as, “Yeah, Python is great in so many ways, too bad it’s not really scalable.”

Ahem. Four. Hundred. Million. Users. Per. Day. Not only has Instagram scaled to become the biggest Python user in the world, but the company recently moved over to Python 3 with zero user experience interruption.”

What are Python's Technical Strengths?

- It's Object-Oriented and Functional
- It's Free
- It's Portable
- It's Powerful
- It's Relatively Easy to Use
- It's Relatively Easy to Learn
- It has extensive Libraries

- It is heavily used in all modern day requirements like – in Data Analytics, in Scientific projects, in Genome projects, in Natural Language processing, in Machine Learning , etc.

Features of Python

- Simple -- Python is a simple and minimalistic language. Reading a good Python program feels almost like reading English, although very strict English! This pseudo-code nature of Python is one of its greatest strengths. It allows you to concentrate on the solution to the problem rather than the language itself.

- Easy to Learn -- As you will see, Python is extremely easy to get started with. Python has an extraordinarily simple syntax, as already mentioned.

- Coherence -- Python is extremely logical and coherent. You can see it was designed by a brilliant computer scientist. Most of the time you can just guess how a method is called, if you don't know it. [You may not realize how important this is right now, especially if you are at the beginning, but this is a major feature. It means less cluttering in your head, less skimming through the documentation, and less need for mapping in your brain when you code.]

Features of Python

- Free and Open Source -- Python is an example of a FLOSS (Free/LibrÃ© and Open Source Software). In simple terms, you can freely distribute copies of this software, read it's source code, make changes to it, use pieces of it in new free programs, and that you know you can do these things. FLOSS is based on the concept of a community which shares knowledge. This is one of the reasons why Python is so good - it has been created and is constantly improved by a community who just want to see a better Python.

- Software Integration -- Another important aspect is that Python can be extended and integrated with many other languages, which means that even when a company is using a different language as their mainstream tool, Python can come in and act as a glue agent between complex applications that need to talk to each other in some way. This is kind of an advanced topic, but in the real world, this feature is very important.

Features of Python

- Developer Productivity -- According to Mark Lutz (Learning Python, 5th Edition, O'Reilly Media), a Python program is typically one- fifth to one-third the size of equivalent Java or C++ code. This means the job gets done faster. And faster is good. Faster means a faster response on the market. Less code not only means less code to write, but also less code to read (and professional coders read much more than they write), less code to maintain, to debug, and to refactor.

- Another important aspect is that Python runs without the need of lengthy and time consuming compilation and linkage steps, so you don't have to wait to see the results of your work.

Features of Python

- High-level Language -- When you write programs in Python, you never need to bother about the low-level details such as managing the memory used by your program, etc.

- Portable -- Due to its open-source nature, Python has been ported (i.e. changed to make it work on) to many platforms. All your Python programs can work on any of these platforms without requiring any changes at all if you are careful enough to avoid any system-dependent features.

- You can use Python on Linux, Windows, FreeBSD, Macintosh, Solaris, OS/2, Amiga, AROS, AS/400, BeOS, OS/390, z/OS, Palm OS, QNX, VMS, Psion, Acorn RISC OS, VxWorks, PlayStation, Sharp Zaurus, Windows CE and even PocketPC !

Features of Python

- Interpreted -- This requires a bit of explanation.
- A program written in a compiled language like C or C++ is converted from the source language i.e. C or C++ into a language that is spoken by your computer (binary code i.e. 0s and 1s) using a compiler with various flags and options. When you run the program, the linker/loader software copies the program from hard disk to memory and starts running it.
- Python, on the other hand, does not need compilation to binary. You just *run* the program directly from the source code. Internally, Python converts the source code into an intermediate form called bytecodes and then translates this into the native language of your computer and then runs it. All this, actually, makes using Python much easier since you don't have to worry about compiling the program, making sure that the proper libraries are linked and loaded, etc, etc. This also makes your Python programs much more portable, since you can just copy your Python program onto another computer and it just works!

Features of Python

- Object Oriented -- Python supports procedure-oriented programming as well as object-oriented programming. In *procedure-oriented* languages, the program is built around procedures or functions which are nothing but reusable pieces of programs. In *object-oriented* languages, the program is built around objects which combine data and functionality. Python has a very powerful but simplistic way of doing OOP, especially when compared to big languages like C++ or Java.

- Extensible -- If you need a critical piece of code to run very fast or want to have some piece of algorithm not to be open, you can code that part of your program in C or C++ and then use them from your Python program.

Features of Python

- Embeddable -- You can embed Python within your C/C++ programs to give 'scripting' capabilities for your program's users.

- Extensive Libraries -- The Python Standard Library is huge indeed. It can help you do various things involving regular expressions, documentation generation, unit testing, threading, databases, web browsers, CGI, ftp, email, XML, XML-RPC, HTML, WAV files, cryptography, GUI (graphical user interfaces), Tk, and other system-dependent stuff. Remember, all this is always available wherever Python is installed. This is called the 'Batteries Included' philosophy of Python.

- Besides the Standard Libraries, there are various other high-quality libraries such as numpy, scipy, matplotlib, pandas, scikit-learn, etc.

Features of Python

- Satisfaction and Enjoyment -- Last but not least, the fun of it! Working with Python is fun. One can code for 8 hours and leave the office happy and satisfied, alien to the struggle other coders have to endure because they use languages that don't provide them with the same amount of well-designed data structures and constructs. Python makes coding fun, no doubt about it. And fun promotes motivation and productivity.

- My personal take : I learnt Python at the age of 54!!! And now I love to code..... Anything now that comes to my mind, I try to think “can it be done in Python?”

Drawbacks of Python

- Probably, the only drawback that one could find in Python, which is not due to personal preferences, is the *execution speed*.

- Typically, Python is slower than its compiled brothers.

- The standard implementation of Python produces, when you run an application, a compiled version of the source code called byte code (with the extension .pyc), which is then run by the Python interpreter.

- The advantage of this approach is portability, which we pay for with a slowdown due to the fact that Python is not compiled down to machine level as are other languages.

- However, Python speed is rarely a problem today, hence its wide use regardless of this suboptimal feature.

Drawbacks of Python

- What happens is that in real life, hardware cost is no longer a problem, and usually it's easy enough to gain speed by parallelizing tasks.

- When it comes to number crunching though, one can switch to faster Python implementations, such as PyPy, which provides an average 7-fold speedup by implementing advanced compilation techniques.

- When doing data science, you'll most likely find that the libraries that you use with Python, such as Pandas and Numpy, achieve native speed due to the way they are implemented.

- If that wasn't a good enough argument, you can always consider that Python is driving the backend of services such as Spotify and Instagram, where performance is a concern.

- Nonetheless, Python does its job perfectly adequately.

Power of Python

- I feel the Power of Python is because of :

1. Very easy to learn
2. List Comprehension
3. Huge availability of Libraries
4. Numpy and Scipy
5. Recommender Systems
6. Natural Language Processing
7. Data Analytics - like Sentiment Analysis
8. BioPython
9. Machine Learning
10. Parallel Programming

Where Can You Use Python

- One significant advantage of learning Python is that it's a general-purpose language that can be applied in a large variety of projects.
- Below are just some of the most common fields where Python has found its use:

1. Data science and Analytics
2. Scientific and mathematical computing
3. Web development
4. Finance and trading
5. System automation and administration
6. Computer graphics
7. Basic game development
8. Security and penetration testing
9. General and application-specific scripting
10. Mapping and geography (GIS software)
11. BioPython
12. Machine Learning

Python Versions

- There are two major versions in Python 2.x and 3.x. Which version is to be used ??

- *Python 2.x is legacy, Python 3.x is the present and future of the language*

- Python 3.0 was released in 2008.
- The final 2.x version 2.7 release came out in mid-2010, with a statement of extended support for this end-of-life release.
- The 2.x branch will see no new major releases after that.

- 3.x is under active development and has already seen over five years of stable releases, including version 3.3 in 2012, 3.4 in 2014, 3.5 in 2015 and 3.6 in 2016 [version - 3.6.1 in 09/2017]
- This means that all recent standard library improvements, for example, are only available by default in Python 3.x.

Python Versions

- Which version you ought to use is mostly dependent on what you want to get done.

- If you can do exactly what you want with Python 3.x, great!

- There are a few minor downsides, such as very slightly worse library support and the fact that some current Linux distributions and Macs are still using 2.x as default (although Python 3 ships with many of them), but as a language Python 3.x is definitely ready.

- As long as Python 3.x is installed on your user's computers (which ought to be easy, since many people reading this may only be developing something for themselves or an environment they control) and you're writing things where you know none of the Python 2.x modules are needed, it is an excellent choice.

Python Versions

- Also, most Linux distributions have Python 3.x already installed, and all have it available for end-users.

- Some are phasing out Python 2 as preinstalled default.

- In particular, instructors introducing Python to new programmers should consider teaching Python 3 first and then introducing the differences in Python 2 afterwards (if necessary).

- Note : In this Course, we will be using Python 3.6 0 the latest version from Anaconda Distribution.

Python Versions

- However, there are some key issues that may require you to use Python 2 rather than Python 3.
  - Firstly, if you're deploying to an environment you don't control, that may impose a specific version, rather than allowing you a free selection from the available versions.
  - Secondly, if you want to use a specific third party package or utility that doesn't yet have a released version that is compatible with Python 3, and porting that package is a non-trivial task, you may choose to use Python 2 in order to retain access to that package.

- There exist an excellent tool called 2to3 [it comes with the Python distribution, which converts Python 2.x compatible code to Python 3.x !!!

Installation of Python

- Python is installed by default in Linux and Mac machines.

- We will consider the Anaconda Distribution for Python as it includes a lot of useful libraries and also can be updated very easily.
- Anaconda is available for Windows, OS X or Linux, 32- or 64-bit, with disk space requirement of about 3 GB HD available..
- The open source version of Anaconda is an easy-to-install high performance Python distribution with a package manager, environment manager and collection of 720+ open source packages with free community support.

Installation of Python

- The basic features of the distribution is :
- License: Anaconda is free to use and redistribute under the terms of the Anaconda End User License Agreement
- Windows Vista or newer, macOS 10.7+, or Linux (Ubuntu, RedHat and others; CentOS 6+)
- 32-bit or 64-bit
- Minimum 3 GB disk space to download and install.
- Over 150 packages are automatically installed with Anaconda.
- Over 250 additional open source packages can be individually installed from the Anaconda repository at the command line, by using the “conda install” command.
- Thousands of other packages are available from Anaconda.org.
- Others can be downloaded using the “pip install” command that is included and installed with Anaconda.

- Current version of Anaconda is 4.4, which features Python 3.6.

Installation of Python - Windows

- Download the installer − Python 3.6:
- [depending on whether 64-bit or 32-bit]

https://repo.continuum.io/archive/Anaconda3-4.4.0-Windows-x86_64.exe

- Or

https://repo.continuum.io/archive/Anaconda3-4.4.0-Windows-x86.exe

- Double-click the **.exe** file to install Anaconda and follow the instructions on the screen.
- For further queries , please lookup :

https://www.continuum.io/downloads#windows

Installation of Python - Linux

- Download the installer − Python 3.6:
- [depending on whether 64-bit or 32-bit]

https://repo.continuum.io/archive/Anaconda3-4.4.0-Linux-x86_64.sh

- Or

https://repo.continuum.io/archive/Anaconda3-4.4.0-Linux-x86.sh

- In your terminal window type one of the below and follow the instructions: **Python 3.6 version**

$ bash Anaconda3−4.4.0−Linux−x86_64.sh

- For further queries , please lookup :

https://www.continuum.io/downloads#windows

Python Interpreter

- As currently implemented, it's also a software package called an *interpreter*.
- An interpreter is a kind of program that executes other programs.
- When you write a Python program, the Python interpreter reads your program and carries out the instructions it contains.
- When the Python package is installed on your machine, it generates a number of components—minimally, an interpreter and a support library.
- Depending on how you use it, the Python interpreter may take the form of an executable program, or a set of libraries linked into another program.

Python Interpreter

- In its simplest form, a Python program is just a text file containing Python statements.

- For example, the following file, named *myexample01.py*, is one of the simplest Python scripts I could dream up, but it passes for a fully functional Python program:

```
print('Hello world')
```

- To go into the Interpreter mode, from Linux $ prompt type : python3
- You will see some messages and then :

```
>>>
```

- At this prompt, you can now type any python commands (either single line or multi-line and then execute, e.g. :

```
>>> print('Hello world')
Hello world
```

Python Interpreter

Interactive Mode

- When commands are read from a tty, the interpreter is said to be in *interactive mode*.
- In this mode it prompts for the next command with the *primary prompt*, usually three greater-than signs (>>>); for continuation lines it prompts with the *secondary prompt*, by default three dots (...).
- The interpreter prints a welcome message stating its version number and a copyright notice before printing the first prompt:

$ python

Python 3.6.1 |Anaconda 4.4.0 (x86_64)| (default, May 11 2017, 13:04:09)

[GCC 4.2.1 Compatible Apple LLVM 6.0 (clang-600.0.57)] on darwin

Type "help", "copyright", "credits" or "license" for more information.

>>>

Python Interpreter

Interactive Mode

- Continuation lines are needed when entering a multi-line construct. As an example, take a look at this if statement:

```
>>> the_world_is_flat = True
>>> if the_world_is_flat:
...         print("Be careful not to fall off!")
...
Be careful not to fall off!
```

Executing Python from Command Line

- When we write a Python program in a file , say, myexample01.py, then we can run this program from the (Unix) command prompt as well, like :

```
$ python  myexample01.py
Hello world
$
```

- One can pass on parameters/arguments to the Python programs as well.

Python Documentation

- Python [both ver 2.7 and 3.6] has extensive documentation
- The official documentation is in :

https://docs.python.org/3/

- You can also 'download' these documentation from :

https://docs.python.org/3/download.html

- A sample :

The if statement is used for conditional execution:

if_stmt ::= "if" expression ":" suite
("elif" expression ":" suite)
["else" ":" suite]

It selects exactly one of the suites by evaluating the expressions one by one until one is found to be true (see section Boolean operations for the definition of true and false); then that suite is executed (and no other part of the if statement is executed or evaluated). If all expressions are false, the suite of the else clause, if present, is executed.

Getting Help in Python

- In general, official documentation page is :

https://www.python.org/about/help/

- From the Python interpreter prompt, type help()

>>> help()

[a lot messages]

help>while

- It will show you the help page for the 'while' statement.

help>random

- It will show you the help page for 'random' module.

Reference

- Python for Informatics – C. Severance
- Think Python – A. B. Downey - [O'Reilly]
- Python Crash Course – Eric Matthes [No starch Press]
- A Byte of Python – Swaroop C H
- Introducing Python – Bill Lubanovic [O'Reilly]

- Learning Python – 5th Ed – Mark Lutz - [O'Reilly]

End of Presentation

Python – S1