Chapter 2 LITERATURE REVIEW

packages on Anaconda Cloud or in a local Anaconda Repository, install them in an environment, run the packages and update them. It is available for Windows, macOS and Linux. Navigator is automatically include with Anaconda version 4.0.0 or higher. The following applications are available by default in Navigator:

* + JupyterLab
  + Jupyter Notebook
  + QtConsole
  + Spyder
  + Glueviz
  + Orange
  + Rstudio
  + Visual Studio Code
* **Reference papers**

 “Release Notes” docs.anaconda.com. 31 May 2018.Retrieved 31 May 2018.

 “Conda-Conda documentation”. Retrieved February 25, 2016.

 “Building Conda Packages for Multiple Operating Systems”. Pydannt.29 January 2015.Retrieved 9 April 2015.

 Lorica, Ben (March 24, 2013).” Python data tools just keep getting better”. O’Reilly

Radar. Retrieved October 30, 2014.

**2.2 Jupyter Notebook**

Jupyter Notebook (Formerly IPython Notebooks) is a web-based interactive computational environment for creating Jupyter notebooks documents. The “notebook” term can colloquially make reference to many different entities, mainly the Jupyter web application, Jupyter python web server, or Jupyter document format depending on context.

A Jupyter Notebook document is a JSON document, following a versioned schema, and containing an ordered list of input/output cells which can contain code, text (using Markdown), mathematics, plots and rich media, usually ending with the “. ipynb” extension. Jupyter notebooks document can be converted to a number of open standard output formats (HTML, presentation, slides, LaTeX, PDF, ReStructuredText, Markdown, Python) through ‘Download As’ in the web interface, via the nbconvert library or ‘jupyter nbconvert’ command line interface in a shell.

|  |  |  |
| --- | --- | --- |
| Department of CSE, RNSIT | 2018-2019 | 6 |

Chapter 2 LITERATURE REVIEW

Jupyter Notebook provides a browser-based REPL built upon a number of popular open-source libraries:

* IPython
* Tornado
* jQuery
* Bootstrap (front-end framework)
* MathJax

Jupyter Notebook can connect to many kernels, to allow programming in many languages. As of the 2.3 release (October 2014), there are currently 49 Jupyter-compatible kernels for as many programming languages, including Python, R, Julia and Haskell.

The Notebook interface was added to IPython in the 0.12 release(December 2011), renamed to Jupyter notebook in 2015. Jupyter Notebook is similar to the notebook interface of other programs such as Maple, Mathematica and SageMath, a computational interface style that originated with Mathematica in the 1980s.

* **Jupyter Kernels**

A Jupyter kernel is a program responsible for handling various types of request, and providing a reply. Kernels talks to the other components of Jupyter using ZeroMQ over the network, and thus can be on the same or remote machines. Usually Kernels are implemented and allow execution of a single language with a couple of exceptions.

* **Jupyter Hub**

JupyterHub is a multi-user server for Jupyter Notebooks. It is designed to support many users by spawning, managing, and proxying many singular Jupyter Notebook servers.

* **Jupyter Lab**

JupyterLab is the next-generation user interface for Project Jupyter. It offers all the familiar building blocks of the classic Jupyter Notebook in a flexible and powerful user interface. The first stable release was announced on February 20, 2018.

* **Reference papers**
  + “Rendering Notebooks on GitHub-Jupyter Blog”. Jupyter Blog.2015-05-07. Retrieved 2018-04-10.
  + “Amazon SageMaker on AWS”. Amazon Web Services,Inc.Retrieved 2018-05-09.
  + “Welcoming to Collaboratory” research.google.com. Retrieved 2018-05-09.

|  |  |  |
| --- | --- | --- |
| Department of CSE, RNSIT | 2018-2019 | 7 |

Chapter 2 LITERATURE REVIEW

**2.3 Python 3**

Python 3.0 (also called "Python 3000" or "Py3K") was released on December 3, 2008. It was designed to rectify fundamental design flaws in the language—the changes required could not be implemented while retaining full backwards compatibility with the 2.x series, which necessitated a new major version number. The guiding principle of Python 3 was: "reduce feature duplication by removing old ways of doing things".

Python 3.0 was developed with the same philosophy as in prior versions. However, as Python had accumulated new and redundant ways to program the same task, Python 3.0 had an emphasis on removing duplicative constructs and modules, in keeping with "There should be one and preferably only one obvious way to do it".

Nonetheless, Python 3.0 remained a [multi-paradigm language.](https://en.wikipedia.org/wiki/Multi-paradigm_programming_language) Coders still had options among [object-orientation,](https://en.wikipedia.org/wiki/Object-oriented) [structured programming,](https://en.wikipedia.org/wiki/Structured_programming) [functional programming](https://en.wikipedia.org/wiki/Functional_programming) and other paradigms, but within such broad choices, the details were intended to be more obvious in Python 3.0 than they were in Python 2.x.

* **Compatibility**

Python 3.0 broke [backward compatibility,](https://en.wikipedia.org/wiki/Backward_compatibility) and much Python 2 code does not run unmodified on Python 3. Python's [dynamic typing](https://en.wikipedia.org/wiki/Dynamic_typing) combined with the plans to change the semantics of certain methods of dictionaries, for example, made perfect [mechanical translation](https://en.wikipedia.org/wiki/Source-to-source_compiler) from Python 2.x to Python 3.0 very difficult. A tool called "2to3" does the parts of translation that can be done automatically. At this, 2to3 appeared to be fairly successful, though an early review noted that there were aspects of translation that such a tool would never be able to handle.

Prior to the roll-out of Python 3, projects requiring compatibility with both the 2.x and 3.x series were recommended to have one source (for the 2.x series), and produce releases for the Python 3.x platform using 2to3. Edits to the Python 3.x code were discouraged for so long as the code needed to run on Python 2.x. This is no longer recommended as of 2012 the preferred approach is to create a single code base that can run under both Python 2 and 3 using compatibility modules

* **Features**

Some of the major changes included for Python 3.0 were:

* Changing print so that it is a built-in function, not a statement. This made it easier to change a module to use a different print function, as well as making the syntax more regular. In Python 2.6 and 2.7 print () is available as a built-in but is masked by the

|  |  |  |
| --- | --- | --- |
| Department of CSE, RNSIT | 2018-2019 | 8 |

Chapter 2 LITERATURE REVIEW

print statement syntax, which can be disabled by entering from future import print function at the top of the file.



* + Removal of the Python 2 input function, and the renaming of the raw input function to input . Python 3's input function behaves like Python 2's raw input function, in that the input is always returned as a string rather than being evaluated as an expression.
  + Adding support for optional function annotations that can be used for informal type declarations or other purposes.
  + Unifying the str/Unicode types, representing text, and introducing a separate immutable bytes type and a mostly corresponding mutable byte array type, both of which represent arrays of [bytes.](https://en.wikipedia.org/wiki/Byte)
  + Removing backward-compatibility features, including old-style classes, string exceptions, and implicit relative imports.
  + A change in [integer division](https://en.wikipedia.org/wiki/Integer_division) functionality: in Python 2, 5/2 is 2; in Python 3, 5/2 is 2.5. (In both Python 2 (2.2 onwards) and Python 3, 5//2 is 2).
* **Reference papers**

 ["PEP 3000 -- Python 3000".](https://www.python.org/dev/peps/pep-3000/) python.org. Retrieved December 27, 2016.

 Guido van Rossum. ["The fate of reduce () in Python 3000".](http://www.artima.com/weblogs/viewpost.jsp?thread=98196) Artima Developer. Retrieved 2007-03-22.

 ["Computer Programming for Everybody".](https://web.archive.org/web/20070329060757/http:/www.python.org/cp4e/) Python Software Foundation. Archived from [the original](https://www.python.org/cp4e/) on March 29, 2007. Retrieved March 22, 2007.

 Neal Norwitz; Barry Warsaw (2006-06-29). ["PEP 361 -- Python 2.6 and 3.0 Release](https://www.python.org/dev/peps/pep-0361/) [Schedule".](https://www.python.org/dev/peps/pep-0361/) Retrieved 2012-10-07.



|  |  |  |
| --- | --- | --- |
| Department of CSE, RNSIT | 2018-2019 | 9 |