Why Python?

Computing ecosystem around Python

Piotr Beling

Uniwersytet Łódzki
(University of Łódź)

2016
last update: 2018

Scientific environments – overview

Some of the most popular commercial products:

- **MATLAB** – a multi-paradigm numerical computing environment and fourth-generation programming language;
- **Maple** – a symbolic and numeric computing environment, and multi-paradigm programming language;
- Wolfram **Mathematica** – a symbolic mathematical computation program.

Some of the most popular free and open-source environments:

- **Python** ecosystem and **SageMath**;
- **R** – language and environment for statistical computing;
- **Julia** – a high-level, high-performance dynamic language for scientific computing;
- **GNU Octave** – mostly compatible with MATLAB;
- **Scilab** – also similar to MATLAB, but less compatible with it than Octave;
- **Maxima** – system for the manipulation of symbolic and numerical expressions.

Scientific environments – overview

Some of the most popular commercial products:

- **MATLAB** – a multi-paradigm numerical computing environment and fourth-generation programming language;
- **Maple** – a symbolic and numeric computing environment, and multi-paradigm programming language;
- Wolfram **Mathematica** – a symbolic mathematical computation program.

Some of the most popular free and open-source environments:

- **Python** ecosystem and **SageMath**;
- **R** – language and environment for statistical computing;
- **Julia** – a high-level, high-performance dynamic language for scientific computing;
- **GNU Octave** – mostly compatible with MATLAB;
- **Scilab** – also similar to MATLAB, but less compatible with it than Octave;
- **Maxima** – system for the manipulation of symbolic and numerical expressions.

Scientific environments – overview

Some of the most popular commercial products:

- **MATLAB** – a multi-paradigm numerical computing environment and fourth-generation programming language;
- **Maple** – a symbolic and numeric computing environment, and multi-paradigm programming language;
- Wolfram **Mathematica** – a symbolic mathematical computation program.

Some of the most popular free and open-source environments:

- **Python** ecosystem and **SageMath**;
- **R** – language and environment for statistical computing;
- **Julia** – a high-level, high-performance dynamic language for scientific computing;
- **GNU Octave** – mostly compatible with MATLAB;
- **Scilab** – also similar to MATLAB, but less compatible with it than Octave;
- **Maxima** – system for the manipulation of symbolic and numerical expressions.

Scientific environments – overview

Some of the most popular commercial products:

- **MATLAB** – a multi-paradigm numerical computing environment and fourth-generation programming language;
- **Maple** – a symbolic and numeric computing environment, and multi-paradigm programming language;
- Wolfram **Mathematica** – a symbolic mathematical computation program.

Some of the most popular free and open-source environments:

- **Python** ecosystem and **SageMath**;
- **R** – language and environment for statistical computing;
- **Julia** – a high-level, high-performance dynamic language for scientific computing;
- **GNU Octave** – mostly compatible with MATLAB;
- **Scilab** – also similar to MATLAB, but less compatible with it than Octave;
- **Maxima** – system for the manipulation of symbolic and numerical expressions.

Scientific environments – overview

Some of the most popular commercial products:

- **MATLAB** – a multi-paradigm numerical computing environment and fourth-generation programming language;
- **Maple** – a symbolic and numeric computing environment, and multi-paradigm programming language;
- Wolfram **Mathematica** – a symbolic mathematical computation program.

Some of the most popular free and open-source environments:

- **Python** ecosystem and **SageMath**;
- **R** – language and environment for statistical computing;
- **Julia** – a high-level, high-performance dynamic language for scientific computing;
- **GNU Octave** – mostly compatible with MATLAB;
- **Scilab** – also similar to MATLAB, but less compatible with it than Octave;
- **Maxima** – system for the manipulation of symbolic and numerical expressions.

Scientific environments – overview

Some of the most popular commercial products:

- **MATLAB** – a multi-paradigm numerical computing environment and fourth-generation programming language;
- **Maple** – a symbolic and numeric computing environment, and multi-paradigm programming language;
- Wolfram **Mathematica** – a symbolic mathematical computation program.

Some of the most popular free and open-source environments:

- **Python** ecosystem and **SageMath**;
- **R** – language and environment for statistical computing;
- **Julia** – a high-level, high-performance dynamic language for scientific computing;
- **GNU Octave** – mostly compatible with MATLAB;
- **Scilab** – also similar to MATLAB, but less compatible with it than Octave;
- **Maxima** – system for the manipulation of symbolic and numerical expressions.

Scientific environments – overview

Some of the most popular commercial products:

- **MATLAB** – a multi-paradigm numerical computing environment and fourth-generation programming language;
- **Maple** – a symbolic and numeric computing environment, and multi-paradigm programming language;
- Wolfram **Mathematica** – a symbolic mathematical computation program.

Some of the most popular free and open-source environments:

- **Python** ecosystem and **SageMath**;
- **R** – language and environment for statistical computing;
- **Julia** – a high-level, high-performance dynamic language for scientific computing;
- **GNU Octave** – mostly compatible with MATLAB;
- **Scilab** – also similar to MATLAB, but less compatible with it than Octave;
- **Maxima** – system for the manipulation of symbolic and numerical expressions.

Scientific environments – overview

Some of the most popular commercial products:

- **MATLAB** – a multi-paradigm numerical computing environment and fourth-generation programming language;
- **Maple** – a symbolic and numeric computing environment, and multi-paradigm programming language;
- Wolfram **Mathematica** – a symbolic mathematical computation program.

Some of the most popular free and open-source environments:

- **Python** ecosystem and **SageMath**;
- **R** – language and environment for statistical computing;
- **Julia** – a high-level, high-performance dynamic language for scientific computing;
- **GNU Octave** – mostly compatible with MATLAB;
- **Scilab** – also similar to MATLAB, but less compatible with it than Octave;
- **Maxima** – system for the manipulation of symbolic and numerical expressions.

Scientific environments – overview

Some of the most popular commercial products:

- **MATLAB** – a multi-paradigm numerical computing environment and fourth-generation programming language;
- **Maple** – a symbolic and numeric computing environment, and multi-paradigm programming language;
- Wolfram **Mathematica** – a symbolic mathematical computation program.

Some of the most popular free and open-source environments:

- **Python** ecosystem and **SageMath**;
- **R** – language and environment for statistical computing;
- **Julia** – a high-level, high-performance dynamic language for scientific computing;
- **GNU Octave** – mostly compatible with MATLAB;
- **Scilab** – also similar to MATLAB, but less compatible with it than Octave;
- **Maxima** – system for the manipulation of symbolic and numerical expressions.

Scientific environments – overview

Some of the most popular commercial products:

- **MATLAB** – a multi-paradigm numerical computing environment and fourth-generation programming language;
- **Maple** – a symbolic and numeric computing environment, and multi-paradigm programming language;
- Wolfram **Mathematica** – a symbolic mathematical computation program.

Some of the most popular free and open-source environments:

- **Python** ecosystem and **SageMath**;
- **R** – language and environment for statistical computing;
- **Julia** – a high-level, high-performance dynamic language for scientific computing;
- **GNU Octave** – mostly compatible with MATLAB;
- **Scilab** – also similar to MATLAB, but less compatible with it than Octave;
- **Maxima** – system for the manipulation of symbolic and numerical expressions.

What is Python?

Python is a **programing language** which is:

- interpreted, object-oriented, and very high level;
- beautiful and easy (syntax);
- general-purpose (with many general and specific use libraries, including libraries for math calculations, web programming, GUI development, databases, ...);
- free and open-source;
- very popular, with huge community;
- portable (Windows, Linux, ...);
- easy to integrate with software written in other languages (it is often use as a 'glue' for combining and controlling different software together).

Thanks to its features, Python has a strong position in scientific computing.

What is Python?

Python is a **programing language** which is:

- interpreted, object-oriented, and very high level;
- beautiful and easy (syntax);
- general-purpose (with many general and specific use libraries, including libraries for math calculations, web programming, GUI development, databases, ...);
- free and open-source;
- very popular, with huge community;
- portable (Windows, Linux, ...);
- easy to integrate with software written in other languages (it is often use as a 'glue' for combining and controlling different software together).

Thanks to its features, Python has a strong position in scientific computing.

What is Python?

Python is a **programing language** which is:

- interpreted, object-oriented, and very high level;
- beautiful and easy (syntax);
- general-purpose (with many general and specific use libraries, including libraries for math calculations, web programming, GUI development, databases, ...);
- free and open-source;
- very popular, with huge community;
- portable (Windows, Linux, ...);
- easy to integrate with software written in other languages (it is often use as a 'glue' for combining and controlling different software together).

Thanks to its features, Python has a strong position in scientific computing.

What is Python?

Python is a **programing language** which is:

- interpreted, object-oriented, and very high level;
- beautiful and easy (syntax);
- general-purpose (with many general and specific use libraries, including libraries for math calculations, web programming, GUI development, databases, ...);
- free and open-source;
- very popular, with huge community;
- portable (Windows, Linux, ...);
- easy to integrate with software written in other languages (it is often use as a 'glue' for combining and controlling different software together).

Thanks to its features, Python has a strong position in scientific computing.

What is Python?

Python is a **programing language** which is:

- interpreted, object-oriented, and very high level;
- beautiful and easy (syntax);
- general-purpose (with many general and specific use libraries, including libraries for math calculations, web programming, GUI development, databases, ...);
- free and open-source;
- very popular, with huge community;
- portable (Windows, Linux, ...);
- easy to integrate with software written in other languages (it is often use as a 'glue' for combining and controlling different software together).

Thanks to its features, Python has a strong position in scientific computing.

What is Python?

Python is a **programing language** which is:

- interpreted, object-oriented, and very high level;
- beautiful and easy (syntax);
- general-purpose (with many general and specific use libraries, including libraries for math calculations, web programming, GUI development, databases, ...);
- free and open-source;
- very popular, with huge community;
- portable (Windows, Linux, ...);
- easy to integrate with software written in other languages (it is often use as a 'glue' for combining and controlling different software together).

Thanks to its features, Python has a strong position in scientific computing.

What is Python?

Python is a **programing language** which is:

- interpreted, object-oriented, and very high level;
- beautiful and easy (syntax);
- general-purpose (with many general and specific use libraries, including libraries for math calculations, web programming, GUI development, databases, ...);
- free and open-source;
- very popular, with huge community;
- portable (Windows, Linux, ...);
- easy to integrate with software written in other languages (it is often use as a 'glue' for combining and controlling different software together).

Thanks to its features, Python has a strong position in scientific computing.

What is Python?

Python is a **programing language** which is:

- interpreted, object-oriented, and very high level;
- beautiful and easy (syntax);
- general-purpose (with many general and specific use libraries, including libraries for math calculations, web programming, GUI development, databases, ...);
- free and open-source;
- very popular, with huge community;
- portable (Windows, Linux, ...);
- easy to integrate with software written in other languages (it is often use as a 'glue' for combining and controlling different software together).

Thanks to its features, Python has a strong position in scientific computing.

What is Python?

Python is a **programing language** which is:

- interpreted, object-oriented, and very high level;
- beautiful and easy (syntax);
- general-purpose (with many general and specific use libraries, including libraries for math calculations, web programming, GUI development, databases, ...);
- free and open-source;
- very popular, with huge community;
- portable (Windows, Linux, ...);
- easy to integrate with software written in other languages (it is often use as a 'glue' for combining and controlling different software together).

Thanks to its features, Python has a strong position in scientific computing.

Computing ecosystem around Python

The different scientific communities have built a whole technical computing ecosystem around Python.

Some of the most important parts of this ecosystem are the:

- NumPy – the fundamental library for numerical computation; defines N-dimensional array objects and operations on them;
- SymPy – library for symbolic mathematics;
- SciPy – a library of numerical algorithms and toolboxes, including signal processing, optimization and statistics;
- SciKits – add-on packages for SciPy, for example for image processing (scikit-image) and machine learning (scikit-learn);
- pandas – library providing high-performance, easy-to-use data structures and data analysis tools;
- matplotlib – a mature 2D and 3D plotting package;
- IPython – provides a rich architecture for interactive computing and includes a powerful interactive shell.

Computing ecosystem around Python

The different scientific communities have built a whole technical computing ecosystem around Python.
Some of the most important parts of this ecosystem are the:

- NumPy – the fundamental library for numerical computation; defines N-dimensional array objects and operations on them;
- SymPy – library for symbolic mathematics;
- SciPy – a library of numerical algorithms and toolboxes, including signal processing, optimization and statistics;
- SciKits – add-on packages for SciPy, for example for image processing (scikit-image) and machine learning (scikit-learn);
- pandas – library providing high-performance, easy-to-use data structures and data analysis tools;
- matplotlib – a mature 2D and 3D plotting package;
- IPython – provides a rich architecture for interactive computing and includes a powerful interactive shell.

Computing ecosystem around Python

The different scientific communities have built a whole technical computing ecosystem around Python.
Some of the most important parts of this ecosystem are the:

- NumPy – the fundamental library for numerical computation; defines N-dimensional array objects and operations on them;
- SymPy – library for symbolic mathematics;
- SciPy – a library of numerical algorithms and toolboxes, including signal processing, optimization and statistics;
- SciKits – add-on packages for SciPy, for example for image processing (scikit-image) and machine learning (scikit-learn);
- pandas – library providing high-performance, easy-to-use data structures and data analysis tools;
- matplotlib – a mature 2D and 3D plotting package;
- IPython – provides a rich architecture for interactive computing and includes a powerful interactive shell.

Computing ecosystem around Python

The different scientific communities have built a whole technical computing ecosystem around Python.
Some of the most important parts of this ecosystem are the:

- NumPy – the fundamental library for numerical computation; defines N-dimensional array objects and operations on them;
- SymPy – library for symbolic mathematics;
- SciPy – a library of numerical algorithms and toolboxes, including signal processing, optimization and statistics;
- SciKits – add-on packages for SciPy, for example for image processing (scikit-image) and machine learning (scikit-learn);
- pandas – library providing high-performance, easy-to-use data structures and data analysis tools;
- matplotlib – a mature 2D and 3D plotting package;
- IPython – provides a rich architecture for interactive computing and includes a powerful interactive shell.

Computing ecosystem around Python

The different scientific communities have built a whole technical computing ecosystem around Python.
Some of the most important parts of this ecosystem are the:

- NumPy – the fundamental library for numerical computation; defines N-dimensional array objects and operations on them;
- SymPy – library for symbolic mathematics;
- SciPy – a library of numerical algorithms and toolboxes, including signal processing, optimization and statistics;
- SciKits – add-on packages for SciPy, for example for image processing (scikit-image) and machine learning (scikit-learn);
- pandas – library providing high-performance, easy-to-use data structures and data analysis tools;
- matplotlib – a mature 2D and 3D plotting package;
- IPython – provides a rich architecture for interactive computing and includes a powerful interactive shell.

Computing ecosystem around Python

The different scientific communities have built a whole technical computing ecosystem around Python.
Some of the most important parts of this ecosystem are the:

- NumPy – the fundamental library for numerical computation; defines N-dimensional array objects and operations on them;
- SymPy – library for symbolic mathematics;
- SciPy – a library of numerical algorithms and toolboxes, including signal processing, optimization and statistics;
- SciKits – add-on packages for SciPy, for example for image processing (scikit-image) and machine learning (scikit-learn);
- pandas – library providing high-performance, easy-to-use data structures and data analysis tools;
- matplotlib – a mature 2D and 3D plotting package;
- IPython – provides a rich architecture for interactive computing and includes a powerful interactive shell.

Computing ecosystem around Python

The different scientific communities have built a whole technical computing ecosystem around Python.
Some of the most important parts of this ecosystem are the:

- NumPy – the fundamental library for numerical computation; defines N-dimensional array objects and operations on them;
- SymPy – library for symbolic mathematics;
- SciPy – a library of numerical algorithms and toolboxes, including signal processing, optimization and statistics;
- SciKits – add-on packages for SciPy, for example for image processing (scikit-image) and machine learning (scikit-learn);
- pandas – library providing high-performance, easy-to-use data structures and data analysis tools;
- matplotlib – a mature 2D and 3D plotting package;
- IPython – provides a rich architecture for interactive computing and includes a powerful interactive shell.

Computing ecosystem around Python

The different scientific communities have built a whole technical computing ecosystem around Python.
Some of the most important parts of this ecosystem are the:

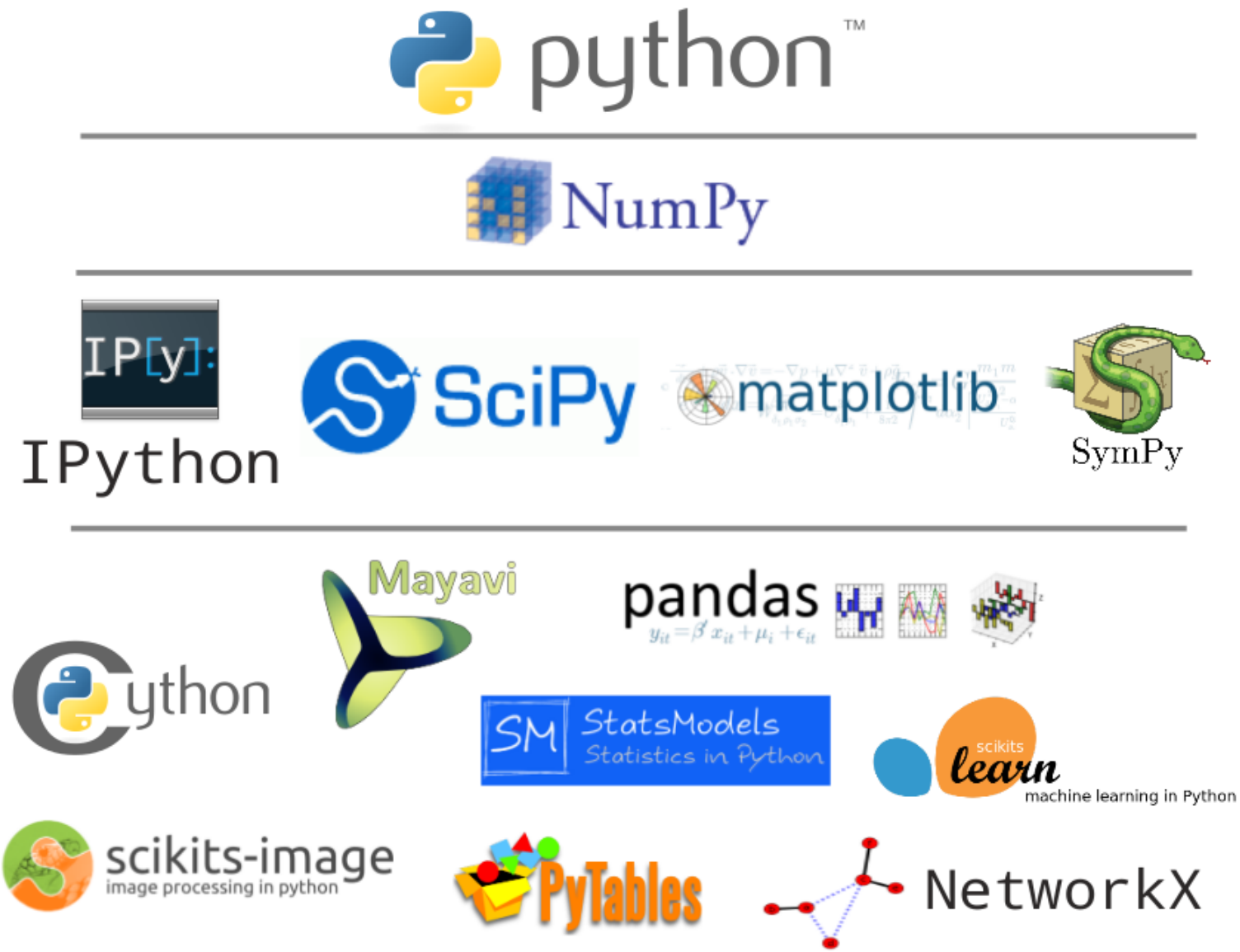
- NumPy – the fundamental library for numerical computation; defines N-dimensional array objects and operations on them;
- SymPy – library for symbolic mathematics;
- SciPy – a library of numerical algorithms and toolboxes, including signal processing, optimization and statistics;
- SciKits – add-on packages for SciPy, for example for image processing (scikit-image) and machine learning (scikit-learn);
- pandas – library providing high-performance, easy-to-use data structures and data analysis tools;
- matplotlib – a mature 2D and 3D plotting package;
- IPython – provides a rich architecture for interactive computing and includes a powerful interactive shell.

Computing ecosystem around Python

The different scientific communities have built a whole technical computing ecosystem around Python.
Some of the most important parts of this ecosystem are the:

- NumPy – the fundamental library for numerical computation; defines N-dimensional array objects and operations on them;
- SymPy – library for symbolic mathematics;
- SciPy – a library of numerical algorithms and toolboxes, including signal processing, optimization and statistics;
- SciKits – add-on packages for SciPy, for example for image processing (scikit-image) and machine learning (scikit-learn);
- pandas – library providing high-performance, easy-to-use data structures and data analysis tools;
- matplotlib – a mature 2D and 3D plotting package;
- IPython – provides a rich architecture for interactive computing and includes a powerful interactive shell.

Scientific Python: a Rich Ecosystem (diagram authored by Fernando Pérez)



SageMath

SageMath (previously Sage or SAGE, System for Algebra and Geometry Experimentation):

- free, open-source competitor to Maple, Mathematica, Magma, and Matlab (popular, proprietary software);
- covers many aspects of mathematics, including algebra, combinatorics, numerical mathematics, number theory, and calculus;
- integrates many specialized mathematics software packages (not only the Python ecosystem, but also Maxima, GAP, FLINT, R and many more) and provides access to their combined power through a Python-based language.

Drawback: There is no native version of SageMath for the Windows operating system. Users of Windows currently have to use virtualization technology such as VirtualBox to run SageMath. For this reason, we will focus on smaller, more portable, pure Python ecosystem which should be good enough for most usages.

SageMath

SageMath (previously Sage or SAGE, System for Algebra and Geometry Experimentation):

- free, open-source competitor to Maple, Mathematica, Magma, and Matlab (popular, proprietary software);
- covers many aspects of mathematics, including algebra, combinatorics, numerical mathematics, number theory, and calculus;
- integrates many specialized mathematics software packages (not only the Python ecosystem, but also Maxima, GAP, FLINT, R and many more) and provides access to their combined power through a Python-based language.

Drawback: There is no native version of SageMath for the Windows operating system. Users of Windows currently have to use virtualization technology such as VirtualBox to run SageMath. For this reason, we will focus on smaller, more portable, pure Python ecosystem which should be good enough for most usages.

SageMath

SageMath (previously Sage or SAGE, System for Algebra and Geometry Experimentation):

- free, open-source competitor to Maple, Mathematica, Magma, and Matlab (popular, proprietary software);
- covers many aspects of mathematics, including algebra, combinatorics, numerical mathematics, number theory, and calculus;
- integrates many specialized mathematics software packages (not only the Python ecosystem, but also Maxima, GAP, FLINT, R and many more) and provides access to their combined power through a Python-based language.

Drawback: There is no native version of SageMath for the Windows operating system. Users of Windows currently have to use virtualization technology such as VirtualBox to run SageMath. For this reason, we will focus on smaller, more portable, pure Python ecosystem which should be good enough for most usages.

SageMath

SageMath (previously Sage or SAGE, System for Algebra and Geometry Experimentation):

- free, open-source competitor to Maple, Mathematica, Magma, and Matlab (popular, proprietary software);
- covers many aspects of mathematics, including algebra, combinatorics, numerical mathematics, number theory, and calculus;
- integrates many specialized mathematics software packages (not only the Python ecosystem, but also Maxima, GAP, FLINT, R and many more) and provides access to their combined power through a Python-based language.

Drawback: There is no native version of SageMath for the Windows operating system. Users of Windows currently have to use virtualization technology such as VirtualBox to run SageMath. For this reason, we will focus on smaller, more portable, pure Python ecosystem which should be good enough for most usages.

SageMath

SageMath (previously Sage or SAGE, System for Algebra and Geometry Experimentation):

- free, open-source competitor to Maple, Mathematica, Magma, and Matlab (popular, proprietary software);
- covers many aspects of mathematics, including algebra, combinatorics, numerical mathematics, number theory, and calculus;
- integrates many specialized mathematics software packages (not only the Python ecosystem, but also Maxima, GAP, FLINT, R and many more) and provides access to their combined power through a Python-based language.

Drawback: There is no native version of SageMath for the Windows operating system. Users of Windows currently have to use virtualization technology such as VirtualBox to run SageMath. For this reason, we will focus on smaller, more portable, pure Python ecosystem which should be good enough for most usages.

Two ways of using Python

1. shell (interactive mode)
   - gives immediate feedback for each statement;
   - useful for experimenting and executing short sequences of commands;
   - tools:
     - **ipython3 qtconsole** (recommended),
     - **Jupyter (IPython) Notebook** (recommended),
     - console in **Spyder** (Scientific PYthon Development EnviRonment) (recommended)
     - IDLE,
     - ipython3,
     - python3 (run without any arguments).
2. source file (normal mode)
   - a script is contained in a text file (usually with *.py* extension)
   - and executed by: `python3 script_file_name.py`
   - Integrated Development Environments (IDEs):
     - **PyCharm** (recommended as a overall IDE);
     - **Spyder** (Scientific PYthon Development EnviRonment) (recommended as a scientific IDE);
     - Eclipse + PyDev (or LiClipse).

Two ways of using Python

1. shell (interactive mode)
   - gives immediate feedback for each statement;
   - useful for experimenting and executing short sequences of commands;
   - tools:
     - **ipython3 qtconsole** (recommended),
     - **Jupyter (IPython) Notebook** (recommended),
     - console in **Spyder** (Scientific PYthon Development EnviRonment) (recommended)
     - IDLE,
     - ipython3,
     - python3 (run without any arguments).
2. source file (normal mode)
   - a script is contained in a text file (usually with *.py* extension)
   - and executed by: `python3 script_file_name.py`
   - Integrated Development Environments (IDEs):
     - **PyCharm** (recommended as a overall IDE);
     - **Spyder** (Scientific PYthon Development EnviRonment) (recommended as a scientific IDE);
     - Eclipse + PyDev (or LiClipse).

Two ways of using Python

1. shell (interactive mode)
   - gives immediate feedback for each statement;
   - useful for experimenting and executing short sequences of commands;
   - tools:
     - **ipython3 qtconsole** (recommended),
     - **Jupyter (IPython) Notebook** (recommended),
     - console in **Spyder** (Scientific PYthon Development EnviRonment) (recommended)
     - IDLE,
     - ipython3,
     - python3 (run without any arguments).
2. source file (normal mode)
   - a script is contained in a text file (usually with *.py* extension)
   - and executed by: `python3 script_file_name.py`
   - Integrated Development Environments (IDEs):
     - **PyCharm** (recommended as a overall IDE);
     - **Spyder** (Scientific PYthon Development EnviRonment) (recommended as a scientific IDE);
     - Eclipse + PyDev (or LiClipse).

Two ways of using Python

1. shell (interactive mode)
   - gives immediate feedback for each statement;
   - useful for experimenting and executing short sequences of commands;
   - tools:
     - **ipython3 qtconsole** (recommended),
     - **Jupyter (IPython) Notebook** (recommended),
     - console in **Spyder** (Scientific PYthon Development EnviRonment) (recommended)
     - IDLE,
     - ipython3,
     - python3 (run without any arguments).
2. source file (normal mode)
   - a script is contained in a text file (usually with *.py* extension)
   - and executed by: `python3 script_file_name.py`
   - Integrated Development Environments (IDEs):
     - **PyCharm** (recommended as a overall IDE);
     - **Spyder** (Scientific PYthon Development EnviRonment) (recommended as a scientific IDE);
     - Eclipse + PyDev (or LiClipse).

Two ways of using Python

1. shell (interactive mode)
   - gives immediate feedback for each statement;
   - useful for experimenting and executing short sequences of commands;
   - tools:
     - **ipython3 qtconsole** (recommended),
     - **Jupyter (IPython) Notebook** (recommended),
     - console in **Spyder** (Scientific PYthon Development EnviRonment) (recommended)
     - IDLE,
     - ipython3,
     - python3 (run without any arguments).
2. source file (normal mode)
   - a script is contained in a text file (usually with *.py* extension)
   - and executed by: `python3 script_file_name.py`
   - Integrated Development Environments (IDEs):
     - **PyCharm** (recommended as a overall IDE);
     - **Spyder** (Scientific PYthon Development EnviRonment) (recommended as a scientific IDE);
     - Eclipse + PyDev (or LiClipse).

Two ways of using Python

1. shell (interactive mode)
   - gives immediate feedback for each statement;
   - useful for experimenting and executing short sequences of commands;
   - tools:
     - **ipython3 qtconsole** (recommended),
     - **Jupyter (IPython) Notebook** (recommended),
     - console in **Spyder** (Scientific PYthon Development EnviRonment) (recommended)
     - IDLE,
     - ipython3,
     - python3 (run without any arguments).
2. source file (normal mode)
   - a script is contained in a text file (usually with *.py* extension)
   - and executed by: `python3 script_file_name.py`
   - Integrated Development Environments (IDEs):
     - **PyCharm** (recommended as a overall IDE);
     - **Spyder** (Scientific PYthon Development EnviRonment) (recommended as a scientific IDE);
     - Eclipse + PyDev (or LiClipse).

Two ways of using Python

1. shell (interactive mode)
   - gives immediate feedback for each statement;
   - useful for experimenting and executing short sequences of commands;
   - tools:
     - **ipython3 qtconsole** (recommended),
     - **Jupyter (IPython) Notebook** (recommended),
     - console in **Spyder** (Scientific PYthon Development EnviRonment) (recommended)
     - IDLE,
     - ipython3,
     - python3 (run without any arguments).
2. source file (normal mode)
   - a script is contained in a text file (usually with *.py* extension)
   - and executed by: `python3 script_file_name.py`
   - Integrated Development Environments (IDEs):
     - **PyCharm** (recommended as a overall IDE);
     - **Spyder** (Scientific PYthon Development EnviRonment) (recommended as a scientific IDE);
     - Eclipse + PyDev (or LiClipse).

Two ways of using Python

1. shell (interactive mode)
   - gives immediate feedback for each statement;
   - useful for experimenting and executing short sequences of commands;
   - tools:
     - **ipython3 qtconsole** (recommended),
     - **Jupyter (IPython) Notebook** (recommended),
     - console in **Spyder** (Scientific PYthon Development EnviRonment) (recommended)
     - IDLE,
     - ipython3,
     - python3 (run without any arguments).
2. source file (normal mode)
   - a script is contained in a text file (usually with *.py* extension)
   - and executed by: `python3 script_file_name.py`
   - Integrated Development Environments (IDEs):
     - **PyCharm** (recommended as a overall IDE);
     - **Spyder** (Scientific PYthon Development EnviRonment) (recommended as a scientific IDE);
     - Eclipse + PyDev (or LiClipse).

Homework – install scientific Python (3.x) ecosystem

Install scientific Python (3.x) ecosystem on your computer:

- Windows: install *Anaconda* (choose *Python 3.x version*) and *PyCharm (community)*;
- Linux: install package with *ipython3 qtconsole* and *PyCharm (community)*. Be ready to install packages with additional Python modules in future.

- **Anaconda** is a high performance distribution of Python. It includes all necessary modules, *ipython qtconsole*, and *spyder* IDE.
  Home page: https://www.continuum.io/
  Download page: https://www.continuum.io/downloads
- **PyCharm** (community) is a very good IDE.
  WWW: https://www.jetbrains.com/pycharm/

Recommended bibliography

About Python and its scientific ecosystem:

- Gaël Varoquaux, et al. *Scipy Lecture Notes* available on http://www.scipy-lectures.org/
- Robert Johansson *Introduction to Scientific Computing in Python*, 2016, book available on http://www.freetechbooks.com/...
- Hans Petter Langtangen *A Primer on Scientific Programming with Python*, 2014, book available on https://hplgit.github.io/...
- Hans Fangohr *Introduction to Python for Computational Science and Engineering (A beginner's guide)*, 2015, book available on http://www.southampton.ac.uk/...

About SageMath:

- Gregory V. Bard *Sage for Undergraduates*, 2014, book available on http://www.gregorybard.com/Sage.html
- George A. Anastassiou, Razvan A. Mezei *Numerical Analysis Using Sage*, 2015, New York, Springer