

Intro to Python — SMM692

Getting Started with Python

Simone Santoni

Bayes Business School

MSc Pre-Course Series

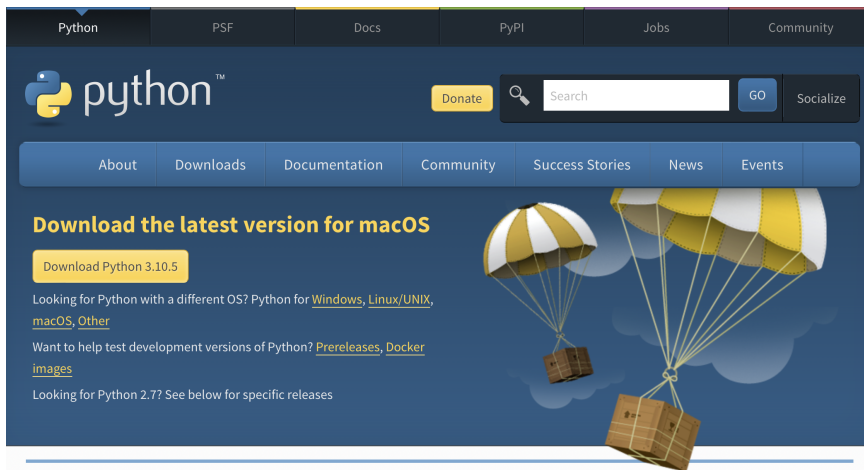
Outline

- 1 Installing Python
 - Options
 - Installation Procedure for Anaconda
- 2 The Anaconda Distribution
 - Distinctive Features
 - The Bundle of Applications
- 3 How Python Runs Programs
- 4 How We Run Python Programs
 - Non-Interactive Approach
 - Interactive Approach
- 5 Managing Python Environments
 - The What and Why of PyEnvs
 - The How of PyEnvs

Outline

- 1 Installing Python
 - Options
 - Installation Procedure for Anaconda
- 2 The Anaconda Distribution
 - Distinctive Features
 - The Bundle of Applications
- 3 How Python Runs Programs
- 4 How We Run Python Programs
 - Non-Interactive Approach
 - Interactive Approach
- 5 Managing Python Environments
 - The What and Why of PyEnvs
 - The How of PyEnvs

Option 1: Official Installer

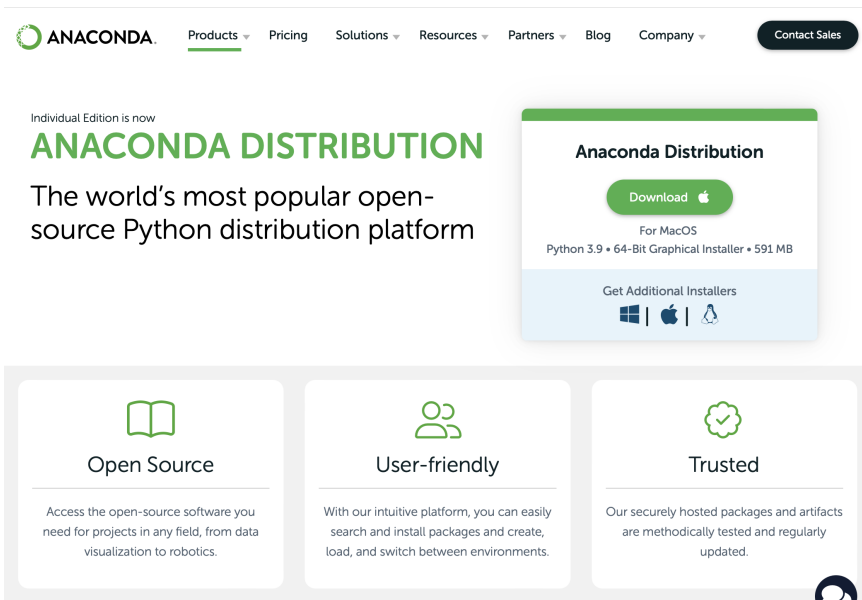
The screenshot shows the Python.org homepage. At the top, there's a navigation bar with links to Python, PSF, Docs, PyPI, Jobs, and Community. Below this is the Python logo and a search bar with a 'GO' button. A 'Donate' button is also visible. A secondary navigation bar contains links to About, Downloads, Documentation, Community, Success Stories, News, and Events. The main content area features a large banner for downloading the latest version for macOS, with a 'Download Python 3.10.5' button. To the right of the banner is an illustration of two parachutes carrying crates. Below the banner, there's text about finding Python for different OSes and links to prereleases and Docker images. At the bottom, there's a section for 'Active Python Releases' with a table of release information.

Active Python Releases

For more information visit the [Python Developer's Guide](#).

Python version	Maintenance status	First released	End of support	Release schedule
3.10	bugfix	2021-10-04	2026-10	PEP 619
3.9	security	2020-10-05	2025-10	PEP 596

Option 2: Anaconda Distro (Preferred Way)



The screenshot shows the Anaconda Distribution website. At the top is a navigation bar with the Anaconda logo, links for Products, Pricing, Solutions, Resources, Partners, Blog, and Company, and a Contact Sales button. The main content area features the text 'Individual Edition is now' followed by 'ANACONDA DISTRIBUTION' in large green letters. Below this is the tagline 'The world's most popular open-source Python distribution platform'. To the right is a download box for macOS with a 'Download' button and details about the installer. Below the main text are three feature boxes: 'Open Source', 'User-friendly', and 'Trusted', each with an icon and a description. At the bottom right is a chat bubble icon and a set of navigation icons.

ANACONDA


Products Pricing Solutions Resources Partners Blog Company Contact Sales

Individual Edition is now

ANACONDA DISTRIBUTION




The world's most popular open-source Python distribution platform


Anaconda Distribution


Download 


For MacOS
Python 3.9 • 64-Bit Graphical Installer • 591 MB

Get Additional Installers

 |  | 


Open Source
Access the open-source software you need for projects in any field, from data visualization to robotics.


User-friendly
With our intuitive platform, you can easily search and install packages and create, load, and switch between environments.


Trusted
Our securely hosted packages and artifacts are methodically tested and regularly updated.

Outline

- 1 Installing Python
 - Options
 - Installation Procedure for Anaconda
- 2 The Anaconda Distribution
 - Distinctive Features
 - The Bundle of Applications
- 3 How Python Runs Programs
- 4 How We Run Python Programs
 - Non-Interactive Approach
 - Interactive Approach
- 5 Managing Python Environments
 - The What and Why of PyEnvs
 - The How of PyEnvs

Steps

- ❶ Download the installer for your operating system (unless you have a very old machine running Win, go for the 64-Bit version)
- ❷ Run the installer
 - For Linux: navigate to the folder where you have downloaded the installer as per step 1, open a shell session, then run `$ bash ./Anaconda3-XXXX.XX-Linux-x86-64.sh`
 - For Win and Mac OS: just run the graphical installer downloaded in step 1
- ❸ Accept the terms proposed by the Anaconda people to use their software, comprising Python, the conda package manager, and a bundle of modules for data science
- ❹ That's it!
 - For Linux users: if you accepted the default installation options, an environmental variable has been created either in your `.bashrc` or `.zshrc`. That means you can access the various pieces of software included in the Anaconda installation (e.g., Anaconda Navigator) from a shell session
 - For Win and Mac OS users: the various pieces of software included in the Anaconda installation are available from the menu of your system

Outline

- 1 Installing Python
 - Options
 - Installation Procedure for Anaconda
- 2 The Anaconda Distribution
 - Distinctive Features
 - The Bundle of Applications
- 3 How Python Runs Programs
- 4 How We Run Python Programs
 - Non-Interactive Approach
 - Interactive Approach
- 5 Managing Python Environments
 - The What and Why of PyEnvs
 - The How of PyEnvs

How is the Anaconda Distro Different?

- Anaconda is 'battery-included' — it comes with a humongous number of modules for data science
 - If you use the official Python installation, then you have to install the modules you need on your own!
- Anaconda is a bundle of various pieces of software:
 - `conda` is the Swiss army knife to manage Python modules and environments
 - `anaconda-navigator` is the graphical interface from within to access Python IDEs and related desktop/web applications

Outline

- 1 Installing Python
 - Options
 - Installation Procedure for Anaconda
- 2 The Anaconda Distribution
 - Distinctive Features
 - **The Bundle of Applications**
- 3 How Python Runs Programs
- 4 How We Run Python Programs
 - Non-Interactive Approach
 - Interactive Approach
- 5 Managing Python Environments
 - The What and Why of PyEnvs
 - The How of PyEnvs

What Software Can I Access from Anaconda Navigator?


[Connect](#)
[Home](#)
[Environments](#)
[Learning](#)
[Community](#)

[End-to-end package security, guaranteed](#)
[Documentation](#)
[Anaconda Blog](#)

 Applications on base (root)
[Channels](#)

DataSpell

DataSpell is an IDE for exploratory data analysis and prototyping machine learning models. It combines the interactivity of Jupyter notebooks with the intelligent Python and R coding assistance of PyCharm in one user-friendly environment.

[Install](#)

DataLore

Online Data Analysis Tool with smart coding assistance by JetBrains. Edit and run your Python notebooks in the cloud and share them with your team.

[Launch](#)

IBM Watson Studio Cloud

IBM Watson Studio Cloud provides you the tools to analyze and visualize data, to cleanse and shape data, to create and train machine learning models. Prepare data and build models, using open source data science tools or visual modeling.

[Launch](#)

JupyterLab
[3.0.14](#)

An extensible environment for interactive and reproducible computing, based on the Jupyter Notebook and Architecture.

[Launch](#)

Notebook
[6.3.0](#)

Web-based, interactive computing notebook environment. Edit and run human-readable docs while describing the data analysis.


IPyT Console
[5.0.3](#)

PyQt GUI that supports inline figures, proper multiline editing with syntax highlighting, graphical calltips, and more.


Spyder
[4.2.5](#)

Scientific Python Development Environment. Powerful Python IDE with advanced editing, interactive testing, debugging and introspection features


VS Code
[1.69.2](#)

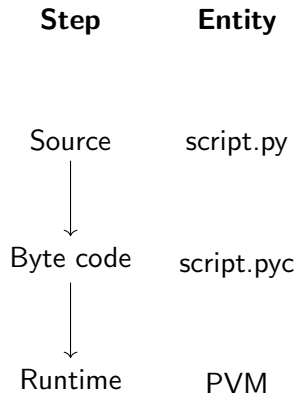
Streamlined code editor with support for development operations like debugging, task running and version control.

Python as a Language and Interpreter

- Typically, we refer to Python as a programming language
- However, Python is 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
 - In effect, the interpreter is a layer of software logic between your code and the computer hardware on your machine

Python Execution Model

- When you instruct Python to run your script (e.g., 'script.py'), there are a few steps the interpreter carries out before your code actually starts crunching away
- First, the code is compiled to something called 'byte code'
 - This step happens behind the scenes (there is nothing to do for the programmer!)
 - A file called 'script.pyc', automatically generated, contains the translation of your code into lower-level code instructions
- Then, the compiled code is routed to something called a 'Python Virtual Machine' (PVM)
 - This step is hidden to the programmer like the previous one
 - Mainly, it iterates through your byte code instructions, one by one, to carry out their operations



Outline

- 1 Installing Python
 - Options
 - Installation Procedure for Anaconda
- 2 The Anaconda Distribution
 - Distinctive Features
 - The Bundle of Applications
- 3 How Python Runs Programs
- 4 How We Run Python Programs**
 - Non-Interactive Approach**
 - Interactive Approach
- 5 Managing Python Environments
 - The What and Why of PyEnvs
 - The How of PyEnvs

Script Preparation → Script Run

Step 1: script preparation

The below displayed Python code achieves two things: i) it prints the string object “Bazinga!”, and ii) it prints the result of the algebraic operation $4 + 2$. Note that all lines starting with # are not considered Python code — instead, they are comments that illustrate/explain the logic of the script.

```
# Print a string object
print("Bazinga")
# Print the result of an algebraic operation
print(2 + 4)
```

Step 2: script run



```
(base) → ~ python simple_script.py
Bazinga
6
(base) → ~
```

Outline

- 1 Installing Python
 - Options
 - Installation Procedure for Anaconda
- 2 The Anaconda Distribution
 - Distinctive Features
 - The Bundle of Applications
- 3 How Python Runs Programs
- 4 How We Run Python Programs**
 - Non-Interactive Approach
 - Interactive Approach**
- 5 Managing Python Environments
 - The What and Why of PyEnvs
 - The How of PyEnvs

Running a Python Shell in the Terminal

```
(base) → ~ python
Python 3.9.7 (default, Sep 16 2021, 08:50:36)
[Clang 10.0.0 ] :: Anaconda, Inc. on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Bazinga")
Bazinga
>>> print(2 + 4)
6
>>> █
```

Running an IPython Shell in the Terminal

```
(base) → ~ ipython
Python 3.9.7 (default, Sep 16 2021, 08:50:36)
Type 'copyright', 'credits' or 'license' for more information
IPython 7.29.0 -- An enhanced Interactive Python. Type '?' for help.

In [1]: print("Bazinga")
Bazinga

In [2]: print(2 + 4)
6

In [3]:
```

Running a Python IDE

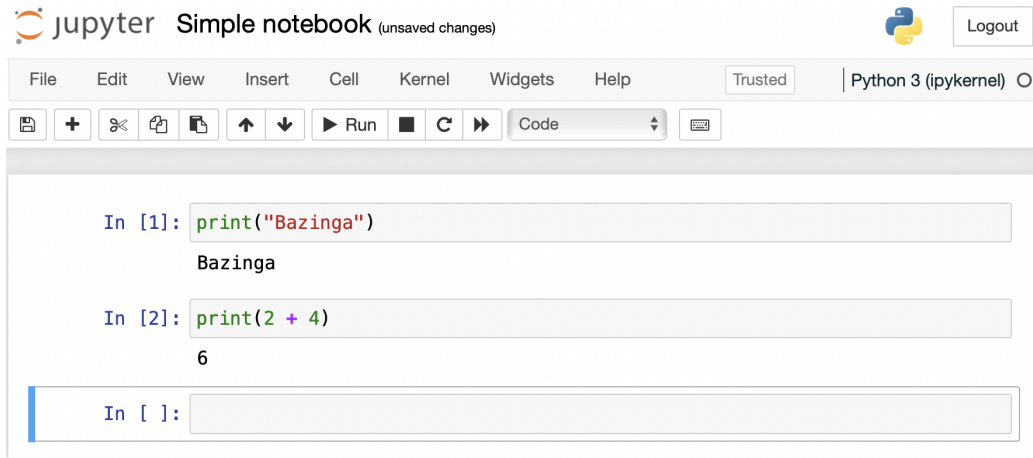
There are plenty of Python IDEs in the market, including:

- Colab (online)
- IDLE
- Datalore (online)
- Jupyter/Jupyterlab
- PyCharm
- Qt Console
- Spyder
- Thonny
- Wing

By installing a couple of plugins, the following (advanced) text editors can turn into Python IDEs:

- Emacs
- Vim/Neovim
- VSCode

Interactive Python Coding with Jupyter



The screenshot shows the Jupyter Simple notebook interface. At the top, the Jupyter logo is followed by the text "Simple notebook (unsaved changes)". To the right is a Python logo and a "Logout" button. Below this is a menu bar with options: File, Edit, View, Insert, Cell, Kernel, Widgets, Help, Trusted, and Python 3 (ipykernel). Under the menu bar is a toolbar with icons for saving, adding, deleting, copying, pasting, undo, redo, and running code. The main area contains three code cells. The first cell has the input `In [1]: print("Bazinga")` and the output `Bazinga`. The second cell has the input `In [2]: print(2 + 4)` and the output `6`. The third cell is empty, showing `In []:`.

jupyter Simple notebook (unsaved changes)

Python 3 (ipykernel)

File Edit View Insert Cell Kernel Widgets Help Trusted Python 3 (ipykernel)

Save Add Delete Copy Paste Undo Redo Run Code

```
In [1]: print("Bazinga")
Bazinga
```

```
In [2]: print(2 + 4)
6
```

```
In [ ]:
```

Interactive Python Coding with VSCode

The screenshot shows the VS Code interface with a file named `simple_script.py` open. The editor contains the following code:

```
1 print("Bazinga")
2 print(2 + 4)
```

The right-hand side of the interface shows the interactive Python environment. It includes a toolbar with 'Clear All', 'Restart', and a terminal icon, along with the text 'tech_sci (Python 3.9.7)'. The output area displays the results of the script execution:

```
Python 3.9.7 (default, Sep 16 2021, 08:50:36)
Type 'copyright', 'credits' or 'license' for more
information
IPython 7.29.0 -- An enhanced Interactive
Python. Type '?' for help.

✓ print("Bazinga") ...
... Bazinga

✓ print(2 + 4) ...
... 6
```

Interactive Python Coding in Jupyter with VSCode

The screenshot displays the VS Code Jupyter interface. The top bar shows the notebook name 'simple_script.ipynb — intro-to-Python-SMM692'. The left sidebar contains icons for Explorer, Search, Source Control, Run and Debug, Testing, Extensions, Python, Remote Explorer, and Docker. The main editor area shows the notebook content. The first cell contains the code `print("Bazinga")`, which has been executed successfully, as indicated by the green checkmark and '0.3s' execution time. The output of this cell is 'Bazinga'. The second cell contains the code `print(2 + 4)`, which has also been executed successfully, with a green checkmark and '0.2s' execution time. The output of this cell is '6'. The third cell is currently empty and has not been executed yet, showing a blue cursor. The bottom status bar indicates the current cell is a Python cell.

simple_script.ipynb — intro-to-Python-SMM692

simple_script.ipynb U

simple_script.ipynb > empty cell

+ Code + Markdown | Run All Clear Outputs of All Cells Restart ... tech_sci (Python 3.9.7)

```
print("Bazinga")
```

[1] ✓ 0.3s Python

... Bazinga

```
print(2 + 4)
```

[2] ✓ 0.2s Python

... 6

[] Python

Outline

- 1 Installing Python
 - Options
 - Installation Procedure for Anaconda
- 2 The Anaconda Distribution
 - Distinctive Features
 - The Bundle of Applications
- 3 How Python Runs Programs
- 4 How We Run Python Programs
 - Non-Interactive Approach
 - Interactive Approach
- 5 Managing Python Environments**
 - The What and Why of PyEnvs**
 - The How of PyEnvs

Why Do We Need a Python Environment?

- ① We do not want to break the system-wise installation of Python!
- ② We want to create collections of Python modules to deploy in specific projects/types of projects

Do Not Touch the System-Wise Installation of Python!

- The large majority of operating systems come with Python installed by default
 - That installation is called 'system Python'
 - 'System Python' runs or manages a large number of essential operating systems processes
- Every time we install a new Python module A , the package manager (e.g, conda) checks the libraries on which A depends
 - If a dependency $D(A)$ is not installed, the package manager will install it for us
- A single module can depend on many libraries
 - The web of dependencies linking Python modules can get complicated
 - If inconsistencies arise within the web of dependencies, we may not be able to install the modules we need
 - ... even worst, some modules previously installed may stop to work!
- Now, you do not want to break some essential features of your operating system
- The implication is that we should be better off not using 'system Python' for our data science work

Create PyEnvs, Ad Hoc Collections of Python Modules!

- Mainly, a Python environment is a Python installation — along with a set of modules — that has nothing to do with ‘system Python’
 - In other words, Python environments are not used by default for any processes regarding the operating systems
- Typically, we create a Python environment to carry out a specific project or a class of projects (e.g., Machine Learning)
- The advantage is twofold:
 - Reliability — the web of dependencies is relatively simple insofar as we install only the few modules that are required by the project or project class. Likely as not, we will not come across installation issues!
 - Reproducibility — the environment explicates the modules necessary to carry out a certain project. Hence, a user who wants to reproduce our project’s results knows which modules to install

Outline

- 1 Installing Python
 - Options
 - Installation Procedure for Anaconda
- 2 The Anaconda Distribution
 - Distinctive Features
 - The Bundle of Applications
- 3 How Python Runs Programs
- 4 How We Run Python Programs
 - Non-Interactive Approach
 - Interactive Approach
- 5 Managing Python Environments**
 - The What and Why of PyEnvs
 - The How of PyEnvs**

Creating PyEnvs from the Command Line (1 out of 2)

```
(base) → ~ conda create -n my_env
Collecting package metadata (current_repodata.json): done
Solving environment: done

## Package Plan ##

  environment location: /Users/sbbk475/opt/anaconda3/envs/my_env

Proceed ([y]/n)? y

Preparing transaction: done
Verifying transaction: done
Executing transaction: done
#
# To activate this environment, use
#
#     $ conda activate my_env
#
# To deactivate an active environment, use
#
#     $ conda deactivate

(base) → ~
```

Creating PyEnvs from the Command Line (2 out of 2)

```
(base) → ~ conda activate my_env  
(my_env) → ~ conda install ipython jupyter numpy matplotlib
```

Creating PyEnvs from within Anaconda Navigator (1 out of 2)

ANACONDA.NAVIGATOR

Home

Environments

Learning

Community

Secure your software supply chain from the source. Upgrade Now

End-to-end package security, guaranteed

Documentation

Anaconda Blog

Create Clone Import Backup Remove

Search Environments

base (root)

Installed Channels Update Index...

Search Packages

Create new environment

Name: my_env

Location: /Users/sbbk475/opt/anaconda3/envs/my_env

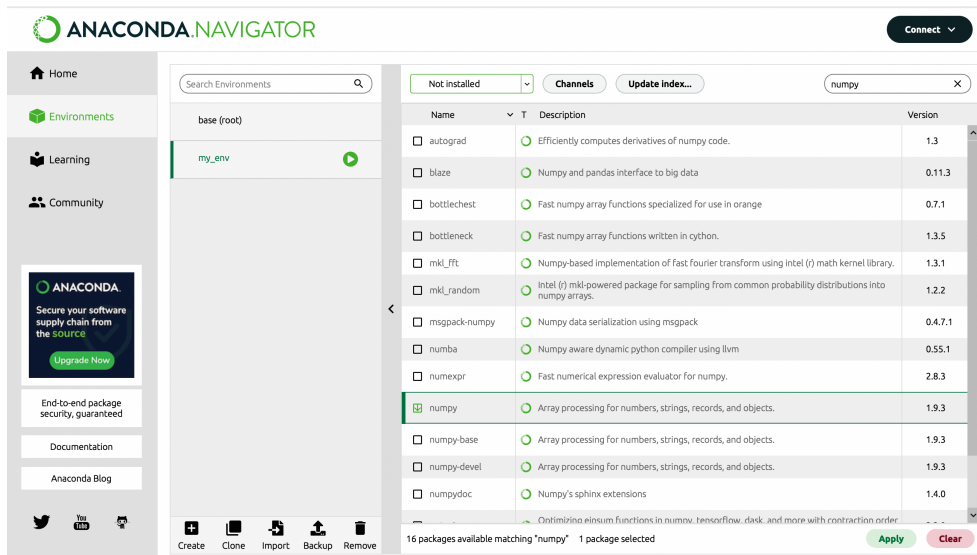
Packages: ☒ Python 3.9.12 ☐ R 3.6.1

Cancel Create

Name	Description	Version
<input checked="" type="checkbox"/> _ipyw_jlab_nb_ex...	A configuration metapackage for enabling anaconda-bundled jupyter extensions	0.1.0
<input checked="" type="checkbox"/> sphinx-theme	able sphinx theme.	0.7.12
<input checked="" type="checkbox"/> and deployment of anaconda		2021.11
<input checked="" type="checkbox"/> ent library		1.9.0
<input checked="" type="checkbox"/> and reproducing data science projects		0.10.1
<input checked="" type="checkbox"/> multiple asynchronous event loop implementations on		2.2.0
<input checked="" type="checkbox"/> rmining appropriate platform-specific dirs.		1.4.4
<input checked="" type="checkbox"/> an app with apple launch services to handle utl and url		0.2.1
<input checked="" type="checkbox"/> appnope	Disable app nap on os x 10.9	0.1.2
<input checked="" type="checkbox"/> appscript	Control applescriptable applications from python.	1.1.2
<input checked="" type="checkbox"/> argh	The natural cli.	0.26.2
<input checked="" type="checkbox"/> argon2-cffi	The secure argon2 password hashing algorithm.	20.1.0
<input checked="" type="checkbox"/> arrow	Better dates & times for python	0.13.1

398 packages available

Creating PyEnvs from within Anaconda Navigator (2 out of 2)



ANACONDA NAVIGATOR

Connect

Home

Environments

Learning

Community

Secure your software supply chain from the source
Upgrade Now

End-to-end package security, guaranteed

Documentation

Anaconda Blog

Twitter YouTube GitHub

Create Clone Import Backup Remove

Search Environments

base (root)

my_env

Not installed Channels Update index...

numpy

Name	Description	Version
<input type="checkbox"/> autograd	Efficiently computes derivatives of numpy code.	1.3
<input type="checkbox"/> blaze	Numpy and pandas interface to big data	0.11.3
<input type="checkbox"/> bottleneck	Fast numpy array functions specialized for use in orange	0.7.1
<input type="checkbox"/> bottleneck	Fast numpy array functions written in cython.	1.3.5
<input type="checkbox"/> mkl_fft	Numpy-based implementation of fast fourier transform using Intel (r) math kernel library.	1.3.1
<input type="checkbox"/> mkl_random	Intel (r) mkl-powered package for sampling from common probability distributions into numpy arrays.	1.2.2
<input type="checkbox"/> msgpack-numpy	Numpy data serialization using msgpack	0.4.7.1
<input type="checkbox"/> numba	Numpy aware dynamic python compiler using llvm	0.55.1
<input type="checkbox"/> numexpr	Fast numerical expression evaluator for numpy.	2.8.3
<input checked="" type="checkbox"/> numpy	Array processing for numbers, strings, records, and objects.	1.9.3
<input type="checkbox"/> numpy-base	Array processing for numbers, strings, records, and objects.	1.9.3
<input type="checkbox"/> numpy-devel	Array processing for numbers, strings, records, and objects.	1.9.3
<input type="checkbox"/> numpydoc	Numpy's sphinx extensions	1.4.0
<input type="checkbox"/> Optimizing einsum functions in numpy, tensorflow, dask, and more with contraction order		

16 packages available matching "numpy" 1 package selected

Apply Clear