**Preface**

**What Is Data Science?**

Data science comprises three distinct and overlapping areas: the skills of a statistician who knows how to model and summarize datasets (which are growing ever larger); the skills of a computer scientist who can design and use algorithms to efficiently store, process, and visualize this data; and the domain expertise—what we might think of as “classical” training in a subject—necessary both to formulate the right questions and to put their answers in context.

A diagram of data science

Description automatically generated

**Why Python**

* NumPy for manipulation of homogeneous array-based data,
* Pandas for manipulation of heterogeneous and labeled data,
* SciPy for common scientific computing tasks,
* Matplotlib for publication-quality visualizations,
* IPython for interactive execution and sharing of code, Scikit-Learn for machine learning

**Jupyter: Beyond Normal Python**

**Getting Started in IPython and Jupyter**

* **IPython shell** for trying out short sequences of commands
* **Jupyter Notebook** for longer interactive analysis and for sharing content with others
* **Interactive development environments (IDEs)** like **Emacs** or **VSCode** for creating reusable Python packages.

**The IPython Shell**

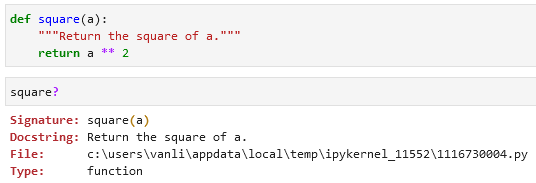
* Start the *IPython Shell* by typing ipython in the *Anaconda Prompt*.
* Launch Jupyter lab (?) $ jupyter lab

**Access documentation** with help() like help(len). The alternative is ? like len?. Get information on objects using ? like in the example below:

A screenshot of a computer

Description automatically generated

You can also get information on functions or other objects you create like in the example below which has a docstring (a description of the function):



?? provides the source code of the object you are curious about:

A screenshot of a computer

Description automatically generated

Use wildcard matching (character \*) to list every object in the namespace whose name ends with *Warning*:

A screenshot of a computer screen

Description automatically generated

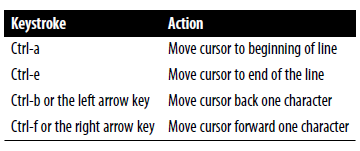
In the example below we area looking for a string method that contains the word *find* somewhere in its name:

A screenshot of a computer

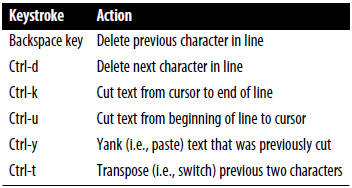
Description automatically generated

**Keyboard Shortcuts in the IPython Shell**

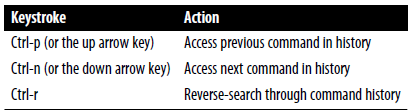
**Navigation Shortcuts**



**Text Entry Shortcuts**



**Command History Shortcuts**

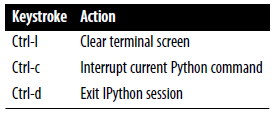


Use *Ctrl-r* to browse the search results. When you’re done press *Enter*.

A close up of text

Description automatically generated

**Miscellaneous Shortcuts**



**Enhanced Interactive Features**

**IPython Magic Commands**

Magic commands are prefixed by the % character.

* **Line magics** are denoted by a single % prefix. They operate on a **single line of input**.
* **Cell magics** are denoted by a double %% prefix and operate on **multiple lines of input**.
* Magic commands documentation %magic.
* Quick and simple list of all available magic functions %lsmagic.

If you have a *.py* file you can execute it / run its script in IPython using %run filename.py:

A screenshot of a computer screen

Description automatically generated

To time code execution use %timeit or %%timeit for multiple lines of code:

A screenshot of a computer

Description automatically generated

You can check you input and output history using In and Out codes. You can also select specific steps like print(In[1]).

A number and numbers on a white background

Description automatically generated

A white background with red text

Description automatically generated

A black text on a white background

Description automatically generated

A screenshot of a computer program

Description automatically generated

An alternative to Out[20] is \_20:



Use print(\_) to get access to the previous output:

A screenshot of a phone

Description automatically generated

You can also use print(\_\_) and print(\_\_\_) to get access to the second/third-to-last outputs.

If you want to suppress you output you can do it using ; at the end of the line:

A white background with black text

Description automatically generated

Use %history to get an overview of your commands, use %history -n to get an numbered overview of your commands. To select a command/commands add a number like 3-5:

A black screen with white text

Description automatically generatedA screenshot of a computer

Description automatically generated

Other useful commands are %rerun (re-execute some portion of the command history) and %save (saves some set of the command history to a file).

**Quick Introduction to the Shell**

Some shell commands:



A screenshot of a computer program

Description automatically generated

echo, pwd, ls, cd, mkdir, mv

A black text on a white background

Description automatically generated

A screenshot of a computer program

Description automatically generated

!echo

A screenshot of a computer

Description automatically generated!pwd !cd

A screenshot of a computer program

Description automatically generated%cd mkdir ls cp rm -r

**Debugging and Profiling**

**Controlling Exceptions using %xmode**

There are 3 formats for errors: Plain, Context, and Verbose.

A screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generatedA screenshot of a computer

Description automatically generated

The last provides the most information but can be bulky.

**To launch a debugger** write %debug.

**Launch the Sdebugger automatically** whenever an exception is raised using

%xmode Plain

%pdb on

A computer code with text

Description automatically generated

**Partial list of debugging commands:**

A screenshot of a computer program

Description automatically generatedl(ist) h(elp) q(uit) c(ontinue) n(ext) <enter> p(rint) s(tep) r(eturn)

**Profiling and Timing Code**

%time Time the execution of a single statement

%timeit Time repeated execution of a single statement for more accuracy

%prun Run code with the profiler

%lprun Run code with the line-by-line profiler

%memit Measure the memory use of a single statement

%mprun Run code with the line-by-line memory profiler

%timeit

A screenshot of a computer code

Description automatically generated

%time

A screenshot of a computer program

Description automatically generated

%prun

A screenshot of a computer

Description automatically generated

A close up of text

Description automatically generated

%lprun

A screenshot of a computer

Description automatically generated%load\_ext line\_profiler %lprun -f

A close up of black text

Description automatically generated

%memit (is a memory-measuring equivalent of %timeit) and %mprun (memory-measuring equivalent of %lprun)

A white rectangular sign with black text

Description automatically generated pip install memory\_profiler

A white rectangular object with black text

Description automatically generated%load\_ext memory\_profiler

A white rectangular object with green text

Description automatically generated with medium confidence %memit

A close-up of a text

Description automatically generated

A screenshot of a computer program

Description automatically generated

A screenshot of a computer program

Description automatically generated%%file %mprun -f

**NumPy**

In some ways, NumPy arrays are like Python’s built-in *list* type, but NumPy arrays provide much more efficient storage and data operations as the arrays grow larger in size.

Check NumPy version: numpy.\_\_version\_\_

**Fixed-Type Arrays in Python**

Example of an array in Python:

A screenshot of a computer code

Description automatically generatedimport array, array.array



Python arrays objects provide efficient storage. NumPy adds to this efficient operations on that data.

**Creating Arrays from Python Lists**

NumPy arrays can only contain data of the same type. This is unlike Python lists.

NumPy arrays can be multidimensional. Python lists are always one-dimensional sequences.

A screenshot of a computer program

Description automatically generated

**Creating Arrays from Scratch**

A close up of words

Description automatically generatedA screenshot of a computer program

Description automatically generated

A screenshot of a computer code

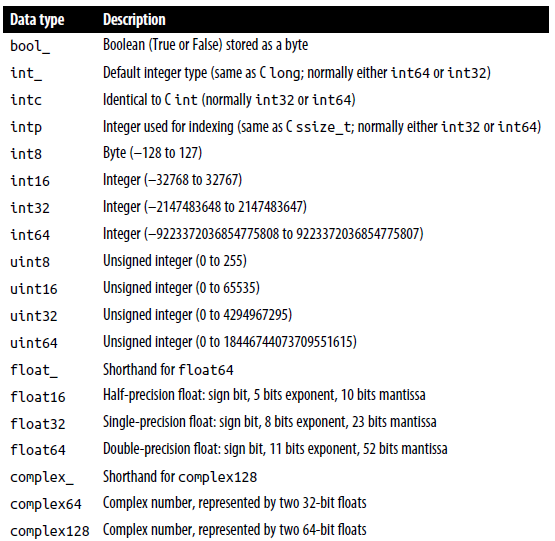
Description automatically generated

**Specify data type when creating an array using dtype=’int16’ or dtype=np.int16:**

A screenshot of a computer

Description automatically generated

**Standard NumPy data types:**



**NumPy Array Attributes**

A screenshot of a computer code

Description automatically generated

ndim, shape, size, dtype, default\_rng(seed= integers

**Array Slicing: Accessing Subarrays**

Follow this logic to access a slice of an array x:

x[start:stop:step]

If any of these are unspecified, they default to the values start=0, stop=<size of dimension>, step=1.



A screenshot of a computer code

Description automatically generated

A white background with black text

Description automatically generatedreverse an array [::-1]

A white background with numbers and symbols

Description automatically generatedreverse a multiarray [::-1, ::-1]

 first column of a multiarray [0, :]

 first row of a multiarray [0] or [0, :]

