DUSTIN MICHELS

DEVX, CARLETON

16 OCTOBER 2017

PYTHON FOR DATA SCIENCE

I. INTRODUCTION

WHAT IS DATA SCIENCE?

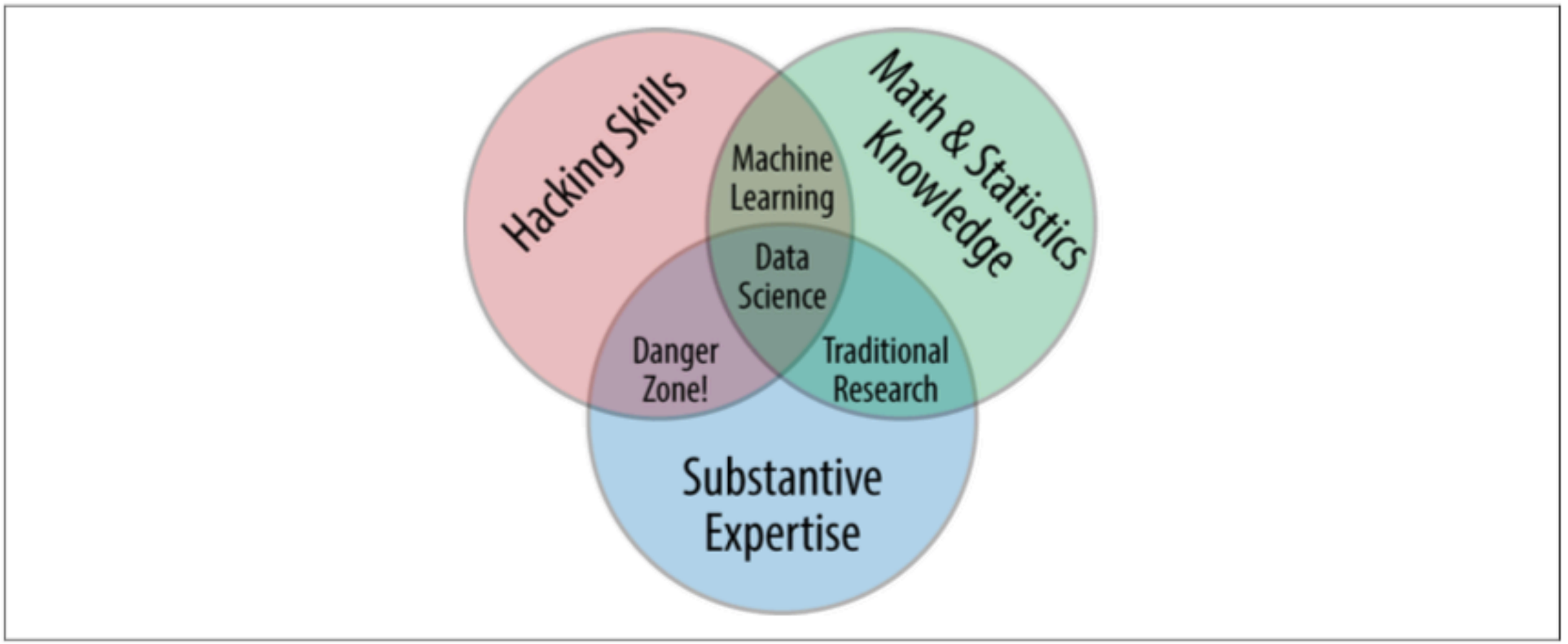


Figure P-1. Drew Conway's Data Science Venn Diagram

"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."

(JAKE VANDERPLAS, PYTHON FOR DATA SCIENCE HANDBOOK, XI)

DRAWING ON WORK OF JAKE VANDERPLAS

- "Why Astronomers Love Python", PyCon 2017, https://youtu.be/lWl6d7mkru4.
- "Python for Data Science Handbook" (2017), O'Reily.

KEY IDEA: PUTTING THE "SCIENCE" IN COMPUTER SCIENCE

- Programming is integral part of scientific work, but not always done scientifically…

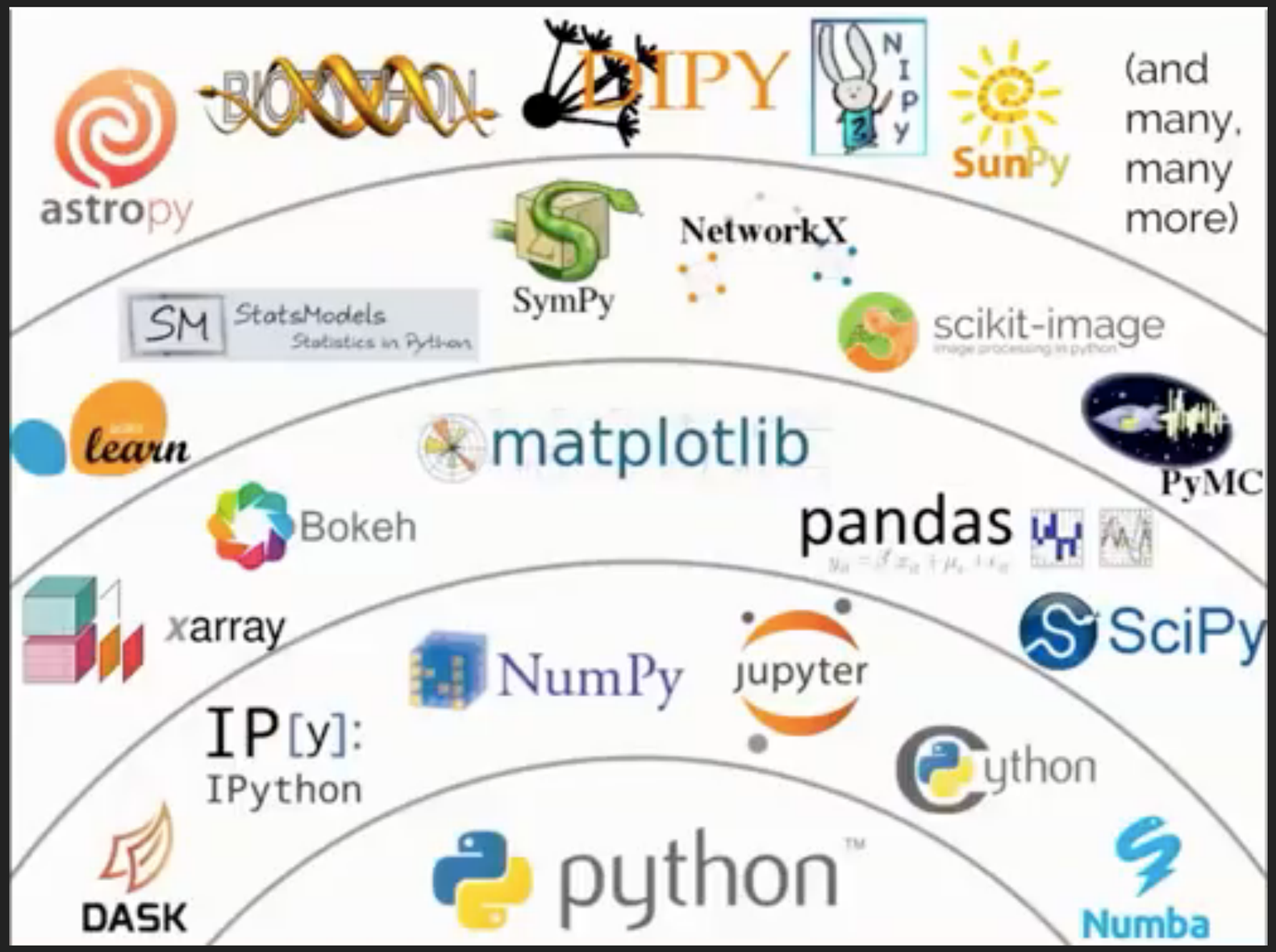
KEY IDEA: PUTTING THE "SCIENCE" IN COMPUTER SCIENCE

- Programming is integral part of scientific work, but not always done scientifically…
- Python and its packages are becoming popular choice
  - Quality
  - Transparency
  - Reproducibly
  - Collaboration

TOOLS

- **Interactivity / collaboration**
  - IPython
  - Jupyter
- **Data wrangling / analysis**
  - Numpy
  - Pandas
- **Visualization**
  - Matplotlib
  - Seaborn

"WHY ASTRONOMERS LOVE PYTHON" (PYCON, 2017)



TODO #1

1. Go to: https://tinyurl.com/devx-data

2. Git clone or download folder

TODO #2

Option A:

Conda/ Miniconda

Option B:

$ pip3 install --upgrade pip`

$ pip3 install numpy matplotlib ipython jupyter

II. INTERACTIVITY / COLLABORATION

IPYTHON / JUPYTER

IPYTHON: IMPROVED "REPL"

```
>>> 2 + 2
4
>>> print("hello")
hello
>>> def hello():
...     print("hello there!")
...
>>> hello()
hello there!
>>>
```

python REPL

```
In [1]: 2 + 2
Out[1]: 4

In [2]: print("hello")
hello

In [3]: def hello():
   ...:     print("hello there!")
   ...:

In [4]: hello()
hello there!
```

ipython REPL

IPYTHON: KEY FEATURES

- Input/output numbering!
- Syntax highlighting!
- Multi-line entry / recall!
- Tab completion!
- Easily pull up documentation with “?” !

IPYTHON: METHOD DISCOVERY

```
In [8]: h = "hello there!"

In [9]: h.
   capitalize()   encode()      format()      isalpha()      islower()
   casefold()     endswith()    format_map()  isdecimal()    isnumeric()
   center()       expandtabs()  index()       isdigit()      isprintable()  >
   count()        find()        isalnum()     isidentifier() isspace()
```

```
In [9]: h.capitalize?
Docstring:
S.capitalize() -> str

Return a capitalized version of S, i.e. make the first character
have upper case and the rest lower case.
Type:      builtin_function_or_method
```

```
In [10]: h.capitalize()
Out[10]: 'Hello there!'
```

IPYTHON: KEYBOARD SHORTCUTS

Keystroke	Action
Ctrl-a	Move cursor to the beginning of the line
Ctrl-e	Move cursor to the end of the line
Ctrl-b (or the left arrow key)	Move cursor back one character
Ctrl-f (or the right arrow key)	Move cursor forward one character

Keystroke	Action
Backspace key	Delete previous character in line
Ctrl-d	Delete next character in line
Ctrl-k	Cut text from cursor to end of line
Ctrl-u	Cut text from beginning fo line to cursor
Ctrl-y	Yank (i.e., paste) text that was previously cut
Ctrl-t	Transpose (i.e., switch) previous two characters

THANKS JAKE!

IPYTHON: REFERENCING PAST IN/OUT

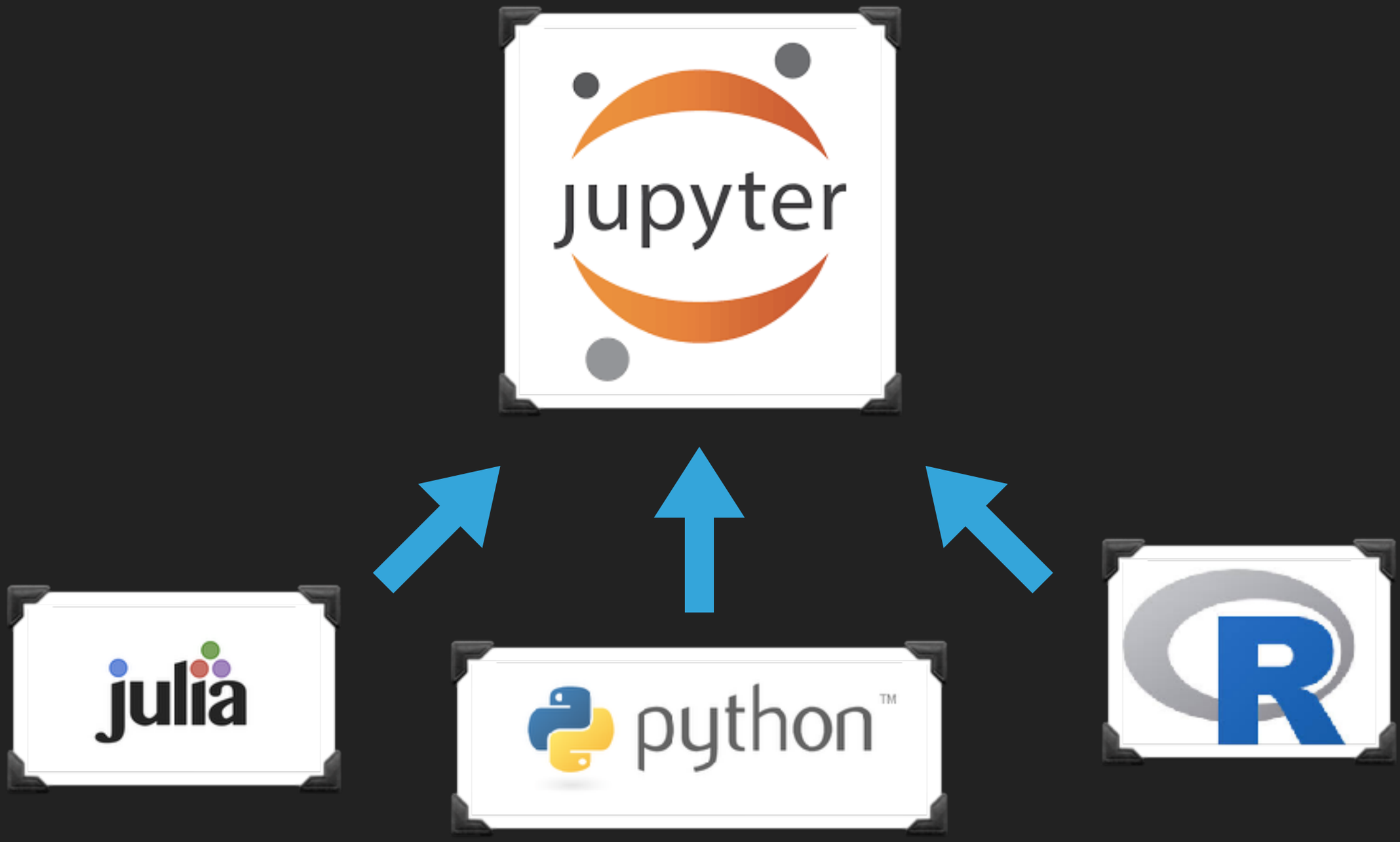
```
In [1]: 2 + 2
Out[1]: 4

In [2]: Out
Out[2]: {1: 4}

In [3]: Out[1]
Out[3]: 4

In [4]: _
Out[4]: 4

In [5]: _ + 3
Out[5]: 7

In [6]: __ + _
Out[6]: 11

In [7]:
```

IPYTHON: MAGIC STUFF!

- Magic Stuff!
  - %paste
  - %timeit
  - %magic

- See: https://jakevdp.github.io/PythonDataScienceHandbook/01.03-magic-commands.html

JUPYTER



JUPYTER

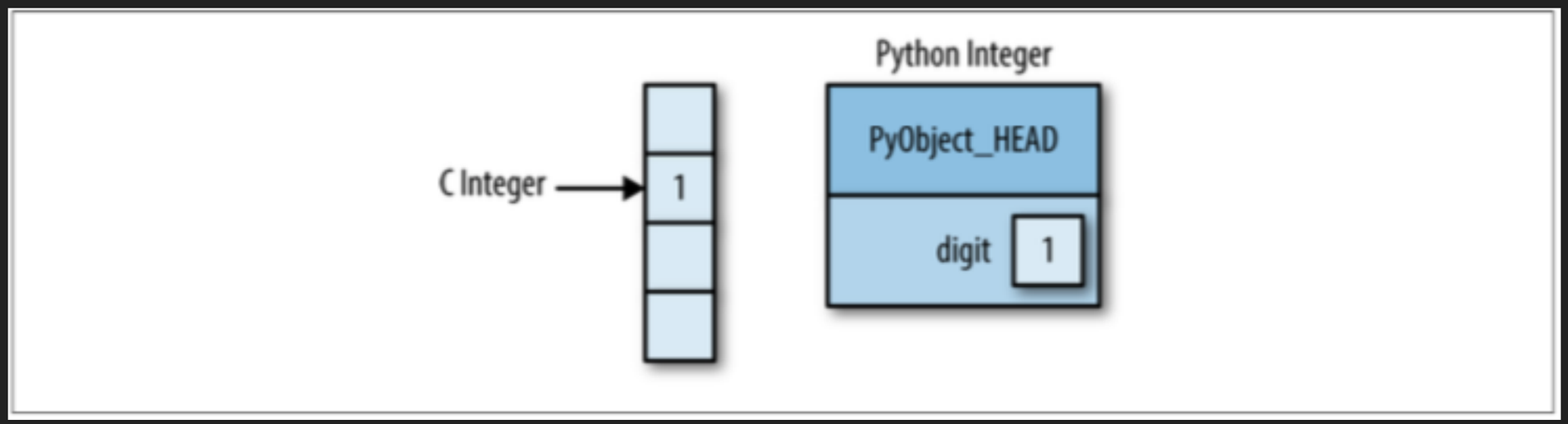
1. Basic navigation: `enter`, `shift-enter`, `up/k`, `down/j`
2. Saving the notebook: `s`
3. Change Cell types: `y`, `m`, `1-6`, `t`
4. Cell creation: `a`, `b`
5. Cell editing: `x`, `c`, `v`, `d`, `z`
6. Kernel operations: `i`, `0` (press twice)

See: https://jupyter-notebook.readthedocs.io/en/stable/examples/Notebook/Notebook%20Basics.html

III. DATA WRANGLING / ANALYSIS

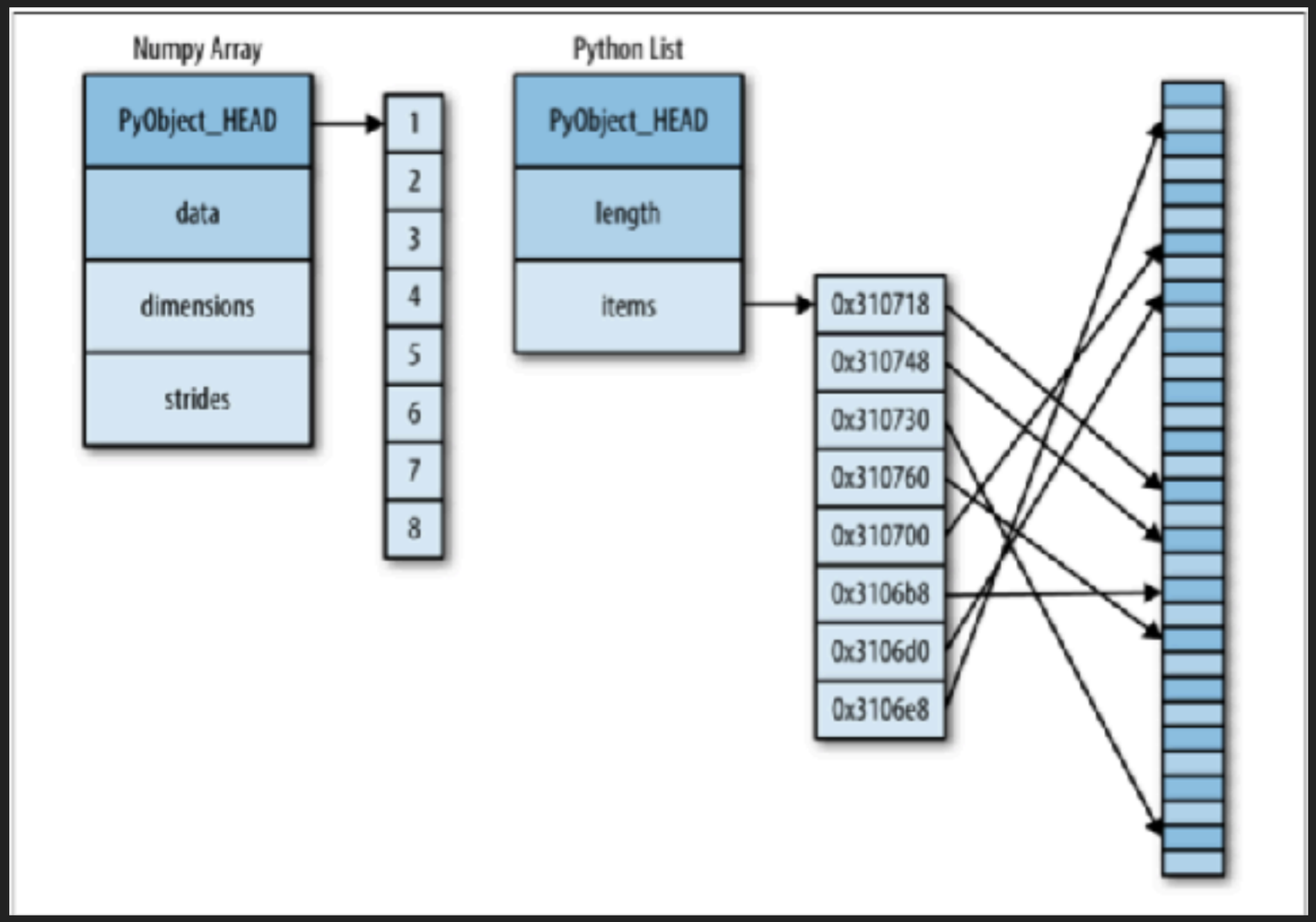
NUMPY / PANDAS

NUMPY



THANKS JAKE!

NUMPY



THANKS JAKE!

NUMPY: DATA TYPES

Table 2-1. Standard NumPy data types

Data type	Description
`bool_`	Boolean (True or False) stored as a byte
`int_`	Default integer type (same as C `long`; normally either `int64` or `int32`)
`intc`	Identical to C `int` (normally `int32` or `int64`)
`intp`	Integer used for indexing (same as C `ssize_t`; normally either `int32` or `int64`)
`int8`	Byte (–128 to 127)
`int16`	Integer (–32768 to 32767)
`int32`	Integer (–2147483648 to 2147483647)
`int64`	Integer (–9223372036854775808 to 9223372036854775807)
`uint8`	Unsigned integer (0 to 255)
`uint16`	Unsigned integer (0 to 65535)
`uint32`	Unsigned integer (0 to 4294967295)
`uint64`	Unsigned integer (0 to 18446744073709551615)
`float_`	Shorthand for `float64`
`float16`	Half-precision float: sign bit, 5 bits exponent, 10 bits mantissa
`float32`	Single-precision float: sign bit, 8 bits exponent, 23 bits mantissa
`float64`	Double-precision float: sign bit, 11 bits exponent, 52 bits mantissa
`complex_`	Shorthand for `complex128`
`complex64`	Complex number, represented by two 32-bit floats
`complex128`	Complex number, represented by two 64-bit floats

THANKS JAKE!

NUMPY: UFUNCS

Table 2-2. Arithmetic operators implemented in NumPy

Operator	Equivalent ufunc	Description
+	np.add	Addition (e.g., 1 + 1 = 2)
-	np.subtract	Subtraction (e.g., 3 - 2 = 1)
-	np.negative	Unary negation (e.g., -2)
*	np.multiply	Multiplication (e.g., 2 * 3 = 6)
/	np.divide	Division (e.g., 3 / 2 = 1.5)
//	np.floor_divide	Floor division (e.g., 3 // 2 = 1)
**	np.power	Exponentiation (e.g., 2 ** 3 = 8)
%	np.mod	Modulus/remainder (e.g., 9 % 4 = 1)

THANKS JAKE!

NUMPY: AGGREGATION FUNCTIONS

Table 2-3. Aggregation functions available in NumPy

Function Name	NaN-safe Version	Description
`np.sum`	`np.nansum`	Compute sum of elements
`np.prod`	`np.nanprod`	Compute product of elements
`np.mean`	`np.nanmean`	Compute median of elements
`np.std`	`np.nanstd`	Compute standard deviation
`np.var`	`np.nanvar`	Compute variance
`np.min`	`np.nanmin`	Find minimum value
`np.max`	`np.nanmax`	Find maximum value
`np.argmin`	`np.nanargmin`	Find index of minimum value
`np.argmax`	`np.nanargmax`	Find index of maximum value
`np.median`	`np.nanmedian`	Compute median of elements
`np.percentile`	`np.nanpercentile`	Compute rank-based statistics of elements
`np.any`	N/A	Evaluate whether any elements are true
`np.all`	N/A	Evaluate whether all elements are true

IV. VISUALIZATION

MATLOTLIB / SEABORN