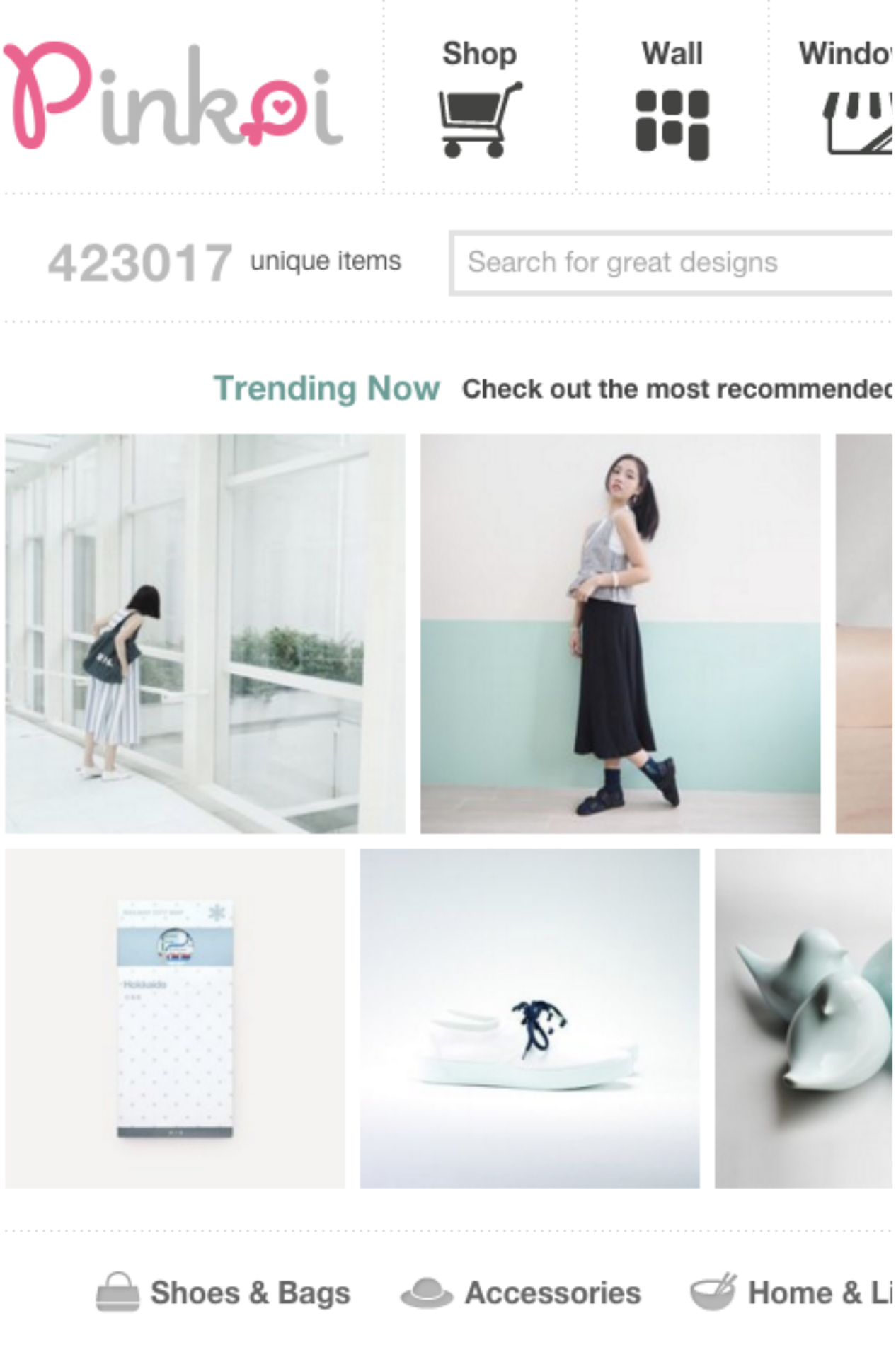
Practicing Python 3

Mosky

Python?

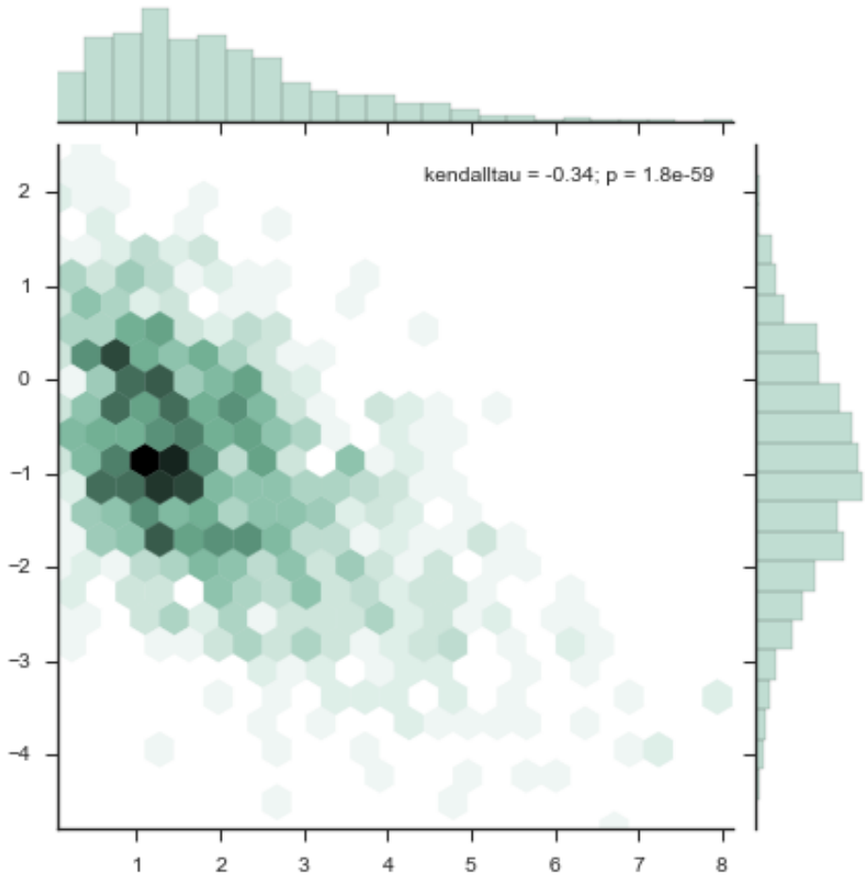


Websites

- Pinkoi
- Google search engine
- Uber
- Pinterest
- use as their back-end language

Desktop Applications

- Dropbox
- BitTorrent
- Blender (3D graphics)
- Disney



Python source code: [download source: hexbin_marginals.py]

```
import numpy as np
from scipy.stats import kendalltau
import seaborn as sns
sns.set(style="ticks")

rs = np.random.RandomState(11)
x = rs.gamma(2, size=1000)
y = -.5 * x + rs.normal(size=1000)

sns.jointplot(x, y, kind="hex", stat_func=kendalltau, color="#4CB391")
```

Science

- NASA
  - gravitational waves
  - etc.
- CERN LHC
- MMTK (Biology)
- use Python to do
  - Data mining
  - Data visualization
  - Scientific programming
  - Simulation

Embedded System

- iRobot uses Python.
- Raspberry Pi supports Python.
- Linux has built-in Python.

Why?

- Python is slower than C, Java
- but much faster to write
- easy to speed up
  - Numba
  - Cython
- has rich libraries
- emphasizes code readability
  - easier to learn
  - easier to co-work.
- “Time is money.”

Showcases

A Website in a Minute

```
from flask import Flask

app = Flask(__name__)

@app.route("/")
def hello():
    return "Hello World!"

if __name__ == "__main__":
    app.run()
```

Symbolic Mathematics

```
from sympy import init_printing
init_printing()

from sympy import symbols
x = symbols('x')
x**2
```

$$x^2$$

```
diff(x**2)
```

$$2x$$

```
from sympy import symbols
from sympy import diff
x = symbols('x')


x**2
diff(x**2)
```

Data Visualization

```
%matplotlib inline

import matplotlib
matplotlib.style.use('ggplot')

import pandas as pd
import numpy as np

ts = pd.Series(np.random.randn(1000), index=pd.date_range('1/
ts = ts.cumsum()
ts.plot()
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x109ba8128>
```


```
import pandas as pd
import numpy as np

ts = pd.Series(
 np.random.randn(1000),
 index=pd.date_range(
  '1/1/2000', periods=1000
 )
)
ts = ts.cumsum()

ts.plot()
```

Mosky

- Python Charmer at Pinkoi.
- Has spoken at
  - PyCons in
    TW, MY, KR, JP, SG, HK,
    COSCUPs, and TEDx, etc.
- Countless hours
  on teaching Python.
- Own Python packages:
- ZIPCodeTW,
  MoSQL, Clime, etc.
- http://mosky.tw/

Outline

- Environment
  - Introduction & Setup
  - “Hello, Python!”
- General Programming Topics
  - Data Type and Control Flow
  - Function
  - Built-in Functions
  - IO and File
  - Methods of String
- The Challenges: Find the big file!

Outline (cont.)

- Fascinating Programming Topics in Python
  - Comprehensions
  - Functional Tools
  - Module and Package
  - Useful Packages
  - Object-oriented Programming in Python
- A Mini Project: Crawl the Internet!

Our Toolbox

We Will Use

(a terminal)	Master the machine
Python 3	not Python 2
Jupyter Notebook	Learn Python with browser
Visual Studio Code	A full-featured source code editor
(other libs)	will be introduced in this slides

Open a Terminal

- Windows:
  - Start Menu / Search / Type “cmd” or “Command Prompt”
  - It has super limited features. If you wanna get a better one:
    - ConEmu
    - Babun
    - Cygwin
- Mac:
  - Spotlight (Cmd-Space) / “terminal”
  - It also has only limited features. Get a better one:
    - iTerm2

Common MS-DOS Commands

`C:`	Go to *C:* drive.
`cd PATH`	Change Directory to *PATH*. *PATH* can be */Users/python/*, *python/*, etc.
`dir`	Display the contents of a directory.
`cls`	Clear the terminal screen.
`python PATH`	Execute a Python program.
`exit`	Exit the current command processor.

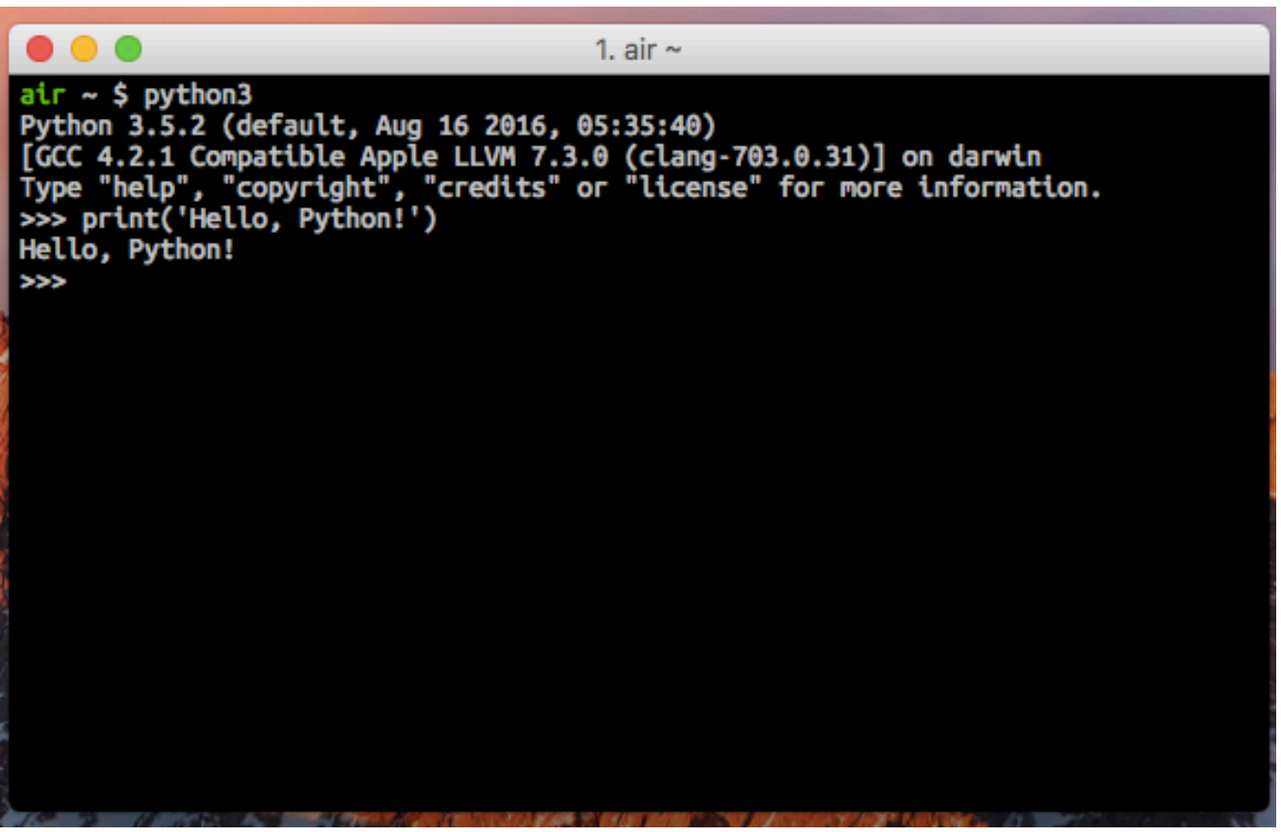
Common Unix-like (Mac) Commands

`cd PATH`	Change Directory to *PATH*. *PATH* can be */Users/python/*, *python/*, etc.
`ls`	List the contents of a directory.
`Ctrl-L`	Clear the terminal screen.
`python PATH` `python3 PATH`	Execute a Python program.
`Ctrl-D`	Exit the current terminal.

Install Python 3 with Miniconda

- Install Python 3 with Miniconda
  - http://conda.pydata.org/miniconda.html
- On Mac and Linux,
  - “bash installer”
    - `chmod +x Miniconda3-latest-MacOSX-x86_64.sh`
    - `./Miniconda3-latest-MacOSX-x86_64.sh`

- If you are a pro,
  - Mac:
    - `$ brew install python3`
  - Ubuntu / Debian (Linux):
    - `$ sudo apt-get install python3`
  - and get pip:
    - https://pip.pypa.io/en/stable/installing/#installing-with-get-pip-py



OS Shell, and Python Shell

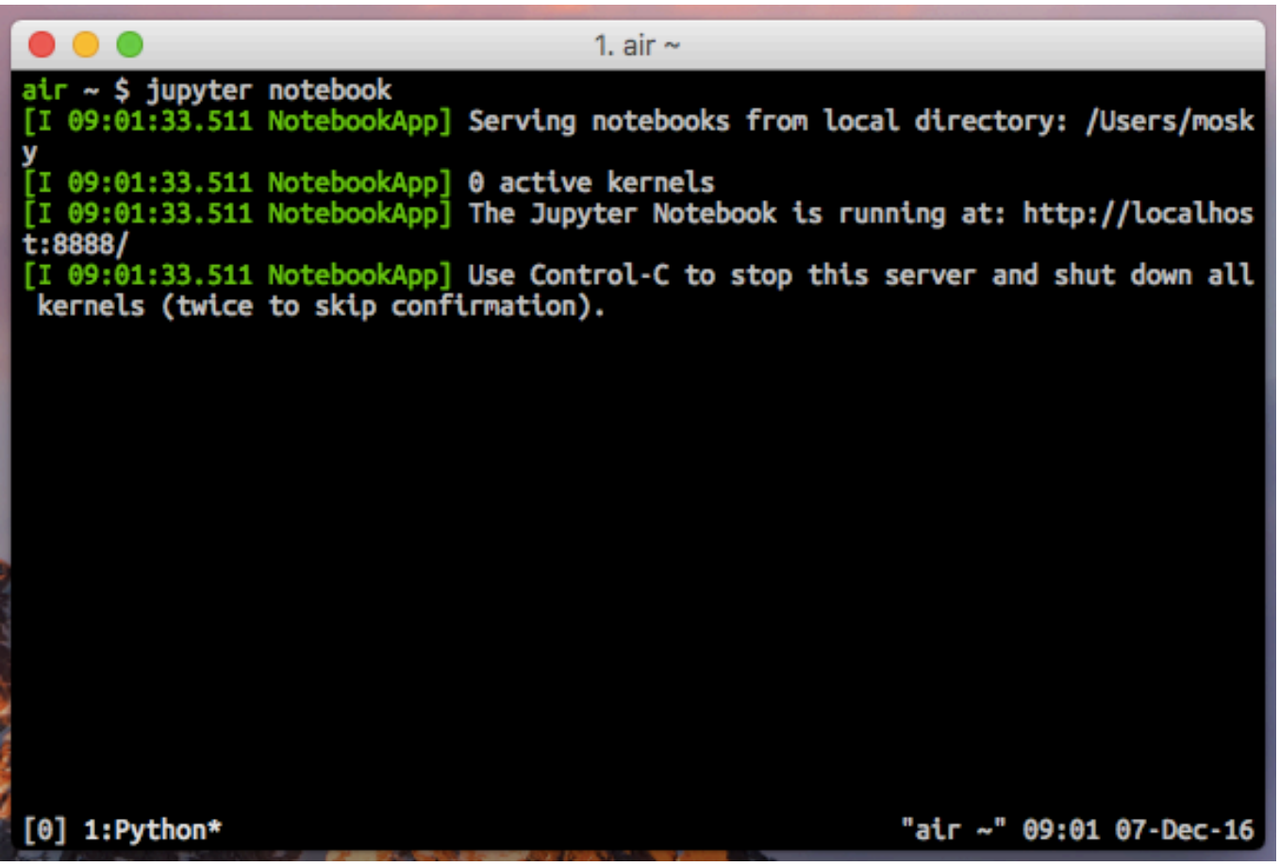
- Give command to ___ in a shell.
- On Mac and Linux,
  default OS shell is Bash
  - $
- On Windows, default OS shell is
  Command Prompt (cmd)
  - >
- Usually we use $ to indicate
  command on an OS shell.
- Python's
  - python or python3
    - >>>
    - exit()

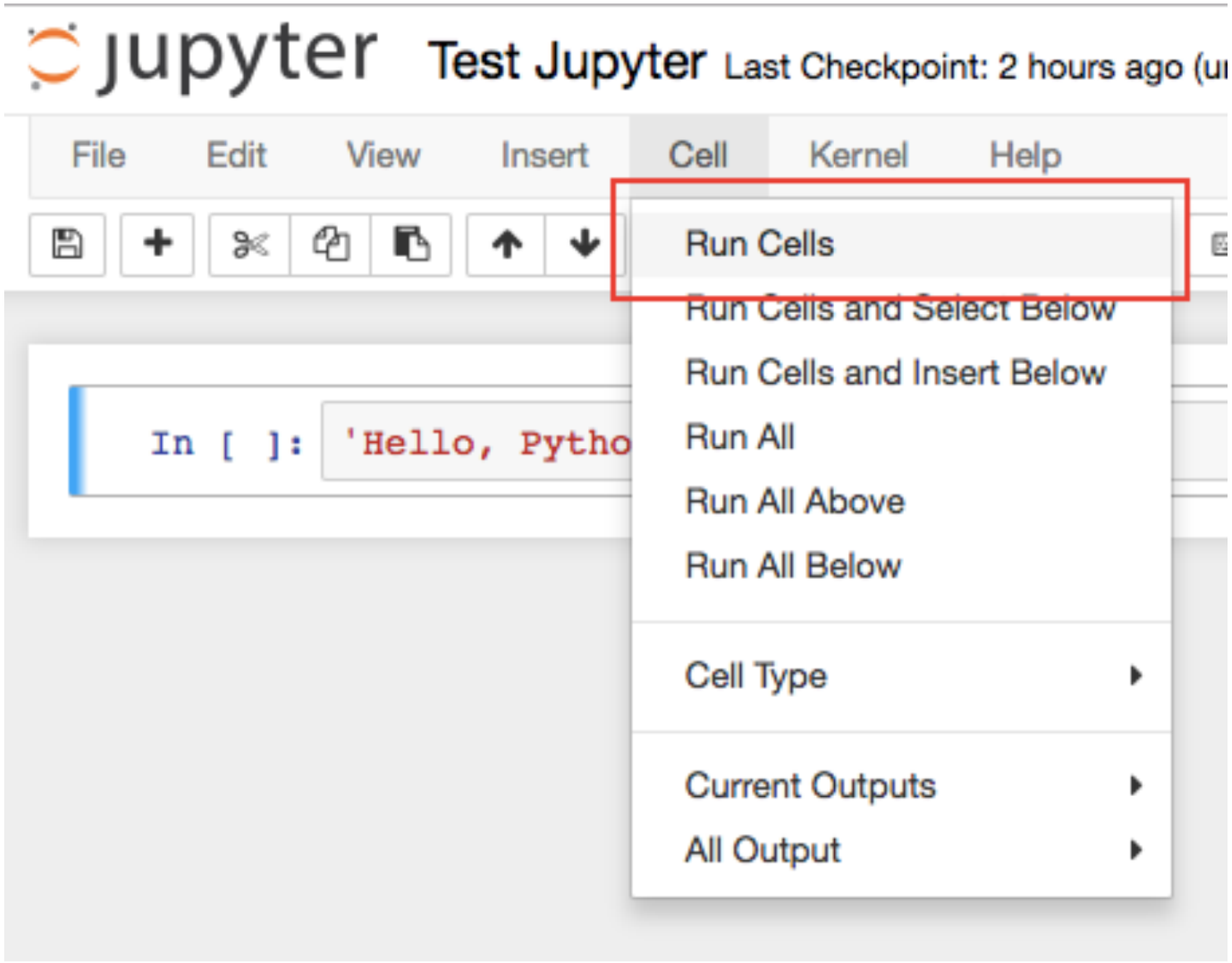
Install Jupyter Notebook and Other Packages

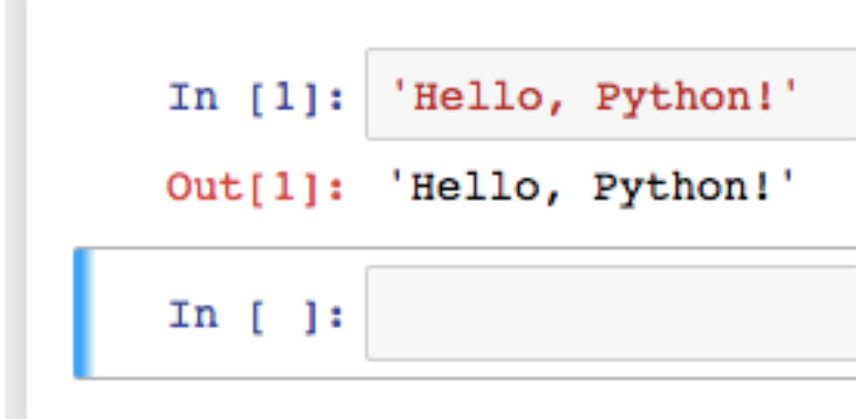
- `$ conda install numpy scipy matplotlib ipython pandas sympy jupyter requests beautifulsoup4 flask`
- Includes
  - SciPy stack
  - Jupyter Notebook
  - Libs for a minimal web crawler: `requests` & `beautifulsoup4`
  - A lightweight web framework: `flask`
- If you are a pro,
  - Use `pip` instead.
  - On Mac, *do not* add `sudo` .

Start Jupyter Notebook

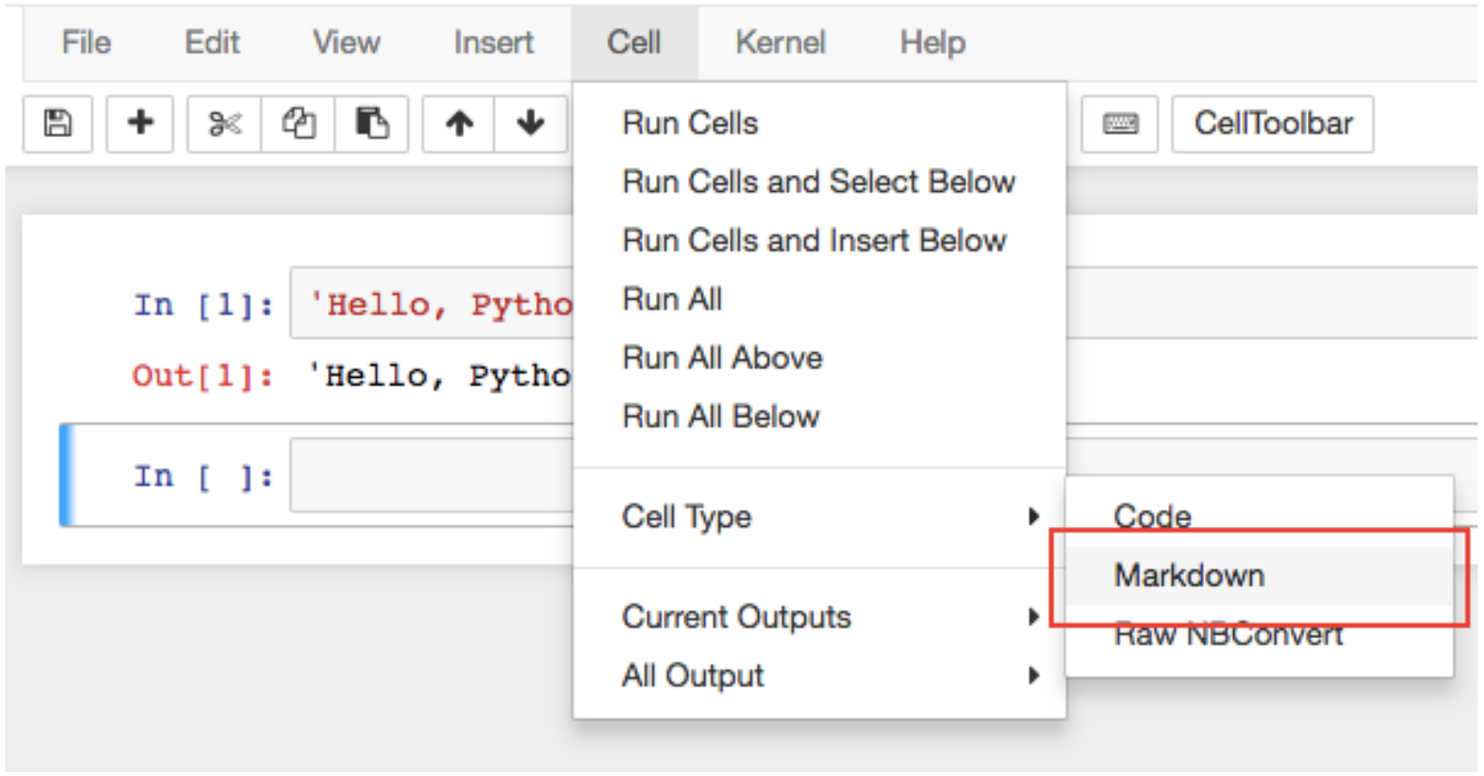
- `$ jupyter notebook`
- It will open up a browser.
- Keep the terminal open.







Jupyter Notebook – Run Cell, "Ctrl-Enter", and "B"



'Hello, Python!'

'Hello, Python!'

Common Shortcuts

1. `Esc` to escape from cell (from edit mode to com
2. `Ctrl-Enter` to run.
3. `B` to insert cell below.
4. `D, D` to delete the current cell.
5. `Shift-Cmd-P` to open command palette.
6. `H` to open keyboard shortcuts.

'Hello, Python!'

'Hello, Python!'

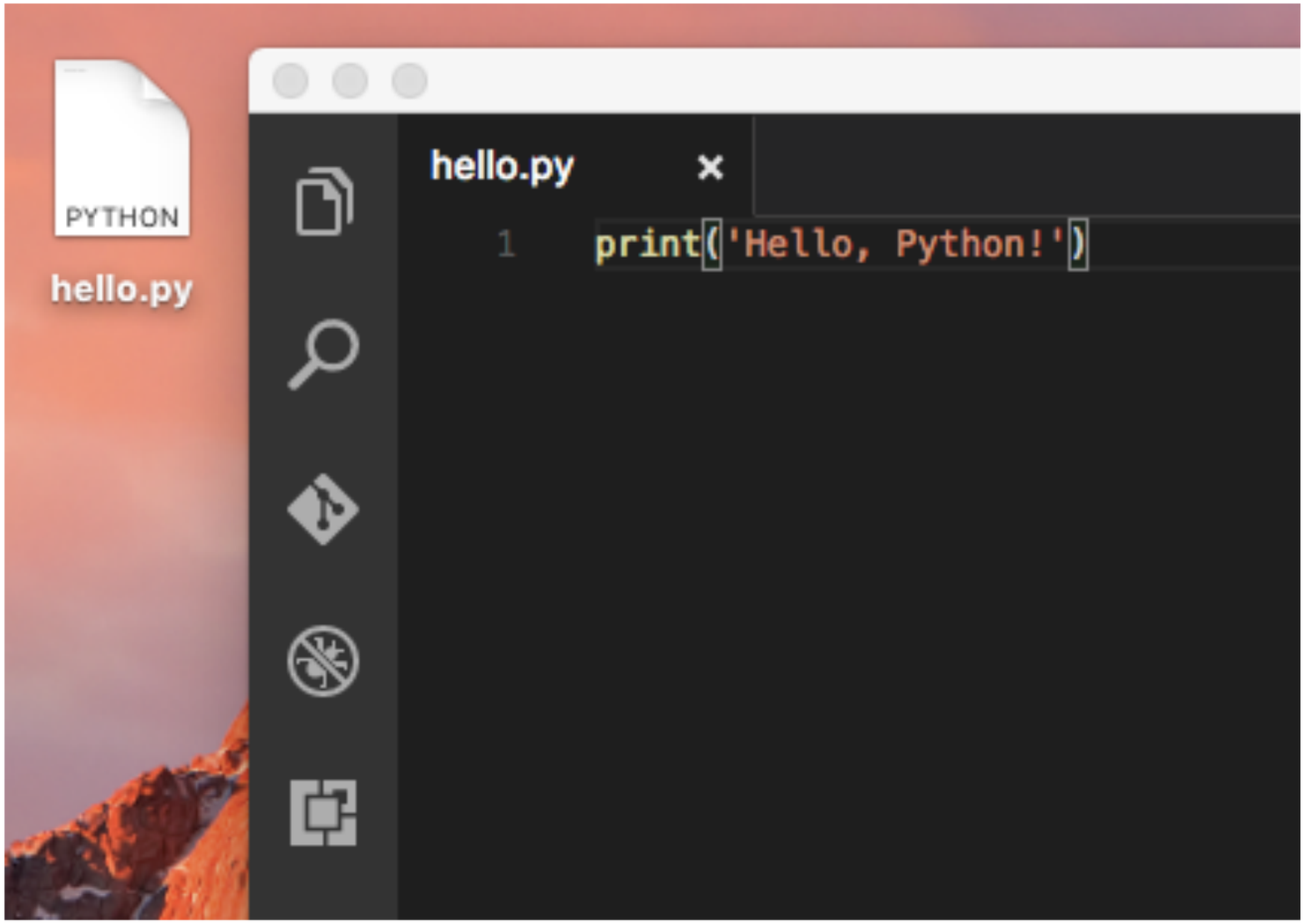
Common Shortcuts

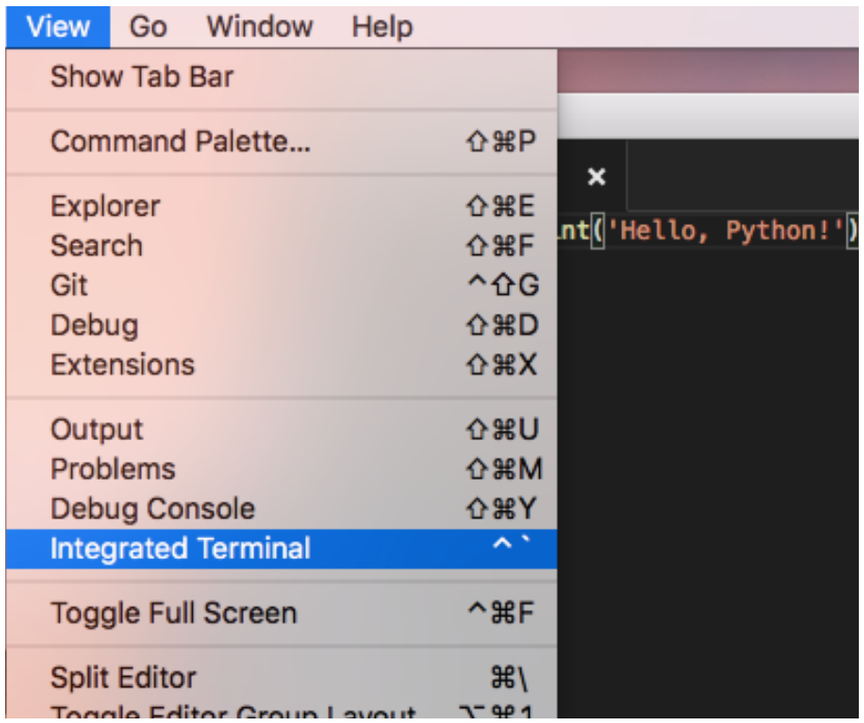
1. `Esc` to escape from cell (from edit mode to comm
2. `Ctrl-Enter` to run.
3. `B` to insert cell below.
4. `D, D` to delete the current cell.
5. `Shift-Cmd-P` to open command palette.
6. `H` to open keyboard shortcuts.

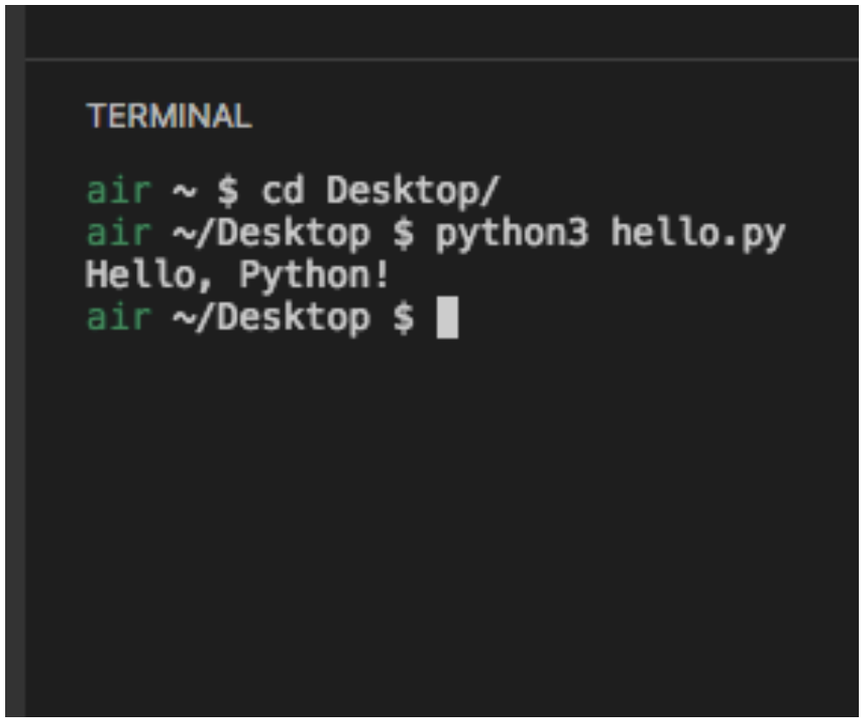
Jupyter Notebook – Markdown and Shortcuts

Install Visual Studio Code

- https://code.visualstudio.com/download
- Just download and install it!
- Execute a program:
  - `$ cd DIR_PATH_OF_PYTHON_FIELS`
  - `$ python hello.py`
  - or
  - `$ python3 hello.py`







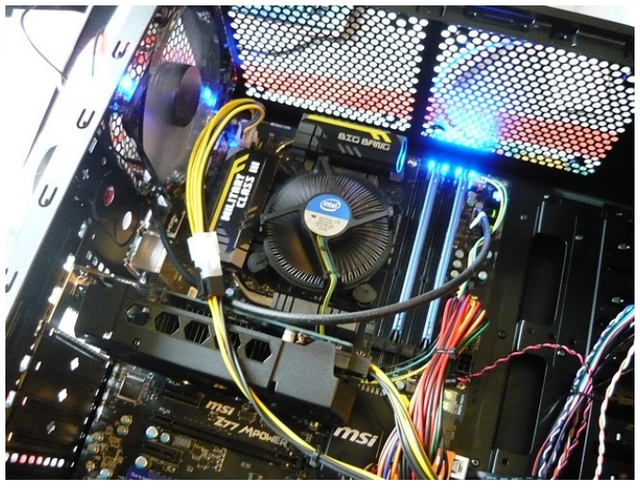
Visual Studio Code – Edit and Run in Integrated Terminal

When to Use What

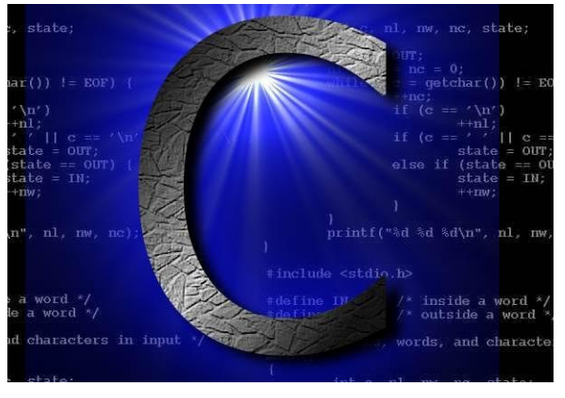
- Jupyter Notebook
  - Interactive programming
  - Shorter (≤ 10 lines per cell)
  - Learning
  - Exploring data
- Visual Studio Code (i.e. an editor with a terminal)
  - “Real” programming
  - Longer
  - Automation scripts
  - Sever-side code

Hello, Python!


User
Software
Hardware




Google
MacBook Pro

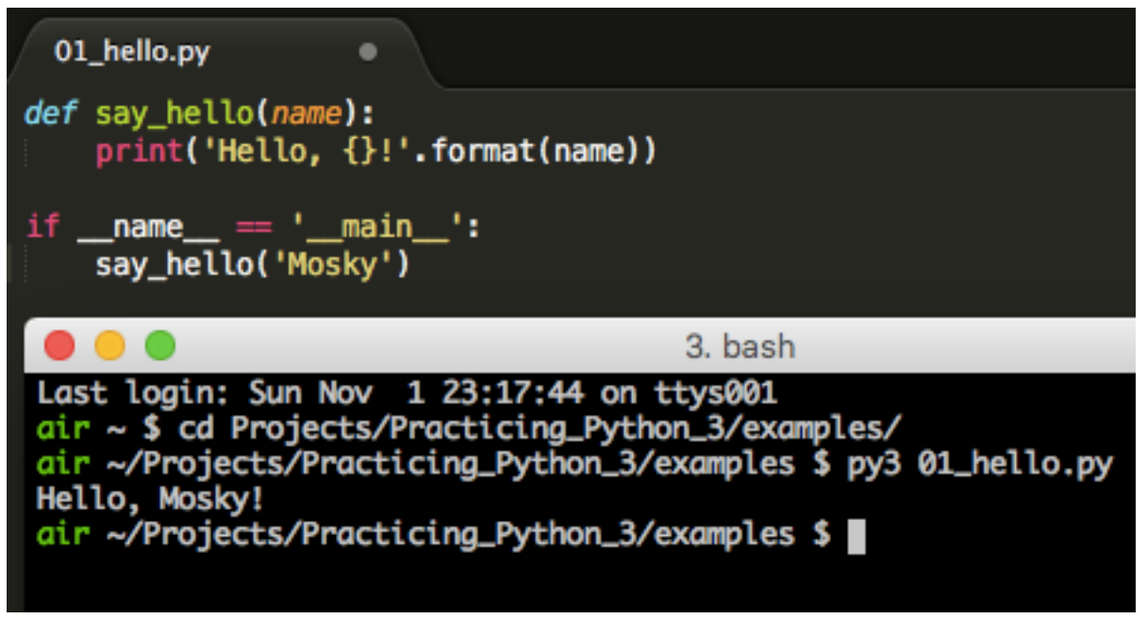


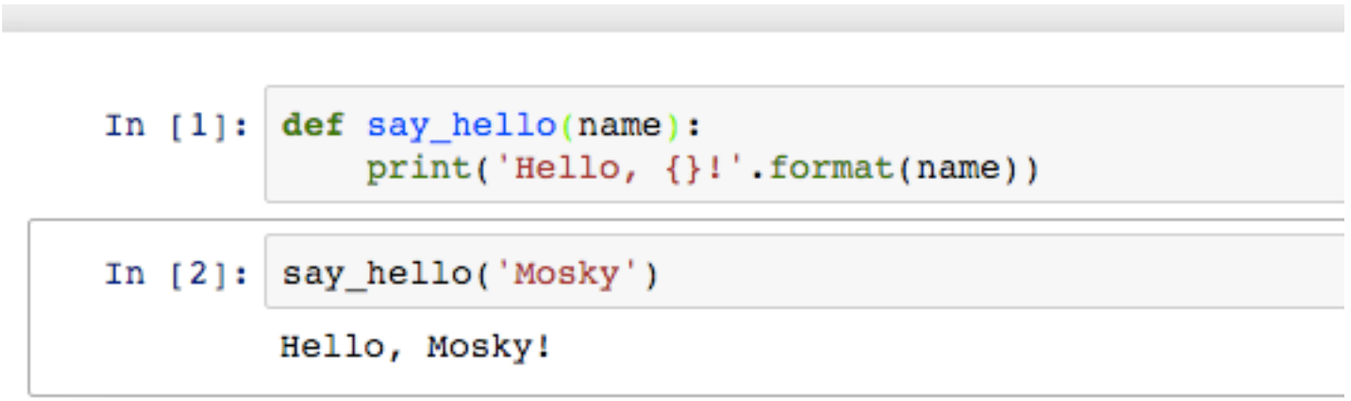
v1 001_hello.py

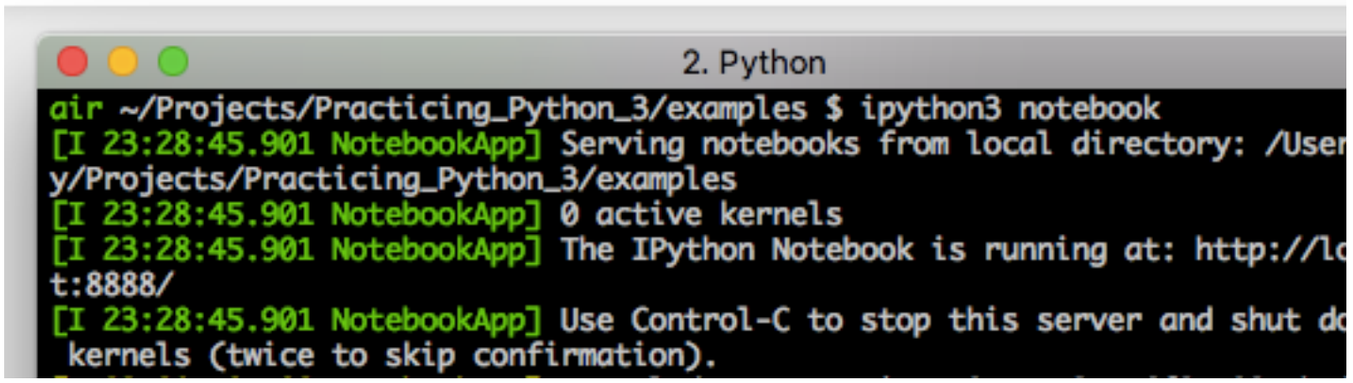
```
print('Hello, Python!')
```

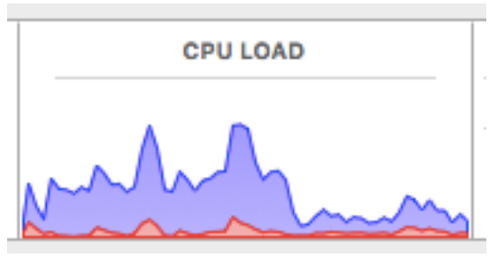
Try It

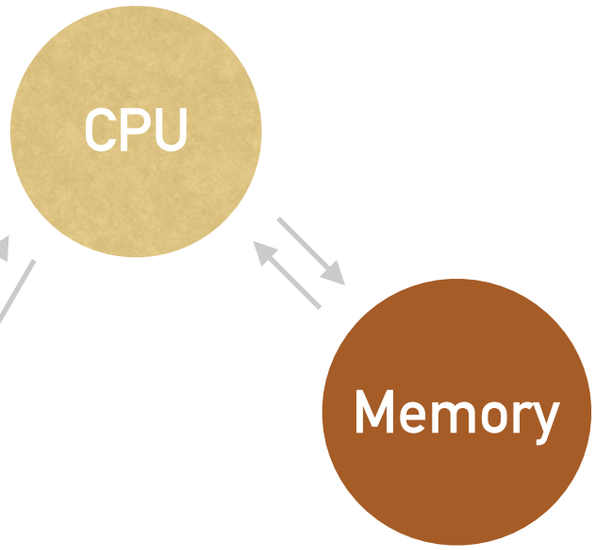
- Type them on
  - Jupyter Notebook, ignore
    - `if __name__ == '__main__':`
    - `def ...`
  - Visual Studio Code
- and execute it.



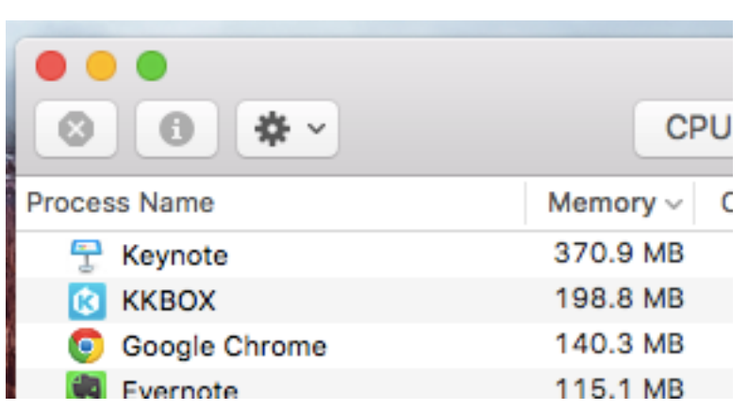





CPU LOAD


CPU
Memory


Storage
PYTHON
01_hello.py


CPU
Process Name
Memory
Keynote
370.9 MB
KKBOX
198.8 MB
Google Chrome
140.3 MB
Evernote
115.1 MB

v1 002_formal_hello.py

```
def say_hello(name):
    print('Hello, {}!'.format(name))

if __name__ == '__main__':
    say_hello('Mosky')
```

Python Tutor

- http://www.pythontutor.com/visualize.html

Syntax

- Each statement ends with a newline.
- A compound statement ends with a colon (:).
- Indentation (   ) identifies a suite, not curly brace ({}).
- An indentation is 4-space, not tab.
- Spaces and blank lines are ignored usually.

v1 003_formal_hello_with_comments.py

```
# the part after `#` is "comment".
# here we define a `say_hello` "function" which accepts an "argument" `name`.
def say_hello(name):

    # the indents distinguish the lines in or out of the function.
    # in this function, we call a bulit-in function `print` with an argument.

    # `'Hello, {}!'` is a string.
    # `.format` is its "method", method is a function belonging an "object".
    # we give `name` as an argument of the format.

    print('Hello, {}!'.format(name))

# `if` is used for conditional execution.
# `__name__` is a bulit-in "variable", it will be the name of a "module".
# if it is executed directly, it will be `'__main__'`.
if __name__ == '__main__':

    # call the function we just defined
    say_hello('Mosky')
```

Checkpoint: Say Hi to Python

```
print('Hello, Python!')
```

Programming?

Abstraction

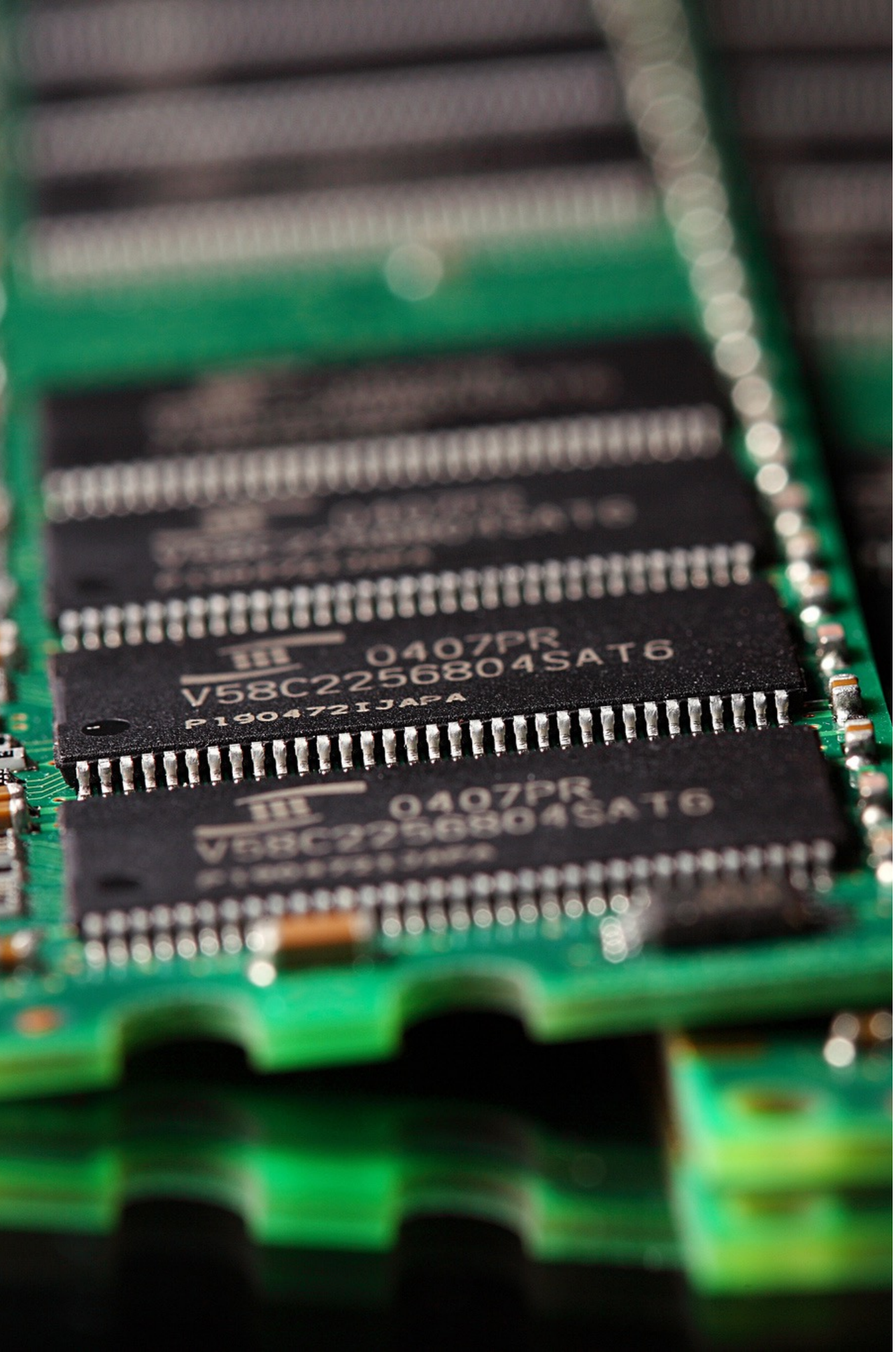
- Programming
  is abstracting.
- Abstract the world with:
  - Data types: *1*.
    - Operations: *1+1*.
  - Control flow: *if … else*.
- They are from:
  - Libraries.
  - Your own code.

Primitives

Data Types

Primitives

int	Integer: 3
float	Floating-point numbers: `3.14`
str	Text Sequence: `'3.14'`
bytes	Binary Sequence: `b'\xe4\xb8\xad\xe6\x96\x87'`
bool	`True` or `False`
None	Represents the absence of a value.



Variables

- Points to an “object”.
- Everything in Python is an object, including class.
- The object:
  - Has a (data) type.
  - Supports an operation set.
  - Lives in the memory.

“

01_data_types_primitives.ipynb

-The Notebook Time

Del

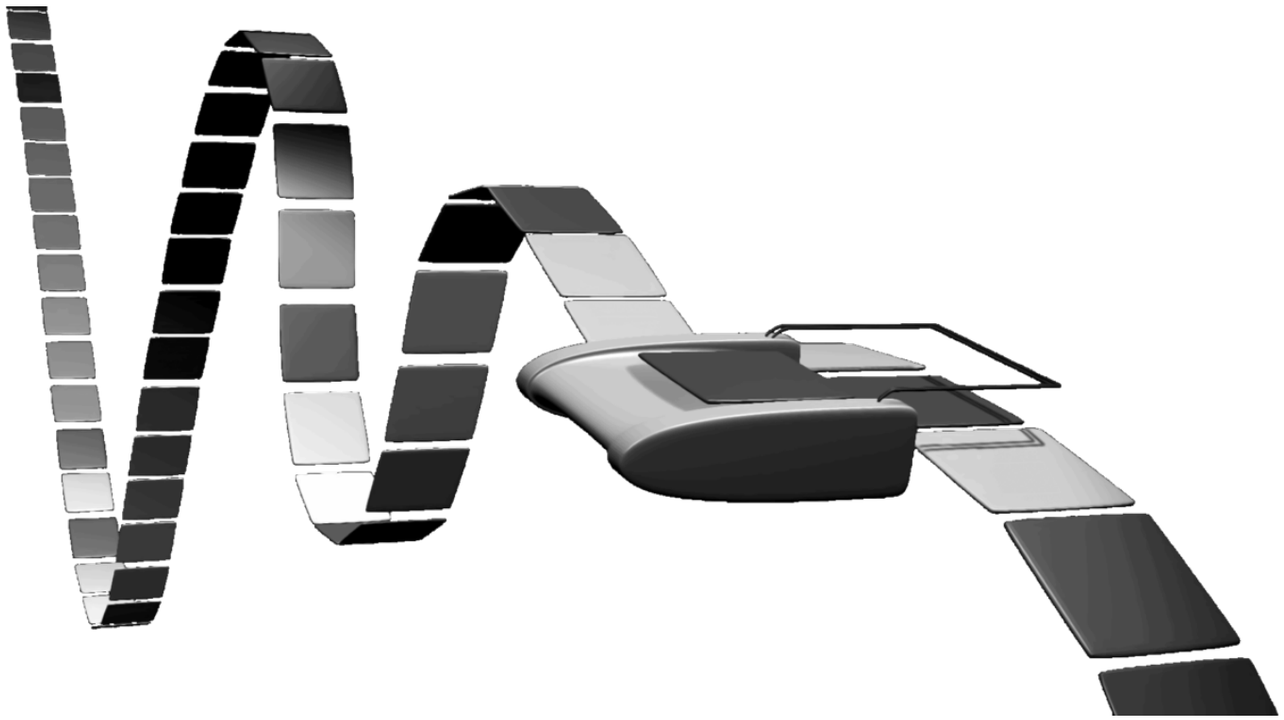
- `del x`
  - Remove the pointing.

Checkpoint: Calculate BMR

- Search “The Mifflin St Jeor Equation” in Basal metabolic rate – Wikipedia.
- Are you healthy enough?

If & While

Control Flow



If & While

- if
  - if … else
  - if … elif … else
- while
  - The while-loop.
  - Do while true.

“

02_control_flow_if_while.ipynb

-The Notebook Time

Collections

Data Types

Collections

list	Contains objects.
tuple	A compound objects has an unique meaning, e.g., a point: (3，1).
dict	Maps an object to another object.
set	Contains non-repeated objects.

“

03_data_types_collections.ipynb

-The Notebook Time

For

Control Flow

For & Related Statements

- `for`
  - The for-loop.
  - In each iteration, get the next item of a collection.
  - Supports str, list, tuple, set, and dict, etc.
  - I.e. iterate an iterable.
- `break`
  - Leave the loop.
- `continue`
  - Go the next iteration.
- `loop … else`
  - If no break happens, execute the *else*.

Pass

- `pass`
  - Do nothing.

“

04_control_flow_for.ipynb

-The Notebook Time

Checkpoint: Calculate Average BMR by Gender

height_cm	weight_kg	age	male
152	48	63	1
157	53	41	1
140	37	63	0
137	32	65	0

Try

Control Flow

Try & Related Statements

- An exception stops the whole execution.
- Catch the exceptions and handle them.
- Sometimes stop is better than a bad output.

“

05_control_flow_try.ipynb

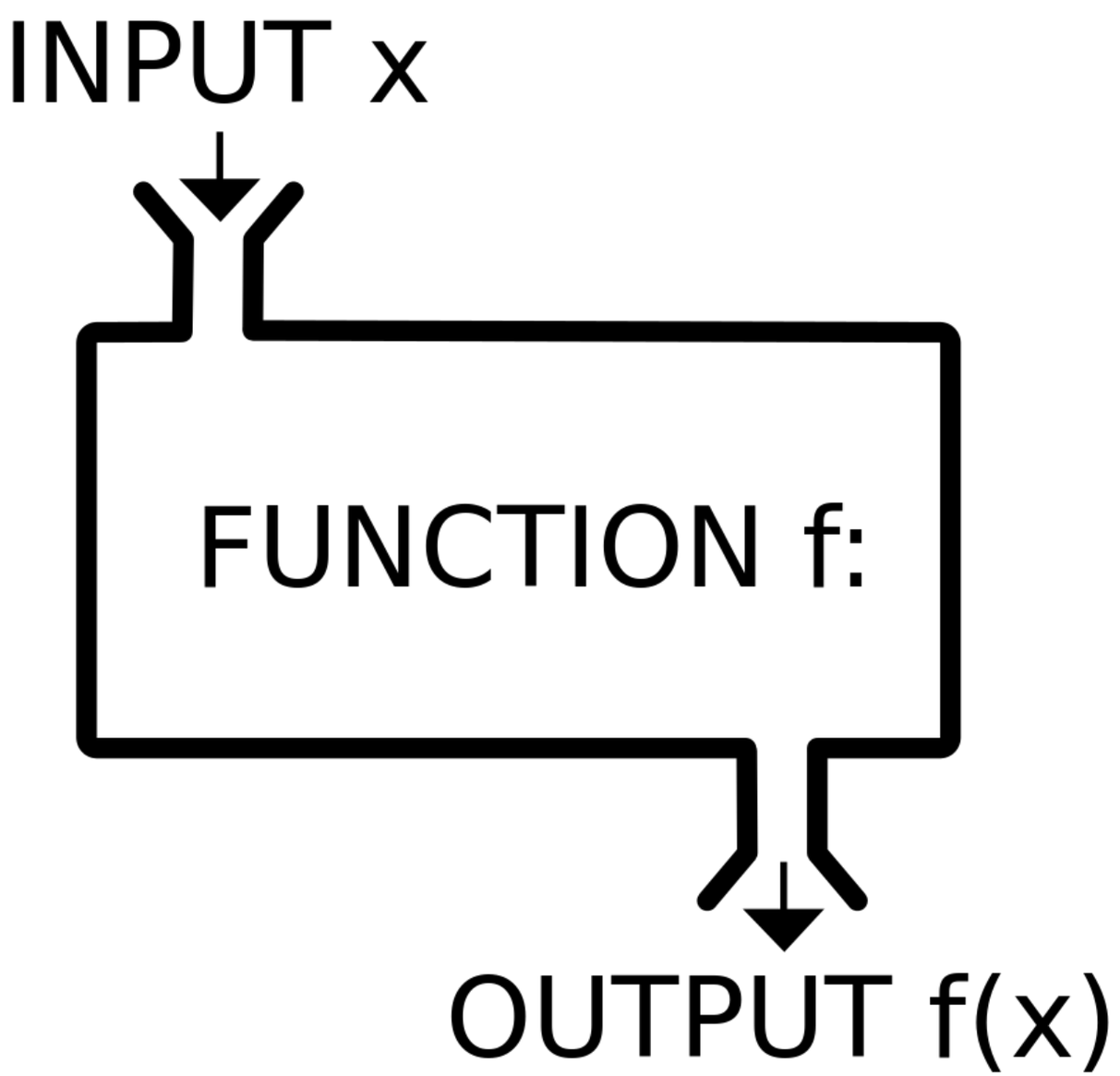
-The Notebook Time

Raise

- `raise`
  - `raise RuntimeError('should not be here')`
    - Raise an customized exception.
  - `raise`
    - Re-raise the last exception.

Function

Control Flow



Function

- Reuse statements.
- Extend built-in operations.
- `def`
  - Take the inputs.
- `return`
  - Give an output.
  - If no return, returns None.
- Method
  - Is a function belonging to an object.

“

06_control_flow_function.ipynb

-The Notebook Time

Recap

- When calling function:
  - Keyword arguments: *f(a=3.14)*
  - Unpacking argument list: *f(*list_like, **dict_like)*
- When defining function:
  - Default values: *def f(a=None): ...*
  - Arbitrary argument list: *def f(*args, **kwargs): ...*
- Using duck typing to max the flexibility.
- Using docstrings to cooperate.
- Small functions – *lambda*.

Common Functions

Libraries

“

07_libraries_
common_functions.ipynb

-The Notebook Time

Input & Output

Libraries

Input & Output

- IO: input & output.
- Standard IO:
  - Output (stdout).
  - Error (stderr).
  - Input (stdin).
- File IO:
  - Including networking.
- Command line arguments
- Always validate user's inputs.

“

08_libraries_input_output.ipynb

-The Notebook Time

Checkpoint: Calculate Average BMR by Gender From Dataset

- The dataset is that CSV.

Command Line Arguments

Libraries

“

09_libraries_
command_line_arguments.py

-The Notebook Time

Recap

- *open(…)* → file object for read or write.
- The *with* will close file after the suite.
- Input
  - stdin → *input()*
  - *for line in <file object>: …*
  - *<file object>.read()*
- Output
  - *print(…)* → stdout
  - *print(…, file=sys.stderr)* → stderr
  - *<file object>.write(…)*

Checkpoint: Write a CLI Program to Calculate BMR Interactively

```
➤ $ python3 calc_bmr.py
➤ Enter height: 152
➤ Enter width: 48
➤ Enter age: 63
➤ Enter gender: M
➤ 1120
```

The Challenges

Find the Largest File

- Input:
  - A path from command line.
- Output:
  - The name of the largest file in the directory.
  - Don't show the traceback to user.
- Snippets:
  - `from os import listdir`
  - `from os.path import join, isfile, getsize`

Find the Largest Directory

- Input and output are same as previous.
- Note the output shall be one of the first level directories.
- Snippets:
  - `from os import walk`
  - `from os.path import isdir, islink`
- Hints:
  - Which directory is the first level directory of a file?