ch01-intro

May 4, 2019

1 Chapter 1: Computing with Python

1.0.1 Overview: a typical Python-based scientific computing stack.

1.1 Interpreter

- The easist way to execute Python code: run the program directly.
- Use Jupyter magic command to write Python source file to disk:

```
In [1]: %%writefile hello.py
        print("Hello from Python!")
```

```
Overwriting hello.py
```

- Use the ! system shell command (included in the Python Jupyter kernel) to interactively run Python with hello.py as its argument.

```
In [1]: !python hello.py
```

```
Hello from Python!
```

```
In [2]: !python --version
```

```
Python 3.6.5 :: Anaconda, Inc.
```

1.2 Input and output caching

- Input & output history can be accessed using **In** (a list) & **Out** (a dictionary). Both can be indexed with a cell number.

```
In [3]: 3 * 3
```

```
Out[3]: 9
```

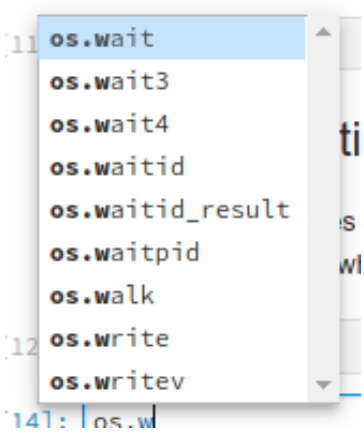
```
In [4]: In[1]
```

```
Out[4]: "get_ipython().system('python hello.py')"
```

software stack

- A single underscore = the most recent output;
- A double underscore = the *next* most recent output.

```
In [5]: 1+1

Out[5]: 2

In [6]: 2+2

Out[6]: 4

In [7]: _, __

Out[7]: (4, 2)

In [8]: # In = a list
        In

Out[8]: ['',
         "get_ipython().system('python hello.py')",
         "get_ipython().system('python --version')",
         '3 * 3',
         'In[1]',
         '1+1',
         '2+2',
         '_, __',
         'In']
```



autocompletion

```
In [9]: # Out = a dictionary
        Out

Out[9]: {3: 9,
         4: "get_ipython().system('python hello.py')",
         5: 2,
         6: 4,
         7: (4, 2),
         8: ['',
          "get_ipython().system('python hello.py')",
          "get_ipython().system('python --version')",
          '3 * 3',
          'In[1]',
          '1+1',
          '2+2',
          '_, __',
          'In',
          'Out']}

In [10]: # Suppress output results by ending statement with a semicolon
         1+2;
```

1.3 Autocompletion

- The **Tab** key activates autocompletion (displays list of symbol names that are valid completions of what has been typed thus far.)

```
In [11]: import os
```

- Results of typing "os.w", followed by :