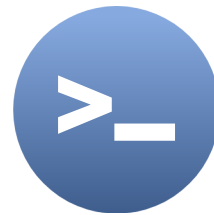


Python on the command line

DATA PROCESSING IN SHELL



Susan Sun
Data Person

Python basics

Python

- comes pre-installed on MacOS, Linux
- needs to be user-install for Windows [instructions here](#)
- can be used with GUI interfaces (e.g Jupyter Notebook, Spyder, PyCharm, etc.)
- can also be accessed directly via the command line interface

Using Python documentation

Documentation:

```
man python
```

```
...  
-V , --version  
    Prints the Python version number of the executable and exits.
```

```
python --version
```

```
Python 3.5.2
```

Using Python documentation

Example 1: using native Python

```
which python
```

```
/usr/bin/python
```

Example 2: using Anaconda Python

```
which python
```

```
/anaconda3/bin/python
```

The Python interactive session

To activate a Python interactive session in the terminal:

```
python
```

```
Python 3.5.2 (default, Nov 23 2017, 16:37:01)
[GCC 5.4.0 20160609] on linuxType "help", "copyright", "credits" or
"license" for more information.
>>>
```

The Python interactive session

Inside the interactive session, only use Python syntax:

```
>>> print('hello world')  
hello world
```

To exit the Python session and return to terminal:

```
>>> exit()  
$
```

Python interactive session alternative

Python interactive session:

- easy to activate, intuitive
- not good for code reproducibility

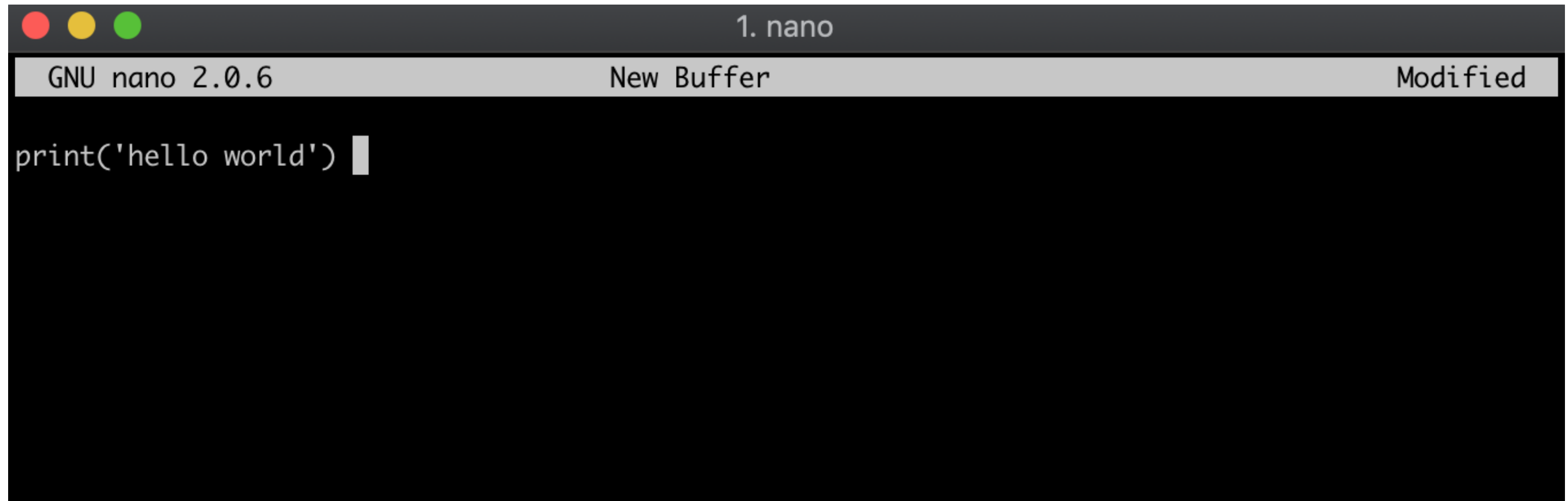
Alternative:

- save Python commands in a Python `.py` script
- execute script by calling `python` + script

Python script execution on the command line

Method 1

- Create a `.py` file using a text editor on the command line (e.g. nano, Vim, Emacs)

A screenshot of a terminal window with a dark background. At the top, there's a title bar with three colored circles (red, yellow, green) on the left and the text "1. nano" on the right. Below the title bar is a light gray status bar with "GNU nano 2.0.6" on the left, "New Buffer" in the center, and "Modified" on the right. The main area of the terminal is black and contains the text `print('hello world')` followed by a white cursor (a vertical bar) at the end of the line.

```
1. nano
GNU nano 2.0.6      New Buffer      Modified
print('hello world') |
```


Python script execution on the command line

Method 2

- Create a `.py` file by `echo`-ing the Python syntax into the `hello_world.py` file, instantiating the Python file in the same step.

```
echo "print('hello world')" > hello_world.py
```

Sanity check file content:

```
cat hello_world.py
```

```
print('hello world')
```

Python script execution on the command line

Make sure in the same directory as the `.py` file:

```
ls
```

```
hello_world.py
```

Execute `.py` file by preceding filename with `python` :

```
python hello_world.py
```

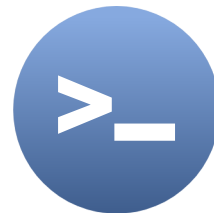
```
hello world
```

Let's practice!

DATA PROCESSING IN SHELL

Python package installation with pip

DATA PROCESSING IN SHELL



Susan Sun
Data Person

Python standard library

Python standard library has a collection of:

- built-in functions (e.g. `print()`)
- built-in packages (e.g. `math`, `os`)

Data science packages like **scikit-learn** and **statsmodel**:

- are **NOT** part of the Python standard library
- can be installed through `pip`, the standard package manager for Python, **via the command line**

Using pip documentation

Documentation:

```
pip -h
```

Usage:

```
pip <command> [options]
```

Commands:

install	Install packages.
uninstall	Uninstall packages.
freeze	Output installed packages in requirements format.
list	List installed packages.

Using pip documentation

Documentation:

```
pip --version
```

```
pip 19.1.1 from /usr/local/lib/python3.5/dist-packages/pip (python 3.5)
```

```
python --version
```

```
Python 3.5.2
```

Upgrading pip

If `pip` is giving an upgrade warning:

```
WARNING: You are using pip version 19.1.1, however version 19.2.1 is available.  
You should consider upgrading via the 'pip install --upgrade pip' command.
```

Upgrade `pip` using itself:

```
pip install --upgrade pip
```

```
Collecting pip  
  |#####| 1.4MB 10.7MB/s  
Successfully installed pip-19.2.1
```


pip list

`pip list` : displays the Python packages in your current Python environment

```
pip list
```

```
Package            Version
-----
agate              1.6.1
agate-dbf          0.2.1
agate-excel        0.2.3
agate-sql          0.5.4
Babel              2.7.0
```

pip install one package

`pip install` installs the package specified and any other dependencies

```
pip install scikit-learn
```

```
Collecting scikit-learn
```

```
  Downloading https://files.pythonhosted.org/packages/1f/af/e3c3cd6f61093830059138624db0/
```

```
    |#####| 6.6MB 32.5MB/s
```

```
Collecting scipy>=0.17.0 (from scikit-learn)
```

```
  Downloading https://files.pythonhosted.org/packages/14/49/8f13fa215e10a7ab0731cc95b0e9/
```

```
    |#####| 25.1MB 35.5MB/s
```

```
...
```

pip install a specific version

By default, `pip install` will always install the latest version of the library.

```
pip install scikit-learn
```

```
Successfully built sklearn
```

```
Installing collected packages: joblib, scipy, scikit-learn, sklearn
```

```
Successfully installed joblib-0.13.2 scikit-learn-0.21.3 scipy-1.3.0 sklearn-0.0
```

pip install a specific version

To install a specific (or older) version of the library:

```
pip install scikit-learn==0.19.2
```

```
Collecting scikit-learn==0.19.2
```

```
  Downloading https://files.pythonhosted.org/packages/b6/e2/a1e254a4a4598588d4fe88b45ab8
```

```
    |#####| 4.9MB 15.6MB/s
```

```
Installing collected packages: scikit-learn
```

```
Successfully installed scikit-learn-0.19.2
```

Upgrading packages using pip

Upgrade the Scikit-Learn package using pip:

```
pip install --upgrade scikit-learn
```

```
Collecting scikit-learn
```

```
  Downloading https://files.pythonhosted.org/packages/1f/af/e3c3cd6f61093830059138624db0/
```

```
    |#####| 6.6MB 41.5MB/s
```

```
Requirement already satisfied, skipping upgrade: numpy>=1.11.0 in /usr/local/lib/python3
```

```
Collecting scipy>=0.17.0 (from scikit-learn)
```

```
Installing collected packages: scipy, joblib, scikit-learn
```

```
Successfully installed joblib-0.13.2 scikit-learn-0.21.3 scipy-1.3.0
```

pip install multiple packages

To `pip install` multiple packages, separate the packages with spaces:

```
pip install scikit-learn statsmodels
```

Upgrade multiple packages:

```
pip install --upgrade scikit-learn statsmodels
```

pip install with requirements.txt

`requirements.txt` file contains a list of packages to be installed:

```
cat requirements.txt
```

```
scikit-learn  
statsmodel
```

Most Python developers include `requirements.txt` files in their Python Github repos.

pip install with requirements.txt

`-r` allows `pip install` to install packages from a pre-written file:

```
-r, --requirement <file>
```

```
Install from the given requirements file. This option can be used multiple times.
```

In our example:

```
pip install -r requirements.txt
```

is the same as

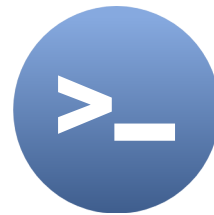
```
pip install scikit-learn statsmodel
```


Let's practice!

DATA PROCESSING IN SHELL

Data job automation with cron

DATA PROCESSING IN SHELL



Susan Sun
Data Person

What is a scheduler?

- Scheduler runs jobs on a pre-determined schedule
- Commercial schedulers: Airflow, Luigi, Rundeck, etc.
- cron scheduler is
 - simple
 - free
 - customizable
 - purely command-line
 - native to MacOS and Linux

What is cron?

Cron:

- is a time-based job-scheduler
- comes pre-installed in MacOS, Unix
- can be installed in Windows via Cygwin or replaced with Windows Task Scheduler
- is used to automate jobs like system maintenance, bash scripts, Python jobs, etc.

What is crontab?

Crontab is a central file to keep track of cron jobs.

```
crontab -l
```

```
no crontab for <username>
```

Documentation:

```
man crontab
```

Add a job to crontab

Method 1: modify crontab using a text editor (e.g. nano, Vim, Emacs)

Method 2: echo the scheduler command into crontab

```
echo "* * * * * python create_model.py" | crontab
```

Check if the job is properly scheduled:

```
crontab -l
```

```
* * * * * python create_model.py
```

Learning to time a cron job

The most frequent schedule for cron jobs is **one minute**.

Breaking down the time component for a cron job:

```
.----- minute (0 - 59)
|  .----- hour (0 - 23)
|  |  .----- day of month (1 - 31)
|  |  |  .----- month (1 - 12) 0R jan,feb,mar,apr ...
|  |  |  |  .---- day of week (0 - 6) (Sunday=0 or 7) 0R sun,mon,tue,wed ...
|  |  |  |  |
*  *  *  *  * command-to-be-executed
```

Learning to time a cron job

```
* * * * * python create_model.py
```

Interpretation:

- Run every minute of every hour of every day of every month and of every day of the week.
- In short, run every minute

Further resources:

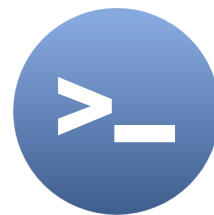
- Use <https://crontab.guru/> to see more ways to schedule a cron job

Let's practice!

DATA PROCESSING IN SHELL

Course recap

DATA PROCESSING IN SHELL



Susan Sun
Data Person

Data downloading on the command line

- How to download data files via `curl` and `wget`
- Documentations, manuals (e.g. `man curl` , `wget --help`)
- Multiple file downloads (e.g. `wget --limit-rate=200k -i url_list.txt`)

Data processing on the command line

- Introduction to command line data toolkit: `csvkit`
- Convert files to csv using `in2csv`
- Print preview using `csvlook` , `csvstat`
- Filter data using `csvcut` , `csvgrep`
- Append multiple data files using `csvstack`

Database manipulation on the command line

- Database manipulation using `sql2csv` , `csvsql`
- Advanced SQL-like ETL commands using `csvkit`

Building data pipelines on the command line

- Execute Python on the command line
- Python package management using `pip`
- Automate Python model and build pipelines with `cron`

Thank you! So long!

DATA PROCESSING IN SHELL