

# Tools Seminar

## Week 6 - Scientific Computing

Hongzheng Chen

Mar 2, 2020

- 1 Introduction
- 2 Package Management
- 3 SciPy
- 4 Math Softwares
- 5 Summary

1

# Introduction

# Scientific Computation

Mostly involve applied mathematics and computational mathematics

- Quantitative Finance (stock)
  - Physical simulation (fluid  $\rightarrow$  CG)
  - Computational biology (gene)
  - Molecular dynamics (protein)
  - Ocean circulation
  - Weather/Climate prediction
  - Epidemics (SARS-CoV-2)
  - Astronomy (black hole  $\rightarrow$  digital image processing)
- \* Supercomputing enables much more complex applications to be done

# Scientific Computation

## Traditional CS

discrete  
integer



## Sci Comp

continuous  
real numbers

# Scientific Computation

## Traditional CS

discrete

integer



## Sci Comp

continuous

**real numbers**

IEEE 754 binary floating point standard

Be careful of the **roundoff error**! → numerical computing

Try `5.2 - 5` in Python

# Scientific Computation

Most of them are **matrix computation!** → **linear algebra**

$$c_{ij} = \sum_k a_{ik} b_{kj}$$

```
for (int i = 0; i < N; i++)
    for (int j = 0; j < N; j++)
        for (int k = 0; k < N; k++)
            C[i][j] += A[i][k] * B[k][j];
```

\* You should know how to store a 2D array in computer memory

$$\begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$$

- Row-major:  $\{\{1,2\},\{3,4\}\}$  (C/C++)
- Column-major:  $\{\{1,3\},\{2,4\}\}$  (Matlab)

# Memory Hierarchy

Recommend to read [CSAPP!](#)

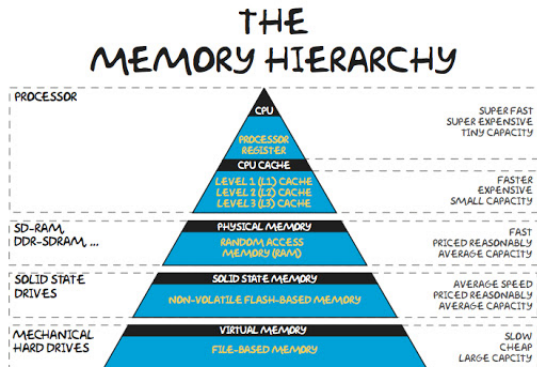


Fig source: <http://computerscience.chemeketa.edu/cs160Reader/ComputerArchitecture/MemoryHierarchy.html>



# Temporal & Spatial Locality

Recommend to read [CSAPP!](#)

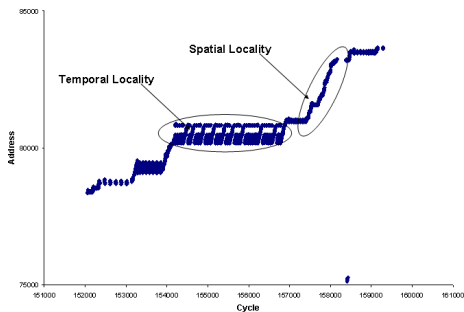


Fig source: <https://stackoverflow.com/a/49325155>

Different data organization affects locality & performance  
Thus, even scientific computing needs knowledge of **computer system & arch!**

2

## Package Management

# Why we need package management

Installing third-party library for C/C++ is very awkward (boost, OpenGL, ...)

- ❶ Download source code with magic version from some unknown webpages
- ❷ Put the code in some system folder that is hard to find
- ❸ Compile the library
- ❹ If gcc/make version not correct, go back to 1
- ❺ If dependency files not found, go back to 1
- ❻ Successfully compiled but the package not found, go back to 2
- ❼ Run the package but get runtime error, go back to 1
- ❽ ...

# Why we need package management

Installing third-party library for C/C++ is very awkward (boost, OpenGL, ...)

- ➊ Download source code with magic version from some unknown webpages
- ➋ Put the code in some system folder that is hard to find
- ➌ Compile the library
- ➍ If gcc/make version not correct, go back to 1
- ➎ If dependency files not found, go back to 1
- ➏ Successfully compiled but the package not found, go back to 2
- ➐ Run the package but get runtime error, go back to 1
- ➑ ...

Thus, we need tools to help us **build, manage, upgrade, remove** different kinds of packages

# Package Management

Fortunately, Python has powerful package management tools!

- Windows: [Anaconda](#) (conda)
  - The complete data science platform
  - If you want to use your own GPU for deep learning in the future, you should install
- Linux: [pip](#) (The Python Package Installer)
  - Be careful of your Python version (2 or 3)
  - `sudo apt install python-pip`
  - `sudo apt install python3-venv python3-pip`
  - `pip3 -V`

# Environment Management

If you need to regularly change Python version, please create a [virtual environment](#)!

```
pip3 install virtualenv
```

- `which python3`
- `virtualenv -p /usr/bin/python3 mpy3`
- `source mpy3/bin/activate`
- `deactivate`

conda has inherent environment management tool:

- `conda create -n your_env_name python=your_py_version`
- `activate env_name`
- `deactivate`

# Environment Management

**pipreqs**: Automatically generate python dependencies

- `pip3 install pipreqs`
- `pipreqs /<your_project_path>/`
- `pip3 install -r requirements.txt`

# Jupyter Notebook

**Jupyter notebook:** A extremely powerful web-based interactive interface

- `pip3 install notebook`
- `jupyter notebook`
  - You should first `cd` to the folder you want to open
- Code, data, figure, notes (Markdown)
- Also valid on Github and VS Code



3

SciPy

# SciPy

**SciPy**: a Python-based ecosystem of open-source software for mathematics, science, and engineering

- NumPy: Base N-dimensional array package
- SciPy library: Fundamental library for scientific computing (FFT, signal, opt, ...)
- Matplotlib: Comprehensive 2-D plotting
- IPython: Enhanced interactive console
- SymPy: Symbolic mathematics
- pandas: Data structures & analysis

\* Anaconda must contain scipy package

For Linux, install by `pip3 install scipy`

# Numpy

**Numpy:** `pip3 install numpy`

- A powerful N-dimensional array object
- Sophisticated (broadcasting) functions
- Tools for integrating C/C++ and Fortran code (core part of numpy is written in C)
- Useful linear algebra, Fourier transform, and random number capabilities
- Indexing, slicing, and iterating functions the same way as in Python

Tutorial:

<https://docs.scipy.org/doc/numpy/user/quickstart.html>

# The Basics

```
import numpy as np
```

- `np.array([[1,2,3],[4,5,6]])`
- `a.shape`, `a.size`, `a.ndim`
  - Be careful of the shape of `np.array([1,2,3])`
  - Differentiate between `(3,)` and `(3,1)`
- `a.dtype`
  - Python is strong-typed
  - `a.astype(np.float64)`
- `a.reshape((3,2))`
  - Input a tuple! Not transpose! Return a new array!
  - Distinguish with `a.T`
  - `a.resize` is inplace

# Array Creation

- `np.zeros((3,4))`
- `np.ones((3,4))`
- `np.arange(1,10,1) [a,b)`
- `np.linspace(0,1,10)`
- `np.random.random((3,4))`

# Stacking

- `np.column_stack((a,b))`, `np.hstack`
- `np.row_stack((a,b))`, `np.vstack`

# Basic Operations

Arithmetic operations are **element-wise** if both are arrays of same size!

- $+$ ,  $-$ ,  $*$ ,  $/$ ,  $**$
- $<$ ,  $>$ ,  $==$

# Broadcasting

**Broadcasting rules:** Examine if two dimensions are compatible

- they are equal, or
- one of them is 1

```
Image (3d array): 256 x 256 x 3
Scale (1d array):          3
Result (3d array): 256 x 256 x 3
```

```
A      (2d array): 5 x 4
B      (1d array):    1
Result (2d array): 5 x 4
```

```
A      (4d array): 8 x 1 x 6 x 1
B      (3d array):    7 x 1 x 5
Result (4d array): 8 x 7 x 6 x 5
```

- $+$ ,  $-$ ,  $*$ ,  $/$ ,  $**$
- $<$ ,  $>$ ,  $==$



# Matrix Product

Notice: `*` for numpy array is the element-wise or **Hadamard product**, denoted as

$$(A \circ B)_{ij} = (A)_{ij}(B)_{ij}$$

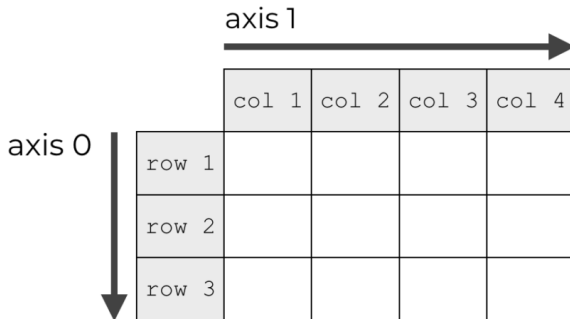
The true matrix product can be called as shown below

```
>>> a @ b                # matrix product
>>> a.dot(b)             # another matrix product
>>> c = np.matrix([[1,2],[3,4]])
>>> c * d                # also matrix product
```

# Clustering Functions

Axis: Extremely important! Ref:

<https://www.sharpsightlabs.com/blog/numpy-axes-explained/>

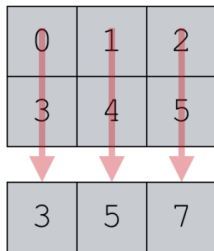


# Clustering Functions

Axis: Extremely important! Ref:

<https://www.sharpsightlabs.com/blog/numpy-axes-explained/>

**WHEN WE SET `axis = 0`, `np.sum()` COLLAPSES THE ROWS AND CALCULATES THE SUM**



`np.sum(a,axis=0)`, `np.mean`, `np.min`, `np.max`, `F.softmax` (pytorch)

# Indexing & Slicing

Basic indexing & Slicing:

- `a[:, a[:6:2], a[::-1]]`
- `b[ : , 1:3], b[ : , 2]`

Fancy indexing: Use array of indices

- `a[b]`
- `a[a>2]`
- Even assignment is allowed, like `a[a>2] = 0`

# Linear Algebra

- `np.eye`
- `a.transpose`
- `np.linalg.inv(a)`
- `np.linalg.trace(a)`
- `np.linalg.solve(a,y)`
- `np.linalg.eig(a)`

## 4

## Math Softwares

# Math Softwares for Scientific Computing

## 3M in Mathematics

- Matlab: Numeric computation, C-like grammar, efficient for engineering
- Mathematica: Symbolic computation, Wolfram Language, fantastic visualization effect, rich documentation (**highly recommended!**)
- Maple: Symbolic computation, few people use now (DONT USE!)

\* Many computation tasks can be done by Python/[Julia](#) now, and the importance of Matlab is sharpen.

# Installation

- Matlab R2019b (student version)
  - Use the campus Internet to download
- Mathematica 12.0
  - \$50 for student version
  - Online version: <https://www.wolframcloud.com/>



# Basic Usage

Similar to Python's interactive window, but with much more powerful functional support

- Type in display mode
  - Ctrl+2: square root
  - Ctrl+4: superscript
  - Ctrl+/: fraction
- Solve[equ,var]
  - == represents equal, = means assignment
  - Symbolic, high-order, parameter, equation systems
  - NSolve for numerical results
  - Reduce for constrained solutions
- D[f], Integral[f,var], Integral[f,{var,x\_min,y\_min}]
  - High-order, multiple variables
  - Display the steps

# Basic Usage

- `Sum[exp,var]`
- `Simplify[exp]`
- `Plot[f,{x,x_min,x_max}]`
  - `ContourPlot`, `ListLinePlot`, `ParametricPlot`
- You can even copy the formulas as  $\text{\LaTeX}$
- The most powerful thing: Enormous database!
- Explore more in
  - <https://www.wolfram.com/mathematica/>
  - <https://www.zhihu.com/question/27834147>

**\* Make the best of the manual!**

5

# Summary

# Summary

- Introduction
- Package management: pip, Anaconda
- Useful tools: virtualenv, pipreqs, jupyter notebook
- Numpy: fancy indexing, broadcasting, linear algebra
- Math software: Matlab, Mathematica

\* We won't cover Python's OOP & stl in the seminar, but please be familiar with it