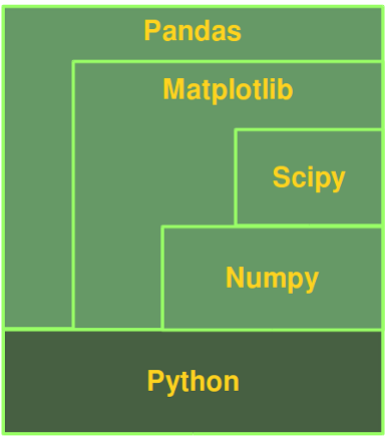
Python Scientific Computing
- numpy, scipy, matplotlib -

2017 – 2019 - 2020

Ando Ki
Future Design Systems
adki@future-ds.com / www.future-ds.com

Table of contents

Related library around numpy



- **Pandas** (Python Data Analysis Library) is a software library written for the Python programming language for data manipulation and analysis
- **Matplotlib**
  - Data visualization routine package for Python
  - It provides MATLAB-like plotting functionality
- **PyLab**
  - Ploting library (along with Matplotlib & SciPy)
- **SciPy**
  - Scientific Python
  - It adds even more MATLAB-like functionalities to Python (scientific algorithms)
- **NumPy**
  - It is numerical Python package
  - It is a library for the Python programming language, adding support for large, multi-dimensional arrays and matrices, along with a large collection of high-level mathematical functions to operate on these arrays. (array mathematics)

FUTURE Design Systems

Installation on Python virtualenv

- Check status and version of additional packages

```
(my_python)$ pip list | grep <package_name>
```

- Install numpy if not installed yet

```
(my_python)$ pip install numpy
```

- Install matplotlib if not installed yet

```
(my_python)$ pip install matplotlib
```

- Install scipy if not installed yet

```
(my_python)$ pip install scipy
```

This may need following packages bef orehand.
$ sudo apt-get install libpng-dev
$ sudo apt-get install libfreetype6-dev

If version error occurs, (my_python)$pi p install –upgrade 'setuptools<45.0.0'.

- If there are any errors while installing additional packages, just install TensorFlow as follows.

```
(my_python)$ pip install –upgrade tfBinaryURL
```

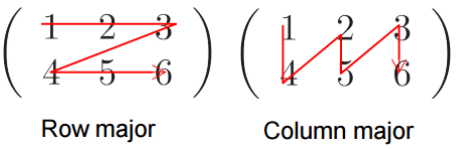
tfBinaryURL for Tensorflow CPU only with Python 2.7:
https://storage.googleapis.com/tensorflow/linux/cpu/tensorflow-1.2.1-cp27-none-linux_x86_64.whl

FUTURE Design Systems

Numpy

- Numpy provides *multidimensional array object* and tools for working with these array.

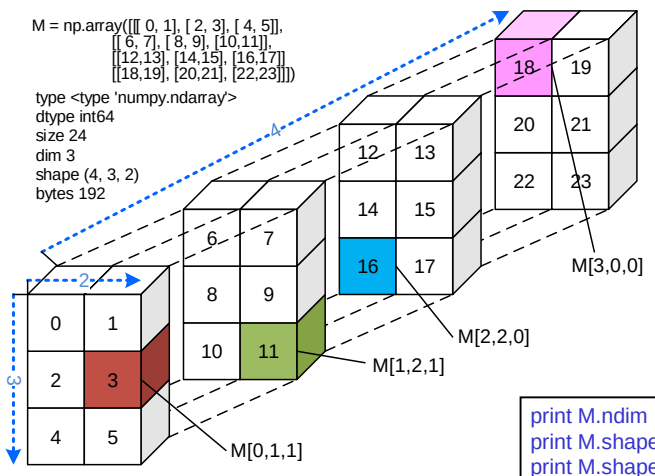
- Numpy array: a grid of values, *all of the same type*.
  - Indexing element: a tuple of nonnegative integer, i.e., 0, 1, …
  - **Rank** of the array: number of dimensions
  - **Shape** of the array: a tuple of integers of the size of the array along each dimension
  - Row or line major array
    - **row-major: C, python**
    - column-major: Fortran, matlab



CuPy: NumPy-like API accelerated with CUDA

FUTURE Design Systems

Numpy array indexing

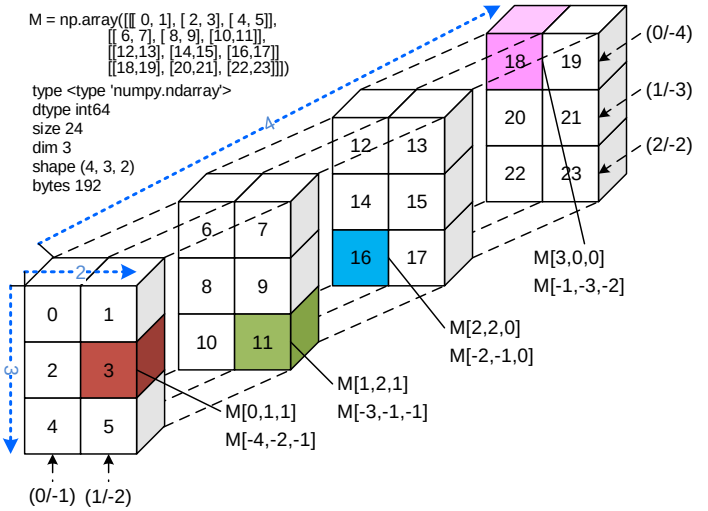



```
print M.ndim
print M.shape
print M.shape[0]
print M.shape[1]
print M.shape[2]
```

```
3
(4,3,2)
4
3
2
```

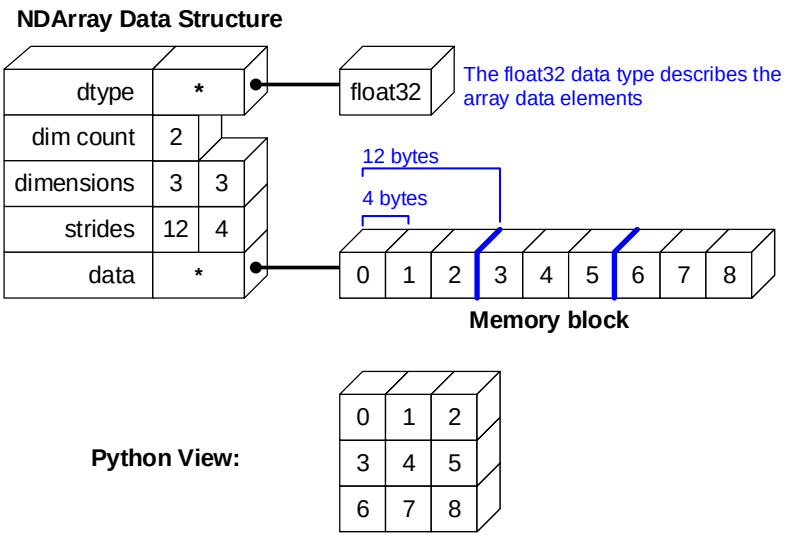

FUTURE Design Systems

Numpy array indexing



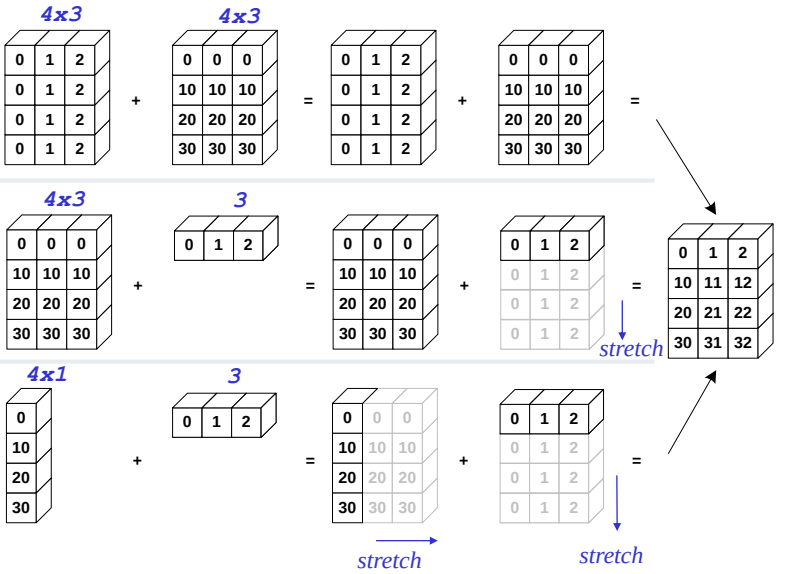
FUTURE Design Systems

Numpy array data structure



FUTURE Design Systems

Numpy array broadcasting



SciPy

- SciPy builds on numpy, and provides a large number of functions that operate on numpy arrays and are useful for different types of scientific and engineering applications.
- An Open Source library of scientific tools for Python
  - Depends on the NumPy library
- Gathers a variety of high level science and engineering modules into one package
- Provides modules for
  - statistics
  - optimization
  - numerical integration
  - linear algebra
  - Fourier transforms
  - signal processing
  - image processing
  - genetic algorithms
  - ODE solvers
  - special functions
  - and more

References

- Python Numpy Tutorials
  - http://cs231n.github.io/python-numpy-tutorial/
- Numpy C-API
  - https://docs.scipy.org/doc/numpy/reference/c-api.html

FUTURE Design Systems

Python Numpy C-API

2017 - 2019

Ando Ki
Future Design Systems
adki@future-ds.com / www.future-ds.com

Table of contents

Numpy C-API: Manual wrapping C/C++ for Numpy

- Step 1: include Python header file
- Step 2: prepare C function to be used
- Step 3: prepare Python callable function, i.e., wrapper
- Step 4: register the callable function within a module's symbol table
- Step 5: write an init function for the module
- Step 6: prepare a setup.py scrip

```
# test.py
import cos_module_np
import numpy as np
import pylab

x = np.arange(0, 2*np.pi, 0.1)
y = cos_module_np.cos_func_np(x)
pylab.plot(x, y)
pylab.show()
```

Import user defined module

Import necessary module

Call C function

Plot

Python C-API: C function to be called by Python

```
#include <Python.h>
#define NPY_NO_DEPRECATED_API NPY_1_7_API_VERSION
#include <numpy/arrayobject.h>
#include <math.h>
//-------------------------------------------------------------------------------
static PyObject *cos_wrapper_np(PyObject *self, PyObject *args) {
    ....
}
//-------------------------------------------------------------------------------
// define functions in module
static PyMethodDef CosMethods[] = {
  {"cos_func_np", cos_wrapper_np, METH_VARARGS,
   "evaluate the cosine ona numpy array"},
  {NULL, NULL, 0, NULL}
};
//-------------------------------------------------------------------------------
// Module initalization
PyMODINIT_FUNC
initcos_module_np(void) {
   (void)Py_InitModule("cos_module_np", CosMethods);
   import_array();
}
```


Python C-API: setup.py

```
from distutils.core import setup, Extension

module1 = Extension('hello', sources = ['hellomodule.c'])

setup ( name = 'Python C-API testing Package'
      , version = '1.0'
      , description = 'This is a tesing package for Python C-API'
      , ext_modules = [module1] )
```

Running ‘cos’ example

- This example shows how to call C function from Python
  - Step 1: go to your project directory and invoke Python virtual environment
    - [user@host] cd $(PROJECT)/codes/python-projects/numpy_C-API/cos
    - [user@host] source ~/my_python/bin/activate
  - Step 2: see the codes
  - Step 3: compile
    - [user@host] make
  - Step 4: run
    - [user@host] make run

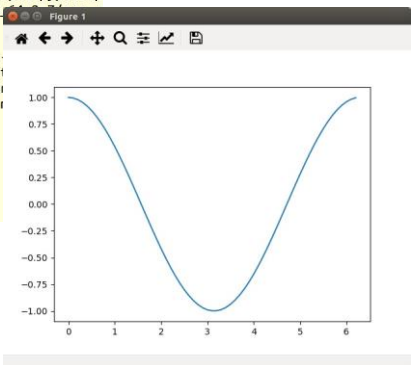
```
# test.py
import cos_module_np
import numpy as np
import pylab

x = np.arange(0, 2*np.pi, 0.1)
y = cos_module_np.cos_func_np(x)
pylab.plot(x, y)
pylab.show()
```

```
[user@host] cd $(PROJECT)/codes/python-projects/numpy_C-API/cos
[user@host] source ~/my_python/bin/activate
(my_python)$ make
(my_python)$ make run
(my_python)$ deactivate
[user@host]
```


Running ‘cos’ example

```
adki@ando-ubuntu: ~/work/seminars/20170822_DeepLearningHWAcc/codes/python-projects/numpy
(tensorflow)$ ls
Clean.bat*  Clean.csh*  Clean.sh*  cos_module_np.c*  Makefile*  Makefile.old*
setup.py*   test.py*
(tensorflow)$ make
python2.7 setup.py build
running build
running build_ext
building 'cos_module_np' extension
creating build
creating build/temp.linux-x86_64-2.7
x86_64-linux-gnu-gcc -pthread -fno-strict-aliasing -DNDEBUG -g -fwrapv -O2 -Wall -Wstric
t-prototypes -fPIC -I/home/adki/tensorflow/local/lib/python2.7/site-packages/numpy/core/
include -I/usr/include/python2.7 -c cos_module_np.c -o build/temp.linux-x86
odule_np.o
creating build/lib.linux-x86_64-2.7
x86_64-linux-gnu-gcc -pthread -shared -Wl,-O1 -Wl,-Bsymbolic-functions -Wl,
nctions -Wl,-z,relro -fno-strict-aliasing -DNDEBUG -g -fwrapv -O2 -Wall -Ws
pes -D_FORTIFY_SOURCE=2 -g -fstack-protector --param=ssp-buffer-size=4 -Wfo
format-security build/temp.linux-x86_64-2.7/cos_module_np.o -o build/lib.li
7/cos_module_np.so
(tensorflow)$ make run
python2.7 test.py
(tensorflow)$
```



References

- Python numpy C-API
  - http://www.scipy-lectures.org/advanced/advanced_numpy/index.html#advanced-numpy
  - http://www.scipy-lectures.org/advanced/interfacing_with_c/interfacing_with_c.html