Python-Numpy

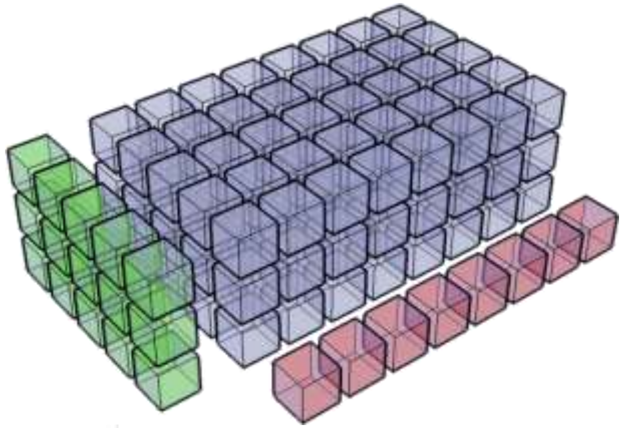
Dr. Sarwan Singh

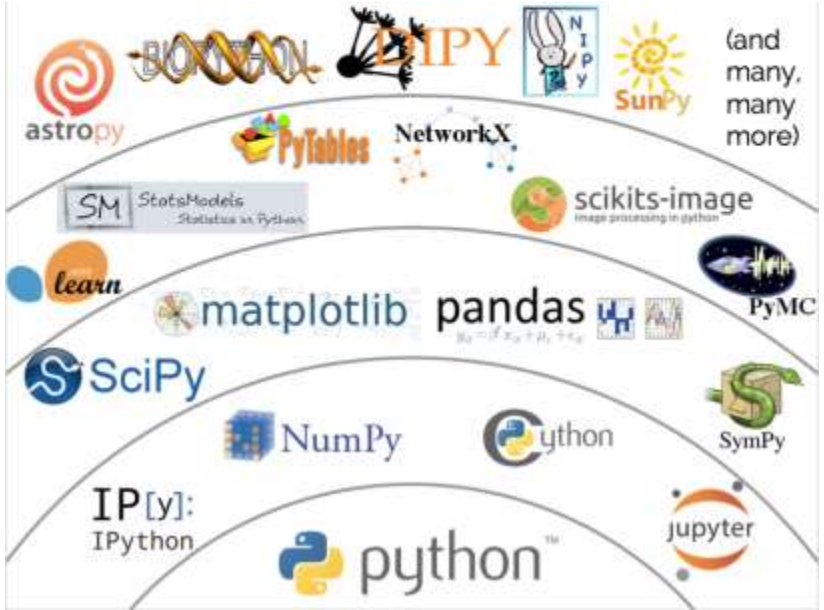
Agenda

- Introduction
- History, usage

NumPy

Numerical Python

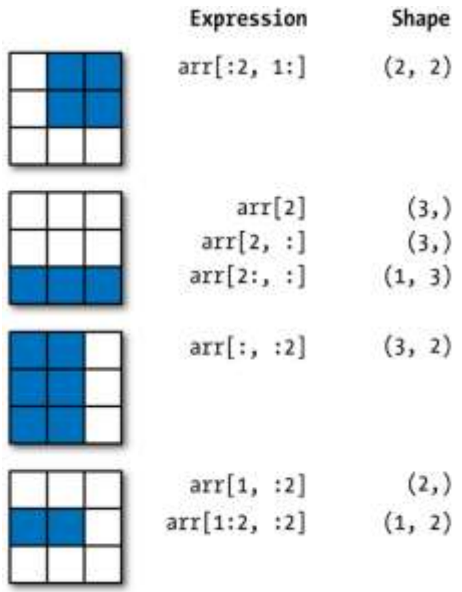


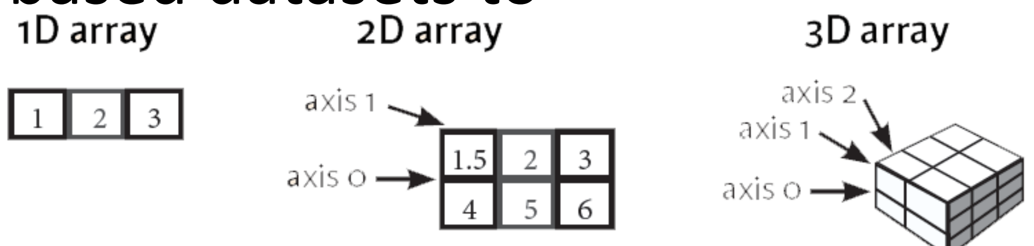


Introduction

- Numerical Python (Numpy) has greater role for numerical computing in Python.
- It provides the data structures, algorithms, and library glue needed for most scientific applications involving numerical data in Python.
- It has fast and efficient multidimensional

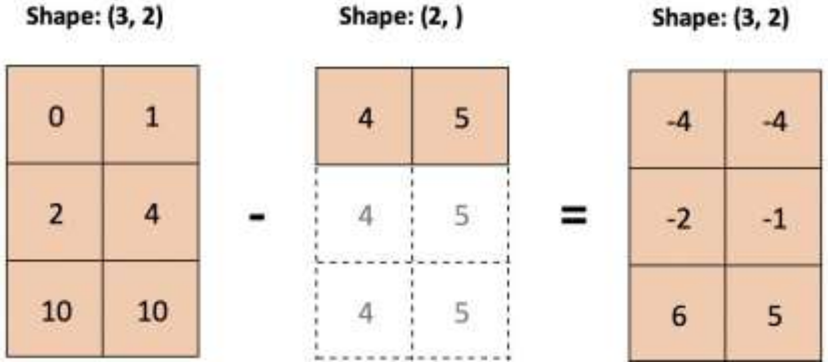
  (N-dimensional) array object ndarray
- Functions for performing element-wise computations with arrays or mathematical operations between arrays
- It has tools for reading and writing array-based datasets to disk





Introduction

- It is useful for Linear algebra operations, Fourier transform, and random number generation
- It has sophisticated (broadcasting) functions
- It has tools for integrating C/C++ and Fortran code
- It provides an efficient interface to store and operate on dense data buffers
- NumPy arrays form the core of nearly the entire ecosystem of data science tools in Python
- Besides its obvious scientific uses, NumPy can also be used as an efficient multi-dimensional container of generic data. Arbitrary data-types can be defined. This allows NumPy to seamlessly and speedily integrate with a wide variety of databases.
- NumPy is licensed under the [BSD license]



History

- **NumPy** derives from an old library called Numeric, which was the first array object built for Python. (written in 2005 launched in 2006)
- Numeric was quite successful and was used in a variety of applications before being phased out.

Jim Fulton

Jim Hugunin

Person	Package	Year
Jim Fulton	Matrix Object in Python	1994
Jim Hugunin	Numeric	1995
Perry greenfield, Rick white, Todd Miller	Numarray	2001
Travis Olipant	NumPy	2005

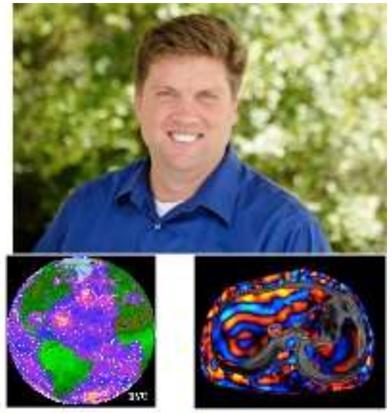

Leaving Microsoft: Software Giant's Key Employee Losse

Jim Hugunin

Jim Hugunin brought his Python skills to Microsoft in 2004 and he left in October 2010 to work for Google. Hugunin delivered IronPython, an implementation of Python for .NET, to Microsoft and helped build the Dynamic Language Runtime. In a notice, he said Microsoft's decision to abandon investment in IronPython led to his decision to leave the company.

Travis Oliphant - CEO

- PhD 2001 from Mayo Clinic in Biomedical Engineering
- MS/BS degrees in Elec. Comp. Engineering
- Creator of **SciPy** (1999-2009)
- Professor at BYU (2001-2007)
- Author of **NumPy** (2005-2012)
- Started **Numba** (2012)
- Founding Chair of **Numfocus** / **PyData**
- Previous PSF Director



NumPy SciPy

NumFOCUS Open Code, Better Science

python

3

- NumPy’s main object is the homogeneous multidimensional array. It is a table of elements (usually numbers), all of the same type, indexed by a tuple of positive integers.
- In NumPy dimensions are called *axes*.
- NumPy defines N-dimensional array type called ndarray(also known by the alias array). It describes the collection of items of the same type. Items in the collection can be accessed using a zero-based index.
- numpy.array is not the same as the Standard Python Library class array.array, which only handles one-dimensional arrays.

Syntax

- numpy.array(
    object,
    dtype = None,
    copy = True,
    order = None,
    subok = False,
    ndmin = 0)

```python
import numpy as np
```

```python
a = np.array([2,3,4])
```

```python
a
```

```
array([2, 3, 4])
```

```python
print("Shape    : " ,a.shape )
print("Size     : " ,a.size )
print("Datatype : " ,a.dtype )
print("ndim     : " ,a.ndim )
```

```
Shape    :  (3,)
Size     :  3
Datatype :  int32
ndim     :  1
```

```python
aa=np.array( [ [1,2,3,4],[5,6,7,8] ])
```

```python
aa
```

```
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
```

```python
print("Shape    : " ,aa.shape )
print("Size     : " ,aa.size )
print("Datatype : " ,aa.dtype )
print("ndim     : " ,aa.ndim )
```

```
Shape    :  (2, 4)
Size     :  8
Datatype :  int32
ndim     :  2
```

dtype : Desired data type of array, optional

copy : Optional. By default (true), the object is copied

order : C (row major) or F (column major) or A (any) (default)

subok : By default, returned array forced to be a base class array. If true, sub-classes passed through

ndmin : Specifies minimum dimensions of resultant array a = np.array(1,2,3,4)   # WRONG

a = np.array([1,2,3,4]) # RIGHT

- The function zeros creates an array full of zeros,
- the function ones creates an array full of ones,
- the function empty creates an array whose initial content is random and depends on the state of the memory.
- By default, the dtype of the created array is float64.
- np.ones((2,3,4), dtype=np.int16)
- np.zeros((3,4))
- np.empty((2,3))

```
a2 = np.array([1, 2, 3, 4], ndmin = 2)
```

```
print("Array    : " ,a2 )
print("Shape    : " ,a2.shape )
print("Size     : " ,a2.size )
print("Datatype : " ,a2.dtype )
print("ndim     : " ,a2.ndim )
```

```
Array    :  [[1 2 3 4]]
Shape    :  (1, 4)
Size     :  4
Datatype :  int32
ndim     :  2
```

```
aa
```

```
array([[1, 2, 3, 4],
       [5, 6, 7, 8]])
```

```
aa[1,3]
```

```
8
```

```
aa[1]
```

```
array([5, 6, 7, 8])
```

```
aa[1,3]=20
```

```
aa
```

```
array([[ 1,  2,  3,  4],
       [ 5,  6,  7, 20]])
```

Printing Array

- When you print an array, NumPy displays it in a similar way to nested lists

```
a = np.arange(6) # 1d array
print(a)
```

```
[0 1 2 3 4 5]
```

```
b = np.arange(12).reshape(4,3) # 2d array
print(b)
```

```
[[ 0  1  2]
 [ 3  4  5]
 [ 6  7  8]
 [ 9 10 11]]
```

```
c = np.arange(24).reshape(2,3,4) # 3d array
print(c)
```

```
[[[ 0  1  2  3]
  [ 4  5  6  7]
  [ 8  9 10 11]]

 [[12 13 14 15]
  [16 17 18 19]
  [20 21 22 23]]]
```

Basic Operations

- Arithmetic operators on arrays apply *elementwise*. A new array is created and filled with the result
  - * , + , +=, *=
  - Dot

```
A-B
```

```
array([[-4, -4],
       [-4, -4]])
```

```
A **2
```

```
array([[0, 1],
       [4, 9]], dtype=int32)
```

```
B <6
```

```
array([[ True,  True],
       [False, False]], dtype=bool)
```

```
A=np.arange(4).reshape(2,2)
B=np.array([[4,5],[6,7]])
print("A \n",A,"\n B\n",B, "\nA+B\n", A+B)
```

```
A
 [[0 1]
 [2 3]]
 B
 [[4 5]
 [6 7]]
A+B
 [[ 4  6]
 [ 8 10]]
```

```
# elementwise product
print("A \n",A,"\n B\n",B, "\nA*B\n", A*B)
```

```
A
 [[0 1]
 [2 3]]
 B
 [[4 5]
 [6 7]]
A*B
 [[ 0  5]
 [12 21]]
```

```
# matrix product
print("A \n",A,"\n B\n",B, "\nA*B\n", A.dot(B) )
```

```
A
 [[0 1]
 [2 3]]
 B
 [[4 5]
 [6 7]]
A*B
 [[ 6  7]
 [26 31]]
```

```
print("A \n",A)
print("B \n",B)
print("A.max " , A.max())
print("A.min " , A.min())
print("A.sum " , B.sum())
```

```
A
 [[0 1]
 [2 3]]
B
 [[4 5]
 [6 7]]
A.max  3
A.min  0
A.sum  22
```

```
B.min(axis=1)
```

```
array([4, 6])
```

```
B.cumsum(axis=1)
```

```
array([[ 4,  9],
       [ 6, 13]], dtype=int32)
```

Universal Functions

- NumPy provides familiar mathematical functions such as sin, cos, and exp. In NumPy, these are called “universal functions”(ufunc).
- Within NumPy, these functions operate elementwise on an array, producing an array as output.

```
np.sqrt(A)
```

```
array([[ 0.        ,  1.        ],
       [ 1.41421356,  1.73205081]])
```

```
np.exp(A)
```

```
array([[  1.        ,   2.71828183],
       [  7.3890561 ,  20.08553692]])
```

```
np.add(A,B)
```

```
array([[ 4,  6],
       [ 8, 10]])
```

Indexing, Slicing and Iterating