Mastering Python

الدرس #7_1

مكتبة الارقام : Numpy

By:

Hussam Hourani

V1.0 - NOV 2019

Agenda

- Why Numpy
- Array
- Array Indexing and Slicing
- Sample implementations
- Array Operations - Sort

Why Numpy

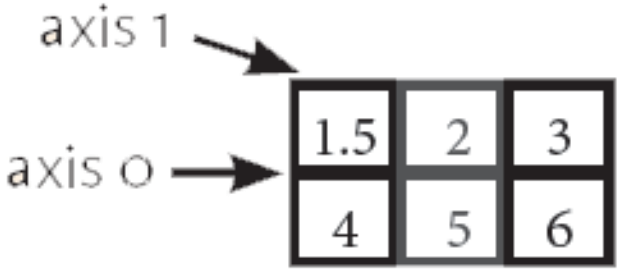
- NumPy is the fundamental package for scientific computing with Python. It contains among other things:
  - a powerful N-dimensional array object
  - sophisticated (broadcasting) functions
  - tools for integrating C/C++ and other languages
  - useful linear algebra, Fourier transform, and random number capabilities
- 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.
- http://www.numpy.org/

Array

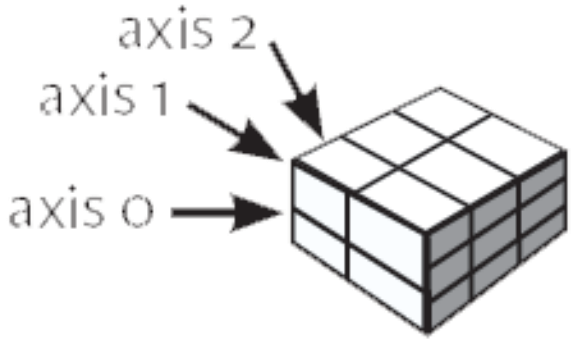
- NumPy is one of Python's that is very important for learning data science because :The library provides creation of an array data structure that holds many benefits over the lists :
  - More compact.
  - Faster access in reading and writing items.
  - More convenient and more efficient.

1D array

2D array



3D array



etc,….N- Dimension

https://www.datacamp.com

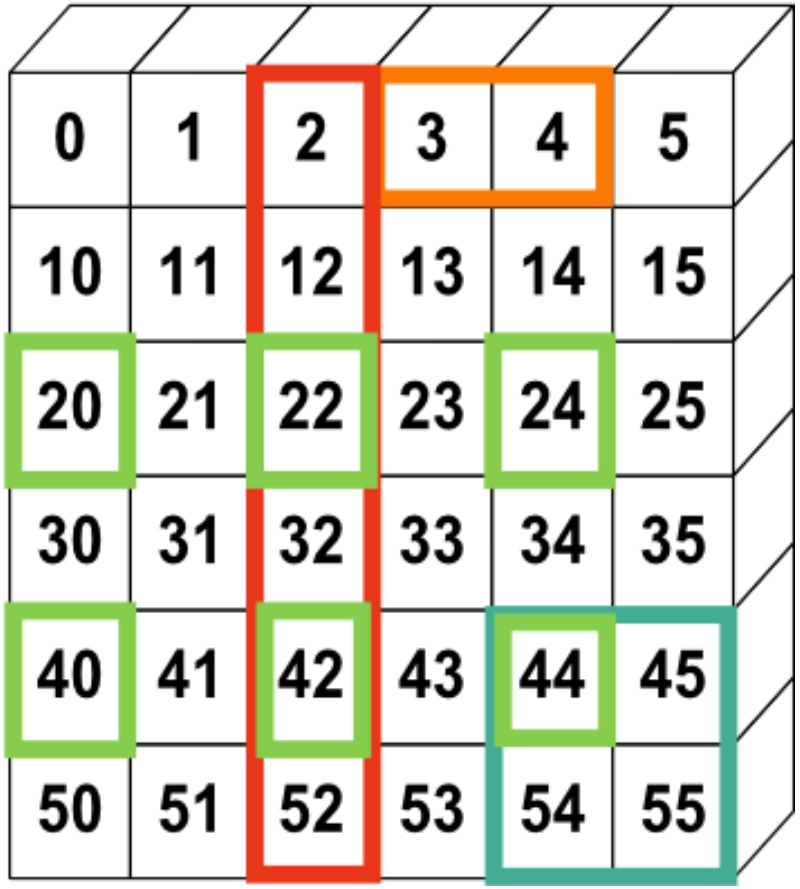
Array Indexing and Slicing

```
>>> a[0,3:5]
array([3,4])


>>> a[4:,4:]
array([[44, 45],
       [54, 55]])

>>> a[:,2]
array([2,12,22,32,42,52])

>>> a[2::2,::2]
array([[20,22,24]
       [40,42,44]])
```



Source: http://www.scipy-lectures.org/intro/numpy/array_object.html

Importing Libraries

```
# 'generic import' of math module
import math
math.sqrt(25)

# import a function
from math import sqrt
sqrt(25)    # no longer have to reference the module

 # import multiple functions at once
from math import cos, floor
 cos(10)
# import all functions in a module (generally discouraged)
from math import *
 cos(10)
# define an alias
import math as m
m.sqrt(25)
```

Array

```
import numpy as np
b = np.array([1, 4, 7, 5])
print(b)
```

Output

```
[1 4 7 5]
```

```
c = np.array([ [1, 4, 7, 5],
               [2, 8, 3, 2]])
print(c)
```

Output

```
[[1 4 7 5]
 [2 8 3 2]]
```

```
a = np.arange(12).reshape(3, 4)
print(a)
print ("a size: ", a.size)
print ("a shape: ", a.shape)
print ("a ndim: ", a.ndim)
print ("a dtype.name: ", a.dtype.name)
print ("a itemsize: ", a.itemsize)    # in bytes
```

Output

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
a size:  12
a shape:  (3, 4)
a ndim:  2
a dtype.name:  int32
a itemsize:  4
```

```
d = np.array([(1.5,2,3), (4,5,6)])
print (d)
print ("d shape: ", d.shape)
```

Output

```
[[1.5 2.  3. ]
 [4.  5.  6. ]]
d shape:  (2, 3)
```

```
f = np.array( [ [1,2], [3,4] ], dtype=complex )
print(f)
```

Output

```
[[1.+0.j 2.+0.j]
 [3.+0.j 4.+0.j]]
```

Array

```
e= np.zeros( (4,10) )
print (e)
```

Output

```
[[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]
 [0. 0. 0. 0. 0. 0. 0. 0. 0. 0.]]
```

```
g= np.ones( (2,4) , dtype=np.int16 )
print (g)
```

Output

```
[[1 1 1 1]
 [1 1 1 1]]
```

```
i = np.arange( 0, 2, 0.3 )
print(i)
```

Output

```
[0.  0.3 0.6 0.9 1.2 1.5 1.8]
```

```
j = np.arange(24).reshape(2,3,4)
print(j)
print ("j shape: ", j.shape)
print ("j ndim: ", j.ndim)
```

Output

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

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

j shape:  (2, 3, 4)
j ndim:  3
```

Array Operations

```
a = np.array([1, -1, 7, 3])
b = np.array([-9, 4, 0, 5])
c =  np.array([[10, 6, 0, 2],[1,2,3,4]])
print("a-b: ",a-b)
print("a*b: ",a*b)
print("a.dot(b): ",a.dot(b))
print("a*2: ",a*2)
print("np.sin(a): ",np.sin(a))
print("a<3: ",a<3)
print("a.sum(): ",a.sum())
print("a.sum(axis=0): ",a.sum(axis=0))
print("c.sum(): ",c.sum())
print("c.sum(axis=0): ",c.sum(axis=0))
print("a.min(): ",a.min())
print("a.max(): ",a.max())
print("a.mean(): ",a.mean())
print("a average(): ",np.average(a))
print("a median(): ",np.median(a))
print("a std(): ",np.std(a))
print("a var(): ",np.var(a))
print("c.cumsum(): ",c.cumsum())
print("a[1:2] :  ",a[1:2])
print("a[2:] :  ",a[2:])
print("c[-1] :  ",c[-1])
```

Output

```
a-b:  [10 -5  7 -2]
a*b:  [-9 -4  0 15]
a.dot(b):  2
a*2:  [ 2 -2 14  6]
np.sin(a):  [ 0.84147098 -0.84147098
0.6569866   0.14112001]
a<3:  [ True  True False False]
a.sum():  10
a.sum(axis=0):  10
c.sum():  28
c.sum(axis=0):  array([11,  8,  3,  6])
a.min():  -1
a.max():  7
a.mean():  2.5
a average():  2.5
a median():  2.0
a std():  2.958039891549808
a var():  8.75
c.cumsum():  [10 16 16 18 19 21 24 28]
a[1:2] :  [-1]
a[2:] :  [7 3]
c[-1] :  [1 2 3 4]
```

General

```
import numpy as np

a = np.array([0,30,45,60,90])

b = np.array([1.0,5.55, 123, 0.567, 25.532])

print ("sin(a): ", np.sin(a*np.pi) )
print ("b around : ", np.around(b) )
print ("b around 1: ", np.around(b, decimals = 1) )
print ( "b around 2: ",np.around(b, decimals = 2) )
print ("b foor: ", np.floor(b) )
print ("b ceil: ", np.ceil(b) )

print ("cos(pi): ", np.cos(np.pi) )
print ("tan(pi): ", np.tan(np.pi) )
print ("arcsin : ", np.arcsin(np.pi/180) )
print ("degrees(pi): ", np.degrees(np.pi) )
```

Output

```
sin(a):  [ 0.00000000e+00 -1.07793678e-14
1.95819692e-15 -2.15587355e-14
 -3.91639383e-15]

b around :  [  1.   6. 123.   1.  26.]
b around 1:  [  1.    5.6 123.    0.6  25.5]
b around 2:  [  1.    5.55 123.    0.57
25.53]
b foor:  [  1.   5. 123.   0.  25.]
b ceil:  [  1.   6. 123.   1.  26.]


cos(pi):  -1.0
tan(pi):  -1.2246467991473532e-16
arcsin :  0.01745417873758517
degrees(pi):  180.0
```

Array Operations - Sort

Syntax:
numpy.sort(a, axis, kind, order)

```
import numpy as np
a =
np.array([[3,7,2,1,8,7,19,15],[10,2,7,4,5,5,
9,1]])

print('a array:')
print (a)
print('\n quicksort:')
print (np.sort(a,kind='quicksort') )
print('\n mergesort')
print (np.sort(a,kind='mergesort') )
print('\n heapsort:')
print (np.sort(a,kind='heapsort') )
print('\n sort as flattened arra:')
print (np.sort(a,axis=None) )
```

Output

```
a array:
[[ 3  7  2  1  8  7 19 15]
 [10  2  7  4  5  5  9  1]]

 quicksort:
[[ 1  2  3  7  7  8 15 19]
 [ 1  2  4  5  5  7  9 10]]

 mergesort
[[ 1  2  3  7  7  8 15 19]
 [ 1  2  4  5  5  7  9 10]]

 heapsort:
[[ 1  2  3  7  7  8 15 19]
 [ 1  2  4  5  5  7  9 10]]

 sort as flattened arra:
[ 1  1  2  2  3  4  5  5  7  7  7  8
9 10 15 19]
```

Spyder (Python 3.6)

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Python\My Stuff

Editor - C:\Python\My Stuff\untitled7.py

fib.py | Bayesian1.py | statistics3.py | statistics4.py | statistics5.py | untitled7.py*

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  def mandelbrot( h,w, maxit=20 ):
4      """Returns an image of the Mandelbrot fractal of size (h,w)."""
5      y,x = np.ogrid[ -1.4:1.4:h*1j, -2:0.8:w*1j ]
6      c = x+y*1j
7      z = c
8      divtime = maxit + np.zeros(z.shape, dtype=int)
9
10     for i in range(maxit):
11         z = z**2 + c
12         diverge = z*np.conj(z) > 2**2            # who is diverging
13         div_now = diverge & (divtime==maxit)  # who is diverging now
14         divtime[div_now] = i                  # note when
15         z[diverge] = 2                        # avoid diverging too mu
16
17     return divtime
18 plt.imshow(mandelbrot(400,400))
19 plt.show()
20
```

... 201 of 212 Aa

Replace with: Replace/find next Replace selection Replace all

Variable explorer

Name | Type | Size | Value

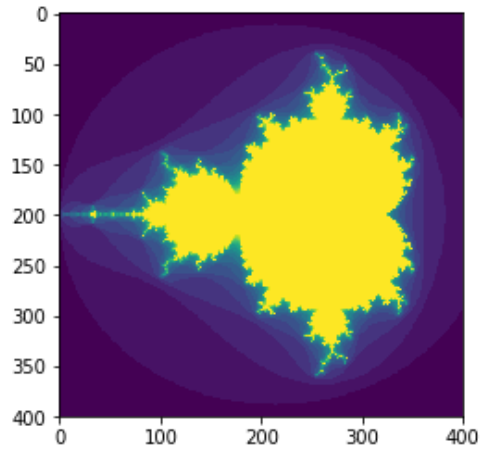
Help | Variable explorer | File explorer

IPython console

Console 1/A

```
In [1]:

In [1]: runfile('C:/Python/My Stuff/untitled7.py', wdir='C:/Python/My
Stuff')
```




```
In [2]:
```

IPython console | History log

Permissions: RW End-of-lines: CRLF Encoding: UTF-8 Line: 20 Column: 1 Memory: 71 %

By : Hussam Hourani

Master in Software Engineering

Hussam Hourani has over 25 years of Organizations Transformation, VROs, PMO, Large Scale and Enterprise Programs Global Delivery, Leadership, Business Development and Management Consulting. His client experience is wide ranging across many sectors but focuses on Performance Enhancement, Transformation, Enterprise Program Management, Artificial Intelligence and Data Science.