Data Analysis and Hypothesis Testing Using the Python ecosystem

Numpy & Matplotlib

Stavros Demetriadis

sdemetri@csd.auth.gr
http://mlab.csd.auth.gr/sdemetri

Numpy: The ‘ndarray' object

- What is 'numpy'?
  - Numpy and ndarray
  - ndarray vs. list (speed, vectorization)
- Array construction
  - The **array()** constructor
  - Array vs. List: measuring the performance with %timeit
  - Constructing 2D, 3D… arrays with array()
- Array construction (cont'd)
  - The **arange()** constructor
  - Array important properties
  - Reshaping arrays
  - Array construction: Special cases

Numpy: The ‘ndarray' object

- Array indexing
  - 1D, 2D and 3D indexing
  - Printing 2D array example
  - Assignment & Broadcasting
  - Fancy indexing
- Array slicing
  - 1D, 2D and 3D slicing
  - View vs. copy
  - Slice with assignment

Numpy: The ‘ndarray' object

- Array operations
  - Element-wise operations
  - Broadcasting
- Array functions
  - Universal functions
  - Statistical functions
  - Axis
  - Difference between a ufunc and a typical function

Review exercises

- ‘ndarray’ object

array(), reshape, shape & broadcasting

- Using the **array()** method and list comprehension to construct the ‘ar’ ndarray object with 10 integers (1 to 10) and shape (2,5)
- Print the **ndim** and **shape** properties of the ‘ar’ array

- Then use reshape() to construct the ‘br’ array based on ‘ar’ but with shape (5,2).
- Try to multiply ar * br. What do you get? Why?

- Change ar and br shapes so that array multiplication is possible.

arange()

- Use the arange() method to construct the ‘ar1’ array object including all integers multiples of 4 (including 4) that are less than 100. Print out how many these numbers are.
- Then reshape the array using the shape method (try at least three different shapes). For each new shape print the ndim property. What is the greatest ndim value that you can achieve with the specific array? Why?

Array indexing

- Construct a 2 X 10 array (ar) with random integers in [1,100] and reshape it as (2,5,2). Then replace all values in the first 5X2 array-item with ‘0’ and the values in the second 5X2 array-item with ‘100’ (no loops).

- Construct two 10-item vectors (ar and br) with random integers in [1,100]. Then swap the values of the two arrays (no loops).

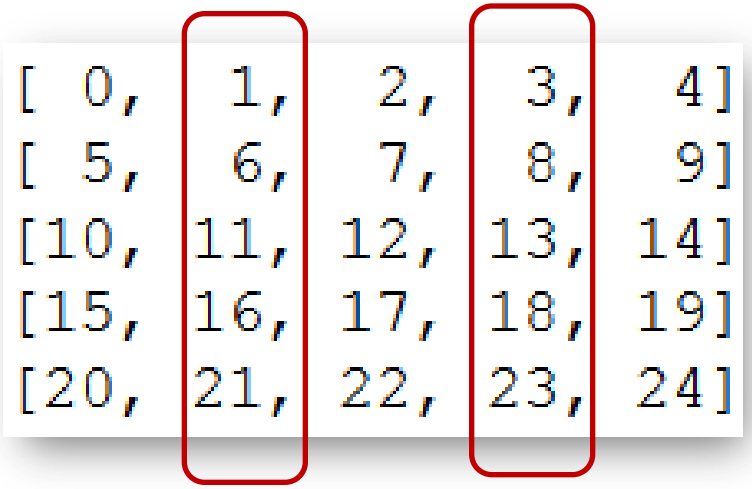
Fancy indexing

- Construct a 10-item vector with random integers in [1,100]. Then (without writing any loop) replace those numbers smaller or equal to 50 with ‘0’ and those greater than 50 with ‘100’.

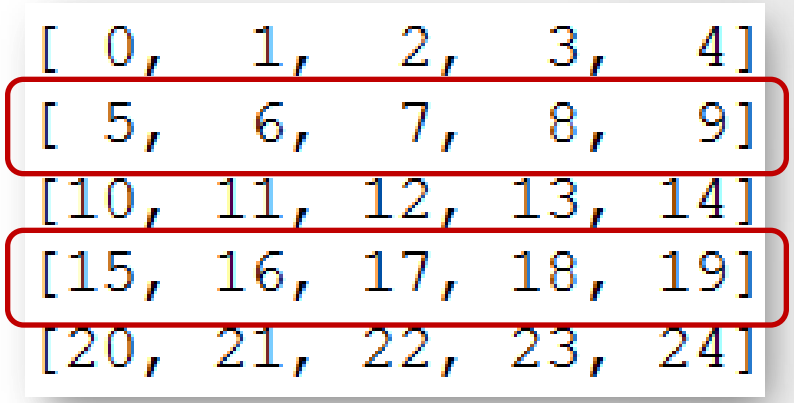
Array slicing

- Write the slices to produce the sub-lists highlighted below

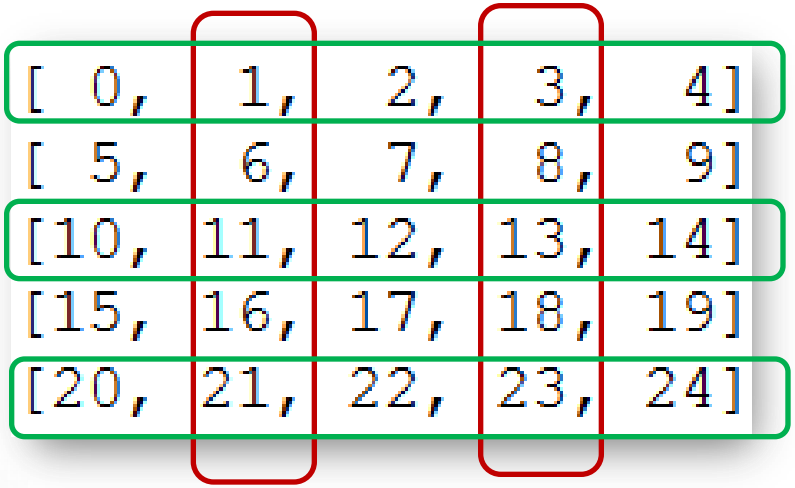
(A)



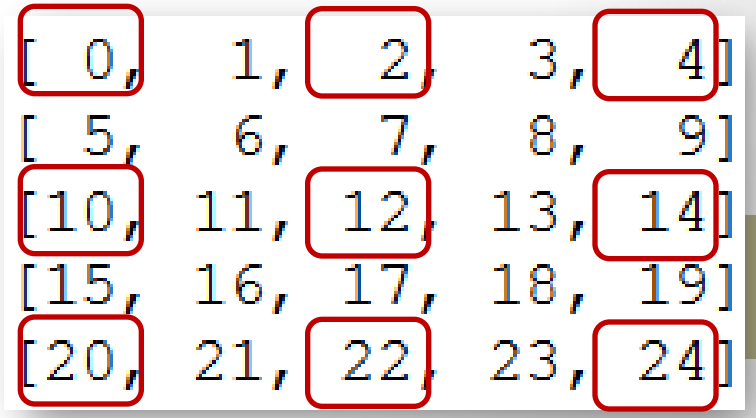
(B)



(C)



(D)



Array slicing

- Construct a 6X6 array with random integers in [1,100]. Then slice the array and assign the value ‘0’ in the cross-section of **even** indexed rows and **odd** indexed columns.

Example output:

```
array([[19,  0, 14,  0, 97,  0],
       [49, 49, 31, 57, 31, 85],
       [72,  0, 97,  0, 74,  0],
       [10, 76, 39, 30, 75,  8],
       [44,  0, 60,  0, 16,  0],
       [50, 77, 42, 34, 94, 12]])
```

Ufuncs vs. numpy Functions

- Construct two (2,5) arrays (named ar and br) with random integers in [1,10]. Then apply appropriate ufuncs (see here) to get:
- A) the **sum** of the arrays (element-wise),
- B) the **square root** of one of the arrays,
- C) an array with '**True**' where 'ar' element is greater than 'br' element

- Then use the 'axis' property and the appropriate statistical function (see here) to find the sum of numbers in rows and columns. What is the difference between ufuncs (Universal functions) and standard numpy functions?

Statistics (advanced)

- Construct an array with 100 pseudo-random integers in [1,100]. Then use **mean**(), **abs**(), and **sqrt**() numpy functions to get the standard deviation of the array according to formula:

$$s = \sqrt{\frac{1}{N-1}\sum_{i=1}^{N}(x_i - M)^2}$$

- x_i : integer values in the array
- M: the mean of the array values
- N: sample size (the size of the array)

- Write your code solution in the form of a function that can get as input an array.
- Then compare the value you get back from your function to the values returned by the **std()** numpy function (see here). Is it the same? Why? *What is the overall conclusion?*

Review exercises

- 100 numpy exercises

Introduction

- The following exercises come from the '**100 numpy exercises**' webpages supported by the numpy community
- These exercises are indicated by the colored number on the right showing the rank of the exercise in the whole list of the 100 exercises for easy access
- The '100 numpy exercises' website includes also the exercise solutions. It is strongly recommended that you work first on the exercise to develop your own solution and then compare the given solution to your own.

arange()

- Create a vector with values ranging from 10 to 49 (6)
- Reverse a vector (first element becomes last) (7)
- Create a 3x3 matrix with values ranging from 0 to 8 (8)

Special form arrays

- Create a null vector of size 10 but the fifth value which is 1 (5)
- Create a 3x3 identity matrix (10)
  - *Hint*: use eye() but alternatively you can also use identity()
- Find indices of non-zero elements from [1,2,0,0,4,0] (9)
  - *Hint*: Use numpy.nonzero but alternatively you can also use numpy.where

Arrays with random values

- Create a 3x3x3 array with random values (11)
- Create a random vector of size 30 and find the mean value (13)
- Create a 10x10 array with random values and find the minimum and maximum values (12)

Functions in numpy

- Open a new notebook and run the following code (24)

```
Import numpy as np
print(sum(range(5),-1))
print(np.sum(range(5),-1))
```

Find the relevant documentation (for the built-in sum and the np.sum functions) and explain the outcome

Matplotlib: plotting in Python

- What is 'matplotlib'?
  - What to import: import matplotlib.pyplot as plt
  - Your first plot
  - Use *%matplotlib inline*
- Basic plotting with 'plot'
  - How to plot an array
  - How to define the values in x-axis
  - How to change the line style
- Customizing the plot
  - Setting axes limits
  - Adding graph legend
  - Moving the axes
  - Annotating the graph

Matplotlib: plotting in Python

- Common plots
  - Bar chart
  - Pie chart
  - Histogram
  - Box-and-Whisker plot
  - Scatter plot
- Multiple plots

Scipy: introduction to scientific computing

- What is Scipy