**Programming for Data Science**

**Practical 02 – Data Manipulation using the Numpy module**

**What you will learn / do in this lab**

1. Learn how to create and manipulate data using the Python Numpy package

Table of Contents

[1. Overview 3](#_Toc67651538)

[A. What you will do for this lab 3](#_Toc67651539)

[B. Intro to Numpy 3](#_Toc67651540)

[C. Why use Numpy? 3](#_Toc67651541)

[2. Installing and updating Numpy 4](#_Toc67651542)

[A. Install Numpy 4](#_Toc67651543)

[B. Update Numpy 4](#_Toc67651544)

[3. Create Numpy arrays 6](#_Toc67651545)

[Task 1: Create 1-d numpy array 6](#_Toc67651546)

[Task 2: Create 2-d numpy array 6](#_Toc67651547)

[4. Inspect Numpy arrays 7](#_Toc67651548)

[Task 1: Use the shape, ndim, size properties and len function 7](#_Toc67651549)

[Task 2: Use the type function, dtype property 7](#_Toc67651550)

[5. Manipulating Array Shapes 8](#_Toc67651551)

[Task 1: Flatten an array 8](#_Toc67651552)

[Task 2: Reshape arrays 8](#_Toc67651553)

[Task 3: Transpose array 8](#_Toc67651554)

[6. Manipulating Array Content 9](#_Toc67651555)

[Task 1: Concatenate arrays (SUBMISSION REQUIRED) 9](#_Toc67651556)

[Task 2: Split arrays 9](#_Toc67651557)

[7. Copying arrays 10](#_Toc67651558)

[Task 1: Make a copy of a Numpy array 10](#_Toc67651559)

[8. Sorting arrays 11](#_Toc67651560)

[Task 1: Basic sorting (SUBMISSION REQUIRED) 11](#_Toc67651561)

[9. Subsetting and Indexing 13](#_Toc67651562)

[Task 1: Basic subsetting 13](#_Toc67651563)

[Task 2: Boolean indexing (SUBMISSION REQUIRED) 13](#_Toc67651564)

[Task 3: Indexing using WHERE 14](#_Toc67651565)

[10. Array Math 15](#_Toc67651566)

[Task 1: Add, subtract, multiply and divide numpy arrays 15](#_Toc67651567)

[11. Statistical methods 16](#_Toc67651568)

[Task 1: sum, mean (SUBMISSION REQUIRED) 16](#_Toc67651569)

[Task 2: std, var 17](#_Toc67651570)

[Task 3: min/ max, argmin/ argmax 18](#_Toc67651571)

[12. File I/O on Numpy arrays 19](#_Toc67651572)

[Task 1: Load a numpy array from a textfile (SUBMISSION REQUIRED) 19](#_Toc67651573)

# Overview

## What you will do for this lab

In this lab, you will learn how to use the **Python Numpy** package to create ndarrays, extract subsets from them, sort them, perform mathematical operations on them and more!

Most importantly, you will learn how to apply Numpy functions to do a simple analysis of a given dataset, e.g. sales orders to find out key characteristics or statistical trends from them.

## Intro to Numpy

**NumPy** is the fundamental package for scientific computing in Python.

It is a Python library that provides a **multi-dimensional array object**, various derived objects (such as masked arrays and matrices), and an assortment of routines for fast operations on arrays, including mathematical, logical, shape manipulation, sorting, selecting, I/O, discrete Fourier transforms, basic linear algebra, basic statistical operations, random simulation and much more.

At the core of the NumPy package, is the **ndarray** object. This encapsulates *n*-dimensional arrays of homogeneous data types, with many operations being performed in compiled code for performance.

## Why use Numpy?

Because Numpy runs much faster!

There are several important differences between NumPy arrays and the standard Python sequences.

* NumPy arrays have a fixed size at creation, unlike Python lists (which can grow dynamically). Changing the size of an *ndarray* will create a new array and delete the original.
* The elements in a NumPy array are all required to be of the same data type, and thus will be the same size in memory. The exception: one can have arrays of (Python, including NumPy) objects, thereby allowing for arrays of different sized elements.
* NumPy arrays facilitate advanced mathematical and other types of operations on large numbers of data. Typically, such operations are executed more efficiently and with less code than is possible using Python’s built-in sequences.
* A growing plethora of scientific and mathematical Python-based packages are using NumPy arrays; though these typically support Python-sequence input, they convert such input to NumPy arrays prior to processing, and they often output NumPy arrays. In other words, in order to efficiently use much (perhaps even most) of today’s scientific/mathematical Python-based software, just knowing how to use Python’s built-in sequence types is insufficient - one also needs to know how to use NumPy arrays.

# Installing and updating Numpy

## Install Numpy

Use pre-built package

In most use cases the best way to install NumPy on your system is by using an pre-built package for your operating system such as those listed below.

In this module, you **do not have to** separately install Numpy as it already comes with Microsoft Azure Notebooks.

/\*tbd

Check out <http://scipy.org/install.html> for links to other available options.

|  |  |
| --- | --- |
| **Anaconda** | A free distribution for the SciPy stack. Supports Linux, Windows & Mac. |
| **Enthought Canopy** | The free and commercial versions include the core SciPy stack packages. Supports Linux, Windows and Mac |
| **Python(x,y)** | A free distribution including the SciPy stack, based around the Spyder IDE. Windows only. |
| **WinPython** | A free distribution including the SciPy stack. Windows only. |
| **Pyzo** | A free distribution based on Anaconda and the IEP interactive development environment. Supports Linux, Windows and Mac. |

\*/

## Update Numpy

Use Anaconda commmand line to update Numpy

|  |  |
| --- | --- |
| If you find that some commands are not working for you, you could be using an outdated version of Numpy.  The latest version of Numpy is 1.13 so you can see I am using an outdated version of it in the screenshot below. |  |

|  |  |
| --- | --- |
| To update your Numpy to the latest version, issue the command to ask Anaconda to download and install the latest release. | conda install -c anaconda numpy |

\*/

# Create Numpy arrays

### Task 1: Create 1-d numpy array

Write Python code to create 1-d Numpy arrays containing the values as follows

1. [0,1,2,3,4,5,6,7]
2. ['Mary', 'John', 'Peter', 'Christine']
3. ['2017-10-01', '2017-10-02', '2017-10-03',….., '2017-10-31']
4. 10 random decimal numbers between 0 and 1

### Task 2: Create 2-d numpy array

Write Python code to create 2-d Numpy arrays containing the values as follows

1. 2x4 array [(1,2,3,4), (5,6,7,8)]
2. 3x3 array [(1.6,1.7,1.8),(2.5,5.0,7.5),(20,40,60)]
3. 3x4 2-d array, with initial value zero
4. 4x3 2-d array with initial value 10.5
5. 4x8 array with random values between 1 and 6

# Inspect Numpy arrays

### Task 1: Use the shape, ndim, size properties and len function

|  |  |
| --- | --- |
| Create 4 arrays with the values as shown in (a) to (d), then use the following properties and function on each array to inspect each one.   * ***shape*** property to find out the shape of the array * ***ndim*** property to find out the number of dimensions of the array * ***size*** property to find out the total number of elements in the array * ***len*** function to find out the number of rows in the array  1. [1.55,1.73,1.9,1.72,1.66,1.8] 2. [('red', 'green', 'yellow', 'blue'),(True,False, False, True), ('2017-10-01', '2017-11-11', '2017-12-23', '2018-03-31')] 3. [[(10,9,8,7),(6,5,4,3)], [(1,2,3,4),(5,6,7,8)]] |  |

### Task 2: Use the type function, dtype property

|  |  |
| --- | --- |
| Apply the ***type*** function and ***dtype*** property on each of the arrays in Task 2 to inspect them    Your output should appear as shown on the right. |  |

# Manipulating Array Shapes

### Task 1: Flatten an array

|  |
| --- |
| 1. Write Python code to convert the multi-dimensional array a, into a 1-D array   **a = np.array([ ['Mary', 'John','Bob'],**  **['Zoe', 'Chris','Ann'],**  **['Leon', 'Kathy','Sam']])** |

### Task 2: Reshape arrays

|  |
| --- |
| 1. Write Python code to convert the following 1-D array a into a 3x5 multi-dimensional array   **a = np.arange(15,30)** |
| 1. Convert the following 2x6 array b into a 3x4 array   **b = np.array([ [1,2,3,4,5,6], [7,8,9,10,11,12]])** |

### Task 3: Transpose array

|  |
| --- |
| 1. Write Python code to transpose the following array   **a = np.array([ [1,2,3,4,5,6], [7,8,9,10,11,12]])** |

# Manipulating Array Content

### Task 1: Concatenate arrays (SUBMISSION REQUIRED)

<https://docs.scipy.org/doc/numpy/reference/generated/numpy.concatenate.html>

|  |
| --- |
| 1. Write Python code to concatenate the following arrays on the **x-axis**   **a = np.array([[ 1,2,3],[ 4,5,6],[7,8,9]])**  **b = np.full((3,3),1.5)**  **c = np.arange(0,15).reshape(5,3)** |
| 1. Write Python code to concatenate the following arrays on the **y-axis**   **a = np.array([[ 1,2,3,4],[ 4,5,6,7],[7,8,9,10],[11,12,13,14]])**  **b = np.random.randint(100,200,(4,6))**  **c = np.arange(0,40).reshape(4,10)** |

### Task 2: Split arrays

<https://docs.scipy.org/doc/numpy/reference/generated/numpy.split.html#numpy.split>

|  |
| --- |
| 1. Write Python code to split the following array into 20 sub-arrays   **x = np.arange(100)** |
| 1. Write Python code to split the following array at these intervals: 10, 25, 45, 75, 95   **x = np.arange(100)** |

# Copying arrays

### Task 1: Make a copy of a Numpy array

|  |
| --- |
| 1. What is the difference between ***y*** and ***z***?   **x = np.array([1, 2, 3])**  **y = x**  **z = np.copy(x)** |
| 1. What is the value of ***x***, ***y*** and ***z*** after executing Line 4?  |  |  | | --- | --- | | **Line** |  | | **1** | **x = np.array([1, 2, 3])** | | **2** | **y = x** | | **3** | **z = np.copy(x)** | | **4** | **x[0] = 10** | |

# Sorting arrays

### Task 1: Basic sorting (SUBMISSION REQUIRED)

|  |
| --- |
| 1. Write Python code to sort the following 1-d array using the sort method   **arr\_1 = np.random.randint(100,200,10)**  Your output should be similar to that below. |
| 1. Write Python code to sort the following 2-d array by columns   **arr\_2 = np.random.randint(1,20,(3,5))**  Your output should be similar to that below. |
| 1. Write Python code to sort the following 2-d array by rows *without affecting the original* .   **arr\_3 = np.random.randint(100,200,(2,5))** |

# Subsetting and Indexing

### Task 1: Basic subsetting

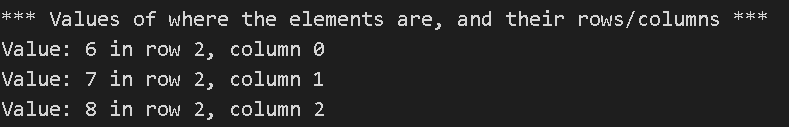
|  |
| --- |
| Given the following two Numpy arrays, answer the questions from (a) to (g) below.  **a = np.arange(1,100,1)**  **b = np.array([[1,2,3,4], [100,200,300,400], [5,6,7,8], [9,10,11,12] ])** |
| 1. What is the output of a[3] and b[3]? What is the datatype of each output? |
| 1. Write Python code to print out 2nd and third rows of the array b (ie [100,200,300,400] and [5,6,7,8] |
| 1. Write Python code to print out all the rows of the last column of array b |
| 1. Write Python code to print out row 2 (i.e.[ 100,200,300,400], first and last columns of array b |
| 1. Write Python code to print out row 3 to end, column 2 to end of the array b |
| 1. Write Python code to print out elements from index 10 to 20 with step 2 from array a |
| 1. Write Python code to select and reverse elements in array a from index 0 to the end, then print it out. |

### Task 2: Boolean indexing (SUBMISSION REQUIRED)

|  |
| --- |
| Given the following Numpy array, answer the questions from (a) to (b) below.  **a = np.array((np.arange(0,10),  np.arange(10,20),  np.arange(20,30),  np.arange(30,40)))**  **b = np.random.randint(100,200,(3,3))** |
| 1. Write Python code to return all the even numbers in **array a** 2. Write Python code to return all the numbers that are greater than 150 in **array b** |

### Task 3: Indexing using WHERE

|  |
| --- |
| Given the following Numpy array, print out the values and the rows and indexes where the element is exactly 5.  b = np.arange(9.).reshape(3, 3)  b5 = np.where( b > 5 )  Your output should look similar to that below: |



# Array Math

### Task 1: Add, subtract, multiply and divide numpy arrays

|  |
| --- |
| Given the following Numpy arrays, answer the questions from (a) to (g) below.  **a = np.arange(100,200,1).reshape(20,5)**  **b = np.arange(1,2000,20).reshape(10,10)**  **c = np.array([ [1,2,3], [4,5,6]])** |
| 1. What happens when you try to perform the following operations on array a and array b?   **print(a+b)** |
| 1. Create an array ***a2*** that has the same shape as array ***a***, containing random numbers between 0 and 100. Print the contents of the resulting array when ***a*** and ***a2*** are added together. |
| 1. Extract a subset of array ***b*** into an array variable ***b1***, such that you can perform a subtraction operation of (***b1 – c)*** successfully. Print out the resulting contents of the array of this operation. |
|  |
| 1. Create an array ***c2*** such that ***c2*** contains the values of array c multiplied by 10. |

# Statistical methods

### Task 1: sum, mean (SUBMISSION REQUIRED)

|  |
| --- |
| **a = np.array((np.arange(0,10),  np.arange(10,20),  np.arange(20,30),  np.arange(30,40)))**   1. What is the sum of all the numbers in array a? 2. What is their mean? 3. What is the sum of all the numbers in each row?   Using the Numpy statistical functions **sum()** and **mean()**, compute the information for array a as specified above. Your output should resemble the screenshot shown below. |

### Task 2: std, var

|  |
| --- |
| **a = np.arange(100,200,0.5)**  **b = np.arange(100,200,0.1)** |
| 1. What is the standard deviation of arrays a and b? |
| 1. What is the variance of arrays a and b? |
| Using the Numpy statistical functions **std()** and **var()**, compute the information for array a as specified. Your output should resemble the screenshot shown below. |

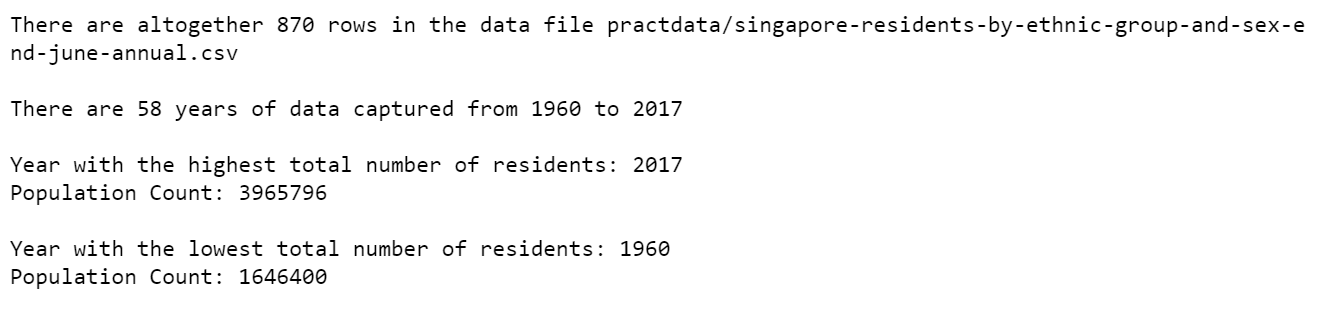
### Task 3: min/ max, argmin/ argmax

|  |
| --- |
| **a = np.random.randint(1,1000,(3,4))** |
| Print out the following information:   * maximum value of each row in array a * maximum value of the entire array a * row and column that has the number with the maximum value in array a |
| * minimum value of each column in array a * minimum value of the entire array a * row and column that has the number with the minimum value in array a |
| Your outputs for parts (a) and (b) should resemble that of the screenshot below. |

# File I/O on Numpy arrays

### Task 1: Load a numpy array from a textfile (SUBMISSION REQUIRED)

Complete the following code that uses the Numpy **genfromtxt** method to read the contents of an external csv file and display pertinent information about the data inside the file as shown on the following sample screen.



|  |
| --- |
| import numpy as np  **### Read the csv file using genfromtxt**  filename = "singapore-residents-by-ethnic-group-and-sex-end-june-annual.csv"  data = np.genfromtxt(filename, dtype=['i2', 'U50', 'i8'],  delimiter=',', names=True)  #print(data)  **### Print out total rows of data in the file**  print(f'There are altogether {\_\_\_\_\_} rows in the data file {\_\_\_\_\_\_}')  print()  **### Print out the number of years of data captured**  data\_years = data['year']  years = np.unique(data\_years)  print(f'There are {\_\_\_\_\_\_} years of data captured from {\_\_\_\_\_\_\_} to {\_\_\_\_\_\_\_\_}')  print()  **## Extract only the rows with Total Residents" - using boolean indexing**  keyword = 'Total Residents'  data\_totalresidents = data[data['level\_1'] == keyword]  #print(data\_totalresidents)  **### Print out the years which has the highest total number of residents**  max\_residents = data\_totalresidents['value'].max()  argmax\_residents = data\_totalresidents['value'].argmax()  print(f'Year with the highest total number of residents: { data\_totalresidents[argmax]['year']}')  print(f'Population Count: { max\_residents } ')  print()  ### Print out the years which has the lowest total number of residents  min\_residents = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  argmin\_residents = \_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_  print(f'Year with the lowest total number of residents: {\_\_\_\_\_\_\_\_\_\_\_}')  print(f'Population Count: { min\_residents } ')  print() |

**-- End of Lab --**