CS506 Programming for Computing

HOS06B– Getting Started with NumPy

06/06/2020 Created by Apiwat Chuaphan

3/28/2022 Revised by Cedric Huang

07/27/2023 Reviewed by Maram Elwakeel

School of Technology & Computing (STC) @ City University of Seattle (CityU)

Before You Start

- Version numbers may not match the most current version at the time of writing. If given the option to choose between the stable release (long-term support) or the most recent, please select the stable release rather than the beta-testing version.
- There might be subtle discrepancies along with the steps. Please use your best judgment while going through this cookbook-style tutorial to complete each step.
- For your working directory, use your course number. This tutorial may use a different course number as an example.
- All the steps and concepts in this tutorial are from the textbook, so if you encounter problems in this tutorial, please try to read and compare the textbook to solve the problem. If you still can't solve the problem, please feel free to contact your course TA.
- Avoid copy-pasting code from the book or the GitHub repository. Instead, type out the code yourself. Resort to copy-pasting only when you are stuck and find that things are not working as expected.

Learning Outcomes

- Understand array object in NumPy
- Understand array manipulation using NumPy's functions

Resources

- NumPy Documentation: https://numpy.org/doc/stable/
- Stanford University. (2020). CS231n: Convolutional Neural Networks for Visual Recognition: NumPy.

Section 1 - NumPy

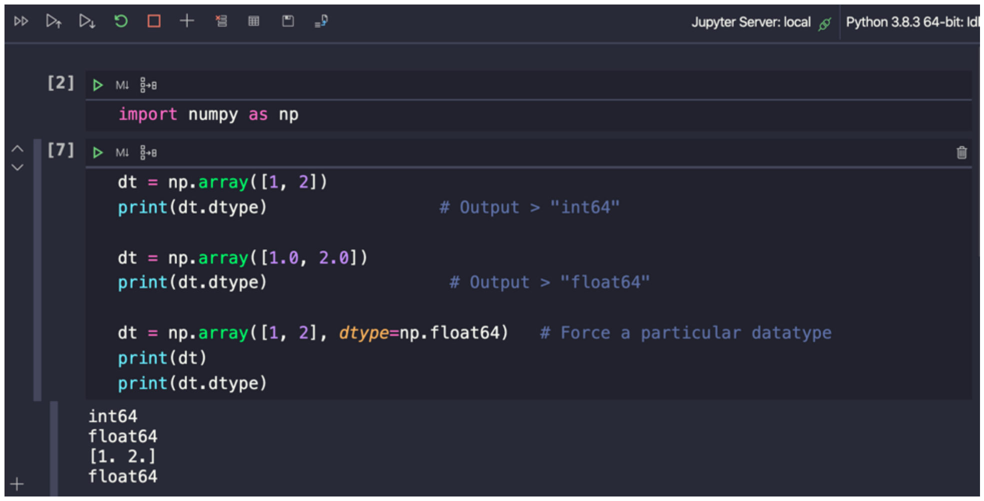
1) Which stands for Numerical Python, is an open-source and the core library for scientific computing in Python. It provides a high-performance multidimensional array object and tools for working with these arrays.
2) NumPy is memory efficient, meaning it can handle a vast amount of data more accessible than any other library. We'll cover the following in this module:
   i. Datatypes
   ii. Arrays
3) Before we begin, let's install NumPy module if you have not done that. Run this command in terminal:
   i. `pip install numpy`
4) Check the version to make sure we have it installed, run:
   i. `pip list | grep numpy`
5) In Visual Studio Code, open the private repository generated when you accepted the HOS06 assignment (If you cannot find that repository in your machine, you might have not cloned the repo, if so, please do so before proceeding).
6) **Datatypes** - The object defined in NumPy is an N-dimensional array type called ndarray, describing the collection of items of the same type. An element in ndarray is an object of data-type object called dtype.
   i. A **dtype** described in the following aspects:
      - Type of data (integer, float, Python object, etc.)
      - Size of the data (how many bytes is in e.g., the integer)

- Byte order of the data (little-endian or big-endian)
- Structured data type (*e.g.*, describing an array item consisting of an integer and a float)

7) Open Jupyter Notebook:

   i. Under module folder, create a new file called `datatypes.ipynb` and simply click on the file to open notebook.

   ii. Type the following to see how NumPy says about data types. Run selected cell to see result for that cell.

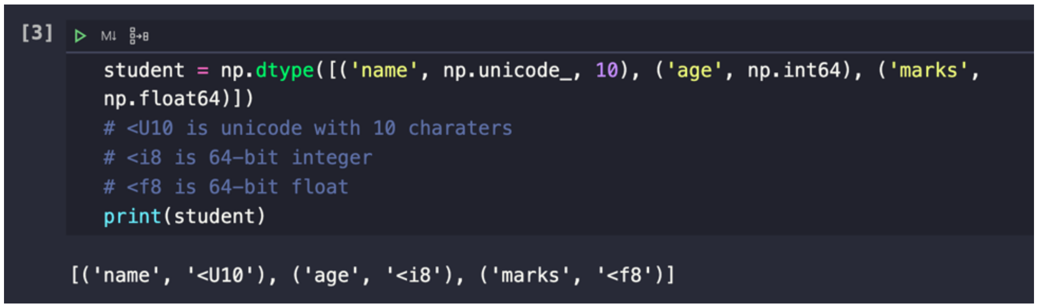


```
import numpy as np
```

```
dt = np.array([1, 2])
print(dt.dtype)                    # Output > "int64"

dt = np.array([1.0, 2.0])
print(dt.dtype)                     # Output > "float64"

dt = np.array([1, 2], dtype=np.float64)   # Force a particular datatype
print(dt)
print(dt.dtype)
```

```
int64
float64
[1. 2.]
float64
```

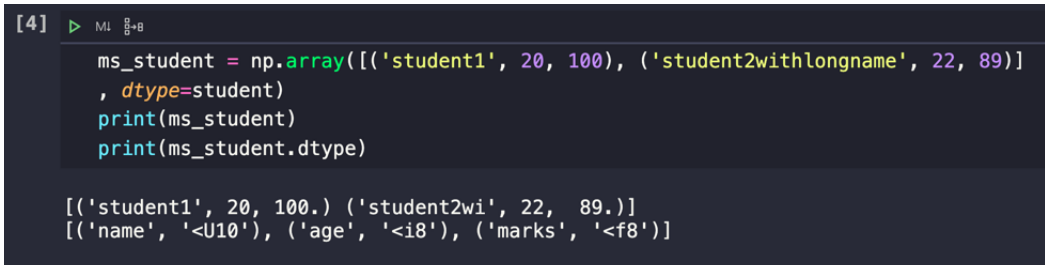
   iii. We also can define a structured data type. For example, we will define a data type named student with the 3 fields: name as string, age as integer, points as float.



```
student = np.dtype([('name', np.unicode_, 10), ('age', np.int64), ('marks',
np.float64)])
# <U10 is unicode with 10 charaters
# <i8 is 64-bit integer
# <f8 is 64-bit float
print(student)
```

```
[('name', '<U10'), ('age', '<i8'), ('marks', '<f8')]
```

   iv. Let's apply to ndarray object.



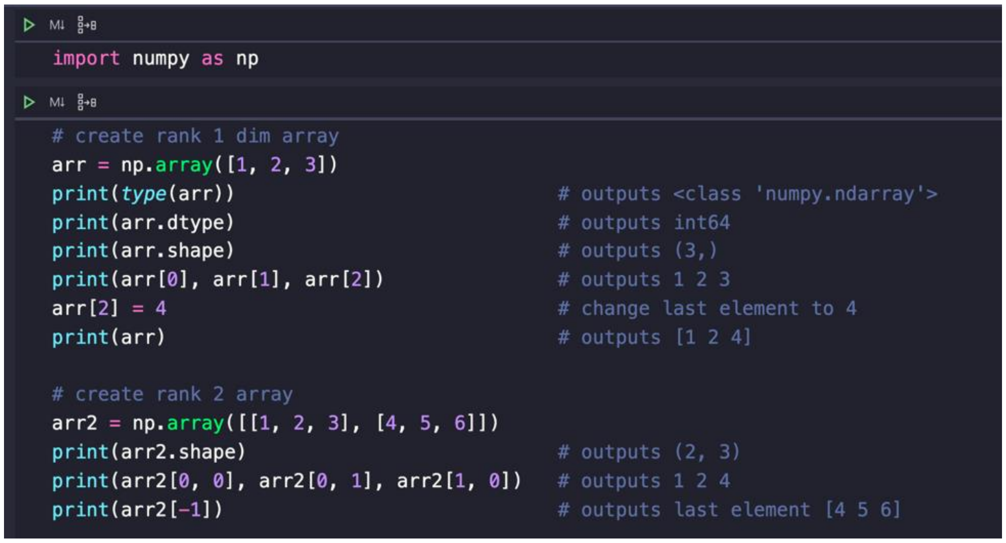
```
[4]
    ms_student = np.array([('student1', 20, 100), ('student2withlongname', 22, 89)]
    , dtype=student)
    print(ms_student)
    print(ms_student.dtype)

[('student1', 20, 100.) ('student2wi', 22,  89.)]
[('name', '<U10'), ('age', '<i8'), ('marks', '<f8')]
```

As you can see here, “*student2withlongname”* has exceeded the string length, only 10 characters acceptance was defined.

8) **Arrays -** A numpy array is a grid of values like Python lists, but all the same type, and is indexed by a tuple of nonnegative integers. The rank and shape are the number of dimensions and a tuple of integers of the size of the array, respectively.

   i. Under module folder, create a new file called `arrays.ipynb` and simply click on the file to open notebook.

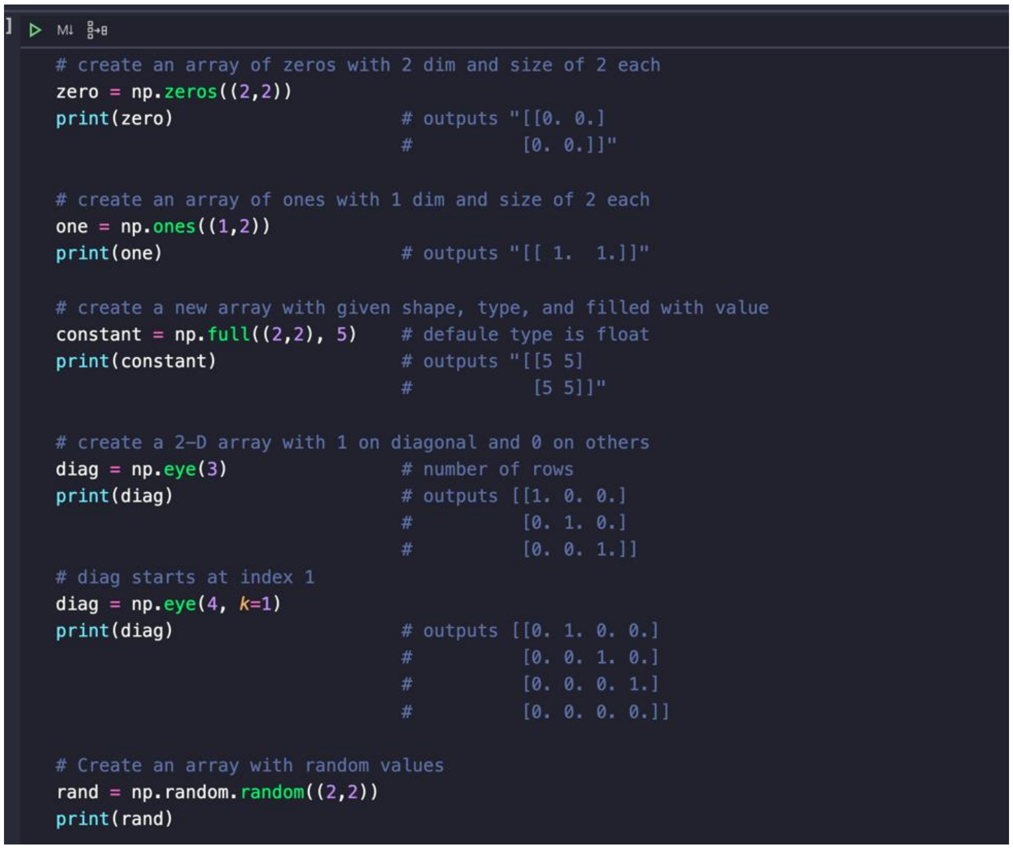
   ii. Type the following to create numpy arrays. Accessing items in array is similar to Python list.



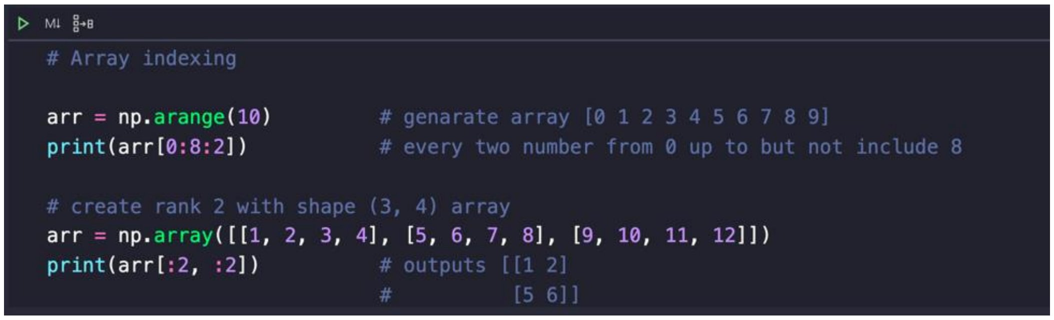
```
import numpy as np
```

```
# create rank 1 dim array
arr = np.array([1, 2, 3])
print(type(arr))                             # outputs <class 'numpy.ndarray'>
print(arr.dtype)                             # outputs int64
print(arr.shape)                             # outputs (3,)
print(arr[0], arr[1], arr[2])                # outputs 1 2 3
arr[2] = 4                                   # change last element to 4
print(arr)                                   # outputs [1 2 4]

# create rank 2 array
arr2 = np.array([[1, 2, 3], [4, 5, 6]])
print(arr2.shape)                            # outputs (2, 3)
print(arr2[0, 0], arr2[0, 1], arr2[1, 0])    # outputs 1 2 4
print(arr2[-1])                              # outputs last element [4 5 6]
```

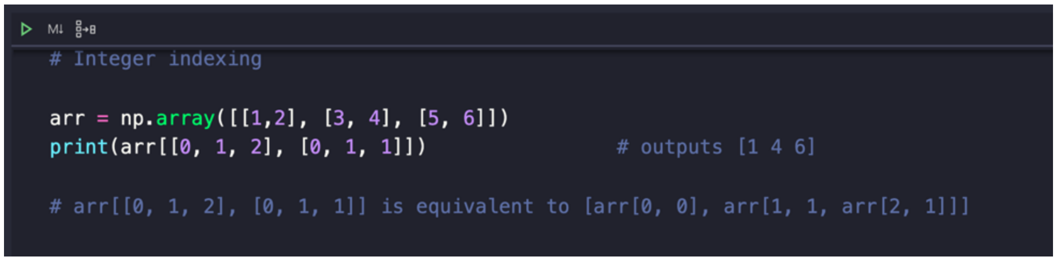
   iii. Stay in same file, type to following functions from numpy to generate arrays in a different way. You will see how it can make your life easier with array.


```
# create an array of zeros with 2 dim and size of 2 each
zero = np.zeros((2,2))
print(zero)                      # outputs "[[0. 0.]
                                 #          [0. 0.]]"

# create an array of ones with 1 dim and size of 2 each
one = np.ones((1,2))
print(one)                       # outputs "[[ 1.  1.]]"

# create a new array with given shape, type, and filled with value
constant = np.full((2,2), 5)     # defaule type is float
print(constant)                  # outputs "[[5 5]
                                 #           [5 5]]"

# create a 2-D array with 1 on diagonal and 0 on others
diag = np.eye(3)                 # number of rows
print(diag)                      # outputs [[1. 0. 0.]
                                 #          [0. 1. 0.]
                                 #          [0. 0. 1.]]
# diag starts at index 1
diag = np.eye(4, k=1)
print(diag)                      # outputs [[0. 1. 0. 0.]
                                 #          [0. 0. 1. 0.]
                                 #          [0. 0. 0. 1.]
                                 #          [0. 0. 0. 0.]]

# Create an array with random values
rand = np.random.random((2,2))
print(rand)
```

iv. Indexing and slicing in NumPy array is similar to Lists in Python. Try the following into a new cell.


```
# Array indexing

arr = np.arange(10)         # genarate array [0 1 2 3 4 5 6 7 8 9]
print(arr[0:8:2])           # every two number from 0 up to but not include 8

# create rank 2 with shape (3, 4) array
arr = np.array([[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]])
print(arr[:2, :2])          # outputs [[1 2]
                            #          [5 6]]
```

v. In NumPy, you can do integer indexing which helps in selecting any arbitrary item in an array using the data from another array like the example below. Add this to new cell.


```
# Integer indexing

arr = np.array([[1,2], [3, 4], [5, 6]])
print(arr[[0, 1, 2], [0, 1, 1]])              # outputs [1 4 6]

# arr[[0, 1, 2], [0, 1, 1]] is equivalent to [arr[0, 0], arr[1, 1, arr[2, 1]]]
```

9) Save your Jupyter Notebook with all Output