Introduction to Computer Programming Lecture 7.1:

Importing Packages: Numpy

User Input

Hemma Philamore
Department of Engineering Mathematics

Import

Import modules written by developers using import keyword

NumPy ("Numerical Python"): scientific computing applications

- Fourier Transform
- Shape Manipulation
- Mathematical and Logical Operations
- Linear Algebra
- Random Number Generation

Advantages

- Fast (uses C and Python)
- Other libraries (data science, machine learning…) based on Numpy
- Extensive built-in functionality

Elementwise Operation

Pure Python

```
1  x = [2, 3, 4, 5, 6]
2  y = [a + 2 for a in x]
3  print(y)
```

```
[4, 5, 6, 7, 8]
```

Simpler way to add 2 to each element

Import numpy module (Often renamed as np)

Numpy

```
1  import numpy as np
2
3  x = np.array([2, 3, 4, 5, 6])
4  y = x + 2
5  print(y)
```

```
[4 5 6 7 8]
```

New type of data structure

Numpy operations act **elementwise**, when applied to a numpy array

Numpy Array

Create 1D array from list

```
1  x = [2, 3, 4, 5, 6]
2  nums = np.array(x)
3  print(nums)
```

```
[2 3 4 5 6]
```

Create 2D array from list of lists

```
1  nums = np.array([[2,4,6], [8,10,12], [14,16,18], [14,16,18]])
2  print(nums)
```

```
[[ 2  4  6]
 [ 8 10 12]
 [14 16 18]
 [14 16 18]]
```

Every inner list becomes a row.
Number of columns is equal to the number of elements in each inner list.

Create nD array from lists of lists

```
1  nums = np.array([[[2,4,6], [8,10,12], [14,16,18]],
2                   [[3,4,8], [2,2,2],   [9,8,1]]])
3  print(nums)
```

```
[[[ 2  4  6]
  [ 8 10 12]
  [14 16 18]]

 [[ 3  4  8]
  [ 2  2  2]
  [ 9  8  1]]]
```

Indexing

1D

```
1  print(nums[3])
```

```
5
```

2D

row

```
1  print(nums[0, 1])
```

column

```
4
```

nD

matrix

row

```
1  print(nums[0, 1])
2  print(nums[0, 1, 2])
```

column

```
[ 8 10 12]
12
```

Numpy Array

Create 1D array from list

```
x = [2, 3, 4, 5, 6]
nums = np.array(x)
print(nums)
```

```
[2 3 4 5 6]
```

Create 2D array from list of lists

```
nums = np.array([[2,4,6], [8,10,12], [14,16,18], [14,16,18]])
print(nums)
```

```
[[ 2  4  6]
 [ 8 10 12]
 [14 16 18]
 [14 16 18]]
```

Create nD array from lists of lists

```
nums = np.array([[[2,4,6], [8,10,12], [14,16,18]],
                 [[3,4,8], [2,2,2],   [9,8,1]]])
print(nums)
```

```
[[[ 2  4  6]
  [ 8 10 12]
  [14 16 18]]

 [[ 3  4  8]
  [ 2  2  2]
  [ 9  8  1]]]
```

Indexing

1D

```
print(nums[3])
```

```
5
```

range

```
print(nums[3:5])
```

```
[5 6]
```

2D

```
print(nums[0, 1])
```

```
4
```

Inner brackets are column

Up to column 1

```
print(nums[1:2, :1])
```

```
[[ 8 10]]
```

nD

```
print(nums[0, 1:])
```

```
[[ 8 10 12]
 [14 16 18]]
```

All rows

```
print(nums[0, :, 1:])
```

```
[[ 4  6]
 [10 12]
 [16 18]]
```

Column 1 to end

Numpy Array

arange

```
1  x = np.arange(1, 10)
2  print(x)
```

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

```
1  x = np.arange(1, 10, 2)
2  print(x)
```

```
[1 3 5 7 9]
```

Step size of 2

zeros

```
1  z = np.zeros(5)
2  print(z)
```

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

```
1  z = np.zeros((5, 2))
2  print(z)
```

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

tuple

ones

```
1  y = np.ones((2, 2))
2  print(y)
```

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

linspace

```
1  u = np.linspace(1, 10, 10)
2  print(u)
```

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

Number of equally-spaced elements

eye

```
1  i = np.eye(4)
2  print(i)
```

```
[[1. 0. 0. 0.]
 [0. 1. 0. 0.]
 [0. 0. 1. 0.]
 [0. 0. 0. 1.]]
```

<u>Identity matrix:</u>
Square matrix with zeros across rows and columns except the diagonal

Re-shaping

```
1 x = np.arange(0, 20)
2 print(x)
```

```
[ 0  1  2  3  4  5  6  7  8  9 10 11 12 13 14 15 16 17 18 19]
```

2D array, 4 rows, 5 columns

```
1 x = x.reshape(4, 5)
2 print(x)
```

```
[[ 0  1  2  3  4]
 [ 5  6  7  8  9]
 [10 11 12 13 14]
 [15 16 17 18 19]]
```

```
1 x = x.reshape(4, 4)
2 print(x)
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
<ipython-input-41-6abf089cf44a> in <module>
----> 1 x = x.reshape(4, 4)
      2 print(x)

ValueError: cannot reshape array of size 20 into shape (4,4)
```

Number of elements in the 1D array not equal to the number of elements in reshaped array (e.g. product of rows and columns)