UCLA College | Social Sciences Economics

Day 6: We ❤️ NumPy

Paul Schumacher, MSc Quantitative Economics

Announcements

Attendance

Midterm Feedback

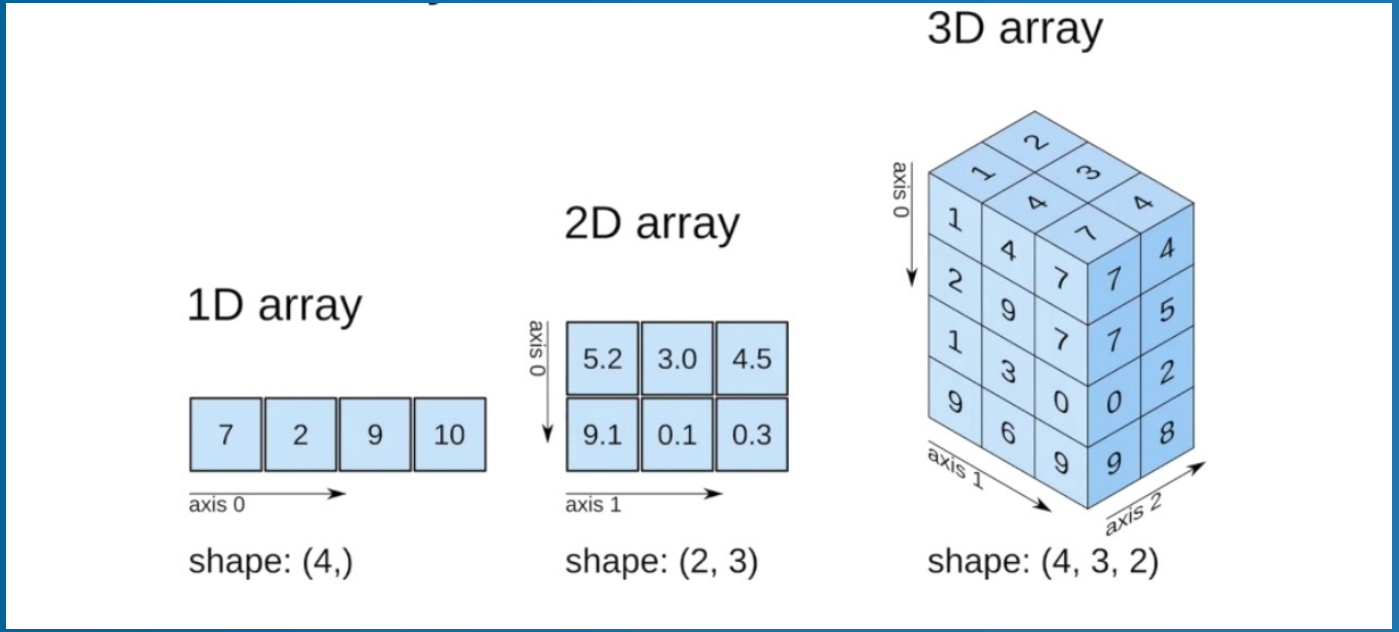
HW1 due Tonight at 11pm

Let’s play Skyfall

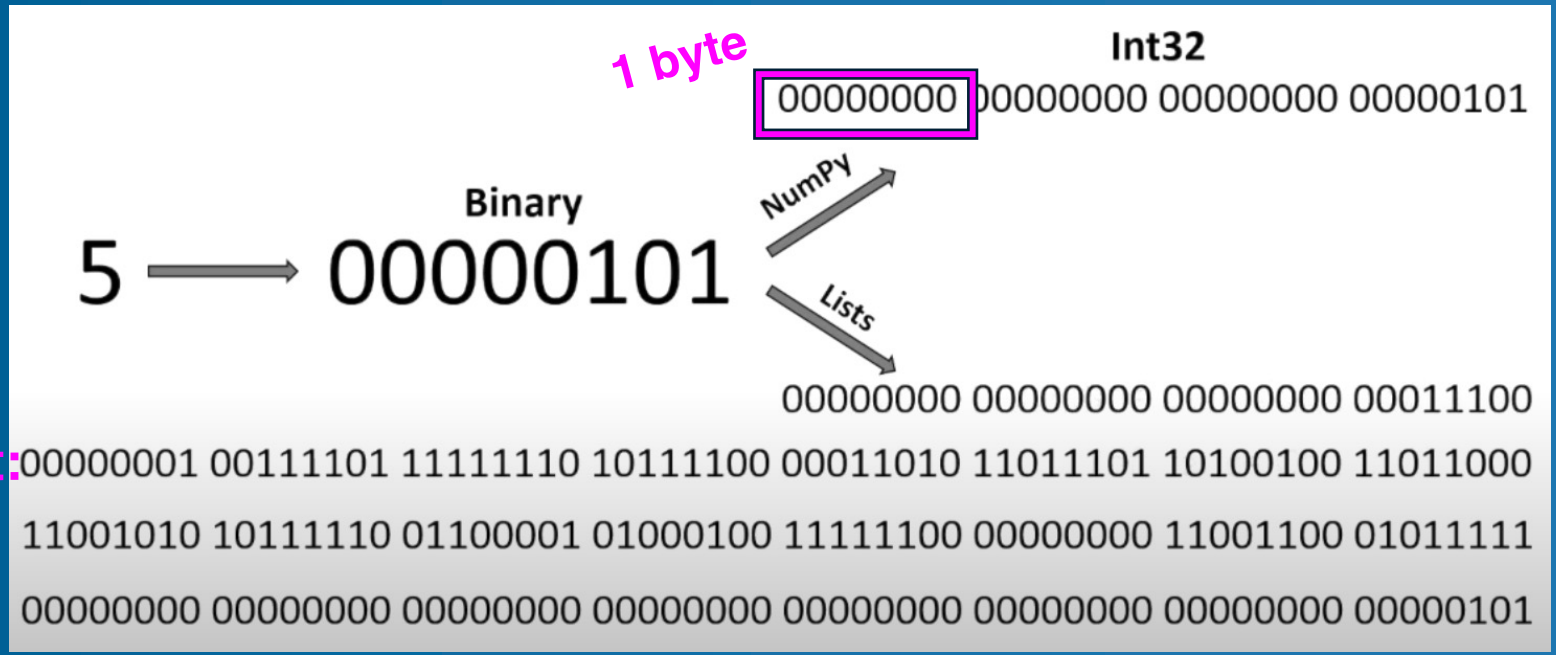
NumPy: Multidimensional *Array* Library

What is an Array?
Collection/ Store of elements:

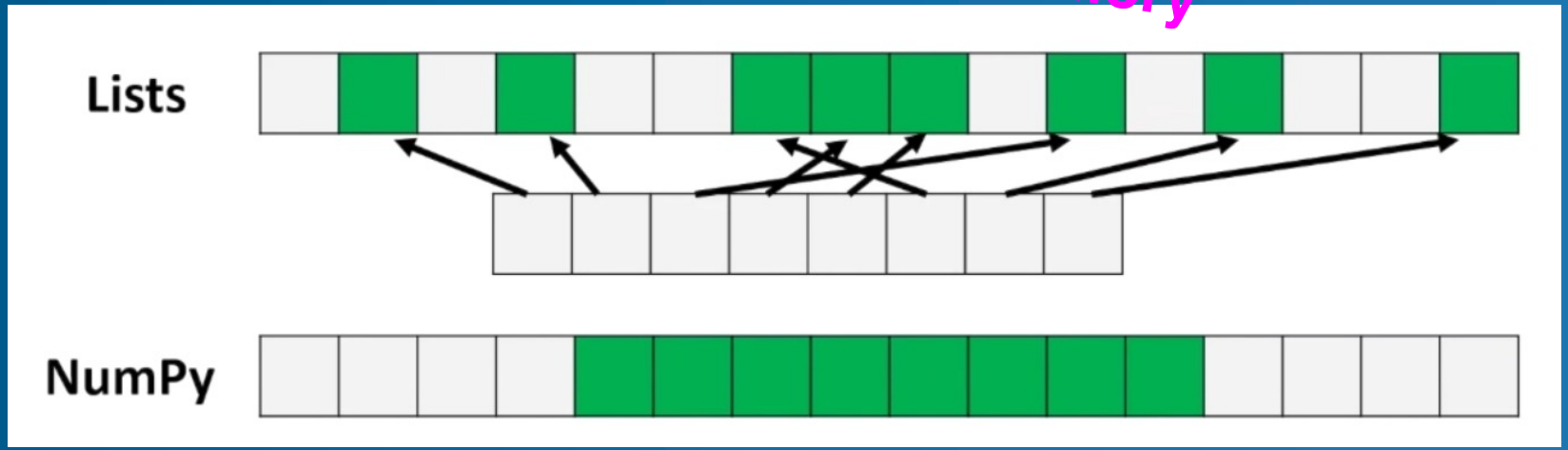
- same data type
- same memory size
- indices



NumPy: NumPy (fast) vs. Lists (slow)



NumPy: NumPy (fast) vs. Lists (slow)



NumPy: NumPy (fast) vs. Lists (slow)

Both are mutable!

```
import numpy as np
v1 = [1,2,3]
v2 = [4,5,6]
v1+v2
np.array(v1) + np.array(v2)
```

```
import numpy as np
v1 = [1,2,3]
v2 = [4,5,6]
```

More applications with NumPy!

```
# Concatenates the two arrays
v1+v2
```

```
[1, 2, 3, 4, 5, 6]
```

```
# Adds the arrays elementwise
np.array(v1) + np.array(v2)
```

```
array([5, 7, 9])
```

NumPy: Basics (creating arrays, shape, size, data type)

```
a = np.array([1,2,3], dtype='int32')
print(a)
```

```
[1 2 3]
```

```
b = np.array([[9.0,8.0,7.0],[6.0,5.0,4.0]])
print(b)
```

```
[[9. 8. 7.]
 [6. 5. 4.]]
```

```
# Get Dimension
a.ndim
```

```
1
```

```
# Get Shape
b.shape
```

```
(2, 3)
```

```
# Get Type
a.dtype
```

```
dtype('int32')
```

```
# Get Size
a.itemsize
```

```
4
```

```
# Get total size
a.nbytes
```

```
12
```

```
# Get number of elements
a.size
```

```
3
```

NumPy: Accessing/Changing Specific Elements, Rows, Columns, etc. (slicing)

```
a = np.array([[1,2,3,4,5,6,7],[8,9,10,11,12,13,14]])
print(a)
```

```
[[ 1  2  3  4  5  6  7]
 [ 8  9 10 11 12 13 14]]
```

```
# Get a specific element [r, c]
a[1, 5]
```

```
13
```

```
# Get a specific row
a[0, :]
```

```
array([1, 2, 3, 4, 5, 6, 7])
```

```
# Get a specific column
a[:, 2]
```

```
array([ 3, 10])
```

```
# Getting a little more fancy [startindex:endindex:stepsize]
a[0, 1:-1:2]
```

```
array([2, 4, 6])
```

```
a[1,5] = 20

a[:,2] = [1,2]
print(a)
```

```
[[ 1  2  5  4  5  6  7]
 [ 8  9  5 11 12 20 14]]
[[ 1  2  1  4  5  6  7]
 [ 8  9  2 11 12 20 14]]
```

NumPy: Accessing/Changing Specific Elements, Rows, Columns, etc. (slicing)

*3-d example

```
b = np.array([[[1,2],[3,4]],[[5,6],[7,8]]])
print(b)
```

```
[[[1 2]
  [3 4]]

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

```
# Get specific element (work outside in)
b[0,1,1]
```

```
4
```

```
# replace
b[:,1,:] = [[9,9,9],[8,8]]
```

```
---------------------------------------------------------------
ValueError                                Traceback (most
<ipython-input-34-db1aebb5daad> in <module>()
      1 # replace
----> 2 b[:,1,:] = [[9,9,9],[8,8]]

ValueError: setting an array element with a sequence.
```

```
b
```

```
array([[[1, 2],
        [9, 9]],

       [[5, 6],
        [8, 8]]])
```

NumPy: Initializing Different Arrays (1s, 0s, full, random, etc...)

```
# All 0s matrix
np.zeros((2,3))
```

```
array([[0., 0., 0.],
       [0., 0., 0.]])
```

```
# All 1s matrix
np.ones((4,2,2), dtype='int32')
```

```
array([[[1, 1],
        [1, 1]],

       [[1, 1],
        [1, 1]],

       [[1, 1],
        [1, 1]],

       [[1, 1],
        [1, 1]]])
```

```
# Any other number
np.full((2,2), 99)
```

```
array([[99., 99.],
       [99., 99.]], dtype=float32)
```

Exam Question

```
# This one is better for initializing
#because the contents are actually set to 0
a1 = np.zeros((3,3))
a1
```

```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

```
# The contents are not always all set to 0
a2 = np.empty((3,3))
a2
```

```
array([[0., 0., 0.],
       [0., 0., 0.],
       [0., 0., 0.]])
```

NumPy: Initializing Different Arrays (1s, 0s, full, random, etc...)

```
# Any other number (full_like)
np.full_like(a, 4)
```

```
array([[4, 4, 4, 4, 4, 4, 4],
       [4, 4, 4, 4, 4, 4, 4]])
```

```
# Random decimal numbers
np.random.rand(4,2)
```

```
array([[0.71965344, 0.32931871],
       [0.1605404 , 0.41693097],
       [0.03546035, 0.70618637],
       [0.09444053, 0.61729817]])
```

```
# Random Integer values
np.random.randint(-4,8, size=(3,3))
```

```
array([[-2, -4, -4],
       [ 6,  6,  3],
       [ 3,  2,  2]])
```

```
# The identity matrix
np.identity(5)
```

```
array([[1., 0., 0., 0., 0.],
       [0., 1., 0., 0., 0.],
       [0., 0., 1., 0., 0.],
       [0., 0., 0., 1., 0.],
       [0., 0., 0., 0., 1.]])
```

```
# Repeat an array
arr = np.array([[1,2,3]])
r1 = np.repeat(arr,3, axis=0)
print(r1)
```

```
[[1 2 3]
 [1 2 3]
 [1 2 3]]
```

NumPy: Be careful when copying arrays!!!

```
a = np.array([1,2,3])
b = a
b[0] = 100

print(a)
```

```
[100   2   3]
```

```
a = np.array([1,2,3])
b = a.copy()
b[0] = 100

print(a)
```

```
[1 2 3]
```

NumPy: Basic Mathematics

```
a = np.array([1,2,3,4])
print(a)
```

```
[1 2 3 4]
```

```
a + 2
```

```
array([5, 6, 7, 8])
```

```
a - 2
```

```
array([-1,  0,  1,  2])
```

```
a * 2
```

```
array([2, 4, 6, 8])
```

```
a / 2
```

```
array([0.5, 1. , 1.5, 2. ])
```

```
b = np.array([1,0,1,0])
a + b
```

```
array([1, 0, 3, 0])
```

```
a ** 2
```

```
array([ 1,  4,  9, 16], dtype=int32)
```

```
# Take the sin
np.cos(a)
```

```
array([ 0.54030231, -0.41614684, -0.9899925 , -0.65364362])
```

NumPy: Linear Algebra

```
a = np.ones((2,3))
print(a)

b = np.full((3,2), 2)
print(b)

np.matmul(a,b)
```

```
[[1. 1. 1.]
 [1. 1. 1.]]
[[2 2]
 [2 2]
 [2 2]]

array([[6., 6.],
       [6., 6.]])
```

```
# Find the determinant
c = np.identity(3)
np.linalg.det(c)
```

```
1.0
```

https://numpy.org/doc/stable/reference/routines.linalg.html

NumPy: Statistics

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

```
array([[1, 2, 3],
       [4, 5, 6]])
```

```
np.min(stats)
```

```
1
```

```
np.max(stats, axis=1)
```

```
array([3, 6])
```

```
np.sum(stats, axis=0)
```

```
array([5, 7, 9])
```

NumPy: Reorganizing Arrays (reshape, vstack, hstack)

```
before = np.array([[1,2,3,4],[5,6,7,8]])
print(before)

after = before.reshape((1,8))
print(after)
```

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

```
before = np.array([[1,2,3,4],[5,6,7,8]])
print(before)

after = before.reshape((2,2,2))
print(after)
```

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

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

```
# Vertically stacking vectors
v1 = np.array([1,2,3,4])
v2 = np.array([5,6,7,8])

np.vstack([v1,v2,v1,v2])
```

```
array([[1, 2, 3, 4],
       [5, 6, 7, 8],
       [1, 2, 3, 4],
       [5, 6, 7, 8]])
```

```
# Horizontal  stack
h1 = np.ones((2,4))
h2 = np.zeros((2,2))

np.hstack((h1,h2))
```

```
array([[1., 1., 1., 1., 0., 0.],
       [1., 1., 1., 1., 0., 0.]])
```

Class exercise: Exploratory Data Analysis (EDA), File Operation

- “PLF Day 6 Worksheet NumPy.ipynb”
- Breakout rooms
- Time: 20 min

Any Questions?