Arrays

Thomas Schwarz, SJ

Example

- Python lists are internally organized as arrays of object pointers
- Python sets are internally organized as hash-tables
- Experiment:
  - Create a set and a list with the same 100,000 random elements
  - Then search for the first 100,000 elements in the set and the list
- Result: List is much slower (timing in sec)
  - array: 126.1766300201416
  - set: 0.0066390037536621O (20,000 times faster)

Array ADT

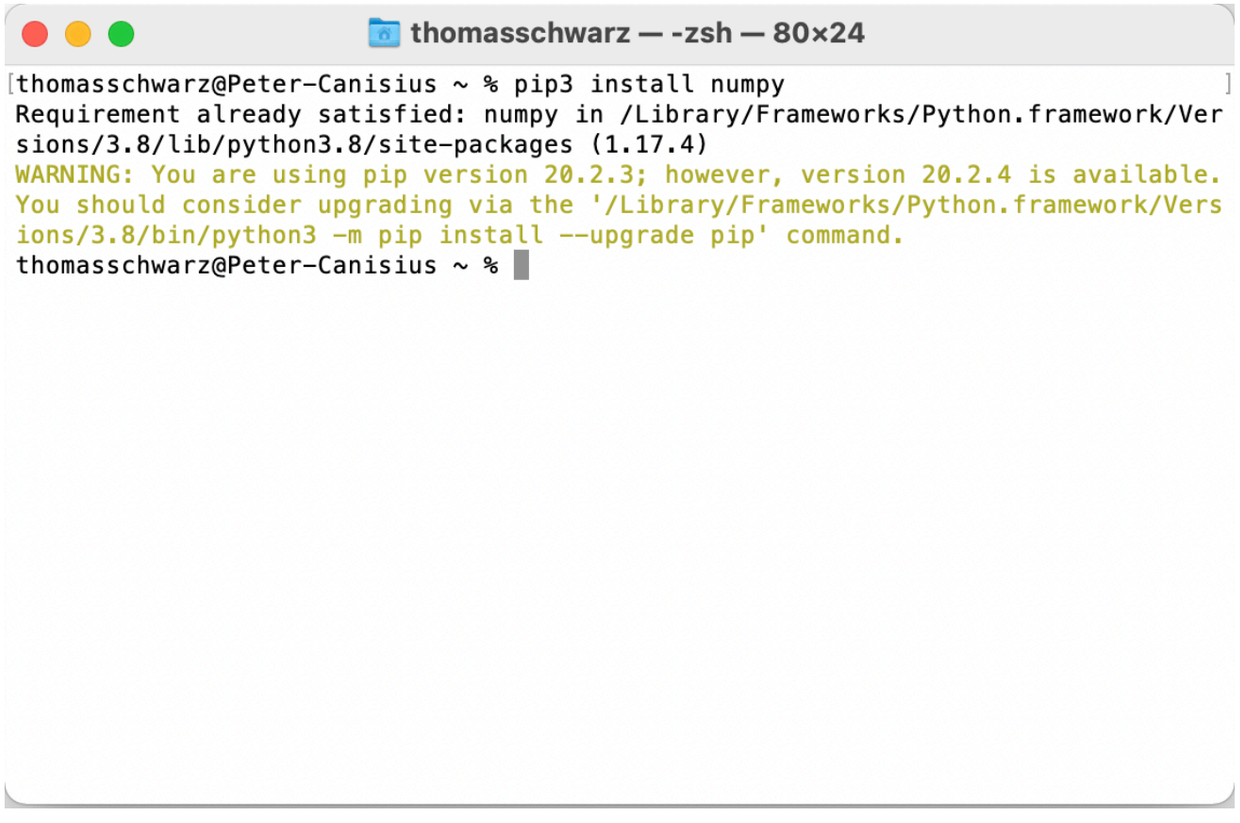
- Classic array:
  - Collection of objects of the same type
  - Accessed by index
- Python implementation:
  - Python has lists, which internally are organized as pointers to objects
- NumPy has a better performing array

NumPy Fundamentals

- Numpy is a module for faster vector processing with numerous other routines
- Scipy is a more extensive module that also includes many other functionalities such as machine learning and statistics

NumPy Fundamentals

- Install numpy with pip3
  - If pip3 does not show up in your terminal window, then you did not set the Python path correctly



NumPy Fundamentals

- Why Numpy?
  - Remember that Python does not limit lists to just elements of a single class
  - If we have a large list $[a_1, a_2, a_3, \ldots, a_n]$ and we want to add a number to all of the elements, then Python will asks for each element:
    - What is the type of the element
    - Does the type support the + operation
    - Look up the code for the + and execute
- This is slow

NumPy Fundamentals

- Why Numpy?
  - Primary feature of Numpy are arrays:
    - List like structure where all the elements have the same type
      - Usually a floating point type
  - Can calculate with arrays much faster than with list
  - Implemented in C / Java for Cython or Jython

NumPy Arrays

- NumPy Arrays are containers for numerical values
- Numpy arrays have dimensions
  - Vectors: one-dimensional
  - Matrices: two-dimensional
  - Tensors: more dimensions, but much more rarely used
- Nota bene: A matrix can have a single row and a single column, but has still two dimensions

NumPy Arrays

- After installing, try out `import numpy as np`
- Making arrays:
  - Can use lists, though they better be of the same type

```
import numpy as np
my_list = [1,5,4,2]
my_vec = np.array(my_list)
my_list = [[1,2],[4,3]]
my_mat = np.array(my_list)
```

Array Creation

- Can generate using lists, tuples, etc. even with a mix of types
  - `np.array([[1,2,3], (1,0,0.5)])`
    - `array([[1. , 2. , 3. ], [1. , 0. , 0.5]])`

Array Creation

- Creating arrays:
  - np.full to fill in with a given value

```
np.full(5, 3.141)

array([3.141, 3.141, 3.141, 3.141, 3.141])
```

Array Creation

- Can also use random values.
  - Uniform distribution between 0 and 1

```
>>> np.random.random((3,2))
array([[0.39211415, 0.50264835],
       [0.95824337, 0.58949256],
       [0.59318281, 0.05752833]])
```

Array Creation

- Or random integers

```
>>> np.random.randint(0,20,(2,4))
array([[ 5,  7,  2, 10],
       [19,  7,  1, 10]])
```

Array Creation

- Or other distributions, e.g. normal distribution with mean 2 and standard deviation 0.5

```
>>> np.random.normal(2,0.5, (2,3))
array([[1.34857621, 1.34419178, 1.977698  ],
       [1.31054068, 2.35126538, 3.25903903]])
```

Array Creation

- fromfunction

```
>>> x = np.fromfunction(lambda i,j: (i**2+j**2)//2, (4
```

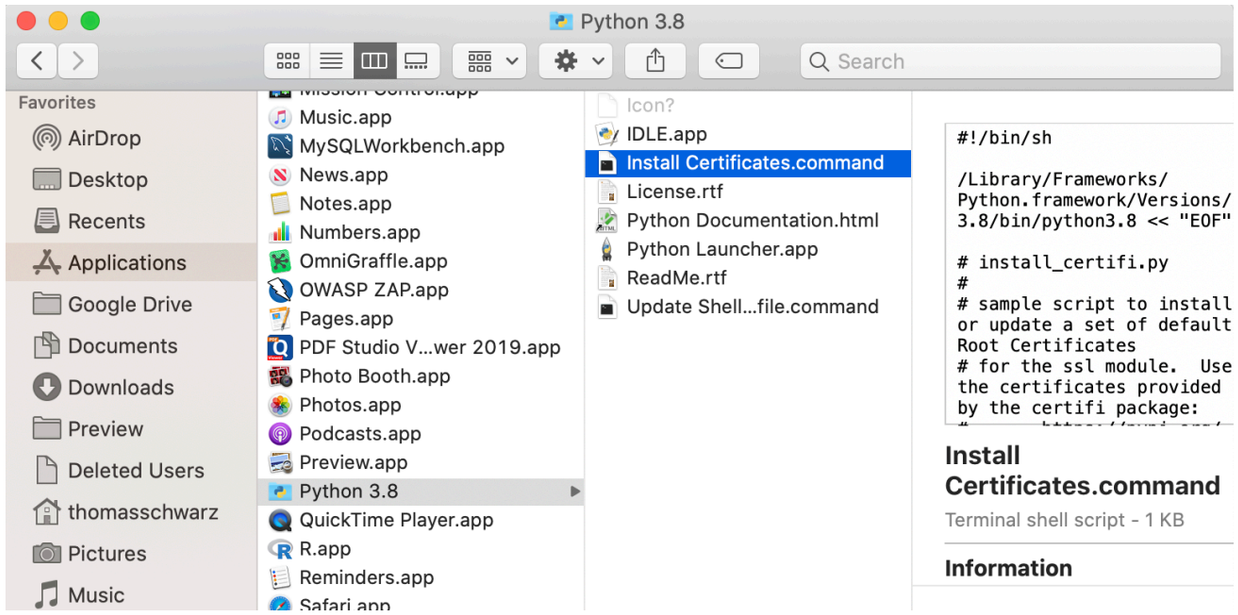
```
>>> x.astype(int)
```

```
array([[ 0,  0,  2,  4,  8],
       [ 0,  1,  2,  5,  8],
       [ 2,  2,  4,  6, 10],
       [ 4,  5,  6,  9, 12]])
```

```
>>> x.shape
(4,5)
```

Array Creation

- Creating from download / file
  - We use urllib.request module
    - If you are on Mac, you need to have Python certificates installed
      - Go to your Python installation in Applications and click on "Install Certificates command"



Array Creation

- Use urllib.request.urlretrieve with website and file name
  - Remember: A file will be created, but the directory needs to exist

```
import urllib.request
urllib.request.urlretrieve(
    url = "https://ndownloader.figshare.com/files/125656
    filename = "avg-monthly-precip.txt"
)
```

  - This is a text file, with one numerical value per line
  - Then create the numpy array using

```
avgmp = np.loadtxt(fname = 'avg-monthly-precip.txt')
print(avgmp)
```

Array Creation

- Example: Get an account at openweathermap.org/appid
- Install requests and import json
  - Use the openweathermap.org api to request data on a certain town
- Result is send as a JSON dictionary

Array Creation

```
import numpy as np
import requests
import json

mumbai=json.loads(requests.get('http://
api.openweathermap.org/data/2.5/weather?
q=mumbai,india&APPID=4561e0cd15ec2ee307bdcfe19ec22ab9').text
)
vasai = json.loads(requests.get('http://
api.openweathermap.org/data/2.5/weather?
q=vasai,india&APPID=4561e0cd15ec2ee307bdcfe19ec22ab9').text)
navi_mumbai = json.loads(requests.get('http://
api.openweathermap.org/data/2.5/weather?
q=navi%20mumbai,india&APPID=4561e0cd15ec2ee307bdcfe19ec22ab9
').text)
chalco = json.loads(requests.get('http://
api.openweathermap.org/data/2.5/weather?
q=Chalco,MX&APPID=4561e0cd15ec2ee307bdcfe19ec22ab9').text)
milwaukee = json.loads(requests.get('http://
api.openweathermap.org/data/2.5/weather?
q=Milwaukee,USA&APPID=4561e0cd15ec2ee307bdcfe19ec22ab9').tex
t)
```

Array Creation

- Can use np.genfromtext
  - Very powerful and complicated function with many different options

Array Creation

- Example

```
converters = {5: lambda x: int(0 if 'Iris-setosa' else 1
'Iris-virginica' else 2) }
my_array = np.genfromtxt('../Classes2/Iris.csv',
                         usecols=(1,2,3,4,5),
                         dtype =[float, float, float, flo
float],
                         delimiter = ',',
                         converters = converters,
                         skip_header=1)
```

- Need a source (the iris file)
- Can specify the columns we need
- Give the dtype U20-unicode string, S20-byte string
- Delimiter
- Skipheader, skipfooter
- converters to deal with encoding

Array Creation

- This is an array of 150 tuples
- Use comprehension to convert to a two-dimensional array

```
m = np.array( [ [row[0], row[1], row[2], row[3], row[4]
for row in my_array ])
```

Array Example

- Python has unlimited precision integers
  - E.g.: 10,000!

```
import math
print(math.factorial(10000))
```

Array Example

- But this is not true for the majority of programming languages
  - Because unlimited precision comes with a performance penalty
  - Can use limited or size-unlimited arrays
    - to implement arbitrary precision

Array Example

- Example:
  - Let's use Numpy arrays
    - Numpy arrays have a type
      - We use np.uint8
        - 8-bit unsigned integers 0, 1, ..., 254, 255
        - Normal operations roll over
          - E.g. 250+10=4

Array Example

- Implement an arbitrary fixed precision counter
  - Contains a Numpy array with type uint8
    - self.array[0] is the least significant "digit"
    - self.array[1] is the second least significant digit
  - Implement an increment function:
    - Raise an exception when attempting to roll over

Array Example

- Need to import Numpy
  - Use usual abbreviation
    - `np.full(n,0)` creates an array of length *n* with values 0
    - `np.zeros(n)` has the same effect
  - Cast the array to np.uint8

```
import numpy as np

class Limited_precision_integer:
    def __init__(self, n):
        self.array = np.full(n,0)
        self.array = self.array.astype(np.uint8)
```

Array Example

- Implementing the increment function
    - We add one to the current "digit"
        - We stop if the increment does not roll over
    - For the last digit:
        - If we are about to roll-over
            - Raise a ValueError

Array Example

```
raise ValueError('LPI rollover to zero')
ValueError: LPI rollover to zero
```

Array Example

```
def incr(self):
        for i in range(len(self.array)-1):
            self.array[i] += 1
            if self.array[i] != 0:
                return
        if self.array[-1] < 255:
            self.array[-1]+=1
        else:
            raise ValueError('LPI rollover to zero')
```

Python Trick

- In a user-defined class
  - We can implement dunder methods
    - `def __getitem__(self, key)`
    - `def __setitem__(self, key, value)`
      - There are two underlines before and after
- For full "container" types, there are more dunder methods
- `https://docs.python.org/3/reference/datamodel.html`

Array ADT

- Array needs to support
  - getting an item at a given location
  - setting an item at a given location

Array ADT Implementation

- Several solutions in order to implement arbitrary length arrays
  - Insertion at the end:
    - Start with an array of fixed length
    - Whenever length is not sufficient:
      - Create a new array with double the length
      - Move array elements to the new array