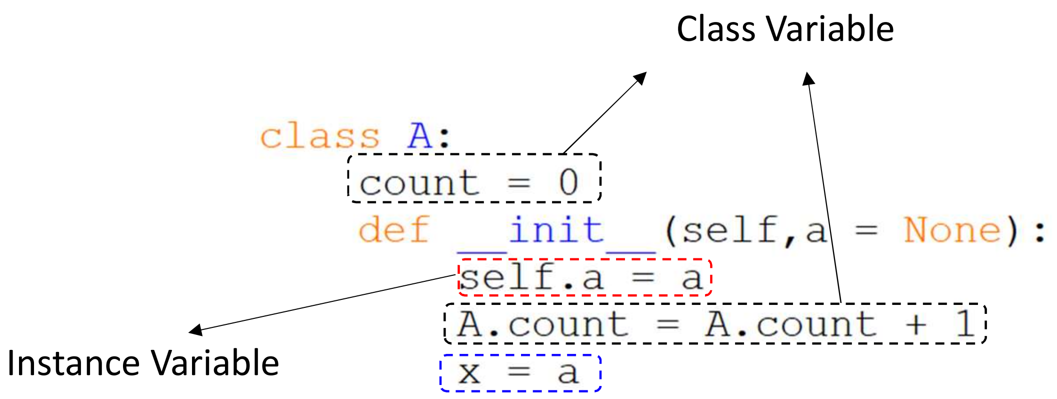
OOP Misc.
and
NumPy Introduction

OOP Misc.

Instance Vs Class Variables



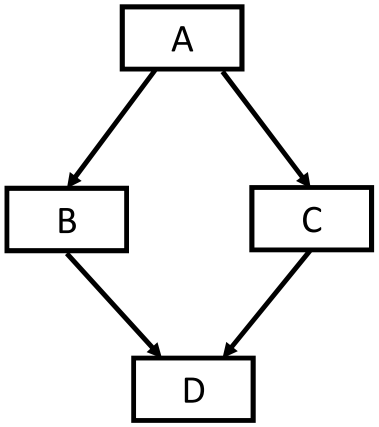
```python
class A:
    count = 0
    def __init__(self,a = None):
        self.a = a
        A.count = A.count + 1
        x = a

a1 = A(1)
a2 = A(2)
a3 = A(3)
```

```
>>> a1.a
1
>>> a2.a
2
>>> a3.a
3
>>> A.count
3
>>> a4 = A()
>>> A.count
4
```

Method Resolution Order

- C3 Linearization

```
class A:
    pass
class B(A):
    pass
class C(A):
    pass
class D(C,B):
    pass
print(D.__mro__)
```



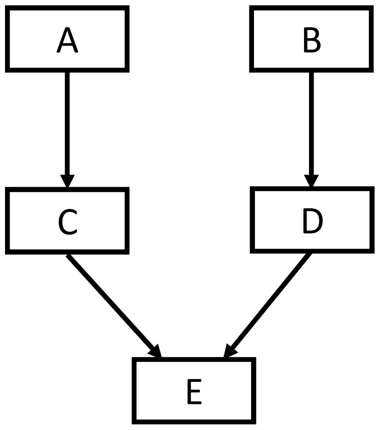
```
>>> D.__mro__
(<class '__main__.D'>, <class '__main__.C'>, <class '__main__.B'>, <class '__main__.A'>, <class 'ob
ject'>)
```

https://en.wikipedia.org/wiki/C3_linearization

Method Resolution Order

- C3 Linearization

```
class A:
    pass
class B:
    pass
class C(A):
    pass

class D(B):
    pass

class E(C,D):
    pass
```



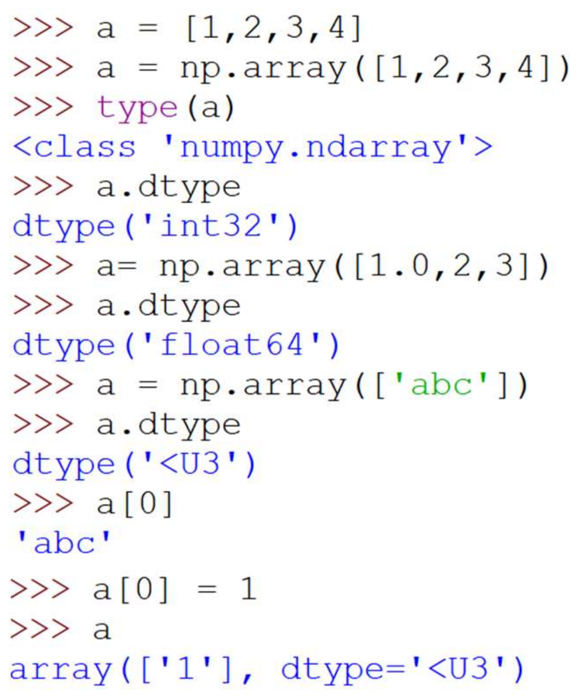
```
>>> E.__mro__
(<class '__main__.E'>, <class '__main__.C'>, <class '__main__.A'>, <class '__main__.D'>, <class '__
main__.B'>, <class 'object'>)
```

NumPy

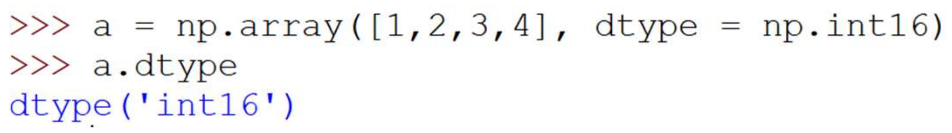
NumPy

- Arrays
  - Mutable sequences

- Stores data as bytes
  - All data should be of same type
  - Can enforce data type
  - Memory efficient
  - Wide range of data types
    - Int32, int64, float64, etc.



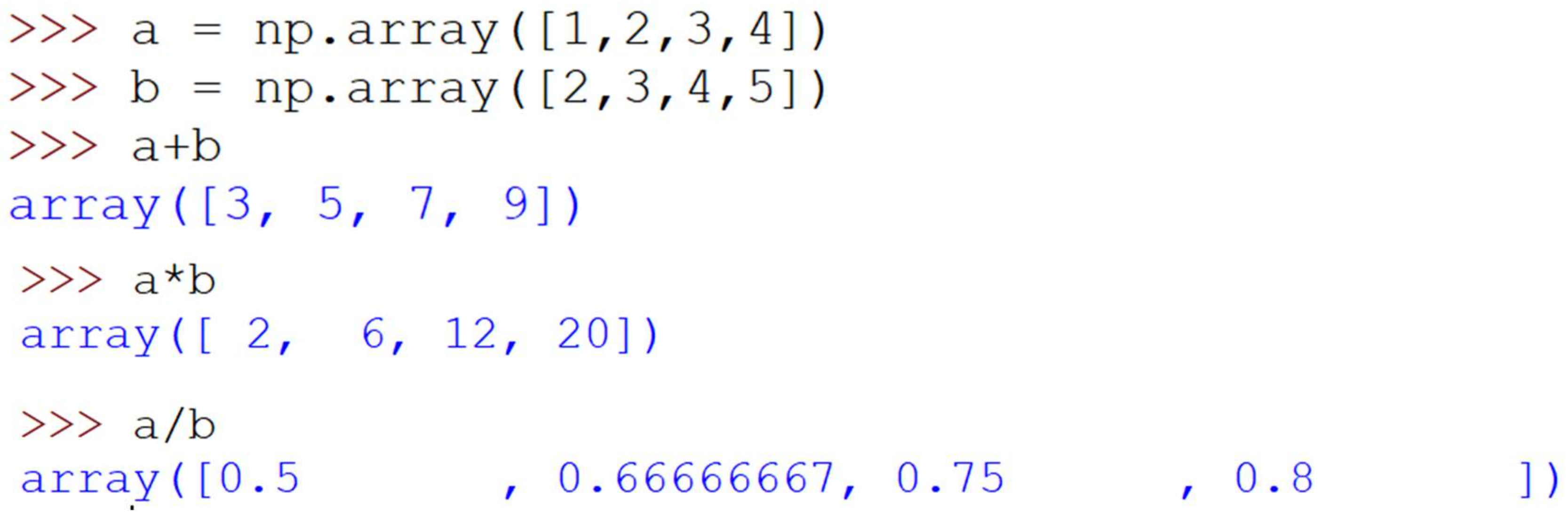
```
>>> a = [1,2,3,4]
>>> a = np.array([1,2,3,4])
>>> type(a)
<class 'numpy.ndarray'>
>>> a.dtype
dtype('int32')
>>> a= np.array([1.0,2,3])
>>> a.dtype
dtype('float64')
>>> a = np.array(['abc'])
>>> a.dtype
dtype('<U3')
>>> a[0]
'abc'
>>> a[0] = 1
>>> a
array(['1'], dtype='<U3')
```



```
>>> a = np.array([1,2,3,4], dtype = np.int16)
>>> a.dtype
dtype('int16')
```

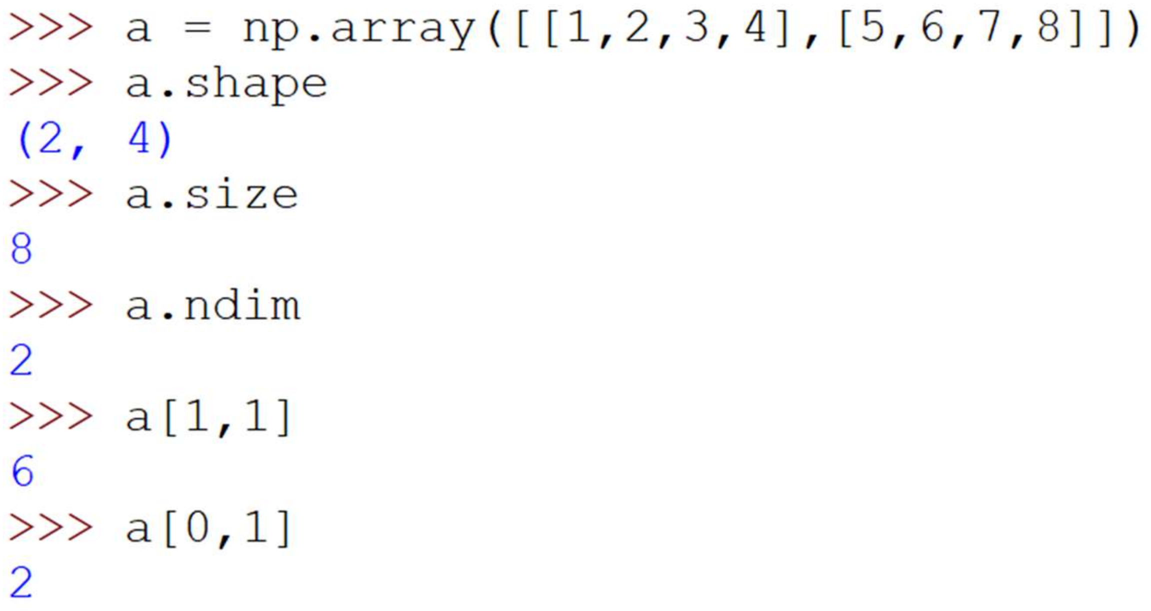
Vectorization

Elementwise operations



```
>>> a = np.array([1,2,3,4])
>>> b = np.array([2,3,4,5])
>>> a+b
array([3, 5, 7, 9])
>>> a*b
array([ 2,  6, 12, 20])
>>> a/b
array([0.5       , 0.66666667, 0.75      , 0.8       ])
```

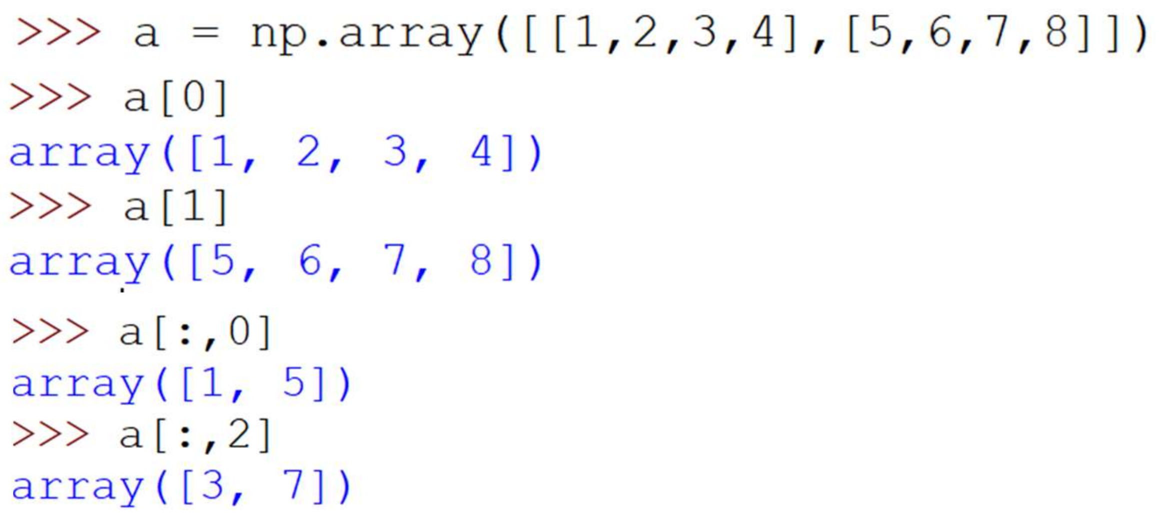
NumPy: MD Arrays



```
>>> a = np.array([[1,2,3,4],[5,6,7,8]])
>>> a.shape
(2, 4)
>>> a.size
8
>>> a.ndim
2
>>> a[1,1]
6
>>> a[0,1]
2
```

NumPy: MD Arrays

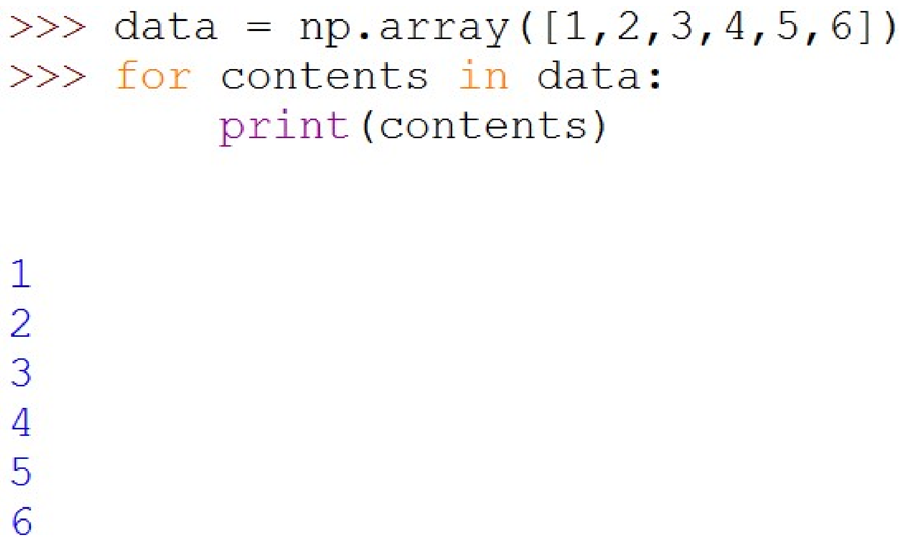
- Slicing of MD Arrays



```
>>> a = np.array([[1,2,3,4],[5,6,7,8]])
>>> a[0]
array([1, 2, 3, 4])
>>> a[1]
array([5, 6, 7, 8])
>>> a[:,0]
array([1, 5])
>>> a[:,2]
array([3, 7])
```

Looping Through a 1D Array

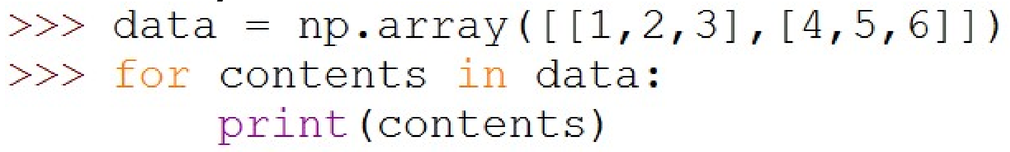
- For a 1D array



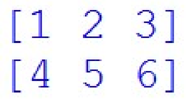
```
>>> data = np.array([1,2,3,4,5,6])
>>> for contents in data:
        print(contents)


1
2
3
4
5
6
```

Looping Through a 2D Array

- But for a 2D array



```
>>> data = np.array([[1,2,3],[4,5,6]])
>>> for contents in data:
        print(contents)
```

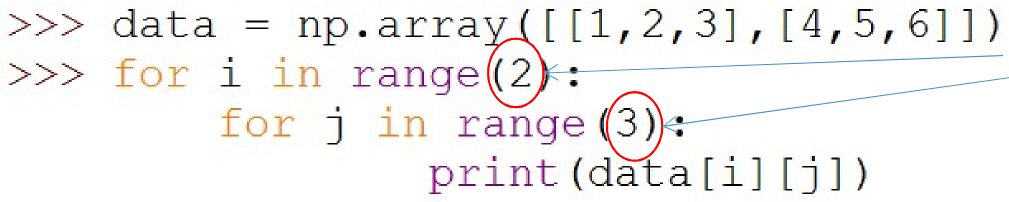


```
[1 2 3]
[4 5 6]
```

- Not every single “item” but the two rows

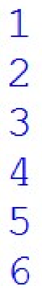
Looping Through a 2D Array

- To investigate every single item



```
>>> data = np.array([[1,2,3],[4,5,6]])
>>> for i in range(2):
        for j in range(3):
            print(data[i][j])
```

What if I don't know how many rows and columns in the array?



```
1
2
3
4
5
6
```

Looping Through a 2D Array

```
>>> data = np.array([[1,2,3],[4,5,6]])
>>> for i in range(data.shape[0]):
        for j in range(data.shape[1]):
            print(data[i][j])
```

```
1
2
3
4
5
6
```