Lecture15

March 28, 2024

[28]:
```
# numpy
# create array: list, arange, linspace, ones, zeros
# attrbutes: shape (reshape), dtype (astype)
# indexing and slicing: a[], a[row, col], a[row][col], a[start:end:step],␣
 ↪a[start:end:step, start:end:step]
# operators (apply between two arrays or broadcasting): Arithmetic (+,-,*,/
 ↪,**,%), comparisons (==,!=,>,<,>=,<=), logical/bitwise (&,|,~)
# functions: np.mean, np.median, np.std, np.var, np.sum, np.unique, np.argmax,␣
 ↪np.argmin, np.argsort, np.argwhere, np.all, np.any, np.hstack, np.vstack
# array vs list: arrays are faster if you want to apply an operation on all␣
 ↪items, +, -, *, ...
# fancy indexing: a[a>3], a[a%2==0], a[b<c]
```

[29]:
```
import numpy as np
a=np.array([True, True, False])
b=np.array([True, False, True])
```

[30]:
```
a&b
```

[30]:
```
array([ True, False, False])
```

[39]:
```
c=np.array([0,1,20,50,10])
d=np.array([1,2,3,50,60])
e=np.array([10,20,30,40,50])
e[c>d]
```

[39]:
```
array([30])
```

[40]:
```
c>d
```

[40]:
```
array([False, False,  True, False, False])
```

[38]:
```
np.hstack([c,d])
np.concatenate([c,d], axis=0)
```

[38]:
```
array([ 0,  1, 20, 50, 10,  1,  2,  3])
```

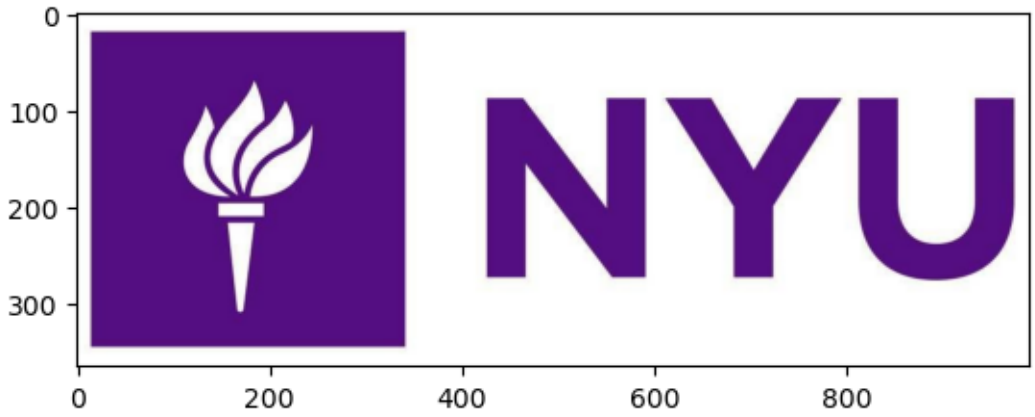
[42]:
```
import matplotlib.pyplot as plt
im=plt.imread('logo.jpg').astype(int)
```

[43]:
```
im.shape
```

[43]:
```
(365, 990, 3)
```

[45]:
```
plt.imshow(im)
```

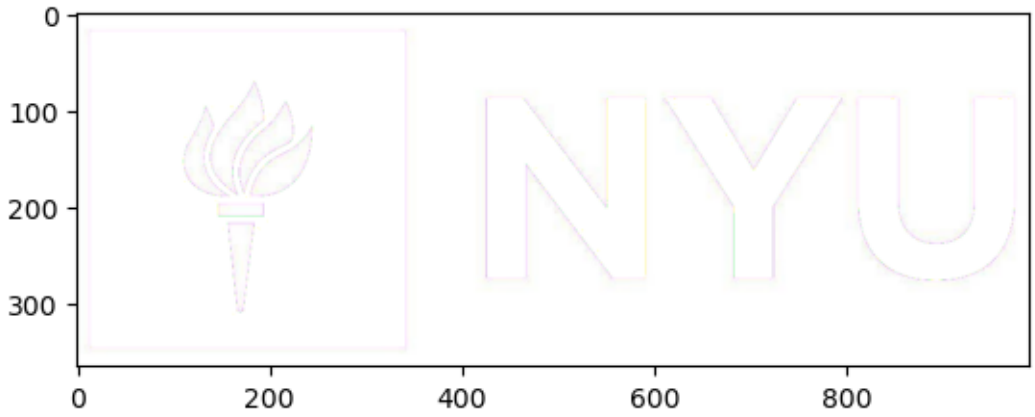
[45]:
```
<matplotlib.image.AxesImage at 0x7f4d44300cd0>
```



[46]:
```
im[im<150]=255
```

[47]:
```
plt.imshow(im)
```

[47]:
```
<matplotlib.image.AxesImage at 0x7f4d0fddb750>
```



1 Matrix

[60]:
```
# create
m=np.matrix([1,2,3])
m=np.matrix(np.array([1,2,3]))
# mat
m
```

[60]:
```
matrix([[1, 2, 3]])
```

[61]:
```
a=np.array([1,2,3])
a
```

[61]:
```
array([1, 2, 3])
```

[62]:
```
m.shape
```

[62]:
```
(1, 3)
```

[63]:
```
a.shape
```

[63]:
```
(3,)
```

[64]:
```
m[0]
```

[64]:
```
matrix([[1, 2, 3]])
```

[67]:
```
a
```

[67]:
```
array([1, 2, 3])
```

[68]:
```
m1=np.matrix(a)
```

[69]:
```
m2=np.mat(a)
```

[72]:
```
a[0]=1000
```

[73]:
```
m1
```

[73]:
```
matrix([[1, 2, 3]])
```

[74]:
```
m2
```

[74]:
```
matrix([[1000,    2,    3]])
```

[76]:
```
# operators: *, T, I
```

[78]:
```
a=np.array([1,2,3])
b=np.array([4,5,6])
a*b
```

[78]:
```
array([ 4, 10, 18])
```

[80]:
```
m1=np.matrix([1,2,3])
m2=np.matrix([4,5,6])
print(m1.shape)
print(m2.shape)
```

```
(1, 3)
(1, 3)
```

[84]:
```
m1.T * m2
```

[84]:
```
matrix([[ 4,  5,  6],
        [ 8, 10, 12],
        [12, 15, 18]])
```

[91]:
```
m1*m1.I
```

[91]:
```
matrix([[1.]])
```

[92]:
```
np.identity(3)
```

[92]:
```
array([[1., 0., 0.],
       [0., 1., 0.],
       [0., 0., 1.]])
```

2 Randomization

[100]:
```
np.random.randn(10)    # generated from standard distribution
np.random.rand(10)     # generated from uniform distribution [0,1]
np.random.randint(0,5, 10)  # generated from range [0,5)
np.random.random_sample(10) # generated from [0,1)
np.random.choice(['hasan', 'james','alma', 'maya', 'tara'], 3)
np.random.choice(['hasan', 'james','alma', 'maya', 'tara'], 3, replace=False)
np.random.choice(['hasan', 'james','alma', 'maya', 'tara'], 3, p=[.6,.1,.1,.1,.
↪1], replace=False)
```

[100]:
```
array(['maya', 'hasan', 'james'], dtype='<U5')
```

2.0.1 Exercise

[101]:
```
#  using invddata1.txt, invdata2.txt
# find if there is a difference between these files
# if yes, find only shared items
def read_file(filename):
    f=open(filename)
    data=f.read()
    f.close()
    return data
```

[111]:
```
d1=read_file('../datafiles/invdata1.txt')
d2=read_file('../datafiles/invdata2.txt')
```

[112]:
```
len(set(d1.split(',')))
```

[112]: 986

[113]:
```
len(set(d2.split(',')))
```

[113]: 987

[116]:
```
set(d1.split(',')) ^ set(d2.split(','))
```

[116]:
```
{'105872287233',
 '109087889605',
 '153989350082',
 '207771969694',
 '228426621711',
 '243847654348',
 '251006115670',
 '259632295385',
 '385815737629',
 '413542006813',
 '413542186813',
 '450474389186',
 '458555138950',
 '478375913042',
 '483777112367',
 '492207218253',
 '518483238936',
 '540852670300',
 '551310033304',
 '693741271476',
 '717781672640',
 '719486334908',
 '752859179142',
 '761269422098',
```

```
'783789011379',
'797221317932',
'805610984521',
'859102729889',
'862953114274'}
```

[ ]:

[ ]: