hw05

February 14, 2019

1 HW 5

1.1 5.1 NumPy arrays

(a) NumPy arrays operate on corresponding elements. Arrays are not treated as matrices.

```
In [200]: import numpy as np

          sx = np.array([[0, 1], [1, 0]])
          sy = np.array([[0, -1j],[1j, 0]])
          sz = np.array([[1, 0], [0, -1]])

          result1b1 = sx * sy * sz
          print(result1b1)
```

```
[[ 0.+0.j  0.+0.j]
 [ 0.+0.j -0.+0.j]]
```

NumPy multiplication operates on corresponding elements.

```
In [201]: result1b2 = np.dot(np.dot(sx,sy),sz)
          print(result1b2)
```

```
[[0.+1.j 0.+0.j]
 [0.+0.j 0.+1.j]]
```

np.dot() treats arrays as matrices and performs matrix multiplication.

```
In [202]: result1b3 = np.dot(sx,sy) - np.dot(sy,sx)
          sz2j = sz * 2j

          if result1b3.all() == sz2j.all():
              print('I deserve an A')
```

```
I deserve an A
```

```
In [203]: v = np.array([1/np.sqrt(2), (-1j)/np.sqrt(2)])
          result1b4 = v.conjugate()
          print(result1b4)
```

```
[ 0.70710678-0.j         -0.        +0.70710678j]
```

1.2 5.2 Coordinate manipulation with NumPy

```
In [243]: import numpy as np
          positions = np.array(\
                [[0.0, 0.0, 0.0], [1.34234, 1.34234, 0.0], \
                 [1.34234, 0.0,  1.34234], [0.0, 1.34234, 1.34234]])
          t = np.array([1.34234, -1.34234, -1.34234])

          print('shape is', positions.shape)
          print('dimension is', positions.ndim)
```

```
shape is (4, 3)
dimension is 2
```

```
In [232]: print('shape is', t.shape)
          print('dimension is', t.ndim)
```

```
shape is (3,)
dimension is 1
```

```
In [233]: print(positions)
          result2c = positions[1]

          print(result2c)
```

```
[[0.      0.      0.     ]
 [1.34234 1.34234 0.     ]
 [1.34234 0.      1.34234]
 [0.      1.34234 1.34234]]
[1.34234 1.34234 0.     ]
```

```
In [234]: result2d = positions[1][1]
          print(type(result2d))
          print('shape is', result2d.shape)
          print('dimension is', result2d.ndim)
```

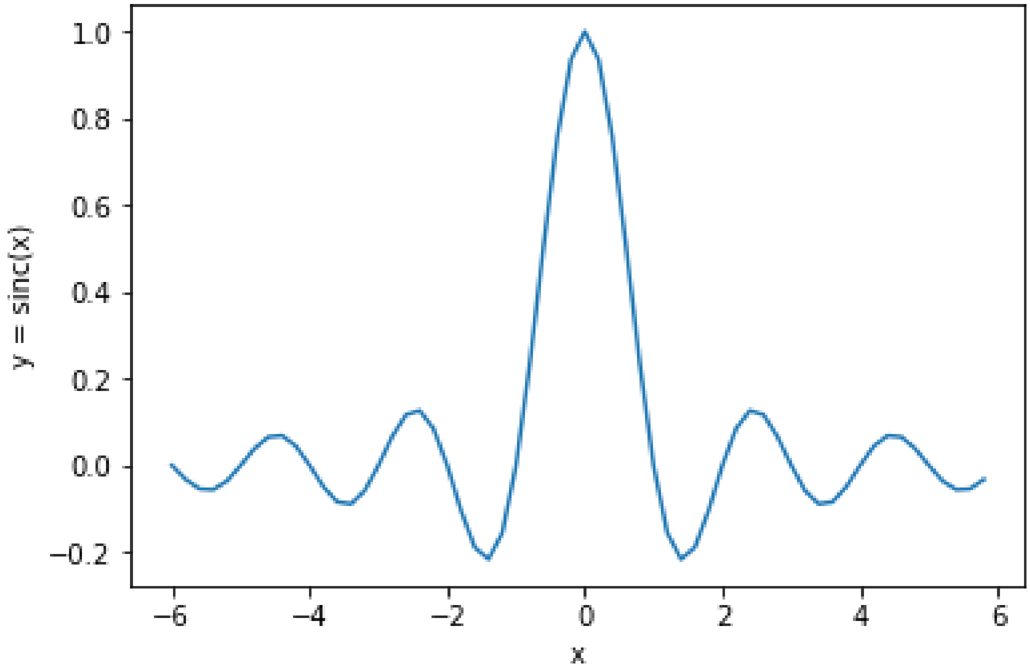
```
<class 'numpy.float64'>
shape is ()
dimension is 0
```

```
In [235]: problem2e = positions[:]
          for i in problem2e:
               i += t
          result2e = problem2e[:]

          print(result2e)

[[ 1.34234 -1.34234 -1.34234]
 [ 2.68468  0.      -1.34234]
 [ 2.68468 -1.34234  0.     ]
 [ 1.34234  0.       0.     ]]
```

```
In [244]: def translate(coordinates, t):
              for i in coordinates:
                  i += t
              return coordinates

          translate(positions, t)

          print(positions)
          positions2 = np.array([[1.5, -1.5, 3], [-1.5, -1.5, -3]])
          t = np.array([-1.5, 1.5, 3])

          translate(positions2, t)
          print(positions2)

[[ 1.34234 -1.34234 -1.34234]
 [ 2.68468  0.      -1.34234]
 [ 2.68468 -1.34234  0.     ]
 [ 1.34234  0.       0.     ]]
[[ 0.  0.  6.]
 [-3.  0.  0.]]
```

1.3 5.3 NumPy functions

```
In [166]: import matplotlib.pyplot as plt
          import numpy as np

          X = np.arange(-6,6,0.2)
          Y = np.sinc(X)
          plt.plot(X, Y)
          plt.xlabel("x")
          plt.ylabel("y = sinc(x)")
          plt.savefig("sinc.png")
```



```
In [215]: import numpy as np

          x = np.arange(1,101)
          y = 6/(x**2)
          mypi = np.sqrt(np.sum(y))

In [ ]:
```