NumPy and Matplotlib II

In this exercise you need to make use of the Python packages numpy and matplotlib.

1. Construct an identity matrix I of dimension 100x100 as numpy.ndarray.
2. Construct a banded matrix A of the form

$$A = \frac{100^2}{4\pi^2}\begin{bmatrix} -2 & 1 & 0 & 0 & \dots & 0 & 0 \\ 1 & -2 & 1 & 0 & \dots & 0 & 0 \\ 0 & 1 & -2 & 1 & \dots & 0 & 0 \\ 0 & 0 & 1 & -2 & \dots & 0 & 0 \\ \dots & \dots & \dots & \dots & \dots & \dots & \dots \\ 0 & 0 & 0 & 0 & \dots & -2 & 1 \\ 0 & 0 & 0 & 0 & \dots & 1 & -2 \end{bmatrix}$$

   of dimension 100x100 as numpy.ndarray.
3. Plot both matrices with the function imshow() of the package matplotlib.pyplot.
4. Construct a vector z with the linspace() function of the numpy package. It should contain a grid of 102 values between 0 and 2π.
5. Use Python's *slicing* capabilities to copy all except from the first and the last value of z into another vector x.
6. Calculate $y = \sin(x)$ and the matrix vector product $d = Ay$ using numpy.sin() and numpy.dot().
7. Plot y and d into the same plot using plot() from the matplotlib.pyplot package.

Hint: You might need to use matplotlib.pyplot.show() in order to guarantee that the notebook shows some output.

Solution:

```python
import numpy as np

dim = 100

## Identity Matrix
# Create matrix with numpy.eye()
I = np.eye(dim)
print(I)
help(np.eye)

## Construct matrix A

### via for loop
A = np.zeros((dim,dim), dtype = float)
# for loop for running through rows and columns
for i in range(dim):
    for j in range(dim):
        # diagonal entries
        if i==j:
            A[i,j] = -2
        # first lower and upper off diagonals
        if np.abs(i-j) == 1:
            A[i,j] = 1

# The factor s
# The number pi is also provided by numpy
s = 100**2/(4*(np.pi**2))
```

```
# scale A
A = s*A
print("A=", A)

### Construct A via np.eye
# numpy.eye() can also create diagonal entries
A = -2*I + np.eye(dim, k=-1) + np.eye(dim, k=1)

# scale A
A = s*A
print("I=", I)
print("\n")
print("A=", A)

## Plot using `imshow`
# Matplotlib provides plotting functionalities
import matplotlib.pyplot as plt
plt.imshow(I)
plt.show()
plt.imshow(A)
plt.show()

## Use `linspace`
# Numpy linspace allows to easily create a regular 1D-grid.
z = np.linspace(0, 2*np.pi, 102)
plt.plot(z,np.zeros(102), 'rx')

## Slicing
# Python allows to easily take slices from arrays.
# The syntax is vector[start:stop:step]
# -1 can be used to get the index of the last entry
x = z[1:-1]

## Apply
y = np.sin(x)

# The @ operator can be used for matrix multiplication of numpy (!) arrays.
d = A@y

## Plot
# We can plot two plots into the same plot
plt.plot(x, y, label = "sin")
plt.plot(x, d, label = "- sin")

# This line is necessary for the legend. It does not show up otherwise.
plt.legend()

# This line plots all figures which were created above.
plt.show()

### Remark
#$\sin''(x) = -\sin(x)$
#The matrix $A$ is a "discretization" of the second derivative
```