Solve the following problems by hand, then implement in `numpy` to check your work and familiarize yourself with the library.

1. Given $\mathbf{u} = \begin{bmatrix} 1 \\ -2 \end{bmatrix}$ and $\mathbf{v} = \begin{bmatrix} 2 \\ -5 \end{bmatrix}$, find:

   (a) $4\mathbf{u} - 3\mathbf{v}$

   Solution: $4\mathbf{u} = \begin{bmatrix} 4 \\ -8 \end{bmatrix}$, $3\mathbf{v} = \begin{bmatrix} 6 \\ -15 \end{bmatrix}$, so $4\mathbf{u} - 3\mathbf{v} = \begin{bmatrix} 4 \\ -8 \end{bmatrix} - \begin{bmatrix} 6 \\ -15 \end{bmatrix} = \begin{bmatrix} -2 \\ 7 \end{bmatrix}$.

```python
import numpy as np
u = np.array([1, -2])
v = np.array([2, -5])
4*u - 3*v
```

   (b) $\mathbf{u} \cdot \mathbf{v}$

   Solution: $\mathbf{u} \cdot \mathbf{v} = 1(2) + (-2)(-5) = 2 + 10 = 12$.

```python
np.dot(u, v) # alternatively, u.dot(v)
```

   (c) The angle θ between $\mathbf{u}$ and $\mathbf{v}$, knowing that $\cos\theta = \dfrac{\mathbf{u} \cdot \mathbf{v}}{\|\mathbf{u}\|\|\mathbf{v}\|}$

   Solution: $\|\mathbf{u}\| = \sqrt{1^2 + (-2)^2} = \sqrt{5}$ and $\|\mathbf{v}\| = \sqrt{2^2 + (-5)^2} = \sqrt{29}$.

   $\theta = \cos^{-1}\left(\dfrac{12}{\sqrt{5}\sqrt{29}}\right) \approx 0.083$ radians or $4.76°$.

```python
theta = np.arccos(np.dot(u, v) / (np.linalg.norm(u) * np.
    linalg.norm(v)))
theta*180/np.pi
```

   (d) Bonus: plot $\mathbf{u}$ and $\mathbf{v}$ using `matplotlib`. Does your angle θ make sense?

   Solution: Something like:

```python
import matplotlib.pyplot as plt
plt.plot([0, u[0]], [0, u[1]], label='u')
plt.plot([0, v[0]], [0, v[1]], label='v')
plt.legend()
```

   which shows that the two vectors are almost parallel, so a small angle makes sense.

2. Consider a matrix $A = \begin{bmatrix} 1 & 3 \\ 2 & -1 \end{bmatrix}$ and a vector $\mathbf{b} = \begin{bmatrix} 1 \\ 0 \end{bmatrix}$.

   (a) Compute $A\mathbf{b}$.

Solution: $A\mathbf{b} = \begin{bmatrix} 1 & 3 \\ 2 & -1 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \end{bmatrix}$

```
A = np.array([
    [1, 3],
    [2, -1]
])
v = np.array([1, 0])
A @ v
```

Note that `@` is the matrix multiplication operator in `numpy` - this is different from the * operator, which performs element-wise multiplication.

   (b) Find A^{-1}. For 2×2 matrix $\begin{bmatrix} a & b \\ c & d \end{bmatrix}$, $A^{-1} = \frac{1}{ad - bc} \begin{bmatrix} d & -b \\ -c & a \end{bmatrix}$.

Solution: $A^{-1} = \frac{1}{-1-6} \begin{bmatrix} -1 & -3 \\ -2 & 1 \end{bmatrix} = \begin{bmatrix} 1/7 & 3/7 \\ 2/7 & -1/7 \end{bmatrix}$

```
print(np.linalg.inv(A))
(1/-7) * np.array([
    [-1, -3],
    [-2, 1]
])
```

I figured it was easier to compare the results of multiplying the adjugate by 1/7 to the `linalg.inv` result rather than writing out all the fractions.

   (c) Solve the equation $A\mathbf{x} = \mathbf{b}$ an unknown vector $\mathbf{x}$.

Solution: $A\mathbf{x} = \mathbf{b}$, multiply both sides by A^{-1}

$\mathbf{x} = A^{-1}\mathbf{b} = \begin{bmatrix} 1/7 & 3/7 \\ 2/7 & -1/7 \end{bmatrix} \begin{bmatrix} 1 \\ 0 \end{bmatrix} = \begin{bmatrix} 1/7 \\ 2/7 \end{bmatrix}$

```
np.linalg.solve(A, v)
```