1. Create two `Numpy` arrays. One named `x` using `linspace`. The other named `y` using `np.arange`.
Adapt the parameters so that the vectors are the same length.

```
x = np.linspace(1,5,5)
y = np.arange(1,6)
```

3. Write a print command that says ”the first three entries of x are” and then the actual values.

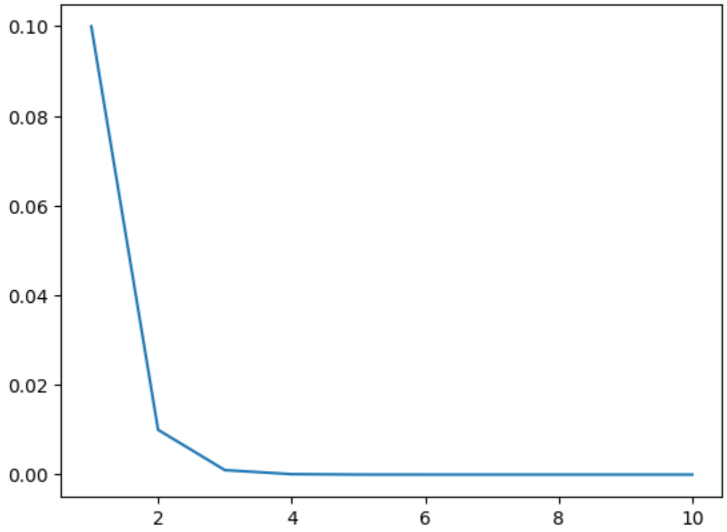
```
print('the first three entries of x are', x[0], x[1], x[2])
```

4. Make a vector as follows:

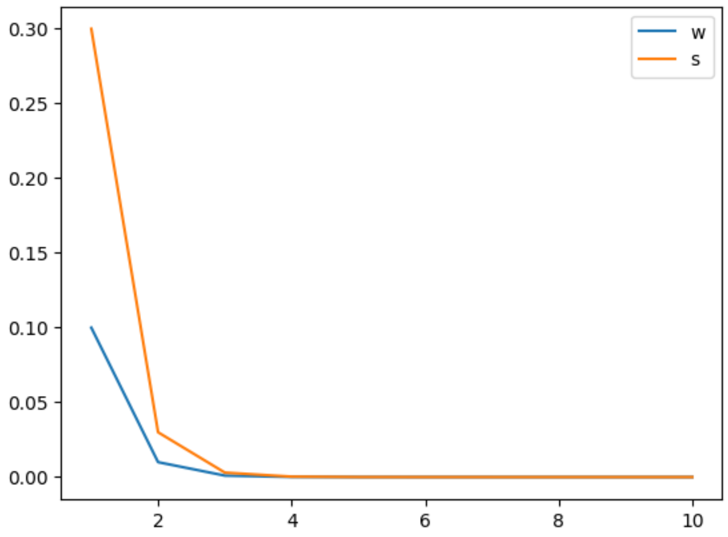
```
w = 10**(-np.linspace(1,10,10))
```

What are the entries of `w`? Make another vector `x` which has integer entries counting from one
to the length of `w`. Plot on a semilogy scale `x` versus `w`. Label the axes.

W is [.1,.01,.001,…,1e-10]



5. Make another vector `s` that is equal to 3 times `w`. add to your previous plot, the plot of `x`
versus `s`



1. Change the vectors in dot product code to ones that are orthogonal. You are welcome to make them smaller vectors.

```
x = np.array([3,4])
y = np.array([-4,3])
dotProduct(x,y,2)
0
```

2. Using the dot product code as a template, write a code that computes a matrix vector multiplication. For this you will need to learn how to make a matrix but other than that it should be straight forward. Test your code with a 2 × 2 matrix and doing the work by hand. Then try it for larger matrices.

```
def matrix_mult(A, B):
    m = len(A)
    n = len(A[0])

    n2 = len(B)
    p = len(B[0])

    # Ensure the number of columns in A is equal to the number of rows in B
    if n != n2:
        raise ValueError("Number of columns in A must be equal to the
number of rows in B.")
```

```
    # Initialize the result matrix with zeros
    result = [[0] * p for _ in range(m)]

    # Perform the matrix-matrix multiplication
    for i in range(m):
        for j in range(p):
            sum = 0
            for k in range(n):
                sum += A[i][k] * B[k][j]
            result[i][j] = sum

    return result

a = [
    [1, 2],
    [3, 4]
]
b = [
    [5, 6],
    [7, 8]
]

result = matrix_mult(a, b)
print(result)
```

```
[[19, 22], [43, 50]]
```

3. Both of these commands are built into `Numpy`. Figure out how to call these commands and
verify that the codes are performing as expected. Which is faster? Your code or `Numpy`?

When using the numpy commands I got the same answer. using np is quicker because it is already optimized