APPM 4600 - Lab 1

Magnus Miller

January 2025

1 Introduction

This is the lab 1 (Jan 14, 2025) report. Completed working with Hannah and Bryce

2 Pre-Lab

2.1 Part I

Figure 1: Python3 installed.

2.2 Part II

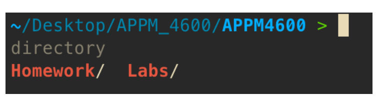

Figure 2: APPM4600 directory created with Homework and Labs subdirectories.

2.3 Part III

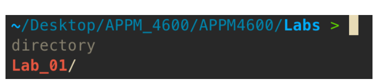

Figure 3: Lab 1 directory created.

2.4 Part IV

I have read through the remaining part of the lab document.

3 Lab-Day: Welcome to Python

The following subsections pertain to Part III of the lab: Welcome to Python.

3.1 Some Basics

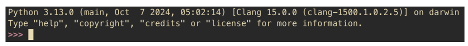

Figure 4: Using the Python3 shell.

3.1.1 Vectors

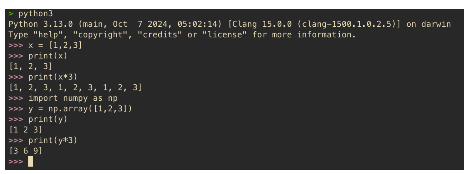

Figure 5: Using generic python lists as well as numpy arrays.

3.1.2 Printing

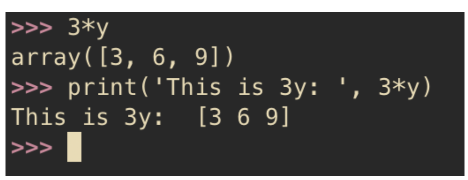

Figure 6: Practicing printing values.

3.2 Plotting

The following is the code producing a plot of a sine function and a cosine function.

```
# Importing Libraries
import numpy as np
import matplotlib.pyplot as plt

# Part 3.2 Code: Plotting
X = np.linspace(0, 2 * np.pi, 100)
Ya = np.sin(X)
Yb = np.cos(X)

plt.plot(X, Ya)
plt.plot(X, Yb)
plt.show()
```

Figure 7: Plotting Code

The above code in **Figure 7** produces the plot below in **Figure 8**.

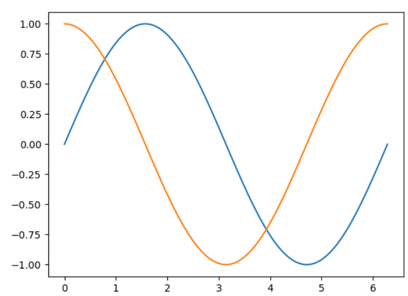

Figure 8: Sine and Cosine Plot.

The variable **X** is essentially a 1-D *numpy* array created using *numpy*'s **linspace** function. **X** is an array of length 100 meaning that it has 100 evenly spaced numbers beginning at 0 and ending at 2π.

3.3 Exercises: The Basics

3.3.1

The following is the code and output for Question 1.

```python
# Question 1:
x = np.linspace(0, 9, 10)
y = np.arange(0, 10, 1)
print(x)
print(y)
```

Figure 9: Code for Question 1.

```
> python3 Lab_01.py
[0. 1. 2. 3. 4. 5. 6. 7. 8. 9.]
[0 1 2 3 4 5 6 7 8 9]
~/Desktop/APPM_4600/APPM4600/Labs/Lab_01 >
directory
venv/
```

Figure 10: Output for Question 1.

3.3.2

The following is the code and output for Question 2.

```python
# Question 2:
print(x[:3])
print(y[:3])
```

Figure 11: Code for Question 2.

```
[0. 1. 2.]
[0 1 2]
~/Desktop/APPM_4600/APPM4600/Labs/Lab_01 >
directory
venv/
```

Figure 12: Output for Question 2.

3.3.3

The following is the code and output for Question 3.

```python
# Question 3:
print("The first three elements of x are: ", x[:3])
```

Figure 13: Code for Question 3.

```
The first three elements of x are:  [0. 1. 2.]
~/Desktop/APPM_4600/APPM4600/Labs/Lab_01 >
directory
venv/
```

Figure 14: Output for Question 3.

3.3.4

The following is the code and output for Question 4.

```python
# Question 4:
w = 10**(-np.linspace(1,10,10))
x1 = np.arange(0, len(w), 1)
plt.semilogy(x,w)
plt.show()
```

Figure 15: Code for Question 4.

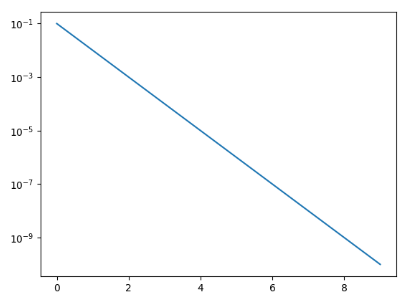

Figure 16: Output for Question 4.

As shown above in the plot in Figure 12, the values of **w** are: 1.e-01, 1.e-02, 1.e-03, 1.e-04, 1.e-05, 1.e-06, 1.e-07, 1.e-08, 1.e-09, and 1.e-10.

3.3.5

The following is the code and output for Question 4.

```python
# Question 5:
s = 3*w
plt.semilogy(x,s)
plt.show()
```

Figure 17: Code for Question 5.

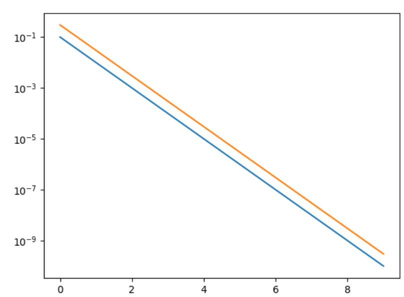

Figure 18: Output for Question 5.

4 Practical Code Design

4.0.1 Question I

The following is the code and the output for Question 1 of Part 4.

```python
'''
driver1()
Param:
Return:
This function serves as the driver for the dot product subroutine
'''
def driver1():
    n = 3
    y = [1,0,0]
    w = [0,1,0]
    # evaluate the dot product of y and w
    dp = dotProduct(y,w,n)
    # print the output
    print('the dot product is : ', dp)
    return
```

Figure 19: Code for Question 1.

```
> python3 testDot.py
the dot product is :  212660.24304678725
the dot product is :  0.0
~/Desktop/APPM_4600/APPM4600/Labs/Lab_01 >
directory
venv/
```

Figure 20: Output for Question 1.

4.0.2 Question II

The following is the code and the output for Question 2 of Part 4.

```python
'''
driver2()
Param:
Return:
This function serves as the driver for the dot product subroutine for matrix
multiplication.
'''
def driver2():
    small_matrix = np.array([[1,2],[3,4]])
    small_vector = np.array([5,6])

    large_matrix = np.random.randint(0,10, (100,100))
    large_vector = np.random.randint(0,10, 100)

    dps = dotProduct(small_matrix, small_vector, 2)
    print('the matrix multiplication is: ', dps)

    dpl = dotProduct(large_matrix, large_vector, 100)
    print('the matrix multiplication is: ', dpl)
    return
```

Figure 21: Code for Question 2.

```
> python3 testDot.py
the dot product is :  212660.24304678725
the dot product is :  0.0
the matrix multiplication is:  [23. 34.]
the matrix multiplication is:  [1960. 2015. 1798. 2136. 1911. 2100. 2106. 2248. 1883. 2005. 2413. 2227.
 2201. 2214. 2483. 1972. 2386. 2441. 1824. 2056. 2142. 2036. 1983. 1877.
 2236. 2116. 2035. 2267. 2003. 2062. 2146. 2131. 2271. 2040. 1870. 2125.
 2093. 2086. 2151. 1937. 2315. 1718. 2367. 1731. 1874. 1925. 1937. 1993.
 2303. 2038. 2119. 1984. 2105. 1897. 1872. 2240. 2377. 2144. 2109. 2128.
 1873. 2067. 2053. 1904. 2149. 2357. 2130. 2076. 2429. 2040. 2175. 2332.
 2322. 2003. 1991. 2199. 1895. 2152. 2025. 2115. 2300. 1986. 2008. 2317.
 2277. 2031. 2143. 2337. 2136. 2019. 1935. 2094. 2051. 2419. 2172. 2400.
 1974. 2210. 2091. 2140.]
~/Desktop/APPM_4600/APPM4600/Labs/Lab_01 >
directory
venv/
```

Figure 22: Output for Question 2.

4.0.3 Question III

Both the dot product and matrix multiplication subroutines can be accomplished using *numpy*'s **dot** function. This function can calculate the inner product of two vectors as well as matrix multiplication of any matrices or matrix vector combination. I believe that the *numpy* function is faster based on the fact it is a widely used and very well maintained python library but I am not certain as I am not too sure how to measure and compare the speed of my code versus the library.