1. For the following problems, write down what each code block would display if executed in a Jupyter cell. If the code generates an error or infinite loop, write `Error`.

   (a)
   ```
   a, b = (0, 2)
   while a < 3:
       print(f’a/b = {a / b:.2f}’)
       a += 1
   ```

   (b)
   ```
   a = np.linspace(0, 2, 3)
   a ** 2 + 1
   ```

   (c)
   ```
   xstr = [’a’, ’b’]
   ystr = [’c’, ’d’]
   for x in xstr:
       for y in ystr:
           print(x + y, end=’,’)
   ```

   (d)
   ```
   cats = [’tiger’, ’lion’, ’liger’, ’tigon’]
   list(zip([x for x in range(4)], cats))
   ```

   Solution:

   (a)
   ```
   a/b = 0.00
   a/b = 0.50
   a/b = 1.00
   ```

   (b) `array([1., 2., 5.])`

   (c) `ac,ad,bc,bd,`

   (d) `[(0, ’tiger’), (1, ’lion’), (2, ’liger’), (3, ’tigon’)]`

2. The dataset `fruits.txt` contains a long series of fruit names, some of which might be repeated:

   ```
   banana,orange,strawberry,orange,...
   ```

   Suppose you read this data into the Python object `fruits` with the code

   ```
   with open(‘fruits.txt’) as wf:
       fruits = wf.read()
   ```

   (a) Write code which creates a set of all unique fruit names in the dataset.

   (b) Write code which creates a dictionary with fruit names as keys, and their frequency in the dataset as values. For example, your dictionary might look like

   ```
   {’banana’:1, ’orange’:2, ...}
   ```

   Solution:

(a)
```
fruitset = set()
fruitlist = fruits.split(’,’)
for name in fruitlist:
    if name not in fruitset:
        fruitset.add(name)

# OR
fruitset = set(fruits.split(’,’))
```

(b)
```
fruitdict = {}
for name in fruitlist:
    if name not in fruitdict:
        fruitdict[name] = 1
    else:
        fruitdict[name] += 1
```

3. Write a **recursive** function `fruits_after(name, fruits)` which takes a name of a fruit `name` (which is a string) and a list `fruits`, and returns the list of elements of `fruits` *after* the first occurrence of `name`.

   For example, the list of fruits from the previous problem might look like

   ```
   fruits_list = [’banana’, ’orange’, ’strawberry’, ’orange’, ...]
   ```

   Using this list, `fruits_after(’banana’, fruits_list)` should return

   ```
   [’orange’, ’strawberry’, ’orange’, ...]
   ```

   and `fruits_after(’orange’, fruits_list)` should return

   ```
   [’strawberry’, ’orange’, ...]
   ```

   Note: you MUST write a recursive function to get full credit on this problem. **Solution:**

4. Write code which would display a plot of the function $f(x) = x\sin(x)$ and its derivative on the same plot, over the domain $[0, 2\pi]$. Make one of the plots red, the other one blue, and give your plot the title `A Function and its Derivative`. Use 100 x-values for your plots.

   Note: you may use NumPy on this question, but it is not required.

   Solution:

   ```
   #4 with NumPy
   xvals = np.linspace(0, 2 * np.pi, 100)
   yvals0 = xvals * np.sin(xvals)
   yvals1 = np.sin(xvals) + xvals * np.cos(xvals)
   plt.plot(xvals, yvals0, ’red’)
   plt.plot(xvals, yvals1, ’blue’)
   plt.title(’A Function and Its Derivative’)
   plt.show()

   #4 without NumPy
   ```

```
xvals = [(x / 100 * 2 * math.pi) for x in range(100)]
yvals0 = [x * math.sin(x) for x in xvals]
yvals1 = [(math.sin(x) + x * math.cos(x)) for x in xvals]
plt.plot(xvals, yvals0, ’red’)
plt.plot(xvals, yvals1, ’blue’)
plt.title(’A Function and Its Derivative’)
plt.show()
```