Week 11: Homework Solutions

3.

```
import numpy as np
import matplotlib.pyplot as plt

theta = np.linspace(0, 2*np.pi, 100)

plt.plot(np.cos(theta), np.sin(theta))
plt.axes().set_aspect("equal")
plt.show()
```

4.

```
import matplotlib.pyplot as plt


def separate_coordinates(list_of_coordinates):
    """
    Take a list of coordinates [[x_1, y_1], ..., [x_n, y_n]]
    and return a pair of lists [x_1, ..., x_n] and [y_1, ..., y_n].
    """
    x_list = []
    y_list = []
    for coord in list_of_coordinates:
        x_list.append(coord[0])
        y_list.append(coord[1])
    return x_list, y_list


long_list = [[2,3], [0, 0], [5, 1]]
x_coords, y_coords = separate_coordinates(long_list)

print(x_coords)
print(y_coords)

plt.plot(x_coords, y_coords)
plt.show()
```

The Python function `zip` does something similar to this (though it is a good exercise to write such a function for yourself). If you have a collection of lists, then `zip` will zip them together to form a collection of lists with a list of all the first entries, a list of all the second entries and so on. (Actually it creates 'tuples' rather than lists, these are essentially immutable lists.) Try the following.

```
>>> print(list(zip([11, 12, 13], [21, 22, 23], [31, 32, 33],
                   [41, 42, 43], [51, 52, 53])))
```

Now given a list of pairs, we need to feed the list into the zip command. Unfortunately, the zip command needs the pairs separated by commas and not the list of the pairs. Compare `zip([[a, b], [c, d], [e, f]])` with `zip([a, b], [c, d], [e, f])`. The first has a single parameter, which a list of pairs; the second has three parameters, each parameter is a pair. If we have a list of coordinates, we have to 'unpack' the list into the arguments of the `zip` command: this is done with the *-operator, which we can call the 'scatter' or 'argument-unpacking' operator.

To see what the unpacking operator * does, compare the outputs of the following three print commands:

```
>>> print("parrot", 42, "spam", sep=': ')
>>> print_list = ["parrot", 42, "spam"]
>>> print(print_list, sep=': ')
>>> print(*print_list, sep=': ')
```

The first and the last give the same output. The second just prints the list out, whereas the third feeds the elements of the list into the print statement as arguments.

We can now use the unpacking operator with the `zip` command. The following will do the same as the above program:

```
import matplotlib.pyplot as plt
long_list = [[2,3], [0, 0], [5, 1]]
x_coords, y_coords = zip(*long_list)
print(x_coords)
print(y_coords)
plt.plot(x_coords, y_coords)
plt.show()
```

We could actually avoid defining `x_coords` and `y_coords` altogether, by using the *-operator to unpack the zipped object directly into the `plot.plt()` function. This leads to the following version, which is more compact but also more opaque:

```
import matplotlib.pyplot as plt
long_list = [[2,3], [0, 0], [5, 1]]
plt.plot(*zip(*long_list))
plt.show()
```