```
plt.show()
```

The syntax `-bo` indicates plotting with line (`-`) with circle marker (`o`) in blue (`b`). To show the plot in the interactive session or the notebook, we need to set the interactive output on with `plt.ion()` and then use the command `plt.show()`.

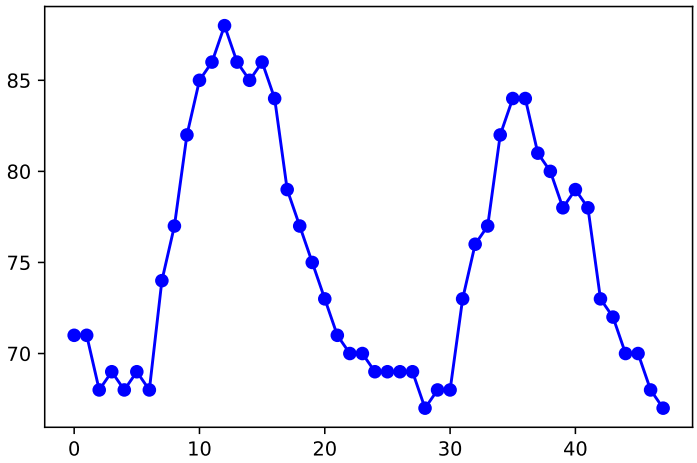

Figure 1.1.: Hourly temperature in downtown Los Angeles on August 5 and 6, 2015 (starting at 12:47AM, ending at 11:47PM).

1.2. Vector addition

Vector addition and subtraction. If `x` and `y` are numpy arrays of the same size, `x+y` and `x-y` give their sum and difference, respectively.

In [ ]:
```
import numpy as np
x = np.array([1,2,3])
y = np.array([100,200,300])
print('Sum of arrays:', x+y)
print('Difference of arrays:', x-y)
```

```
Sum of arrays: [101 202 303]
Difference of arrays: [ -99 -198 -297]
```

Sometimes when we would like to print more than one value, we may add a piece of

string in front of the value, followed by a comma. This allows us to distinguish between the values are we printing.

1.3. Scalar-vector multiplication

Scalar-vector multiplication and division. If `a` is a number and `x` is a numpy array (vector), you can express the scalar-vector product either as `a*x` or `x*a`.

In [ ]:
```
import numpy as np
x = np.array([1,2,3])
print(2.2*x)
```

```
[2.2 4.4 6.6]
```

You can carry out scalar-vector division as `x/a`.

In [ ]:
```
import numpy as np
x = np.array([1,2,3])
print(x/2.2)
```

```
[0.45454545 0.90909091 1.36363636]
```

Remark: For Python 2.x, integer division is used when you use the operator `/` on scalars. For example, `5/2` gives you 2. You can avoid this problem by adding decimals to the integer, *i.e.*, `5.0/2`. This gives you 2.5.

Scalar-vector addition. In Python, you can add a scalar `a` and a numpy array (vector) `x` using `x+a`. This means that the scalar is added to each element of the vector. This is, however, NOT a standard mathematical notation. In mathematical notations, we should denote this as, e.g. $x + a\mathbf{1}$, where x is an n-vector and a is a scalar.

In [ ]:
```
import numpy as np
x = np.array([1,2,3,4])
print(x + 2)
```

```
[3 4 5 6]
```

Elementwise operations. In Python we can perform elementwise operations on numpy arrays. For numpy arrays of the same length `x` and `y`, the expressions `x*y`, `x/y` and `x**y` give the resulting vectors of the same length as x and y and ith element $x_i y_i$,