Storing Collections of Data Using Lists

1. a. kingdoms[0]
b. kingdoms[5]
c. kingdoms[:3]
d. kingdoms[2:5]
e. kingdoms[4:]
f. kingdoms[1:0] (many other solutions)

2. a. kingdoms[6]
b. kingdoms[-1]
c. kingdoms[-6:-3]
d. kingdoms[-4:-1]
e. kingdoms[-2:]
f. kingdoms[-1:-2] (many other solutions)

3. a. appointments.append('16:30')
b. appointments += ['16:30']
c. The approach in (a) modifies the list. The one in (b) creates a new list.

4. a. ids.remove(3382)
b. ids.index(9362)
c. ids.insert(ids.index(9362) + 1, 4499)
d. ids = ids + [5566, 1830] OR:
   ids.append(5566)
   ids.append(1830)
e. ids.reverse()
f. ids.sort()

5. a. alkaline_earth_metals = [4, 12, 20, 38, 56, 88]
b. alkaline_earth_metals[5], alkaline_earth_metals[-1]
c. len(alkaline_earth_metals)
d. max(alkaline_earth_metals)

6. a. temps = [25.2, 16.8, 31.4, 23.9, 28, 22.5, 19.6]
b. temps.sort()
c.
   cool_temps = temps[0:2]
   warm_temps = temps[2:]
d. temps_in_celsius = cool_temps + warm_temps
7.

def same_first_last(L):
    """ (list) -> bool
    Precondition: len(L) >= 2
    Return True if and only if first item of the list is the same as the last.
    >>> same_first_last([3, 4, 2, 8, 3])
    True
    >>> same_first_last(['apple', 'banana', 'pear'])
    False
    >>> same_first_last([4.0, 4.5])
    False
    """
    return L[0] == L[-1]

8.

def is_longer(L1, L2):
    """ (list, list) -> bool

    Return True if and only if the length of L1 is longer than the length of L2.
    >>> is_longer([1, 2, 3], [4, 5])
    True
    >>> is_longer(['abcdef', 'ab', 'cd', 'ef'])
    False
    >>> is_longer(['a', 'b', 'c'], [1, 2, 3])
    False
    """
    return len(L1) > len(L2)

9.

10. a. units[0]
b. units[-1] or units[1]
c. units[0][0]
d. units[1][0]
e. units[0][1:]
f. units[1][0:2]

11. a. units[-2]
b. units[-1]
c. units[-2][-3]
d. units[-1][-3]
e. units[-2][-2:]
f. units[-1][-1:]