codility

WE TEST CODERS

Chapter 2

Arrays

Array is a data-structure that can be used to store many items in one place. Imagine that we have a list of items; for example, a shopping list. We don't keep all the products on separate pages; we simply list them all together on a single page. Such a page is conceptually similar to an array. Similarly, if we plan to record air temperatures over the next 365 days, we would not create lots of individual variables, but would instead store all the data in just one array.

2.1. Creating an array

We want to create a shopping list containing three products. Such a list might be created as follows:

```
shopping = ['bread', 'butter', 'cheese']
```

(that is, `shopping` is the name of the array and every product within it is separated by a comma). Each item in the array is called an element. Arrays can store any number of elements (assuming that there is enough memory). Note that a list can be also empty:

```
shopping = []
```

If planning to record air temperatures over the next 365 days, we can create in advance a place to store the data. The array can be created in the following way:

```
temperatures = [0] * 365
```

(that is, we are creating an array containing 365 zeros).

2.2. Accessing array values

Arrays provide easy access to all elements. Within the array, every element is assigned a number called an index. Index numbers are consecutive integers starting from 0. For example, in the array `shopping = ['bread', 'butter', 'cheese']`, 'bread' is at index 0, 'butter' is at index 1 and 'cheese' is at index 2. If we want to check what value is located at some index (for example, at index 1), we can access it by specifing the index in square brackets, e.g. `shopping[1]`.

2.3. Modifying array values

We can change array elements as if they were separate variables, that is each array element can be assigned a new value independently. For example, let's say we want to record that on the 42nd day of measurement, the air temperature was 25 degrees. This can be done with a single assignment:

```
temperatures[42] = 25
```

If there was one more product to add to our shopping list, it could be appended as follows:

```
shopping += ['eggs']
```

The index for that element will be the next integer after the last (in this case, 3).

2.4. Iterating over an array

Often we need to iterate over all the elements of an array; perhaps to count the number of specified items, for example. Knowing that the array contains of N elements, we can iterate over consecutive integers from index 0 to index $N-1$ and check every such index. The length of an array can be found using the `len()` function. For example, counting the number of items in shopping list can be done quickly as follows:

```
N = len(shopping)
```

Let's write a function that counts the number of days with negative air temperature.

2.1: Negative air temperature.

```
def negative(temperatures):
    N = len(temperatures)
    days = 0
    for i in xrange(N):
        if temperatures[i] < 0:
            days += 1
    return days
```

Instead of iterating over indexes, we can iterate over the elements of the array. To do this, we can simply write:

```
for item in array:
    ...
```

For example, the above solution can be simplified as follows:

2.2: Negative air temperature — simplified.

```
def negative(temperatures):
    days = 0
    for t in temperatures:
        if t < 0:
            days += 1
    return days
```

In the above solution, for every temperature, we increase the number of days with a negative temperature if the number is lower than zero.

2.5. Basic array operations

There are a few basic operations on arrays that are very useful. Apart from the length operation:

```
len([1, 2, 3]) == 3
```

and the repetition:

```
['Hello'] * 3 == ['Hello', 'Hello', 'Hello']
```

which we have already seen, there is also concatenation:

```
[1, 2, 3] + [4, 5, 6] == [1, 2, 3, 4, 5, 6]
```

which merges two lists, and the membership operation:

```
'butter' in ['bread', 'butter', 'cheese'] == True
```

which checks for the presence of a particular item in the array.

2.6. Exercise

Problem: Given array A consisting of N integers, return the reversed array.

Solution: We can iterate over the first half of the array and exchange the elements with those in the second part of the array.

2.3: Reversing an array.

```
1 def reverse(A):
2     N = len(A)
3     for i in xrange(N // 2):
4         k = N - i - 1
5         A[i], A[k] = A[k], A[i]
6     return A
```

Python is a very rich language and provides many built-in functions and methods. It turns out, that there is already a built-in method `reverse`, that solves this exercise. Using such a method, array A can be reversed simply by:

```
1 A.reverse()
```

Every lesson will provide you with programming tasks at `http://codility.com/programmers`.