

## OVERVIEW ON ARRAYS AND LOOPS — LOTTO

The Belgian “Loterie Nationale” organizes many games, one of them is the Lotto. In this game, there are 45 different numbers on 45 different balls, and 6 balls are chosen, randomly. When you play this game, you choose 6 numbers, and you win if you have chosen the 6 numbers that are on the 6 random balls. See [https://en.wikipedia.org/wiki/Six-number\\_lottery\\_game](https://en.wikipedia.org/wiki/Six-number_lottery_game).

The screenshot shows the Belgian Lotto website interface. At the top, there's a red banner with the Lotto logo, a clock indicating 'Encore 2j 4h 30min' until the draw, and a large prize amount of '€ 1.000.000 à gagner' for Wednesday, April 28th. Below the banner, there are navigation tabs: 'Simple' (selected), 'Multi', 'Quick Pick rapide', 'Votre bulletin favori', and 'Votre dernier bulletin', along with a link for 'Infos formules de jeu'. The main content area shows a selection of 6 numbers per grid, with a note that you can already play with 1 grid for €1.25. Below this, there are seven columns of lottery grids, each containing numbers 1-45 in a 7x7 layout. Some numbers are highlighted in red, indicating they have been selected. Each grid has a 'Quickpick' button at the bottom.

Source: <https://www.e-lotto.be/FR/drawGames/lotto/play/single/board>

The objective of this project is to program different ways to play this game.

## 1 Without Python

First, we consider the function “naive\_random\_integers”, in Figure 1.

Variables:

$i$  is an integer.

$chosenIntegers$  is an array of integers.

Instructions of the function:

- 1  $chosenIntegers \leftarrow 6 * [0]$  (An array of size 6 filled with 0)
- 2 For  $i$  from 0 to 5
- 3      $chosenIntegers[i] \leftarrow randint(1, 45)$  (A random integer between 1 and 45)
- 4 End For
- 5 Return  $chosenIntegers$

Figure 1: Algorithm “naive\_random\_integers”.

1. Explain what does the algorithm “naive\_random\_integers”. What constraint of the real Lotto is missing here? What should this algorithm do to overcome this limitation? (it is not asked to modify the algorithm)

To overcome the limitation of our first algorithm, we will use an auxiliary array to remember which numbers were already chosen by the algorithm, to avoid choosing them once more.

We now consider the function “random\_booleans”, in Figure 2. In this algorithm, `nbIntegers` represents the number of integers chosen; `result` represents a random integer; `hasBeenChosen` is an array of 45 booleans: it is initially filled with False, and a cell is changed to True when the number corresponding to the index of the cell is chosen by the algorithm).

---

Variables:  
*nbIntegers* and *result* are integers.  
*hasBeenChosen* is an array of booleans.

Instructions of the function:

```

1  hasBeenChosen ← 45 * [False] (An array of size 45 filled with False)
2  nbIntegers ← 0
3  While nbIntegers < 6
4      result ← randint(1, 45)
5      If not(hasBeenChosen[result - 1]), Then
6          hasBeenChosen[result - 1] ← True
7          nbIntegers ← nbIntegers + 1
8      Enf If
9  End While
10 Return hasBeenChosen

```

---

Figure 2: Function “random\_booleans”.

2. What is the purpose of the condition “If not(`hasBeenChosen`[`result` - 1])” on line 5?
3. Please complete the function “random\_integers” in Figure 3. This function uses the function “random\_booleans” given in the previous question.

---

Variables:  
*idBool* and *idInt* are integers.  
*randomBooleans* is an array of booleans.  
*randomIntegers* is an array of integers.

Instructions of the function:

```

1  randomBooleans ← random_booleans() (A call to the previous algorithm)
2  randomIntegers ← [0, 0, 0, 0, 0, 0]
3  idInt ← 0
4  For idBool from 0 to 44, Do
5      If randomBooleans . . . . ., Then
6          randomIntegers[idInt] ← idBool + 1
7          idInt ← idInt + 1
8      Enf If
9  End For
10 Return randomIntegers

```

---

Figure 3: Function “random\_integers”.

## 2 With Python

4. Write in Python the functions “random\_booleans” and “random\_integers”.
5. Write in Python a new function that asks the user 6 numbers (the ones they wish to play), runs the function “random\_integers” to get another array of 6 random numbers, compare the two arrays, and displays the numbers that are in both arrays.

BONUS Output an error when the user gives twice the same number in their input.