INFO-H-600 - Python
TP 3 - Sequences

Ex. 1. Write a function that finds the minimum in a list of numbers.

```
print(min_list([4, 3, 5, 6, 7]))      # prints 3
print(min_list([3, 12, 3, 2, 10]))    # prints 2
```

Ex. 2. Write a function that finds the name of the oldest person in a list of tuples.

```
print(elder([('Thomas', 20), ('Philippe', 24), ('Axel', 22)]))     # prints Philippe
print(elder([('Jean', 34), ('Laurent', 27), ('Michel', 33)]))      # prints Jean
```

Ex. 3. Write a function that prints a list `[1,2,3]` in the following way : `[1 -> 2 -> 3]`.

```
myprint([11, -2, 35, -46, 7])          # prints [11 -> -2 -> 35 -> -46 -> 7]
myprint([1, 2, 3, 5, 7, 11, 13, 17])   # prints [1 -> 2 -> 3 -> 5 -> 7 -> 11 -> 13 -> 17]
```

Ex. 4. Write a function that replace the elements of a list by their absolute value

```
li1 = [3, -4, -5, 6, -4]
replace_abs(li1)
print(li1)                  # prints [3, 4, 5, 6, 4]
```

Ex. 5. Write a function that swaps elements of a list according to their indices

```
li1 = [3, -4, -5, 6, -4]
swap(li1, 0, 3)
print(li1)                # prints [6, -4, -5, 3, -4]
```

Ex. 6. Write a function that receives a string composed of alphabetic characters and returns a string composed of the same letters but with the opposite case.

```
a = 'HelloWorld'
print(case_swap(a))      # prints hELLOwORLD
```

Ex. 7. Write a function that shuffles the elements of a list in the following manner: for each element, swap it with itself or with a following element chosen at random. Reminder: the function `randint(a,b)` from the `random` library returns an integer choosen at random in $[a, b]$.

```
li1 = [7, -8, 12, -44, 5]
shuffle(li1)
print(li1)   # prints for instance [5, -44, 7, -8, 12]
```

Ex. 8. Write a function that returns the longest list from a list of lists.

```
print(longest_list([[11, -2, 35, -46, 7],[2,1],[14, -2, 3, 1]]))   # prints [11, -2, 35, -46, 7]
print(longest_list([[4, -45, 1, 3], [1, 4], [7, -8, 12, -44, 5]])) # prints [7, -8, 12, -44, 5]
```

Ex. 9. Write a function that returns the longest increasing streak in a list.

```
print(longest_streak([11, -2, 35, -46, 7, -2, 3, 11]))  # prints [-2, 3, 11]
print(longest_streak([4, -45, 1, 3, 1, 4]))             # prints [-45, 1, 3]
```

Ex. 10. Write a function that returns the flattens a two dimensional list

```
print(flatten_list([[11, -2, 35], [-46, 7, -2], [3, 1]]))   # prints [11, -2, 35, -46, 7, -2, 3, 1]
print(flatten_list([[4, -45, 1, 3, 1], [4]]))               # prints [4, -45, 1, 3]
```

Ex. 11. Write a function that returns the index of the first occurence of a character in a string. If this character is not in the string, return *-1*. Try to write a function that is using only one return instruction.

Ex. 12. Write a function that receives as argument a list and two values, `old` and `new`, and that remplaces by `new` the first occurence of `old` in the list.

Ex. 13. Write a function that determines if a sequence composed of numbers is strictly increasing.

```
>>> increasing([1, 2, 5, 8]) # -> True
>>> increasing((1, 8, 5, 2)) # -> False
>>> increasing([1, 2, 2, 5]) # -> False
>>> increasing([5])          # -> True
>>> increasing([])           # -> True
>>> increasing("1258")       # -> True
```

Ex. 14. Write a function that determines weither a string given as argument is a palindrome. A palindrome is a sentence that can be read in both directions: the order of the letters is symetric.

```
>>> palindrome("hello")
False
>>> palindrome("radar")
True
```

INFO-H-600 - Programmation
TP 3 - Sequences
Corrections

Solution to the exercise 1:

```python
def min_list(numbers):
    if (len(numbers) == 0):
        raise Exception("empty list")

    minimum = numbers[0]
    for nombre in numbers:
        if nombre < minimum :
            minimum = nombre
    return minimum

print(min_list([4, 3, 5, 6, 7]))       # prints 3
print(min_list([3, 12, 3, 2, 10]))     # prints 2
```

Solution to the exercise 2:

```python
def elder(list_names):
    if (len(list_names) == 0):
        raise Exception("empty list")

    (nameEldest, ageEldest) = list_names[0]
    for (name,age) in list_names:
        if age > ageDoyen:
            (nameEldest, ageEldest) = (name, age)

    return nomDoyen

print(elder([('Thomas', 20), ('Philippe', 24), ('Axel', 22)]))      # prints Philippe
print(elder([('Jean', 34), ('Laurent', 27), ('Michel', 33)]))       # prints Jean
```

Another solution :

```python
def elder(list_names):
    if (len(list_names) == 0):
        raise Exception("empty list")

    iMax = 0                   # indice du doyen
    for i in range(1,len(list_names)):
        if list_names[i][1] > list_names[iMax][1]:
            iMax = i

    return list_names[iMax][0]
```

Solution to the exercise 3:

```python
def myprint(li):
    res = "["
    if len(li) > 0:
        res += str(li[0])
    for element in li[1:]:
        res += ' -> ' + str(element)

    res += "]"
    print(res)

myprint([11, -2, 35, -46, 7])          # prints [11 -> -2 -> 35 -> -46 -> 7]
myprint([1, 2, 3, 5, 7, 11, 13, 17])   # prints [1 -> 2 -> 3 -> 5 -> 7 -> 11 -> 13 -> 17]
```

Solution to the exercise 4:

```python
def replace_abs(listeNombres):
    for i in range(len(listeNombres)):
        if listeNombres[i] < 0:
            listeNombres[i] = -listeNombres[i]
```

```python
li1 = [3, -4, -5, 6, -4]
replace_abs(li1)
print(li1)                    # prints [3, 4, 5, 6, 4]
```

Solution to the exercise 5:

```python
def swap(li, i1, i2):
    li[i1], li[i2] = li[i2], li[i1]

li1 = [3, -4, -5, 6, -4]
swap(li1, 0, 3)
print(li1)                # prints [6, -4, -5, 3, -4]
```

Solution to the exercise 6:

```python
def case_swap(word):
    res = ""
    for letter in word:
        if letter.lower() == letter:
            res += letter.upper()
        else:
            res += letter.lower()
    return res

a = 'HelloWorld'
print(case_swap(a))      # prints hELLOwORLD
```

Solution to the exercise 7:

```python
import random

def swap(li, i1, i2):
    li[i1], li[i2] = li[i2], li[i1]

def shuffle(list_numbers):
    for i in range(len(list_numbers)-1):
        place = random.randint(i,len(list_numbers)-1)
        swap(list_numbers,i,place)


li1 = [7, -8, 12, -44, 5]
shuffle(li1)
print(li1)   # prints for instance [5, -44, 7, -8, 12]
```

Solution to the exercise 8:

```python
def longest_list(lists):
    if (len(lists) == 0):
        raise Exception("empty list")

    long_list = lists[0]
    for liste in lists:
        if len(liste) > len(long_list):
            long_list = liste

    return long_list

print(longest_list([[11, -2, 35, -46, 7],[2,1],[14, -2, 3, 1]]))   # prints [11, -2, 35, -46, 7]
print(longest_list([[4, -45, 1, 3], [1, 4], [7, -8, 12, -44, 5]])) # prints [7, -8, 12, -44, 5]
```

Solution to the exercise 9:

```python
def longest_streak(li):
    if len(li) == 0:
        return []

    longest_streak = [li[0]]
    streak = [li[0]]

    for i in range(1, len(li)):
        print(i, li[i], streak, longest_streak)
        if li[i] >= streak[-1]:    # non strictly increasing
            streak.append(li[i])
        else:
            if len(streak) > len(longest_streak):
                longest_streak = streak
            streak = [li[i]]

    if len(streak) > len(longest_streak):
        longest_streak = streak
    return longest_streak

print(longest_streak([11, -2, 35, -46, 7, -2, 3, 11]))  # prints [-2, 3, 11]
print(longest_streak([4, -45, 1, 3, 1, 4]))             # prints [-45, 1, 3]
```

Solution to the exercise 10:

```
def flatten_list(2Dlist):
    res = []
    for li in 2Dlist:
        for element in li:
            res.append(element)
    return res

print(flatten_list([[11, -2, 35], [-46, 7, -2], [3, 1]]))   # prints [11, -2, 35, -46, 7, -2, 3, 1]
print(flatten_list([[4, -45, 1, 3, 1], [4]]))               # prints [4, -45, 1, 3]
```

Solution to the exercise 11:

```
def find1(ls, c):
    i = 0
    while i < len(ls) and ls[i] != c:
        i += 1
    #Affirmation: i == len(ls) ou ls[i] == c
    if i == len(ls):
        i = -1
    return i
```

Solution to the exercise 12:

```
def replace1(ls, old, new):
    i = 0
    while i < len(ls) and ls[i] != old:
        i += 1
    #Affirmation: i == len(ls) ou ls[i] == old
    if i < len(ls):
        ls[i] = new
```

Solution to the exercise 13:

```
def increasing(ls):
    i = 1
    while i < len(ls) and ls[i] > ls[i - 1]:
        i += 1
    #Affirmation: i >= len(ls) ou ls[i] <= ls[i - 1]
    return i >= len(ls)
```

Solution alternative...

```
def increasing(seq):
    result = True
    if len(seq) >= 2:
        i = 1
        while i < len(seq) and result:
            result = seq[i] > seq[i - 1]
            i += 1
    return result
```

Solution to the exercise 14:

```
def palindrome(word):
    g = 0
    d = len(word)-1
    while g<d and word[g] == word[d]:
        g+=1
        d-=1
    return word[g] == word[d]
```

Another alternative...

```
def palindrome(word):
    i = 0
    ok = True
    while i < len(word)/2 and ok:
        if word[i] != word[len(word)-i-1]:
            ok = False
        i+=1

    return ok
```