TP 2 : Fonctions, listes, boucles imbriquées

2 Premières fonctions.

Exercice 1.

```
def absolue(x):
    if x>0:
        return x
    else:
        return -x

def fact(n):
    p=1
    for i in range(1,n+1):
        p=p*i
    return p
```

Exercice 2. *Utilisation d'une fonction dans une autre fonction.*

1.
```
def max2(a,b):
    if a>b:
        return a
    else:
        return b
```

2. Le plus efficace :

```
def max3(a,b,c):
    return max2(a,max2(b,c))
```

3. Rappel : pour calculer le maximum d'une liste, utiliser une variable contenant le maximum temporaire, initialisée par exemple au premier élément de la liste). Ensuite, parcourir la liste et comparer tous les éléments à ce maximum temporaire.

```
def max_liste(L):
    m=L[0]
    for i in range(1,len(L)):
        m=max2(m,L[i])
    return m
```

3 Fonctions basiques sur les listes

Exercice 3. *Création de listes.*

1. Par compréhension, ou en utilisant `append` :

```
def carres(n):
    return [i**2 for i in range(n)]
```

```
def carres(n):
    L=[]
    for i in range(n):
        L.append(i**2)
    return L
```

2. Même chose :

```
def non_div_3(n):
    return [i for i in range(n) if i%3!=0]
```

```
def non_div_3(n):
    L=[]
    for i in range(n):
        if i%3>0:
            L.append(i)
    return L
```

3.
```
def addition(L1,L2):
    return [L1[i]+L2[i] for i in range(len(L1))]
```

On aurait pu également utiliser `append` ici.

4 Algorithmes de tri et boucles imbriquées

Exercice 4. *Tri par sélection du minimum.* Il suffit de suivre le pseudo-code. Le calcul de `m` cache une boucle `for` : le code est très semblable à celui du calcul du maximum, bien comprendre la différence et ne pas confondre les indices et les éléments.

```
def tri_selection(L):
    n=len(L)
    for i in range(n-1):
        m = i
        for j in range(i+1,n):
            if L[j]<L[m]:
                m=j
        if m>i:
            x=L[i]
            L[i]=L[m]
            L[m]=x
```

Exercice 5. *Tri à bulles.*

1. Attention, il n'y a que `len(L)-1` couples successifs d'indice, écrire `len(L)` à la place de `len(L)-1` mène à un dépassement d'indice lorsque `i` atteint `len(L)-1` à cause de l'accès à `L[i+1]`.

```
def passage(L):
    for i in range(len(L)-1):
        if L[i]>L[i+1]:
            x=L[i]
            L[i]=L[i+1]
            L[i+1]=x
```

2.

```
def tri_bulles(L):
    for i in range(len(L)-1):
        passage(L)
```

Exercice 6. Le but était d'écrire des boucles imbriquées. Certains ont feinté (c'est naturel mais ce n'était pas le but) en utilisant le fait que `'*'*5` vaut `'*****'`, par exemple.

```
def carre(n):
    for i in range(n):
        for j in range(n):
            print('*',end="")
        print()

def triangle(n):
    for i in range(n):
        for j in range(i+1):
            print('*',end="")
        print()

def triangle2(n):
    for i in range(n-1,-1,-1):
        for j in range(i+1):
            print('*',end="")
        print()

def carre2(n):
    for i in range(n):
        print('*',end="")
    print()
    for i in range(n-2):
        print('*',end="")
        for j in range(n-2):
            print('o',end="")
        print('*',end="")
        print()
    for i in range(n):
        print('*',end="")
    print()
```