```
# ========================================== #
# ========================================== #
# IPT MATHS SPE                              #
# Introduction (Séance I de 2017-2018)       #
# Exercices de niveau I - Propositions de code #
# Septembre 2017                             #
# ========================================== #
# ========================================== #

# ============ #
# EXERCICE I-1 #
# ============ #

"""
L'occasion de revoir quelques syntaxes bien pratiques...
"""

# A l'aide de définitions explicites
print([2*i for i in range(20)])
print([2*i+1 for i in range(20)])

# A l'aide de sous-listes
L = [i for i in range(40)]
print(L[::2])
print(L[1::2])

# ============ #
# EXERCICE I-2 #
# ============ #

"""
Ici, on valide fondamentalement deux points :
(1) La liste L est de longueur n
(2) Tout entier compris entre 1 et n appartient à la liste L
"""

def ValidPermut(n,L):
    if len(L) != n:
        return False
    for i in range(1,n+1):
        if i not in L:
            return(False)
    return(True)

# ============ #
# EXERCICE I-3 #
# ============ #

def ASC_List(L):
    for i in range(len(L)-1):
        if L[i+1] < L[i]:
            return(False)
    return(True)

"""
Version récursive...
"""

def ASC_List_rec(L):
    if len(L) == 1:
```

```
        return(True)
    else:
        if L[1] >= L[0]:
            return(ASC_List_rec(L[1:]))
        else:
            return(False)

"""
Comme suggéré dans l'énoncé, nous utilisons dans les deux codes ci-après la
fonction ASC_List.
Dans le premier code, la liste L est inversée, ce qui est loin d'être
souhaitable mais permet, si besoin est, de (re)prendre conscience du fait
que
c'est bien l'adresse d'une liste qui est passée en argument d'une
fonction...
Le second code, en revanche, commence par copier la liste passée en argument
et
travaille sur la copie. La liste initiale est donc conservée. Ceci dit, du
fait
de la copie, l'ocupation mémoire est doublée...
"""

def DESC_List(L):
    L.reverse()
    return(ASC_List(L))

def DESC_List_2(L):
    M = list(L)
    M.reverse()
    return(ASC_List(M))

# ============ #
# EXERCICE I-4 #
# ============ #

"""
Dans le code proposé ci-dessous, on fait appel à la méthode pop. Celle-ci
renvoie ET détruit l'élément de la liste dont on a précisé l'indice.
"""

def PermCirc(L):
    if len(L) != 1:
        L.insert(0, L.pop(-1))

# ============ #
# EXERCICE I-5 #
# ============ #

def LenStrings(L):
    L2 = []
    for e in L:
        L2.append(len(e))
    return(L2)

# ============ #
# EXERCICE I-6 #
# ============ #

def IntTrap(f,a,b,N):
    if a == b:
```

```
        return(0)
    else:
        h = (b - a) / N
        I = (f(a) + f(b)) * 0.5
        for k in range(1,N):
            I += f(a + k*h)
        return(I * h)

# Pour tester :
def f1(x):
    return(x**2)

# La valeur affichée doir être proche de 1/3 ...
print(IntTrap(f1,0,1,1000))
```