```python
# Subtilités Python pour variables mutables.py

## Question 1 sur l'affichage associé à un tuple de variables mutables
# Les fonctions sont d'abord évaluées de gauche à droite. Leur résultat est mis dans le programme principal et pointe toujours sur la même liste. L'affichage est ensuite effectuée de gauche à droite. Voir ci-dessous.
def f(l):
    l.append(2)
    return l

def g(l):
    l.append(5)
    return l

t = [0]
print(t, f(t)) # Python évalue f(t) et renvoie au programme principal un nom avec une adresse qui pointe sur t; une fois le résultat obtenu, il affiche toutes les variables : [0, 2] [0, 2]

t = [0]
print(f(t), t) # Python évalue f(t) et renvoie au programme principal un nom avec une adresse qui pointe sur t; une fois le résultat obtenu, il affiche toutes les variables : [0, 2] [0, 2]

t = [0]
print(t, f(t), g(t)) # Python évalue f(t) et renvoie au programme principal un nom avec une adresse qui pointe sur t; ensuite il évalue g(t) (dans cet ordre de gauche à droite) et renvoie au programme principal un nom avec une adresse qui pointe sur t; une fois les résultats obtenus, il affiche toutes les variables : [0, 2, 5] [0, 2, 5] [0, 2, 5]

t = [0]
print(t, g(t), f(t), g(t)) # [0,5,2,5] [0,5,2,5] [0,5,2,5] [0,5,2,5]


## Question 2 sur la concaténation associée à un tuple de variables mutables
#  t + l + m = (t + l) + m. Si les variables sont le résultat de fonctions, elles sont évaluées progressivement de gauche à droite. La concaténation se fait au fur et à mesure. On rappelle que la concaténation crée une nouvelle liste issue de la copie de 2 listes. Voir ci-dessous.

def f(l):
    l.append(2)
    return l

def g(l):
    l.append(5)
    return l

t = [0]
print(t + f(t)) # Python évalue f(t) et renvoie au programme principal un nom avec une adresse qui pointe sur t; puis concatène avec t (qui a donc changé, ce qui donne [0, 2, 0, 2])

t = [0]
print(f(t) + t) # Python évalue f(t) et renvoie au programme principal un nom avec une adresse qui pointe sur t; puis concatène avec t (qui a donc changé, ce qui donne [0, 2, 0, 2])

t = [0]
print(f(t) + t + g(t)) # Python l'interprète comme-suit : print((f(t) + t) + g(t)). Ainsi, f(t) est évalué, la concaténation est effectuée, g(t) est évalué et concaténé avec ce qui précède, ce qui donne [0, 2, 0, 2, 0, 2, 5]

t = [0]
print(t + f(t) + g(t)) # Python l'interprète comme-suit : print((t + f(t)) + g(t)). Ainsi, f(t) est évalué, la concaténation est effectuée, g(t) est évalué et
```

```
concaténé avec ce qui procède, ce qui donne [0, 2, 0, 2, 0, 2, 5]
46|
47| t = [0]
48| print(f(t) + g(t) + t) # Python l'interprète comme-suit : print((f(t) + g(t)) +
t). Ainsi, Python évalue f(t) et renvoie au programme principal un nom avec une
adresse qui pointe sur t (maintenant [0,2]); Python évalue g(t) et renvoie au
programme principal un nom avec une adresse qui pointe sur t (maintenant [0,2,5]);
puis la concaténation est effectuée; enfin le résultat est concaténé avec t, ce qui
donne [0, 2, 5, 0, 2, 5, 0, 2, 5]
49|
50| t = [0]
51| print(f(t) + g(t) + t + g(t)) # Python l'interprète comme-suit : print(((f(t) +
g(t)) + t) + g(t)). On obtient: [0, 2, 5, 0, 2, 5, 0, 2, 5, 0, 2, 5, 5].
```