Manipulation de fonctions

Exercice - niveau intermédiaire

On vous demande d'écrire une fonction `numbers` qui prend en argument une liste d'entiers, et qui retourne un tuple contenant

- la somme
- le produit
- le minimum
- le maximum

des éléments de la liste

```
def numbers(l):
    "<votre_code>"


# pour vérifier votre code
from corrections.w4_functional import exo_numbers
exo_numbers.correction(numbers)
```

Exercice - niveau avancé

À présent nous allons écrire une version très simplifiée de l'outil qui est utilisé dans ce cours pour corriger les exercices. Vous remarquerez que les fonctions de correction, comme par exemple `exo_numbers.correction` ci-dessus, prennent en argument la fonction qu'il est question de corriger.

On vous demande d'écrire une fonction `validation` qui prend en argument

- deux fonctions `f` et `g`; imaginez que l'une d'entre elles fonctionne et qu'on cherche à valider l'autre; dans cette version simplifiée toutes les fonctions acceptent exactement un argument
- une liste d'entrées `entrees`; vous pouvez supposer que chacune de ces entrées est dans le domaine de `f` et de `g` (dit autrement, on peut appeler `f` et `g` sur chacune des entrées sans craindre qu'une exception soit levée);

Le résultat attendu pour le retour de `validation` est une liste qui contient autant de booléens que d'éléments dans entrées, chacun indiquant si avec l'entrée correspondante on a pu vérifier que `f(entree) == g(entree)`.

Dans cette première version de l'exercice vous pouvez enfin supposer que les entrées ne sont

pas modifiées par `f` ou `g`.

```
def validation(f, g, entrees):
    "<votre code>"


# pour vérifier votre code
from corrections.w4_functional import exo_validation
exo_validation.correction(validation)
```

Pour information:

- `factorial` correspond à `math.factorial`
- `fact` et `broken_fact` sont des fonctions implémentées par nos soins, la première est correcte alors que la seconde retourne 0 au lieu de 1 pour l'entrée 0.