Programmation Fonctionnelle : des concepts aux applications web (NFP119)

Le type `Optional`

En Python, le type `Optional[T]` est équivalent à `T | None` (ou `Union[T, None]`). Dans ce TP, nous utiliserons la notation `Optional[T]` (même si par la suite nous utiliserons plutôt `T | None` qui est désormais le choix recommandé).

Implanter les fonctions suivantes :

1. `get_opt[A](opt: Optional[A], a: A) -> A`
   `get_opt(opt, a)` renvoie `a` si `opt` est `None`, et renvoie `v` si `opt` est une valeur `v`.

2. `is_some[A](opt: Optional[A]) -> bool`
   `is_some(opt)` renvoie `False` si `opt` est `None`, et renvoie `True` sinon.

3. `val_of[A](opt: Optional[A]) -> A`
   `val_of(opt)` lève l'exception prédéfinie `ValueError` si `opt` est `None` et renvoie `v` si `opt` est une valeur `v`.

4. `filter[A](f: Callable[[A], bool], a: A) -> Optional[A]`
   `filter(f, a)` renvoie `a` si `f(a)` est `True` et `None` sinon.

5. `map[A, B](f: Callable[[A], B], opt: Optional[A]) -> Optional[B]`
   `map(f, opt)` renvoie `None` si `opt` est `None` et `f(v)` si `opt` est une valeur `v`.