Ce TP se concentre sur la gestion d'erreur et les décorateurs.

1 Les erreurs

Question 1

Écrire une fonction *le_juste_prix* qui prend en entrée un entier N, génère un nombre aléatoire entre 0 et N, puis invite l'utilisateur à deviner ce nombre, en indiquant si un nombre entré par l'utilisateur est plus grand ou plus petit. On utilisera la fonction *randint* du module *random* et la commande *input()*

Question 2

Que se passe-t-il si l'utilisateur n'entre pas un nombre ? Corriger cela avec les commandes *try* et *except*.

2 Les décorateurs

Un décorateur a la forme suivante et s'appelle comme suit :

```
def decorateur(fonction):
    def fonction_aux(*args): #*args signifie qu'on ne sait pas a priori combien d'argument notre fonction auxiliaire prendra
        print("Quel␣jolie␣fonction␣!") #Instructions
        val = fonction(*args) #Il faut executer notre fonction a un moment et stocker son resultat
        print("Elle␣s'appelle",  fonction.__name__) #Instructions
        return val #En general on veut quand meme le resultat de notre fonction
    return fonction_aux

@decorateur
def plus(x):
    return x+1
```

Question 3

Écrire un décorateur *temps* qui affiche le temps de calcul d'une fonction. On pourra utiliser la fonction *time* du module *time*.

Question 4

Écrire un décorateur *stock* qui enregistre les valeurs déjà calculées par une fonction afin de ne pas les calculer plusieurs fois en cas de plusieurs appels. L'essayer avec le code suivant :

```
@stock
def fibo(n):
    if n <= 1 :
        return n
    return fibo(n-1) + fibo(n-2)
```

Question 5

Écrire un décorateur *trace* qui affiche les différents appels qu'effectue une fonction récursive. Exemple de résultat attendu :

```
>>> fibo(3)
 fibo <- 3
   fibo <- 2
     fibo <- 1
     fibo -> 1
     fibo <- 0
     fibo -> 0
   fibo -> 1
   fibo <- 1
   fibo -> 1
 fibo -> 2
```

Question 6

Écrire un décorateur *erreur* qui tente d'effectuer un calcul, et génère un message d'erreur plus joli que celui de python en cas d'erreur.