Parler aux tortues

Consignes

Afin de réaliser ce TP, vous devez recopier le fichier `langage-tortue.py` qui se trouve dans le dossier de votre groupe dans `Echange` et renommez le en `langage-tortue-NOM-PRENOM.py` si vous êtes seul ou `langage-tortue-NOM1-NOM2.py` si vous êtes deux.
Les consignes sont réparties sur ce document et sur la feuille papier. La plupart des exercices papier ont pour but de vous aider à écrire les fonctions. Il vaut donc mieux les faire avant de passer à l'exercice de la feuille sur ordinateur, sauf mention contraire.
À la fin de la séance vous devez rendre la feuille papier, une par élève, et votre fichier, qui doit être copié dans de dossier `devoir` qui se trouve dans le dossier `Echange` de votre groupe.
Vous pouvez rajouter en commentaire les tests que vous avez fait pour chacune des fonctions qui renvoie une valeur, comme dans cet exemple :

```
def remplacer(texte, v1, v2):
    res = ''
    for v in texte:
        if v == v1:
            res = res + v2
        elif v == v2:
            res = res + v1
        else:
            res = res + v
    return res

"""
>>> remplacer("chien", "c", "h")
'hcien'
>>> remplacer("chien", "j", "h")
'cjien'
>>> remplacer("chien", "j", "k")
'chien'
"""
```

Vous n'avez pas à vous limiter aux tests donnés dans les exemples.

Rappels sur la tortue

Pour utiliser la tortue il faut mettre ces lignes (qui sont déjà dans le fichier) :

```
from turtle import * # pour utiliser la tortue
setup(640, 480) # taille de la fenetre
```

Lorsque vous exécutez le programme, une fenêtre s'ouvre et vous pouvez voir un petit triangle au milieu. C'est la tortue que vous pouvez contrôler pour faire des dessins.
Pour fermer la fenêtre de la tortue, il faut appuyer sur le bouton "Stop" ou taper :

```
>>> bye()
```

Les commandes de base de la tortue sont :

Commande Python	Remarques
`forward(N)`, `fd(N)`	Avancer de `N` pixels
`backward(N)`, `back(N)`, `bk(N)`	Reculer de `N` pixels
`right(N)`, `rt(N)`	Tourner à droite de `N` degrés
`left(N)`, `lt(N)`	Tourner à gauche de `N` degrés
`up()`	Lever le crayon
`down()`	Baisser le crayon
`reset()`	Tout réinitialiser
`setheading(angle)`	Tourne la tortue vers `angle`^{o}

La commande `setheading(angle)` permet de diriger la tortue vers l'angle indiqué : 0 pour la droite, 90 pour le haut, 180 pour la gauche et −90 ou 270 pour le bas.

Des mots pour tracer

Nous allons construire de nouvelles fonctions permettant de tracer des figures à partir de commandes données à l'aide de textes. Dans la suite de ce document on appellera **instructions** un texte indiquant les actions à effectuer, comme `'hgbdh'`.

```
LONGUEUR = 30

def interactif():
    rep = ""
    while rep != ".":
        rep = input("Nouvelle direction h/b/g/d ? ")
        if rep == "h":
            setheading(90)
            forward(LONGUEUR)
        elif rep == "b":
            setheading(-90)
            forward(LONGUEUR)
        elif rep == "g":
            setheading(180)
            forward(LONGUEUR)
        elif rep == "d":
            setheading(0)
            forward(LONGUEUR)
```

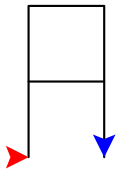
La variable `LONGUEUR` sera utilisée tout au long du fichier pour indiquer la longueur des déplacements. Elle ne sera jamais changée pendant l'exécution des fonctions. On appelle cela une **constante**. La valeur peut être changée dans le fichier, mais cela nécessite de recharger le script.

Exercice 1 : Tester la fonction `interactif()` et déterminer ce qu'il faut rentrer pour arrêter son exécution. Indiquer votre réponse sur la feuille.

Exercice 2 : Modifier la fonction `interactif` pour qu'elle renvoie à la fin de l'exécution un texte composé des commandes tapées. Il faut rajouter une nouvelle variable `instructions` qui sera mise à jour à chaque tour de boucle. Vous pouvez utiliser cette fonction pour faire l'exercice de la feuille.

Exercice 3 : Compléter le code ci-dessous pour que `executer(instructions)` fasse le dessin indiqué par `instructions`. Il faut rajouter plusieurs lignes et il n'y a pas de `return` à mettre à la fin.

```
def executer(instructions):
    for instr in instructions:
        if instr == "h":
            ...
        elif ...:
            ...
        ...
```



Le code ci-dessous permet d'obtenir la figure ci-contre.

```
>>> executer("hhdbgdb")
```

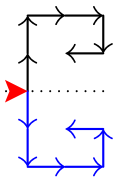
Exercice 4 : Rajouter dans les fonctions `interactif` et `executer` la possibilité de lever le stylo avec `'l'` et de le remettre un position d'écriture avec `'e'`. Les commandes Python sont `up()` et `down()`. La tortue ne doit pas se déplacer avec ces intructions.

Répétitions et symétries

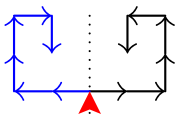
Exercice 5 : Compléter la fonction `repeter(instructions, n)` qui renvoie une nouvelle suite d'instructions où `instructions` est répétée `n` fois.

```
>>> repeter('hghgbd', 3)
'hghgbdhghgbdhghgbd'
```

Exercice 6 : Compléter la fonction `sym_h(instructions)` qui renvoie une nouvelle suite d'instructions qui correspond au symétrique horizontal de la figure produite par `instructions`.



Exercice 7 : Compléter la fonction `sym_v(instructions)` qui renvoie une nouvelle suite d'instructions qui correspond au symétrique vertical de la figure produite par `instructions`.



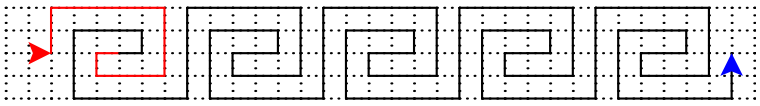
Exercice 8 : Compléter la fonction `inverser(instructions)` qui renvoie une nouvelle suite d'instructions qui inverse l'ordre des instructions de `instructions`. Il faut modifier les commandes lever et écrire, mais pas les autres. Il faut par contre recopier les instructions de droite à gauche.

```
>>> inverser('hhgggbdd')
'ddbggghh'
>>> inverser('glhheddd')
'dddlhheg'
```

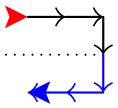
Exercice 9 : Déterminer la suite d'instructions à donner pour obtenir la figure suivante avec ces commandes, et la mettre sur la feuille :

```
base = ...
executer(repeter(base + inverser(base), 5))
```

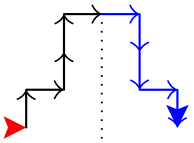
Les instructions de `base` correspondent à la partie en rouge.



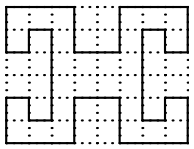
Exercice 10 : Écrire une fonction `sym_h_i(instructions)` qui renvoie une nouvelle suite d'instructions qui correspond au symétrique horizontal de la figure produite par `instructions`, mais en partant du point d'arrivée. Il faut donc également inverser l'ordre des instructions et faire attention avec les commandes lever et écrire.



Exercice 11 : Écrire une fonction `sym_v_i(instructions)` qui renvoie une nouvelle suite d'instructions qui correspond au symétrique vertical de la figure produite par `instructions`, mais en partant du point d'arrivée.



Exercice 12 : En utilisant les fonctions précédentes et le motif de base, dessiner la figure ci-contre. Il faut utiliser la plus petite suite d'instructions possible.



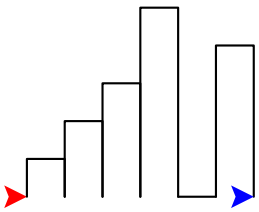
Tracer des nombres

Dans cette partie, nous allons essayer de tracer des "histogrammes" à partir de textes contenant des chiffres. Pour convertir en nombre un texte correspondant à un entier, il faut utiliser la commande `int(nombre)` :

```
>>> int("145")
145
>>> int("9")
9
```

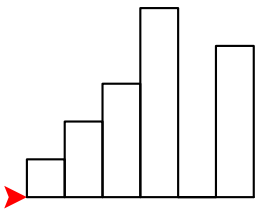
Pour déplacer la tortue, vous n'utiliserez que les commandes `forward(longueur)`, `backaward(longueur)`, `left(angle)` et `right(angle)`.

Exercice 13 : Écrire une fonction `histogramme(nombre)` qui trace un histogramme correspondant au `nombre`, donné sous forme de texte. Chaque barre correspond à un chiffre. La hauteur correspond à ce chiffre multiplié par `LONGUEUR` et la largeur est `LONGUEUR`.



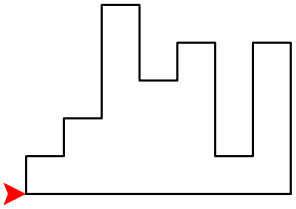
```
>>> histogramme('123504')
```

Exercice 14 : Modifier la fonction `histogramme` pour que la tortue revienne à sa position initiale, et se tourne vers la droite. Le nombre de chiffres est obtenu avec `len(nombre)`.



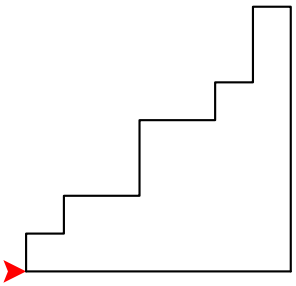
```
>>> histogramme('123504')
```

Exercice 15 : Compléter la fonction `histogramme2(nombre)` qui fonctionne comme `histogramme` mais qui ne revient pas à la hauteur initiale à chaque fois. Au contraire, elle ne fait que les déplacements verticaux nécessaires. La tortue revient à sa position initiale à la fin.



```
>>> histogramme2('1253414')
```

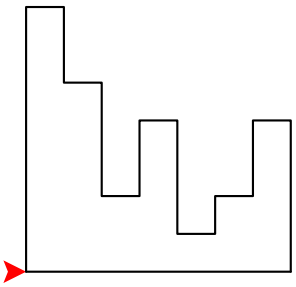
Exercice 16 : Compléter la fonction `histomax(nombre)` qui fonctionne comme `histogramme2` sauf que la tortue ne redescent jamais, sauf à la fin. Si le chiffre traité est inférieur au précédent, la tortue reste à la même hauteur. La tortue revient à sa position initiale à la fin.



```
>>> histomax('1214257')
```

Exercice 17 : Compléter la fonction `histogramme3(nombre)`, qui fait comme `histogramme2`, sauf que les chiffres de `nombre` sont lus de droite à gauche.

```
def histogramme3(nombre):
    ...
    for i in range(len(nombre)-1, -1, -1):
        #le chiffre est nombre[i]
        ...
```



```
>>> histogramme3('4214257')
```