Modélisation et robotique

Exercices 4

1 Révisions du cours

Question A. Valeurs de retour ou effet de bord ? Selon la définition choisie parmi : 0.5 pt 3 min

1.
```
def double(x):
    return 2 * x
```
2.
```
def double(x):
    print(2 * x)
```
3.
```
def double(x):
    print(2 * x)
    return 42
```

dire ce qui sera affiché par l'instruction `print("double de trois %s" % double(3))`.

Question B. Fonctions retournant une valeur. Quel est le rôle de `a` dans la définition de la fonction suivante ? Donner des exemples d'utilisation de cette fonction. 0.5 pt 3 min

```
def avancer_toute():
    a = 0
    while not obstacle():
        av()
        a += 1
    return a
```

Question C. Aller-retour. Définir une fonction `aller_retour()` qui fait que le personnage avance dans la direction actuelle jusqu'à un premier obstacle puis revient à la position initiale. 1 pt 6 min

Question D. Orientation absolue. Au début du programme le personnage est orienté vers l'ouest. Le but de cet exercice est de définir trois fonctions : `tgo()` qui tourne à gauche, `tdo()` qui tourne à droite et `quelle_orientation()` qui retourne l'orientation actuelle sous la forme d'une chaîne de caractère parmi '`sud`', '`nord`', '`est`', '`ouest`'. 2 pt 12 min

Pour cela nous avons commencé à écrire le code suivant qu'il ne reste plus qu'à compléter.

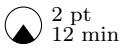
```
orientation = 0

def tgo():
    ...
    orientation += 1
    tg()

def quelle_orientation():
    if orientation % 4 == 1:
        return 'nord'
    ...
```

— Quel est le sens du symbole `%` dans ce code ?
— Que doit-on écrire à la première ligne de `tgo()` et pourquoi ?
— Compléter le programme.

Question E. Effet de bord ou passage de valeur ? Grâce à votre code, il est possible d'exécuter le programme suivant :

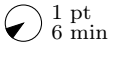
```
def dto():
    tgo(); tgo()

def trajet():
    av(); tgo(); av(); av(); tgo(); av(): av(); dto()

trajet(); print(quelle_orientation())
```

Proposer une solution pour écrire le même programme sans utiliser de variable globale. Pourquoi est-ce mal adapté ici ?

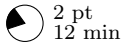


2 Petits programmes

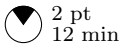
Question F. Orientation initiale. Ajouter une fonction `orienter(point_cardinal)` qui prend en entrée un point cardinal ('`sud`', '`est`', etc.) et oriente le personnage dans cette direction.



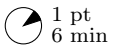
Question G. Compteur kilométrique. Sur le même modèle que l'orientation absolue, proposer une fonction `avk()` qui remplace `av()` et une fonction `kilometrage()` qui retourne le nombre de cases parcourues par le personnage depuis le début du programme.



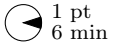
Question H. Ingrédients. Définir une fonction `types_elements(panier)` qui prend en entrée un liste de légumes (par exemple, la liste `['chou', 'chou', 'salade']`) et retourne une liste où chaque élément du panier apparaît une seule fois.



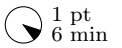
Question I. Pas de fruits. Le panier peut contenir toute sorte de légumes mais aussi quelques fruits parmi une liste connues : `types_de_fruits`. Écrire une fonction `legumes` qui prend en entrée le panier et la liste des types de fruits et retourne les éléments du panier qui ne sont pas des fruits.



Question J. Pas de légumes. Écrire une fonction `fruits` qui, à partir des mêmes paramètres, `panier` et `types_de_fruits`, retourne tous les fruits du panier.



Question K. Les légumes puis les fruits. Toujours à partir des mêmes paramètres, définir une fonction `ranger` qui retournera une liste contenant d'abord tous les légumes du panier puis tous les fruits.



3 Problèmes

Vous pouvez utiliser les fonctions définies jusqu'à maintenant.

Question L. Trouver le mur. Votre personnage est dans une cour rectangulaire, le long d'un des murs, orienté vers l'est. Trouvez de quel côté est le mur. 1.5 pt 9 min

Recommandation : définir et utiliser une fonction `obstacles()` qui retourne toutes les directions dans lesquelles se trouve un obstacle dans la case adjacente au personnage. Votre fonction retournera son résultat sous la forme d'une liste de points cardinaux. Cette fonction est utile pour cette question et elle le sera encore plus, plus tard.

Question M. Revenir au point de départ. Choisissez une direction et faites le tour de la cour en revenant exactement au point de départ du personnage et en respectant son orientation initiale. 1.5 pt 9 min

Question N. Retenir la sortie. On se donne un système de coordonnées cartésiennes pour repérer le personnage sur la grille de la simulation. On utilise pour cela des paires d'entiers (x, y) où la coordonnée `x` augmente de `1` à chaque déplacement vers l'est et la coordonnée `y` augmente de `1` à chaque déplacement vers le nord (ces coordonnées respectives diminuent dans les directions opposées).

Une porte de sortie vient de s'ouvrir dans l'un des murs de la cour (elle n'est pas près d'un coin). En supposant que votre personnage est initialement aux coordonnées (40, 67), faites lui trouver les coordonnées de la sortie et revenir à sa position initiale. Votre fonction retournera la sortie sous la forme d'une paire de coordonnées. 3 pt 18 min

Question O. Généraliser (bonus). Généraliser votre fonction au cas où il y a plusieurs sorties (mais aucune dans un coin) en retournant la liste des sorties. Cette fois-ci on veut les coordonnées de la sortie mais aussi la direction qu'il faut suivre pour l'emprunter. Par exemple, si on a une sortie dans le mur nord de coordonnées (45, 70) on retiendra dans la liste le triplet `(45, 70, 'nord')`.

Le personnage a maintenant la liste des sorties et il est retourné à son point de départ. Il veut emprunter la sortie qui nécessitera le moins de changements de case de sa part. Aidez-le !

Même question si le personnage pouvait se déplacer en diagonale à l'aide de fonctions `avancer_gauche()` et `avancer_droite()`. 3 pt 18 min