Thèmes du programme abordés dans ce TD :

- Constructions élémentaires d'un langage de programmation
- Structures de contrôle : boucles et tests
- Fonctions
- Utilisation d'une bibliothèque

Méthode

Le Rurple logiciel permet de déplacer un robot sur une scène en utilisant le langage de programmation Python augmenté de quelques fonctions spécifiques.

Tout programme Rurple commence par l'import de toutes les fonctions du module / bibliothèque `rurple` avec l'instruction `from rurple import *`.

Des exemples et des tutoriels sont disponibles sur ce site :

http://rurple.free.fr/index.html

Le logiciel Rurple peut être téléchargé depuis ce lien

http://rurple.free.fr/outils/rurple.msi

Voici les principales fonctions qui permettent de déplacer le robot ou de le faire interagir avec son environnement :

- « `avance( )` » (ou appui sur touche A) : le robot avance d'une case devant lui.
- « `gauche( )` » (ou appui sur touche G) : le robot effectue un quart de tour vers la gauche.
- « `depose( )` » (ou appui sur touche D) : le robot dépose une bille sur la case où il se trouve.
- « `prends( )` » (ou appui sur touche P) : le robot ramasse une bille sur la case où il se trouve.

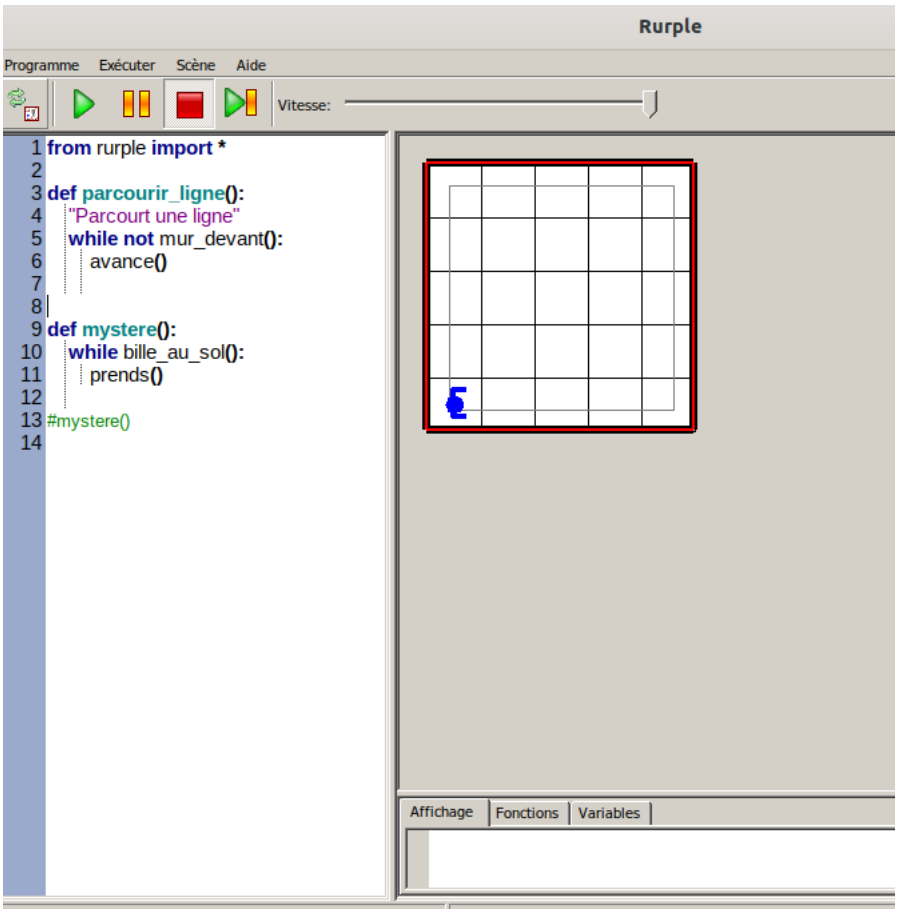
L'interface graphique est composée d'une barre de menu classique en haut et de trois fenêtres :

- La fenêtre de gauche accueille le programme (ou script) des *instructions* que l'on souhaite faire exécuter au robot. Celui-ci lit le programme de haut en bas et exécute les instruc*tions de façon séquentielle. Certaines instructions comme les conditions ou les boucles* permettent de faire des sauts en avant ou en arrière dans le *code source* du programme. On peut enregistrer/ouvrir/créer un nouveau un programme/script (au format `*.py` comme un script Python) en sélectionnant le menu Programme. Il faut enregistrer un nouveau programme avant de l'exécuter.
- La fenêtre en haut à droite contient la grille d'évolution du robot, on peut régler sa taille à partir du menu Scène et il est possible d'insérer des murs ou des billes dans la grille en cliquant dessus avec le bouton gauche. On peut ouvrir/créer une nouvelle/enregistrer (au format *.wld) une scène depuis le menu Scène.

  Pour réinitialiser la scène, on peut cliquer sur le bouton correspondant de la barre d'outils ou effectuer la combinaison de touches sur `CTRL + R`.

- La fenêtre en bas à droite contient trois volets Affichage/Fonctions/Variable. Dans Affichage seront écrites les messages du robot (c'est le programmeur qui le fait parler ...). Dans Fonctions se trouvent la liste des fonctions spécifiques à Rurple en plus des fonctions classiques de Python. Dans Variables on pourra observer l'évolution du contenu des variables.

En dessous de la barre de menu se trouve un barre d'outils avec trois icônes cliquables pour contrôler l'exécution d'un programme (lecture puis interprétation par le robot) selon quatre fonctionnalités : Exécuter, Suspendre, Stopper, Exécuter pas à pas (fait une pause après chaque ligne/instruction).



Dans ce TD, on définira essentiellement des fonctions sans paramètres qui modifient l'état du programme par effet de bord (comme `print`*), on les désigne souvent comme des procédures.*

1. Lancer le logiciel Rurple depuis le partage réseau `T`.
2. Créer un répertoire `TD-Fonction-2020-10-01` dans le répertoire `NSI` de son espace personnel.
3. Télécharger l'archive qui contient toutes les scènes sur lesquelles nous allons travailler et la copier dans le répertoire `TD-Fonction-2020-10-01`.
4. Ouvrir la scène `scene0.wld` depuis le menu scène et le script `Rurple-TD-Fonctions.py`.

   Exécuter la fonction `parcourir_ligne` puis réinitialiser la scène avec `CTRL + R`.

```
from rurple import *

def parcourir_ligne():
    "Parcourt une ligne"
    while not mur_devant():
```

```
        avance()

parcourir_ligne()
```

Commenter l'instruction `parcourir_ligne()` puis ajouter au programme une fonction `tour_grille(n)` qui fait réaliser au robot `n` tours d'une grille rectangulaire sans murs intérieurs.

Tester la fonction.

5. Commenter l'appel de la fonction `tour_grille` puis ouvrir la scène `scene1.wld`.

Exécuter la fonction `mystere`.

Que fait la fonction `mystere` ? Ajouter une chaîne de caractères juste en-dessous de l'en-tête de la fonction pour spécifier le rôle de cette fonction : ainsi on peut documenter sa fonction avec une *docstring*.

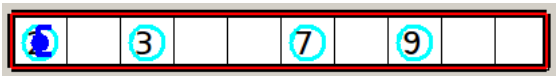
```
from rurple import *

def mystere():
    #ajouter la docstring ici
    while bille_au_sol():
        prends()

mystere()
```

6. Commenter l'appel de fonction `mystere()` dans le programme puis ouvrir la scène `scene2.wld`.

Compléter le programme avec une fonction `aspirateur_ligne()` qui permet au robot de récolter toutes les billes déposées sur une ligne de dimensions $n \times 1$ avec éventuellement plus d'une bille par case. Le robot est positionné initialement à une extrémité de la scène.

On utilisera les fonctions `mur_devant()` qui retourne un booléen indiquant s'il y a un mur devant le robot et `bille_au_sol()` qui retourne un booléen indiquant si la case contient au moins une bille.

Tester la fonction `aspirateur_ligne()` sur la scène `scene2.wld`.



7. Le module `rurple` propose une fonction pour faire tourner le robot à gauche mais pas pour le faire tourner à droite. Ajouter une fonction `droite()` au programme.

8. Ajouter au programme une fonction `quart_tour(ligne)` qui fait tourner le robot d'un quart de tour vers la gauche si le nombre `ligne` est impair et vers la droite sinon.

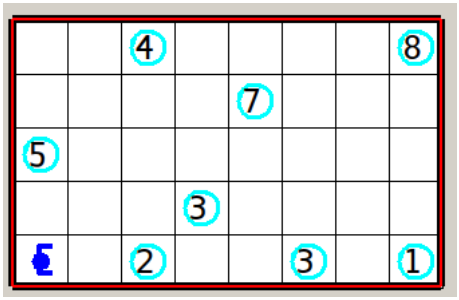
9. Ouvrir la scène `scene3.wld`.

Compléter le programme avec une fonction `aspirateur_grille()` qui permet au robot de récolter toutes les billes déposées sur une grille de dimensions $n \times m$ (sans mur intérieur) avec éventuellement plus d'une bille par case. Le robot est positionné initialement dans le coin inférieur gauche de la grille.

Tester la fonction `aspirateur_grille()` sur la scène `scene3.wld`.

```
def aspirateur_grille():
    "Ramasse toutes les billes sur une grille sans mur intérieur"
    ligne = 1
```

```
    aspirateur_ligne()
    quart_tour(ligne)
    #à  compléter
```



10. Ouvrir la scène `scene0.wld` et commenter le dernier appel de fonction.

a. Écrire une fonction `demi_tour()` qui fait faire un demi-tour sur lui-même au robot.

b. Écrire une fonction `bille_devant()` qui fait avancer le robot sur la case devant (en supposant qu'un test précédent a permis de déterminer qu'elle existe), teste la présence d'une bille au sol avec `bille_au_sol()` puis le ramène à sa position initiale (avec la même orientation).

c. Écrire une fonction `petit_poucet()` qui permet au robot de déposer une bille sur chacune des cases d'une scène carrée de dimensions $n \times n$, où n est un entier impair. Le robot doit parcourir la scène en spirale depuis le coin inférieur gauche (Voir figure 2 ci-dessous).

On commencera par placer n^2 billes ($n = 5$ pour `scene0.wld`) dans la poche du robot depuis le menu `Scène/Nombre de billes`.