

Séquence 3

Algorithme et programmation



Introduction à la programmation avec

Ropy



LA FORGE
des communs
numériques
éducatifs



[Philippe Roy <philippe.roy@ac-grenoble.fr>](mailto:philippe.roy@ac-grenoble.fr)

Présentation de Ropy et de son environnement de programmation

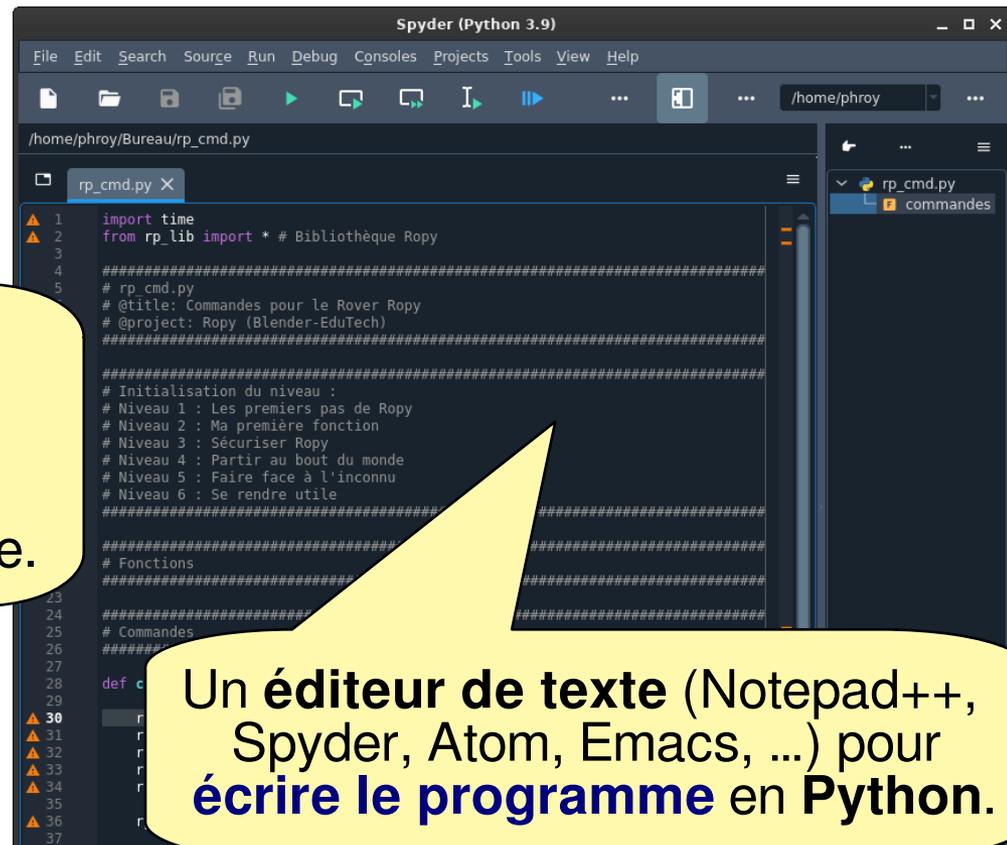
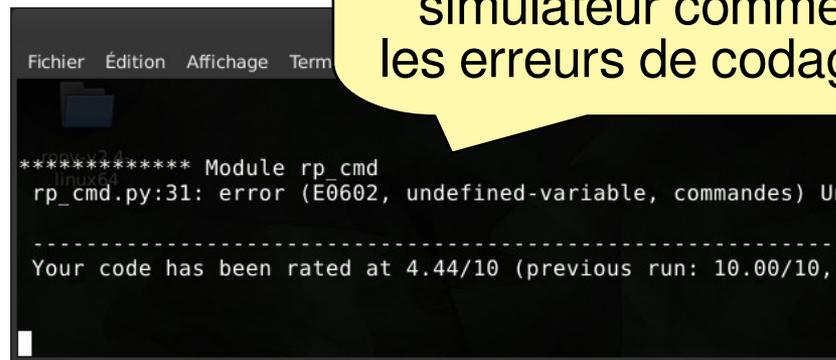


Ropy est un rover martien qui se commande grâce au langage **Python**. L'interface de programmation se décompose en **3 fenêtres** : un éditeur de texte, le simulateur et la console.

Le **simulateur** permet de **visualiser l'évolution du Rover**.



La **console** pour **visualiser les informations** du simulateur comme les erreurs de codage.

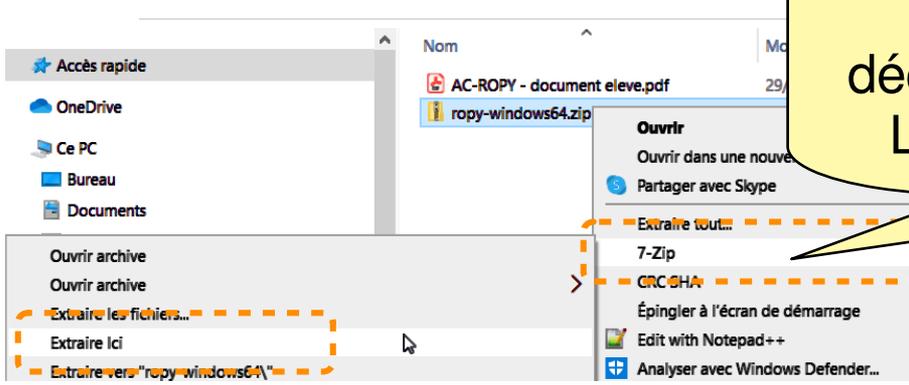


Un **éditeur de texte** (Notepad++, Spyder, Atom, Emacs, ...) pour **écrire le programme** en **Python**.

Mettre en place l'environnement de développement

1 : Récupérer l'archive **ropy-windows64.zip** et la décompresser avec **7-Zip** sur le **bureau**. L'extraction va créer le répertoire **ropy**

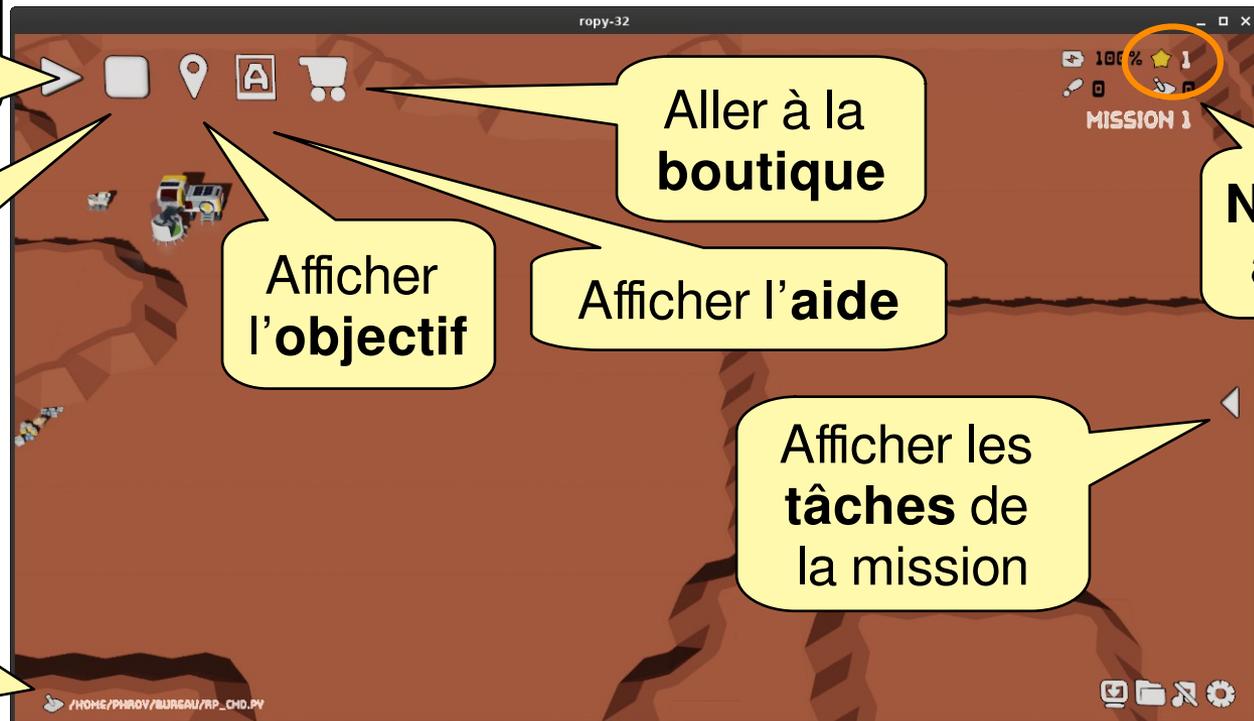
Le **simulateur** et la **console** se lancent en même temps avec **ropy.bat** (situé dans le répertoire créé).



Exécuter
le programme

Arrêter et réinitialiser

Fichier de commandes



Aller à la **boutique**

Afficher l'**objectif**

Afficher l'**aide**

Niveau
actuel

Afficher les **tâches** de la mission

Mettre en place l'environnement de développement

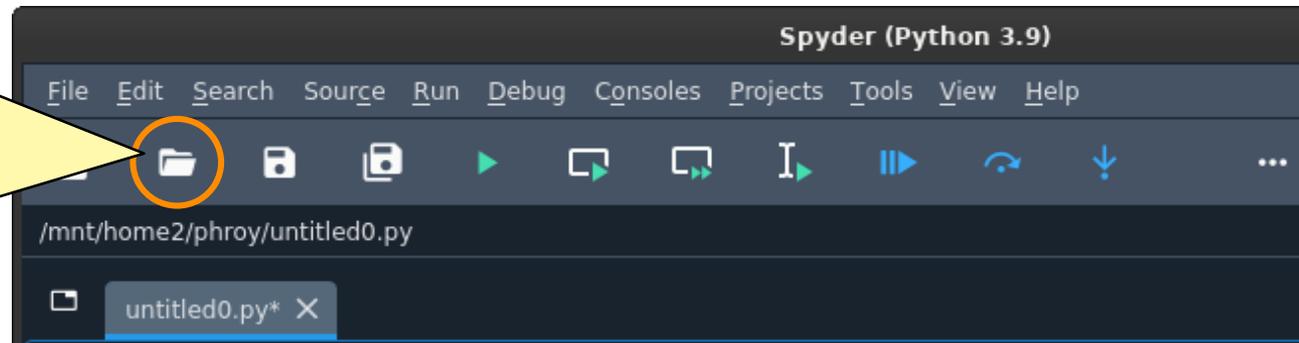


2 : Copier **dans votre répertoire** le fichier de commandes : **ropy_cmd.py** (ropy commandes).

3 : Lancer **Spyder**.



4 : Dans **Spyder** ouvrir le fichier de commandes qui a été précédemment copié dans votre répertoire.



6 : Le nom de votre fichier doit apparaître ici.

/HOME/PHROY/BUREAU/ROP_CMD.PY

5 : Dans le **simulateur**, définir votre fichier comme fichier de commandes.



Contenu du fichier rp_cmd.py



Le fichier `rp_cmd.py` comporte 4 sections.

```
import bge # Bibliothèque Blender Game Engine (UPBGE)
import time
from rp_lib import * # Bibliothèque Ropy

#####
# rp_cmd.py
# @title: Commandes pour le Rover Ropy
# @project: Ropy (Blender-EduTech)
#####

#####
# Initialisation du niveau :
# Niveau 1 : Les premiers pas de Ropy
# Niveau 2 : Ma première fonction
# Niveau 3 : Sécuriser Ropy
# Niveau 4 : Partir au bout du monde
# Niveau 5 : Faire face à l'inconnu
# Niveau 6 : Se rendre utile
#####

#####
# Fonctions
#####

#####
# Commandes
#####

def commandes():
    ➔ rp_gauche()
    rp_avancer()
    rp_avancer()
    rp_avancer()
    rp_avancer()

    rp_fin() # A garder

#####
# En: Externals calls << DONT CHANGE THIS SECTION >>
# Fr: Appels externes << NE PAS MODIFIER CETTE SECTION >>
#####

if __name__=='start':
    thread_cmd_start(commandes)
if __name__=='stop':
    thread_cmd_stop()
```

Le code doit être indenté (décalé sur la droite) avec la touche Tab

} **Import des bibliothèques**
Ne pas modifier cette section

} **Fonctions** : section pour le codage de **vos fonctions**

} **Commandes** : section pour le codage des commandes du robot

} **La commande `rp_fin()` est à conserver.**

} **Appels du simulateur**
(Blender Game Engine)
Ne pas modifier cette section

Mission 2 - Ma première fonction

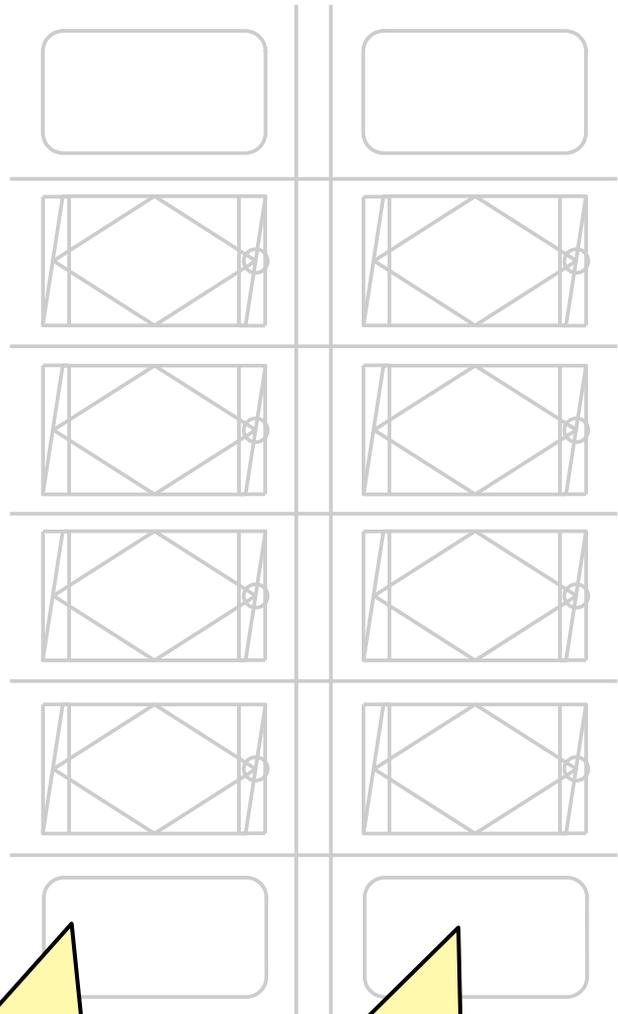
Création d'une fonction



Objectif 2 : Créer la fonction `mrp_avancer()` regroupant `avancer` et `marquer`.

```
#####  
# Fonctions  
#####
```

```
#####  
# Commandes  
#####
```



Représenter que le premier appel

Représenter la fonction

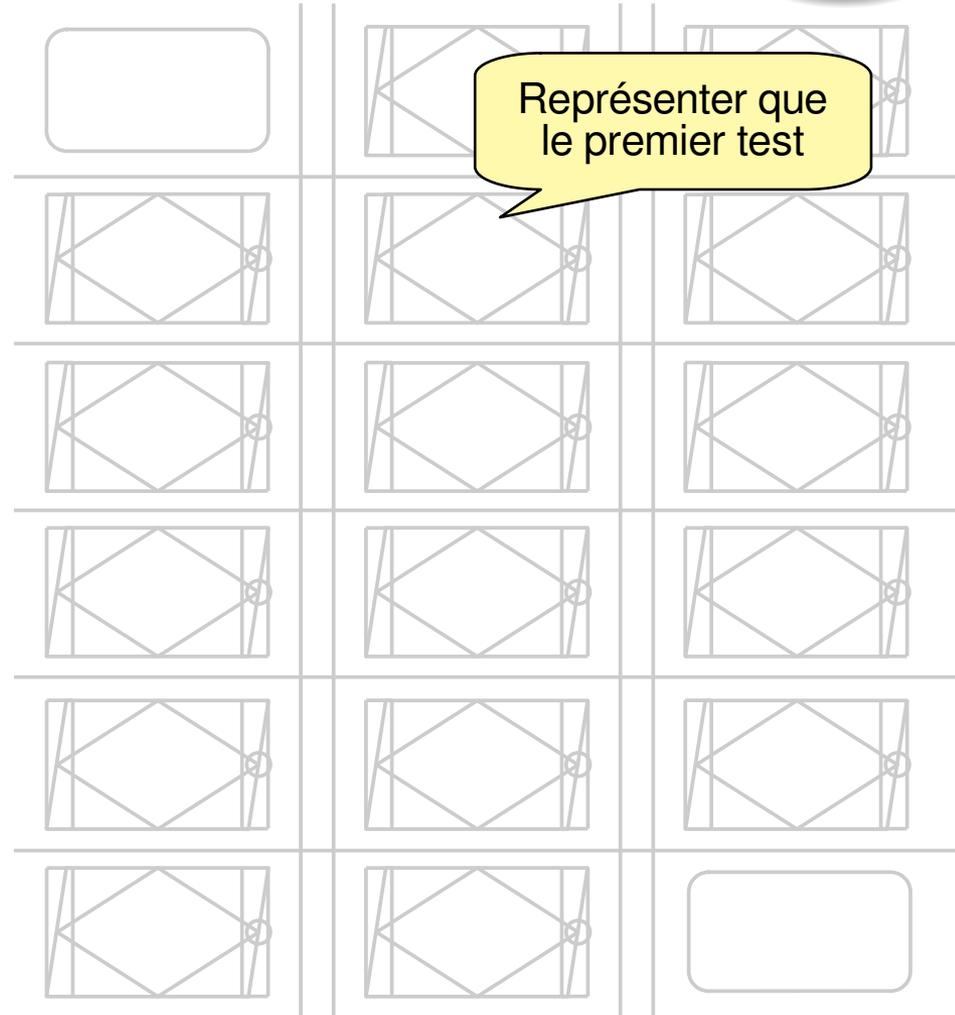
Mission 3 – Apprendre le danger

Structure conditionnelle (si, alors, sinon)



Objectif 3.1 : Aller à la mission 3, provoquer une collision avec un obstacle en avançant et observer ce qui se passe. Il vous faut donc sécuriser l'avance du robot.

```
#####  
# Commandes  
#####
```



La fonction pour **détecter un mur** est : `rp_detect ()`. La fonction retourne **True** si il a un mur et **False** si il n'y a pas de mur.

