EXEMPLE DE JEU POUR DÉCOUVRIR LE LANGAGE DE BASE EN PYTHON

https://py-rates.fr/

python™

ACADÉMIE
DE NORMANDIE
Liberté
Égalité
Fraternité

2021-2022
Maths-Sciences

SOMMAIRE

- Présentation Page 3
- Prise en main du jeu Page 4
- Description du jeu Pages 5 à 6
- Exemple Niveau 1 Pages 7 à 9
- Guide pédagogique Page 10
- Notions travaillées par niveau Page 10
- Exemples de solutions Pages 11 à 12

Ce projet est issu d'une thèse en didactique de l'informatique portant sur l'enseignement-apprentissage de la programmation informatique dans l'enseignement secondaire.

Cette thèse est co-financée par la Région Bretagne et l'Université de Bretagne Occidentale .

Vous pouvez contacter l'auteur à l'adresse suivante :

matthieu.branthome@univ-brest.fr

Ou sur Twitter: @MattBranthome

PRÉSENTATION

L’application Pyrates a été conçue pour accompagner la transition du collège au lycée dans l’apprentissage de la programmation informatique. Une attention particulière a été donnée au passage de la programmation par blocs à la programmation en ligne de code.

Cette application est destinée à des élèves de seconde ou de première et vise une première approche de la programmation en Python. Les élèves peuvent l'utiliser en relative autonomie, un effort ayant été fait en ce sens au niveau des contenus et de la progression pédagogique.

L'application est, de plus, conforme au RGPD : aucune inscription n’est nécessaire, aucune donnée personnelle n’est collectée. L’élève reçoit un code unique généré aléatoirement pour retrouver sa partie lorsqu'il quitte le jeu.

Ainsi Pyrates propose :

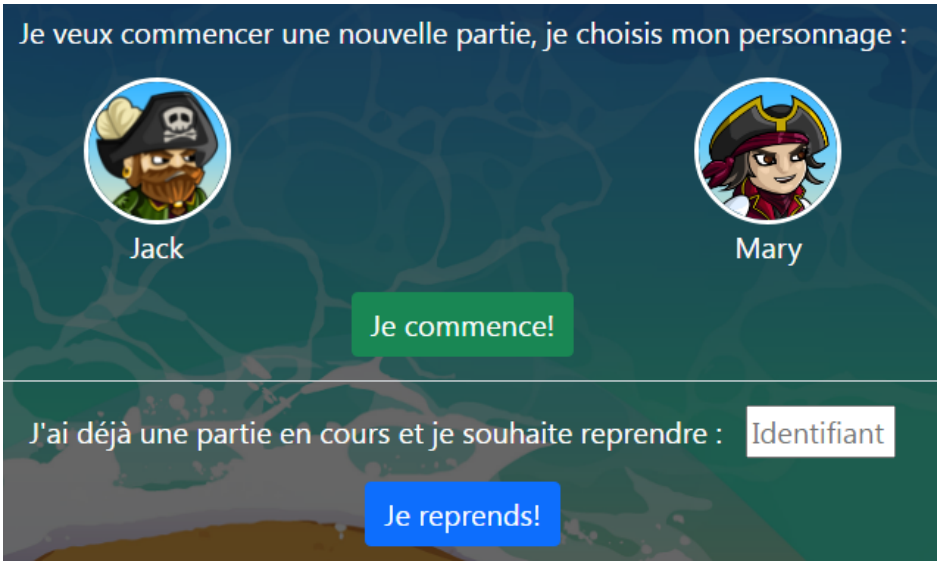
- un « Guide de démarrage » (bouton vert) permettant de découvrir le fonctionnement de l'application en autonomie ;
- un réinvestissement en Python des notions algorithmiques abordées au collège (variable, boucles et conditionnelles) facilité par un « Mémo programmation » (partie bleue) présentant ces notions en s’appuyant sur une comparaison avec les blocs de Scratch ;
- l’intégration d’un analyseur syntaxique issu de la recherche (Kohn, 2017) produisant des messages d’erreur en français dont la formulation est adaptée aux débutants ;
- un environnement de développement simple et tout intégré se rapprochant de ce que les élèves connaissent avec Scratch ;
- des activités ludiques sous la forme d’un jeu de plateforme favorisant l’engagement et la motivation des élèves.

Notons que l'application Pyrates introduit les appels de fonction dans leurs différentes configurations (retours et paramètres) mais n’aborde pas leurs définitions.

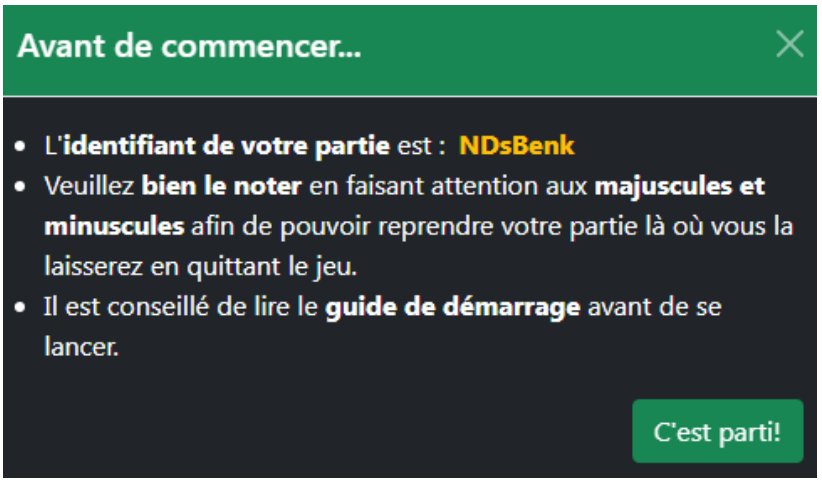
Au cours du jeu, les notions algorithmiques sont mises en jeu dans les différents niveaux sur le modèle des situations adidactiques (Brousseau, 1998). C’est-à-dire que les élèves doivent agir de leur propre mouvement, guidés uniquement par la logique interne des situations et non pour répondre aux intentions affichées de l’enseignant. Cette approche doit permettre de donner du sens aux apprentissages. Ainsi, dans chaque niveau du jeu, la situation ludique proposée doit amener les élèves à mettre en œuvre certaines notions algorithmiques sans pour autant que cela soit explicite. Ils peuvent pour cela s’appuyer sur les éléments présents dans l’environnement, en particulier le « Mémo programmation » Python qu’ils ont à disposition.

PRISE EN MAIN DU JEU

Lors de la première utilisation, choisir son avatar **Jack** ou **Mary** et cliquer sur Je commence!



Une nouvelle page s'ouvre dans laquelle une fenêtre apparaît:

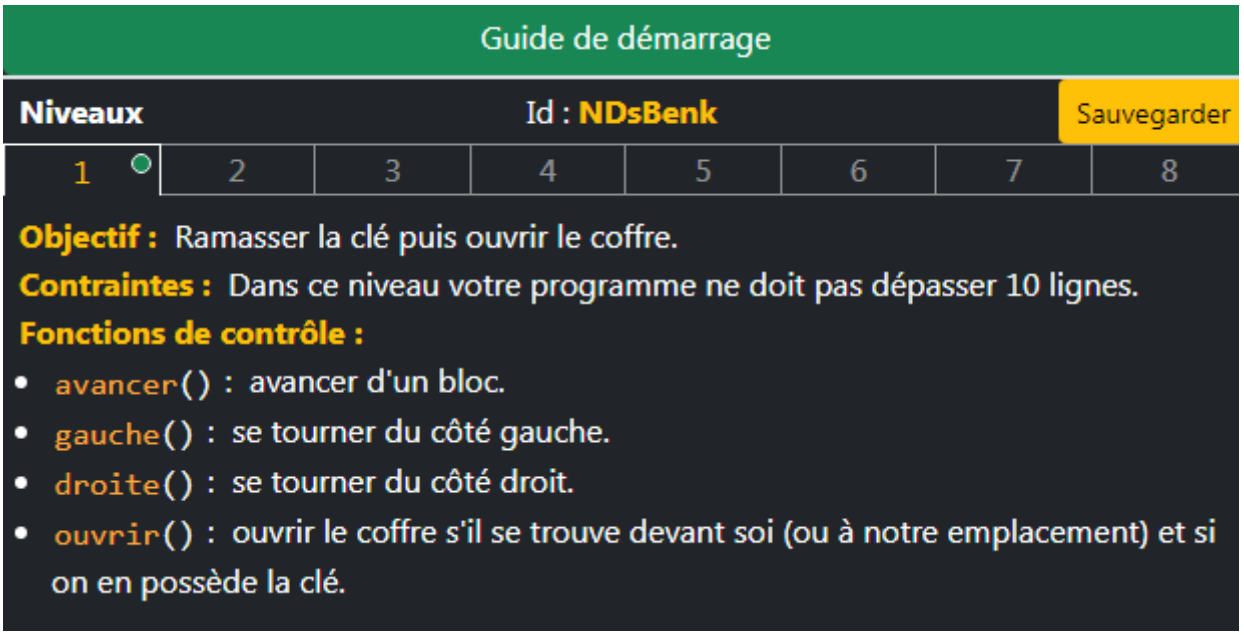


Bien noter l'identifiant de votre partie, cela permet de reprendre la partie au dernier niveau sauvegardé.

Cliquer sur C'est parti! pour débuter le jeu.

DESCRIPTION DU JEU

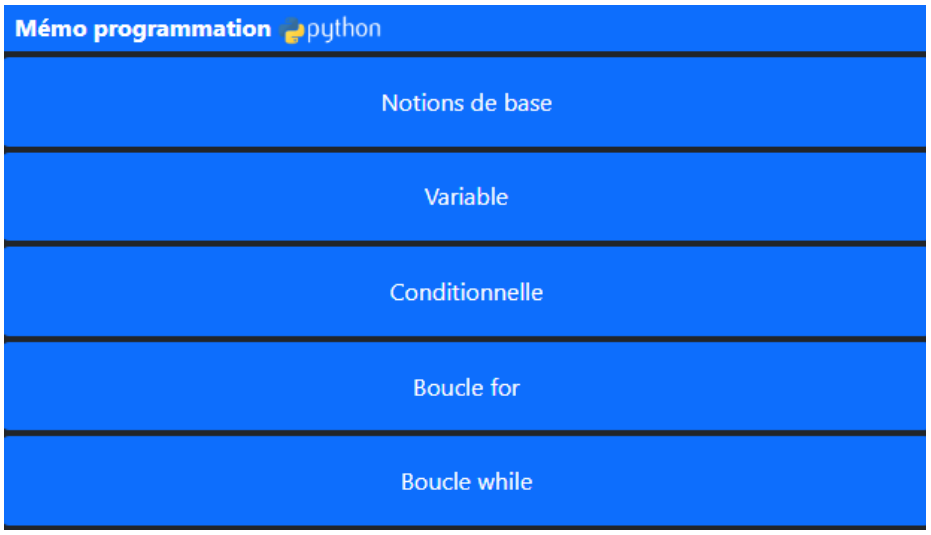
Au début de chaque niveau, il faut commencer par lire l'encadré suivant



Dans cet encadré, on retrouve:

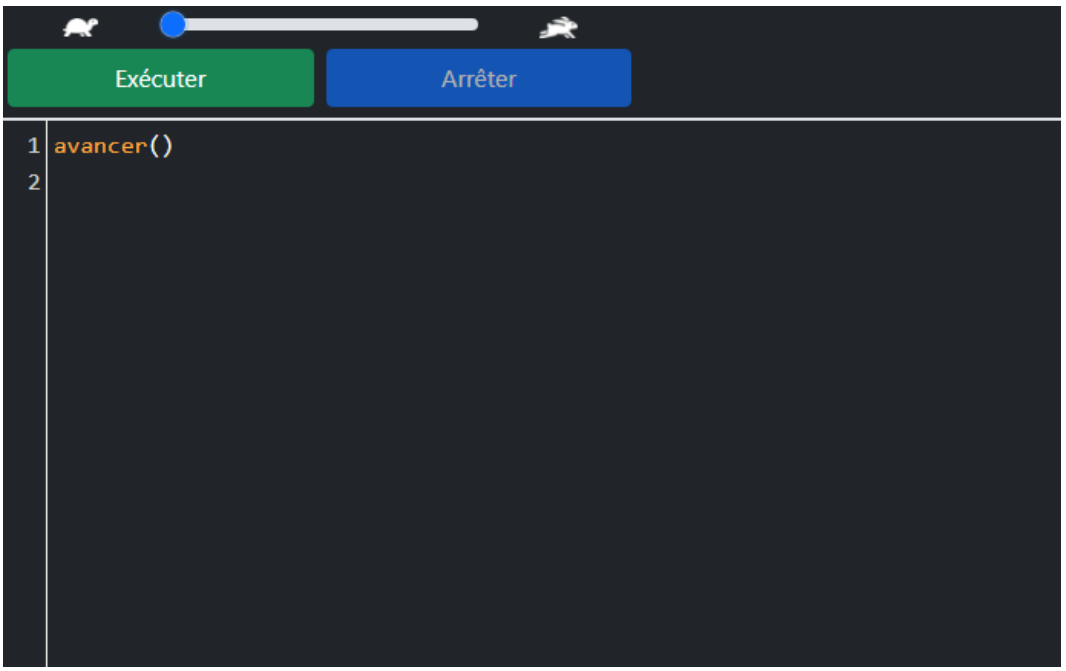
- **Guide de démarrage** (cliquer sur l'encadré vert)
- **L'identifiant de votre partie**
- **l'onglet sauvegarder** permet de mémoriser votre progression
- **L'objectif** (ici: Ramasser la clef puis ouvrir le coffre)
- **Les contraintes** (ici: Ne pas dépasser 10 lignes)
- **Les fonctions de contrôle:** actions que peut réaliser l'avatar

Sous l'encadré précédent, on retrouve les mémos suivant:



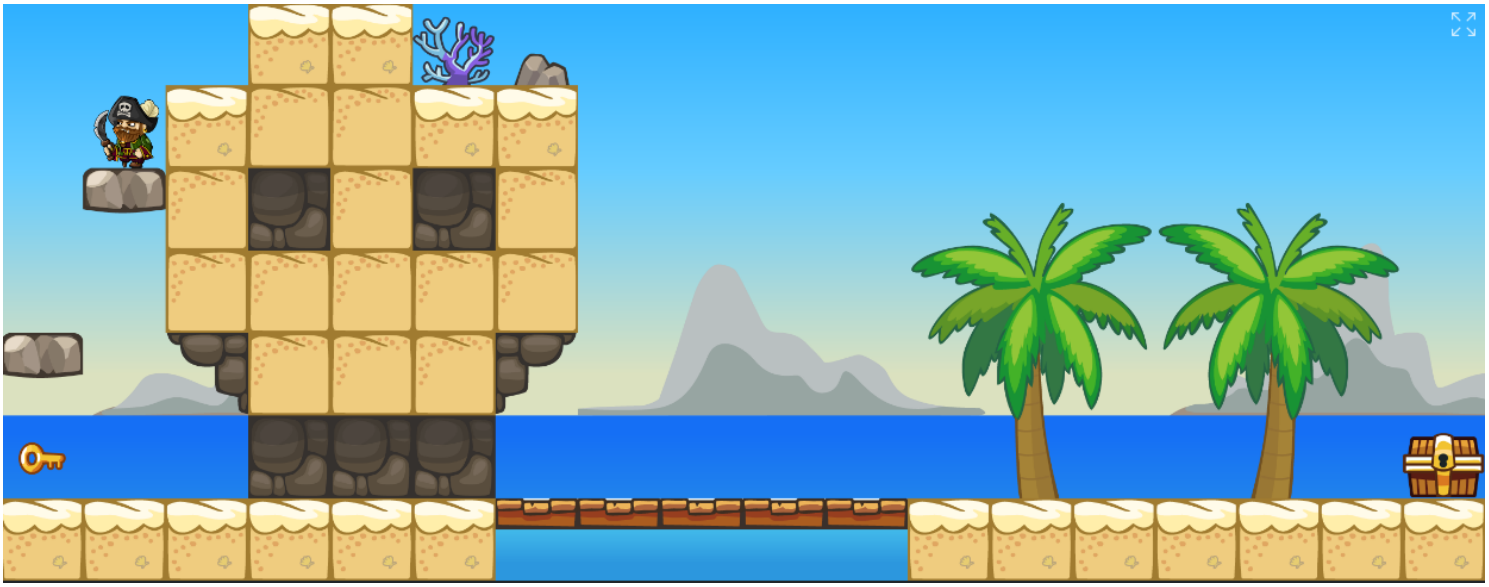
Ces mémos permettent de comprendre certaines fonctionnalités du code.

Dans cette même fenêtre, on trouve une partie console où vous devez écrire le programme qui permet de remplir l'objectif du niveau.



Il est possible de modifier la vitesse de réalisation du programme en jouant sur le curseur 

Enfin, on trouve une fenêtre de jeu dans laquelle l'avatar accomplit les commandes rentrées dans la partie console:

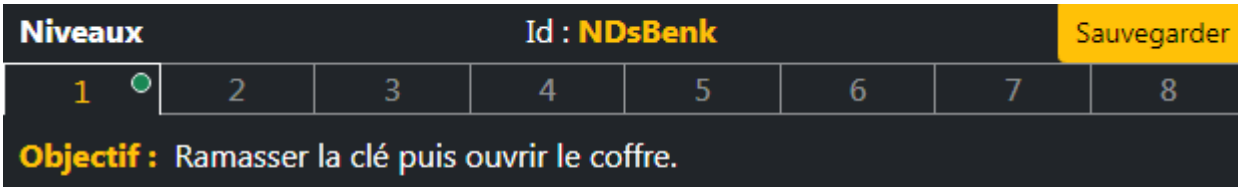


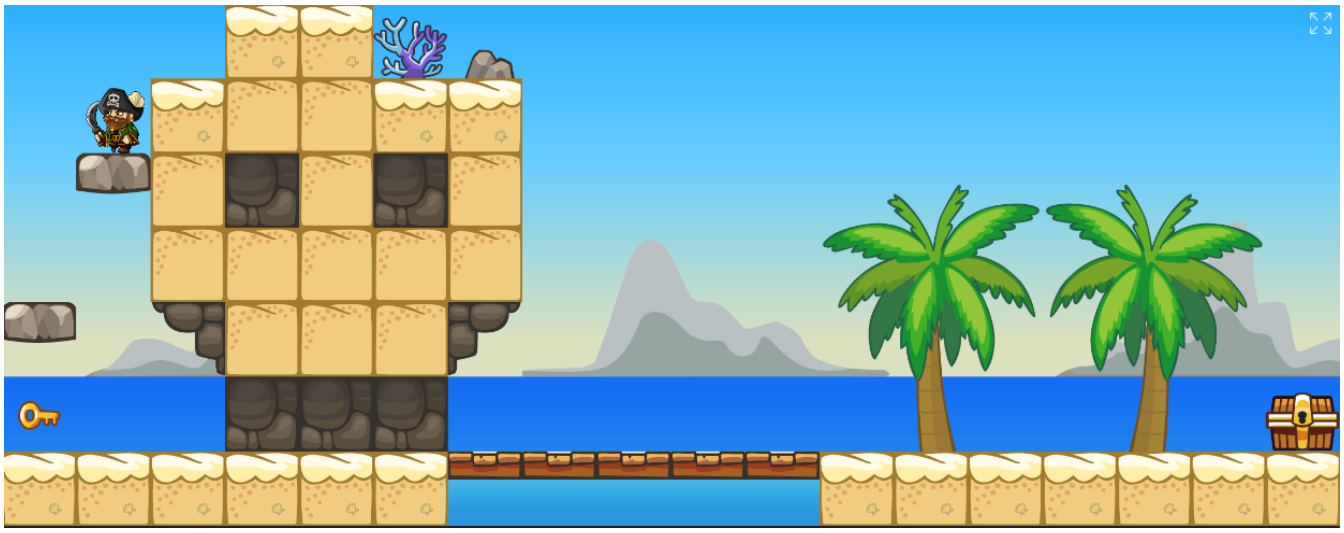
Pour bien débuter chaque niveau, il est conseillé d'essayer séparément les fonctions de contrôle afin de voir l'action que l'avatar réalise.

EXEMPLE: NIVEAU 1

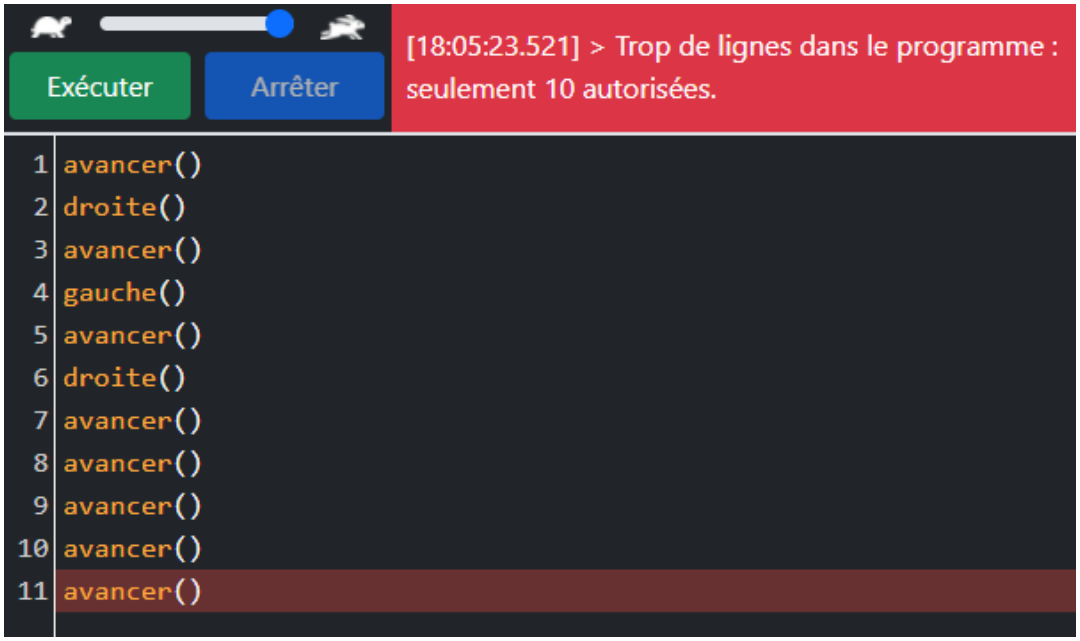
Les objectifs de ce niveau sont dans l'ordre:

- **Ramasser** la clé
- **Ouvrir** le coffre

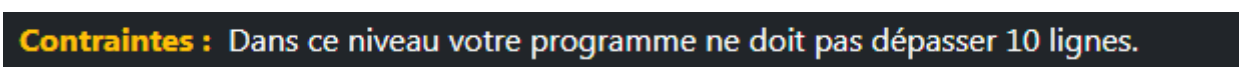




Les élèves vont commencer par écrire le code suivant en mettant les consignes les unes après les autres, c'est alors qu'un message d'erreur va apparaître:

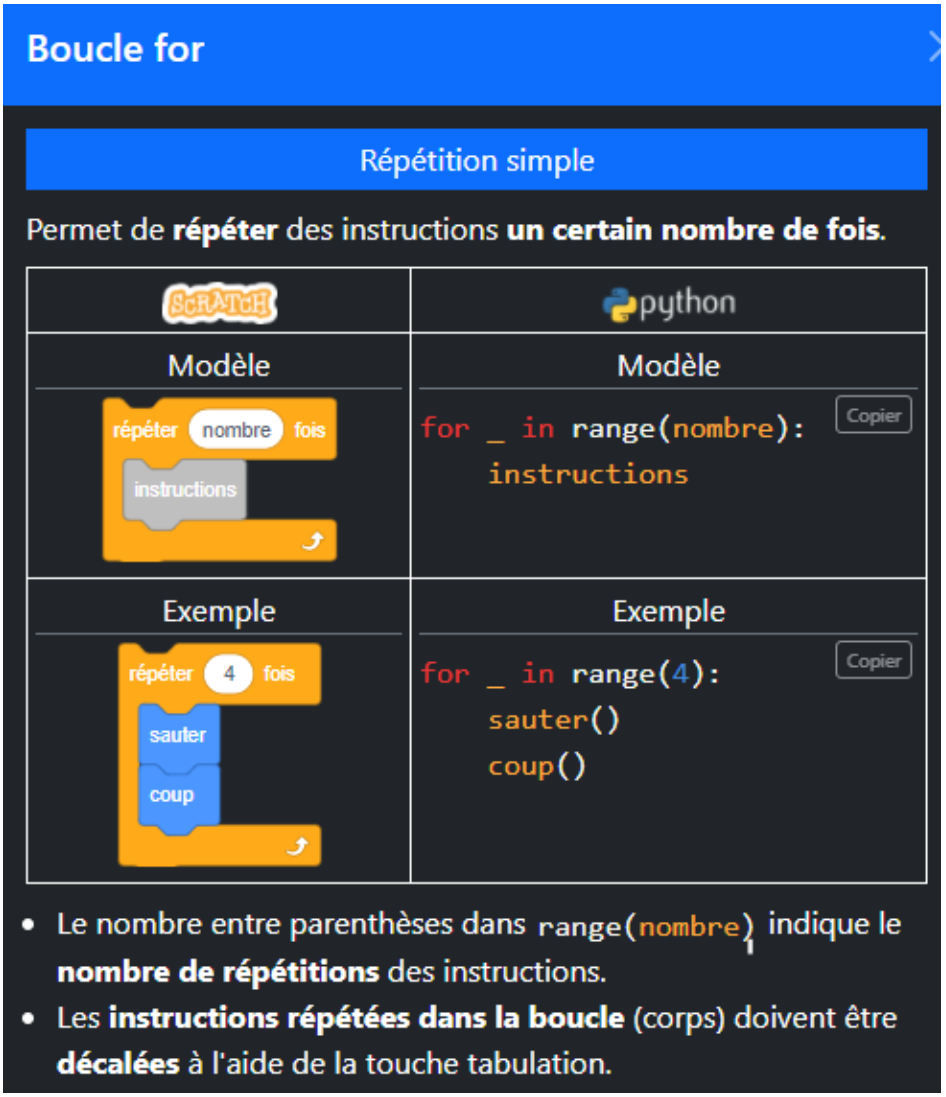


En effet, ils n'ont pas respecté la contrainte de 10 lignes:



Ils vont devoir utiliser les boucles pour pouvoir avancer 16 fois vers la droite une fois la clef récupérée.

En cliquant sur le mémo “Boucle for” L’aide suivante apparaît:



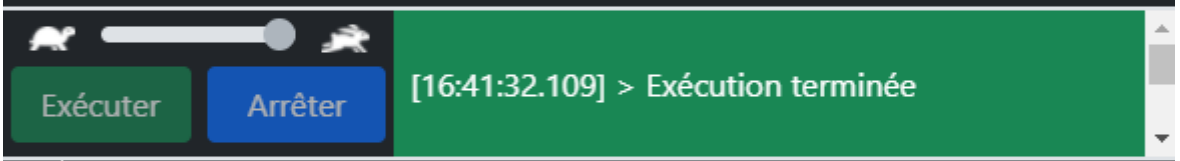
Il est donc possible de copier le modèle et de l'insérer dans le programme:

```
1 avancer()
2 droite()
3 avancer()
4 gauche()
5 avancer()
6 droite()
7 for _ in range(nombre):
8     instructions
```

Il ne reste alors qu'à écrire les bonnes instructions ainsi que le nombre de fois où on les répète.

```
5  avancer()
6  droite()
7  for _ in range(16):
8      avancer()
9  ouvrir()
10
```

Une fois l'étape précédente terminée, il suffit de vérifier le bon déroulement du programme:




```
1  avancer()
2  droite()
3  avancer()
4  gauche()
5  avancer()
6  droite()
7  for _ in range(16):
8      avancer()
9  ouvrir()
```

N'oublier pas de sauvegarder avant de passer au niveau suivant.

GUIDE PÉDAGOGIQUE

Afin de découvrir les fonctions et les solutions des niveaux suivants, consulter le guide disponible via le lien suivant :

https://py-rates.fr/guide/FR/index.html#pedagogical-guide

NOTIONS TRAVAILLÉES PAR NIVEAU:

NIVEAU 1	• notion “boucle for”
NIVEAU 2	• notion “boucle for”
NIVEAU 3	• notion “boucle for” • notion “variable”
NIVEAU 4	• notion “boucle for” • notion “variable” • notion “boucle while”
NIVEAU 5	• notion “boucle for” • notion “variable” • notion “boucle while”
NIVEAU 6	• notion “boucle for” • notion “variable” • notion “boucle while”
NIVEAU 7	• notion “boucle for” • notion “variable” • notion de boucle while
NIVEAU 8	• notion “boucle for” • notion “variable” • notion “boucle while”

EXEMPLES DE SOLUTIONS

NIVEAU 1

```
1  avancer()
2  droite()
3  avancer()
4  gauche()
5  avancer()
6  droite()
7  for _ in range(16):
8      avancer()
9  ouvrir()
```

NIVEAU 2

```
1  for _ in range(6):
2      sauter()
3      avancer()
4  avancer()
5  gauche()
6  avancer()
7  avancer()
8  for _ in range(9):
9      coup()
10     avancer()
11 avancer()
12 ouvrir()
```

NIVEAU 3

```
1  sauter_hauteur(4)
2  avancer()
3  sauter_hauteur(3)
4  avancer()
5  avancer()
6  for _ in range(4):
7      message = lire_nombre()
8      avancer()
9      sauter_hauteur(message)
10     avancer()
11     avancer()
12 ouvrir()
```

```
1  sauter_hauteur(4)
2  avancer()
3  sauter_hauteur(3)
4  avancer()
5  for _ in range(4):
6          avancer()
7          n=lire_nombre()
8          avancer()
9          sauter_hauteur(n)
10         avancer()
11 ouvrir()
```

NIVEAU 4

```
1  avancer()
2  for _ in range(5):
3      message = lire_chaine()
4      if message == "gau":
5          gauche()
6      else:
7          droite()
8      avancer()
9      avancer()
10 ouvrir()
```

```
1  for _ in range(5):
2          avancer()
3          message=lire_chaine()
4          if message == 'droi':
5                  droite()
6                  avancer()
7          elif message == 'gau':
8                  gauche()
9                  avancer()
10 avancer()
11 ouvrir()
```

NIVEAU 5

```
1  avancer()
2  sauter_haut()
3  for _ in range(5):
4      hauteur = mesurer_hauteur()
5      if hauteur == 0:
6          avancer()
7      elif hauteur == 1:
8          sauter()
9      else:
10         sauter_haut()
11     avancer()
12     avancer()
13 ouvrir()
```

```
1  avancer()
2  sauter_haut()
3  for _ in range(5):
4          h = mesurer_hauteur()
5          if h == 1:
6                  sauter()
7          elif h == 2:
8                  sauter_haut()
9          else:
10                 avancer()
11         avancer()
12         avancer()
13 ouvrir()
```

NIVEAU 6

```
1  for compteur in range(6):
2      sauter_hauteur(compteur)
3      avancer()
4  ouvrir()
```

```
1  for i in range(6):
2          sauter_hauteur(i)
3          avancer()
4  ouvrir()
```

NIVEAU 7

```
1  for compteur in range(2,9):
2      tirer(compteur)
3      tourner()
4  tirer(3)
```

```
1  for i in range(7):
2          tirer(i+2)
3          tourner()
4  tirer(3)
```

NIVEAU 8

```
1  avancer()
2  obstacle = detecter_obstacle()
3  while obstacle == True:
4          coup()
5          obstacle = detecter_obstacle()
6  for _ in range(3):
7          avancer()
8  droite()
9  for _ in range(5):
10         avancer()
11 obstacle = detecter_obstacle()
12 while obstacle == True:
13         coup()
14         avancer()
15         obstacle = detecter_obstacle()
16 ouvrir()
```