

HARRY POTTER CONTRE LA FORCE OBSCURE



Harry Potter



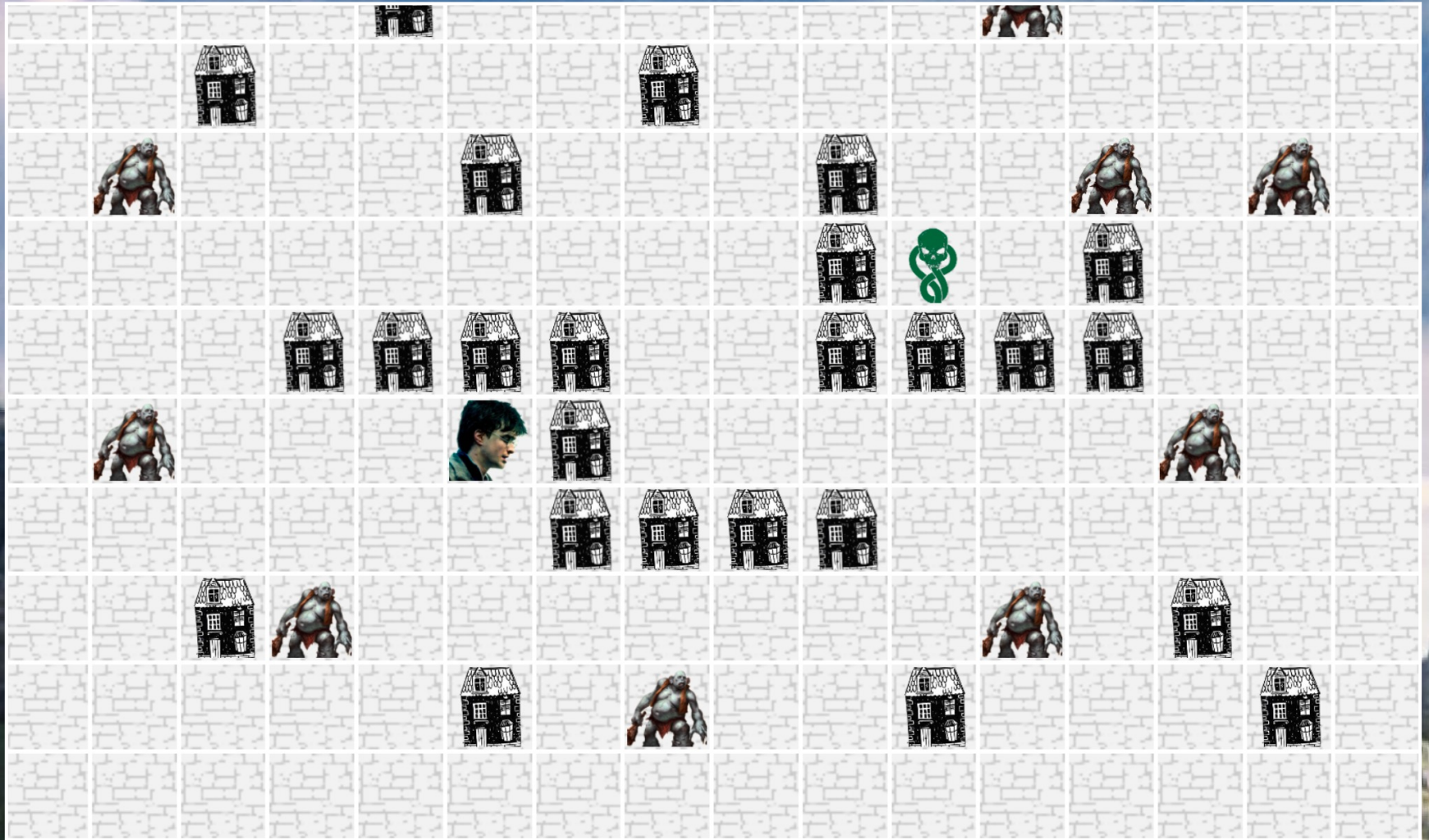
La force obscure



Troll





Bâtiment



Mission : Trouvez la force obscure

Le problème

Trouver le chemin le plus court

de  à 
en évitant les  et les 

Trouver le chemin le plus court entre deux objets, en évitant les obstacles.

Concepts clés de cette mission

algorithme

opérations
primitives

conditions

test *if*
(SI ... ALORS ...
SINON)

boucle *while*
(TANT QUE ...
FAIRE ...)

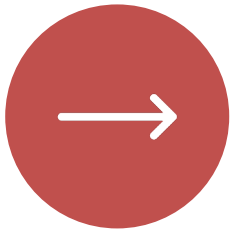
sous-
problème

Algorithme

Un *algorithme* est une description d'une procédure systématique décrivant des instructions à exécuter.

Trouver le chemin le plus court entre deux objets, en évitant les obstacles.

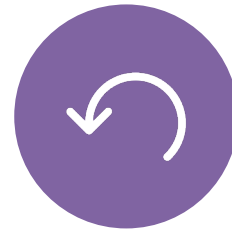
Opérations primitives



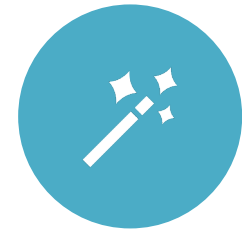
MOVE()



TURN_RIGHT()



TURN_LEFT()



DESTROY_DARK_FORCE ()

Conditions

CAN_MOVE()

IS_IN_FRONT_OF_ENEMY()

IS_ON_TARGET()

Possibilité de
combiner des
conditions

Conditions

Logique

condition **et** condition

condition **ou** condition

non condition

Vrai

Faux

Python

cond1 **and** cond2

cond1 **or** cond2

not cond

True

False

Un peu de logique

Vrai **et** Vrai

Vrai

Vrai **et** Faux

Faux

Faux **et** Vrai

Faux

Faux **et** Faux

Faux

Un peu de logique

Vrai **ou** Vrai

Vrai

Vrai **ou** Faux

Vrai

Faux **ou** Vrai

Vrai

Faux **ou** Faux

Faux

Un peu de logique

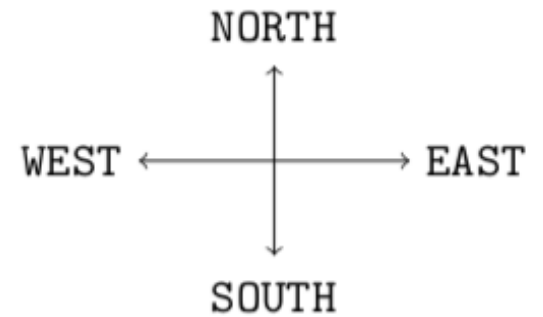
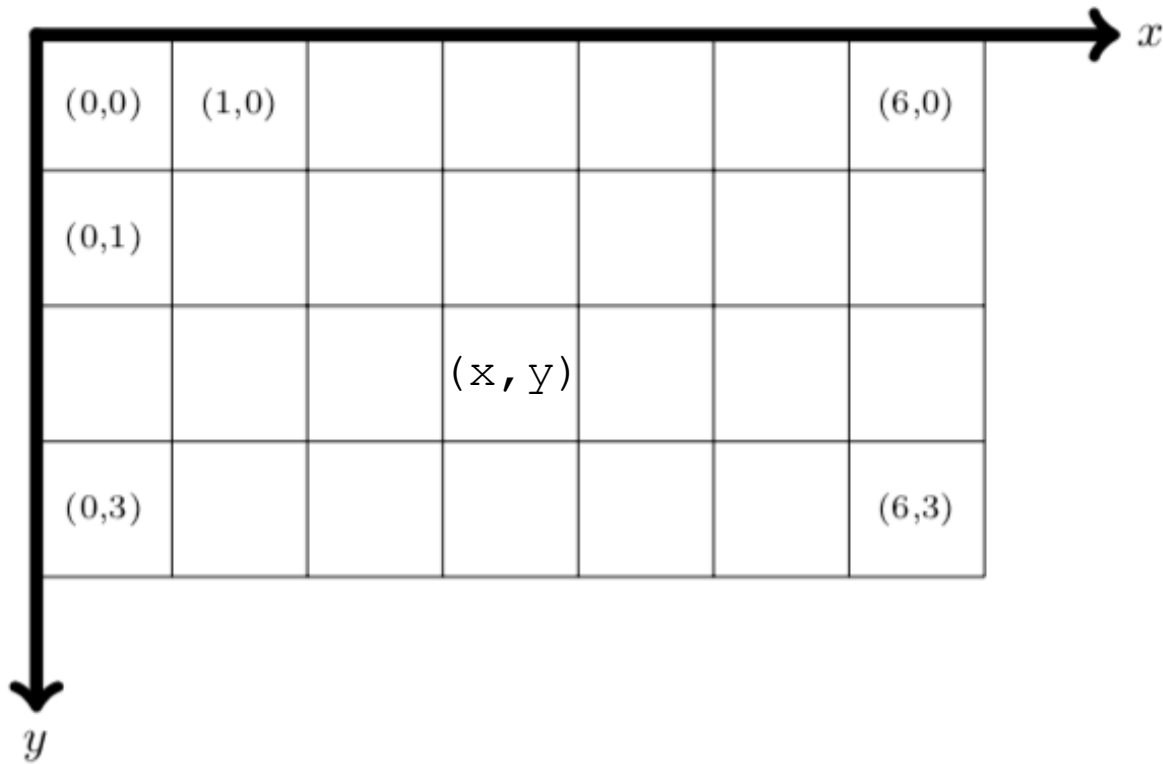
non Vrai

non Faux

Faux

Vrai

Coordonnées et directions



Récupération de l'état du terrain

`get_direction()`

Retourne EAST, WEST, SOUTH ou NORTH

`get_x()`, `get_y()`

Retourne la coordonnée x ou y de



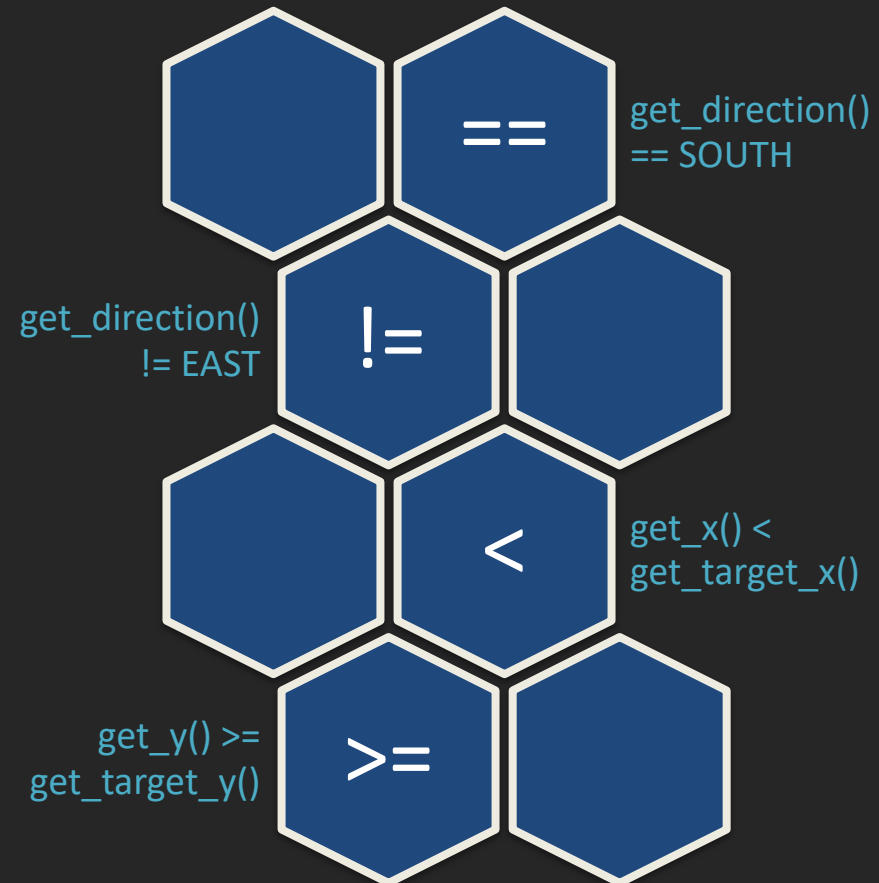
`get_target_x()`, `get_target_y()`

Retourne la position x ou y de



Conditions

Opérateurs de comparaison

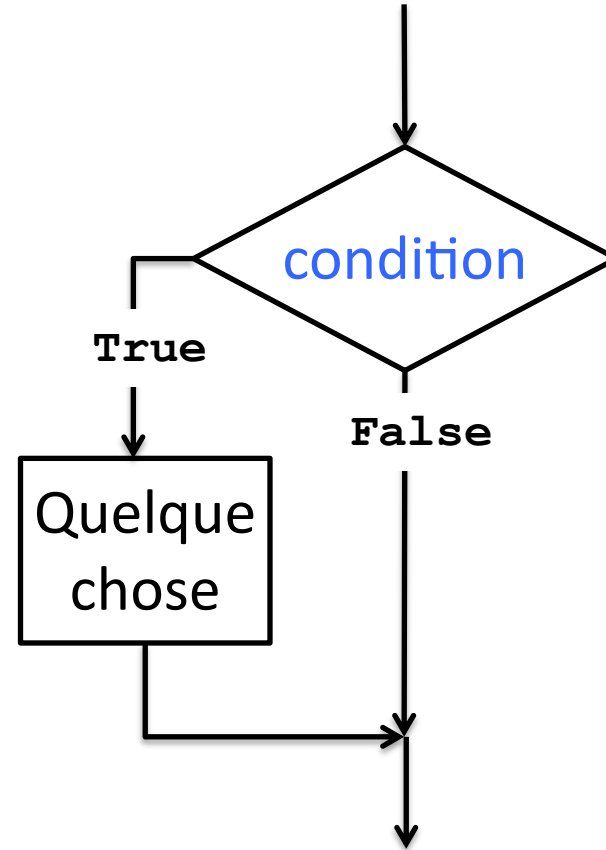


Test (1 branche)

SI condition

ALORS Quelque chose

```
if cond:  
    qqchose
```



Test (2 branches)

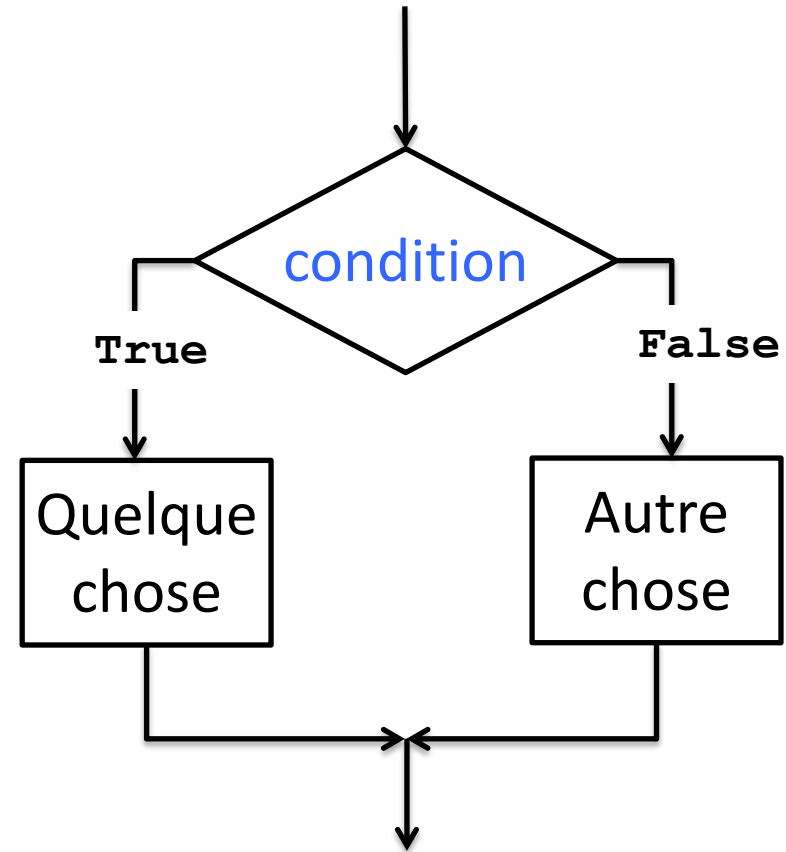
SI condition

ALORS Quelque chose

SINON Autre chose

```
if cond:  
    qqchose
```

```
else:  
    autrechose
```



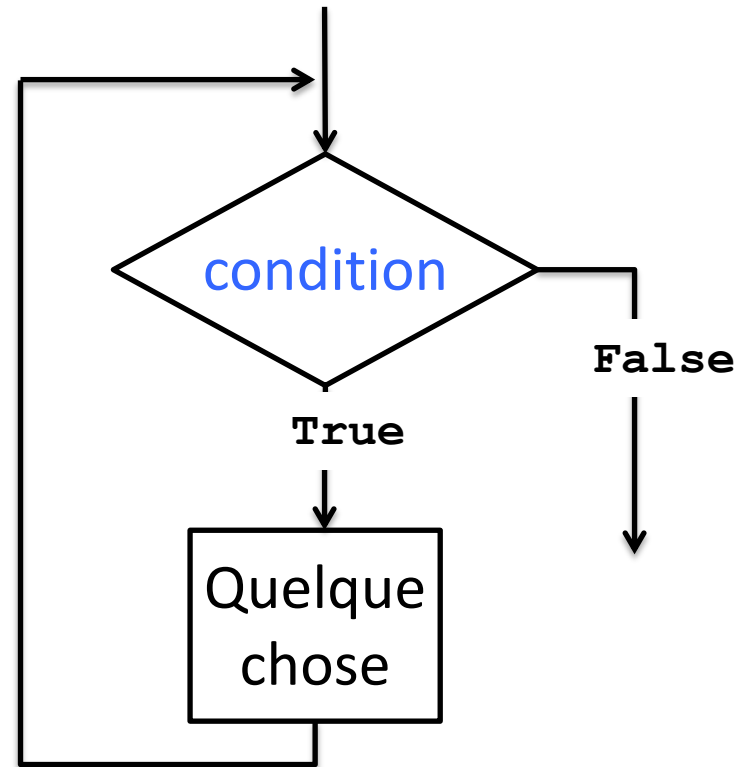
Exemple d'un test avec 2 branches

```
if Com move
| Move
else
Turn Left
Move
Turn Right
Move
Move
Turn Left
Move
Turn Right
endif
```

Boucle

TANT QUE condition
FAIRE Quelque chose

```
while cond:  
    qqchose
```



Exemple d'une boucle

```
TANT QUE Harry peut bouger FAIRE  
|  
|   avancer d'une case  
|
```

Pseudo code

```
while can_move():  
    move()
```

Code

Combiner une boucle et un test

```
TANT QUE Harry peut bouger FAIRE
|
|   avancer d'une case
|   SI Harry peut bouger ALORS
|   |
|   |   avancer d'une case
|   |
|   SINON
|   |
|   |   tourner vers la gauche
```

Pseudo code

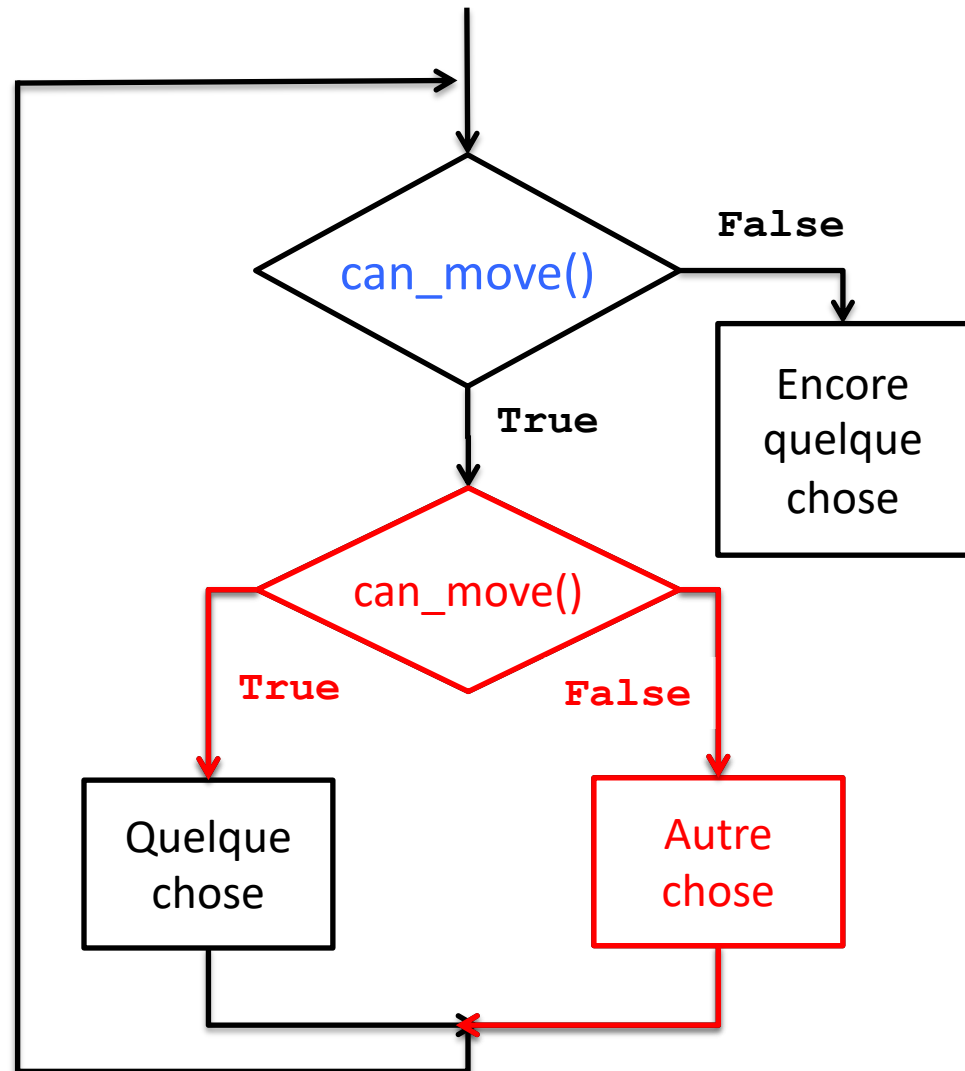
```
while can_move():
    move()
    if can_move():
        move()
    else:
        turn_left()
```

Code

Combiner une boucle et un test

Quel est le problème avec le programme suivant?

```
while can_move():  
    if can_move():  
        #qqchose  
    else:  
        #autrechose  
#encoreqqchose
```

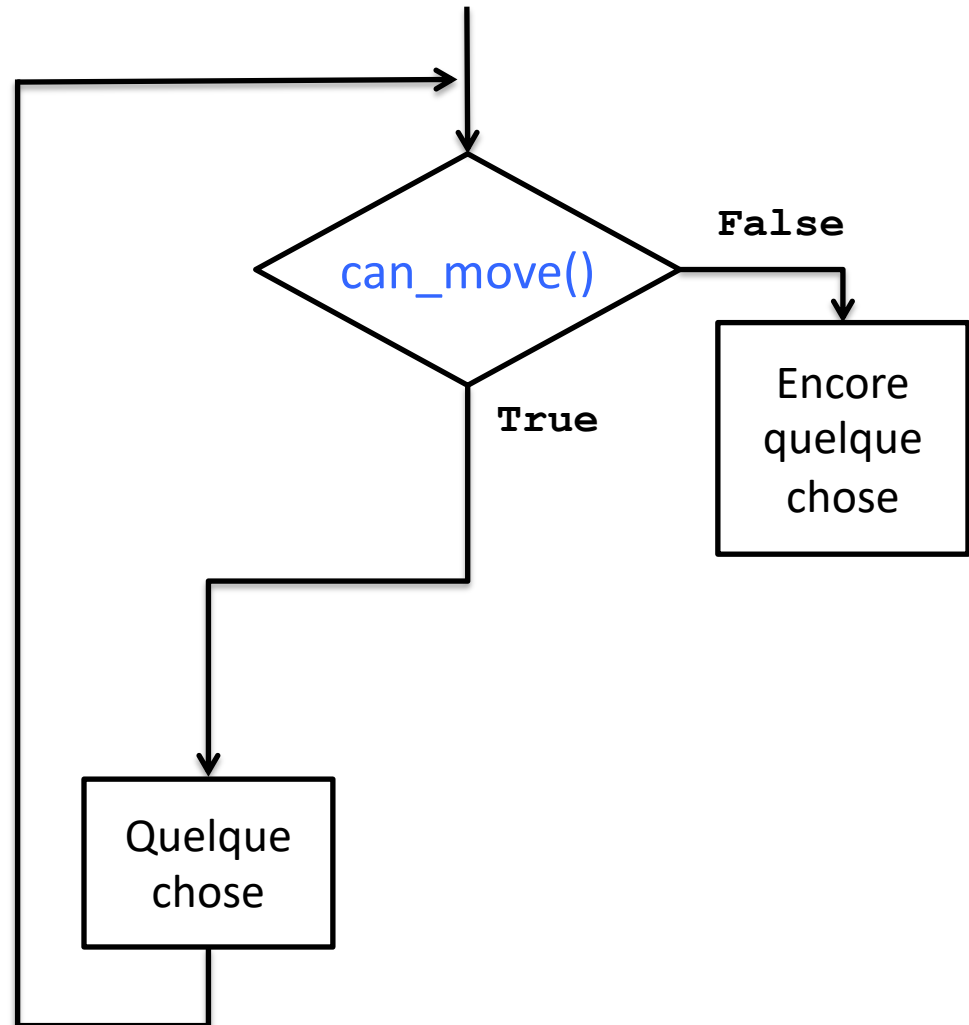


Combiner une boucle et un test

```
while can_move():
```

```
    #qqchose
```

```
    #encoreqqchose
```



Phase
d'implémentation

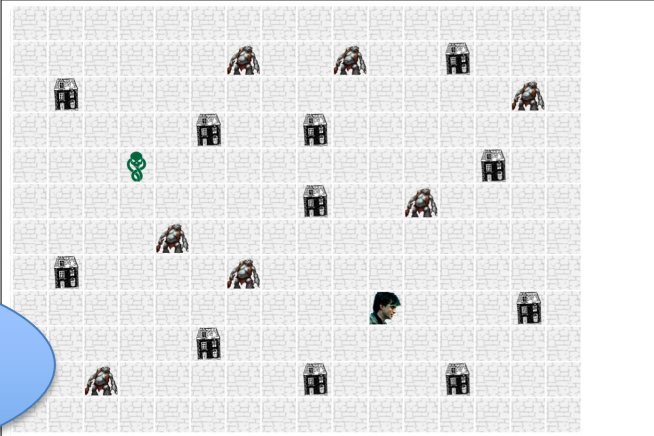
INGenius > APPO-2019 > APPO-2019 : soumission du code source

APPO : soumission du code source

Cette tâche vous soumettre votre code source.

Il y a des erreurs dans votre réponse. Votre note est de 0.0%. [Soumission #5f4ff06d6779dd113234d3ac]

Vous n'avez pas résolu cette instance : Le personnage emprunte un chemin inexistant.



Rejouer le scénario

Votre code

Placez ici votre code

```
1 while get_x() != get_target_x():
2     if get_x() < get_target_x():
3         while get_direction() != EAST:
4             turn_left()
5     else:
6         while get_direction() != WEST:
7             turn_left()
8     move()
9
10 spy_on_target()
```

Soumettre

Informations

Auteur(s)	Benoît Duhoux, Maxime Piraux & Céline Deknop
Date limite	20/09/2019 13:00:00
Etat	Succeeded
Note	100%
Poids de la note	1.0
Nombre d'essais	21
Limite de soumission	Pas de limite

Administration

Ce cours n'est pas affiché aux étudiants. Vous pouvez changer cela en modifiant l'option "accessibilité" dans les paramètres du cours.

- Voir les soumissions
- Editer l'exercice
- Informations de débogage

Scoreboard: Classement de l'APPO 2019

Pour évaluation

- Meilleure soumission
- 02/09/2020 17:14:27 - 100.0%

Historique des soumissions

02/09/2020 21:20:13 - 0.0%
02/09/2020 17:14:27 - 100.0%
02/09/2020 17:04:51 - 0.0%
02/09/2020 17:01:49 - 0.0%
02/09/2020 17:00:36 - 0.0%

Attention pour les boucles infinies

Votre soumission a pris trop de temps à exécuter. Votre note est de 0.0%. [Soumission #5f4ff2be6779dd11333bc620]

Votre code

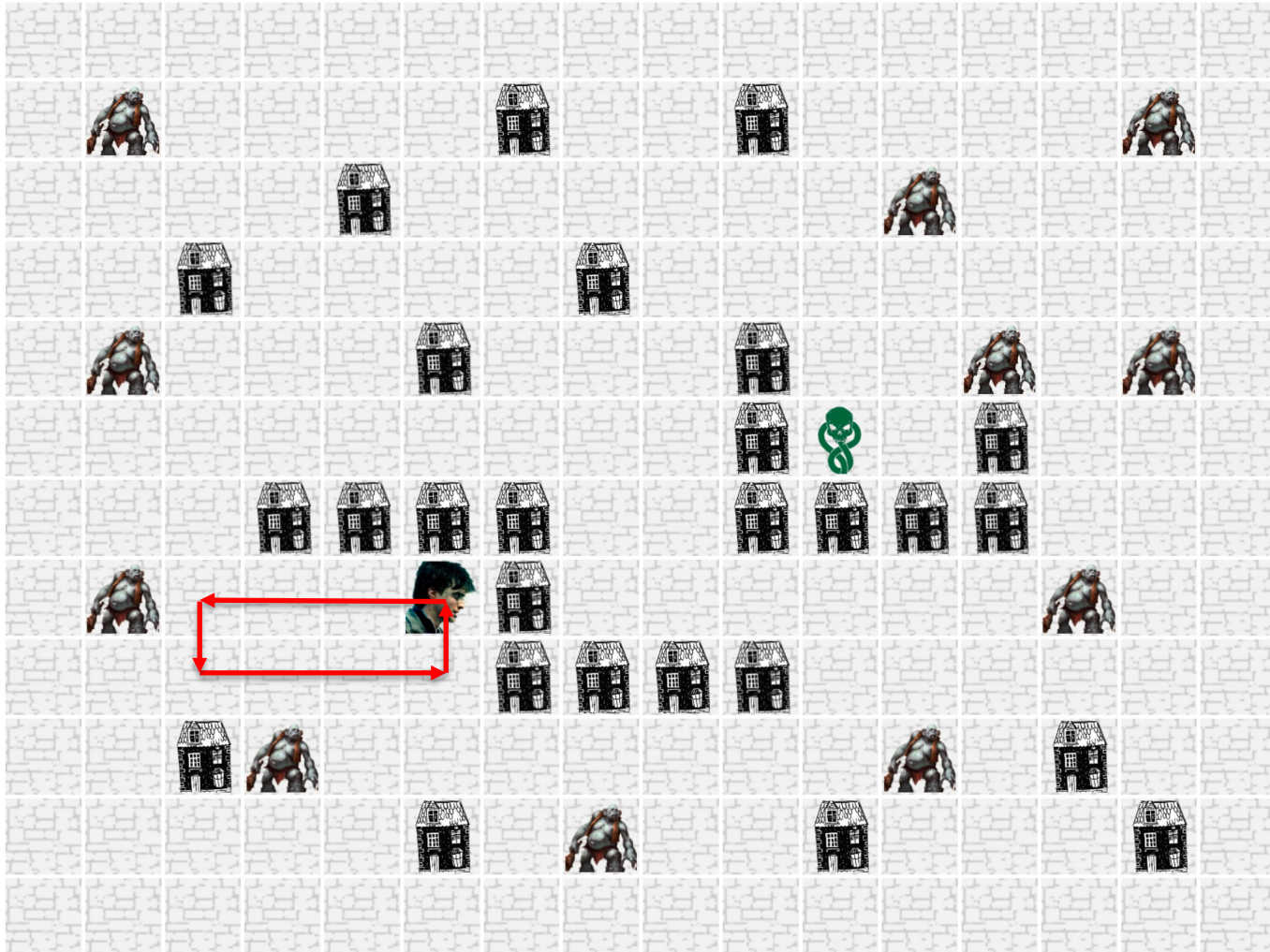
Placez ici votre code

```
1 while get_x() != get_target_x() and get_y() != get_target_y() :
2     while can_move():
3         move()
```

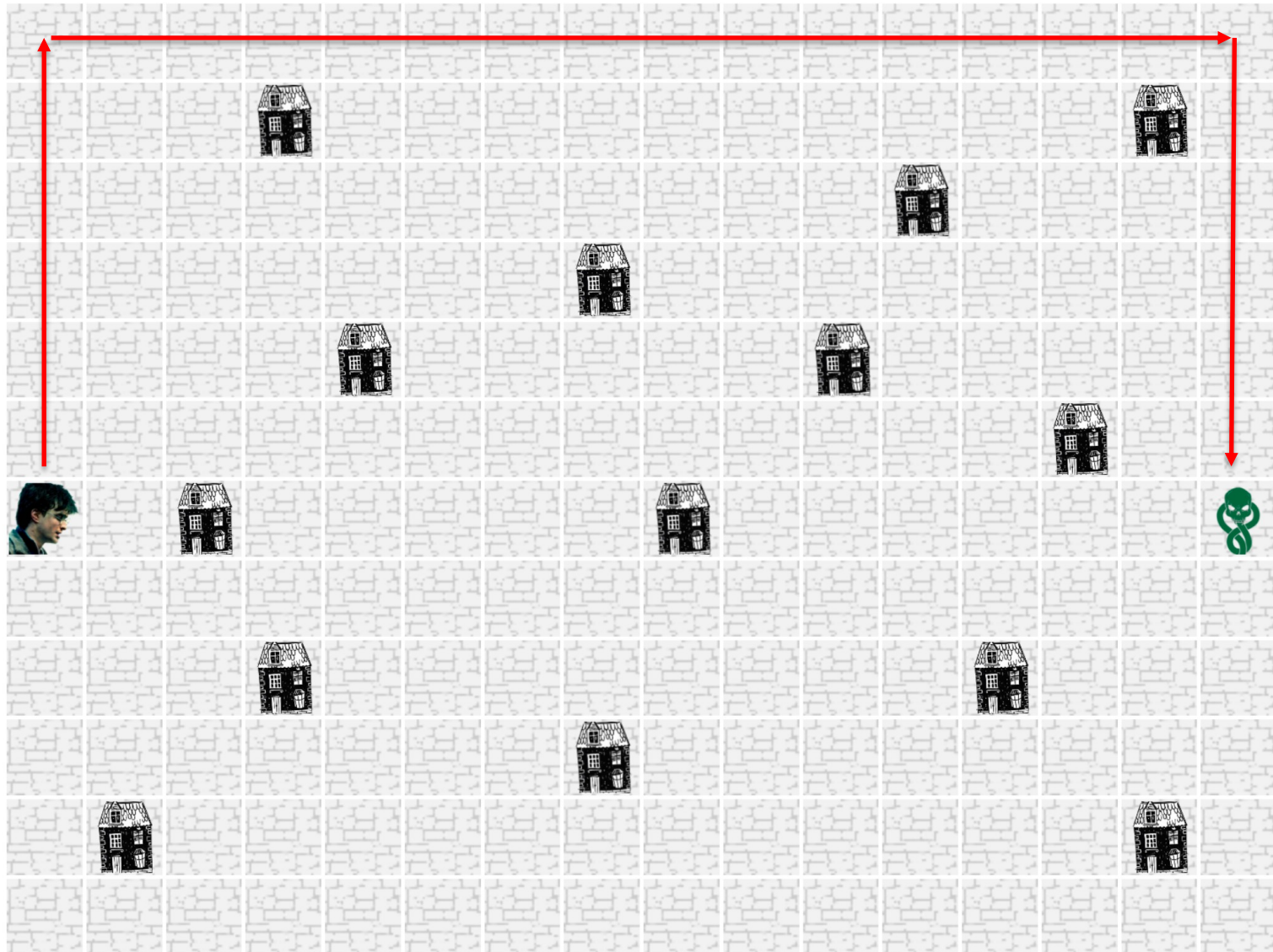
Soumettre

>_

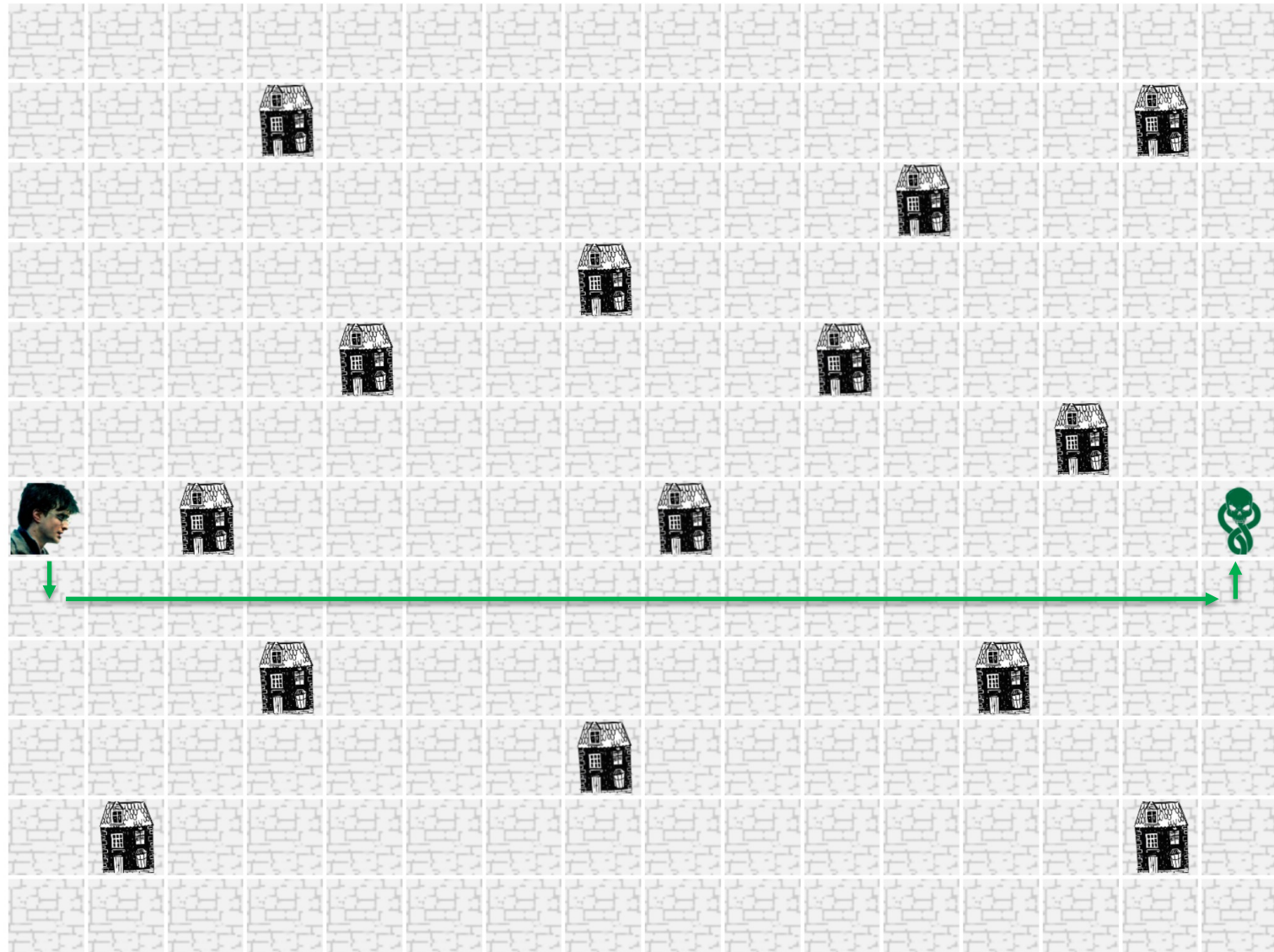
Attention pour les boucles infinies



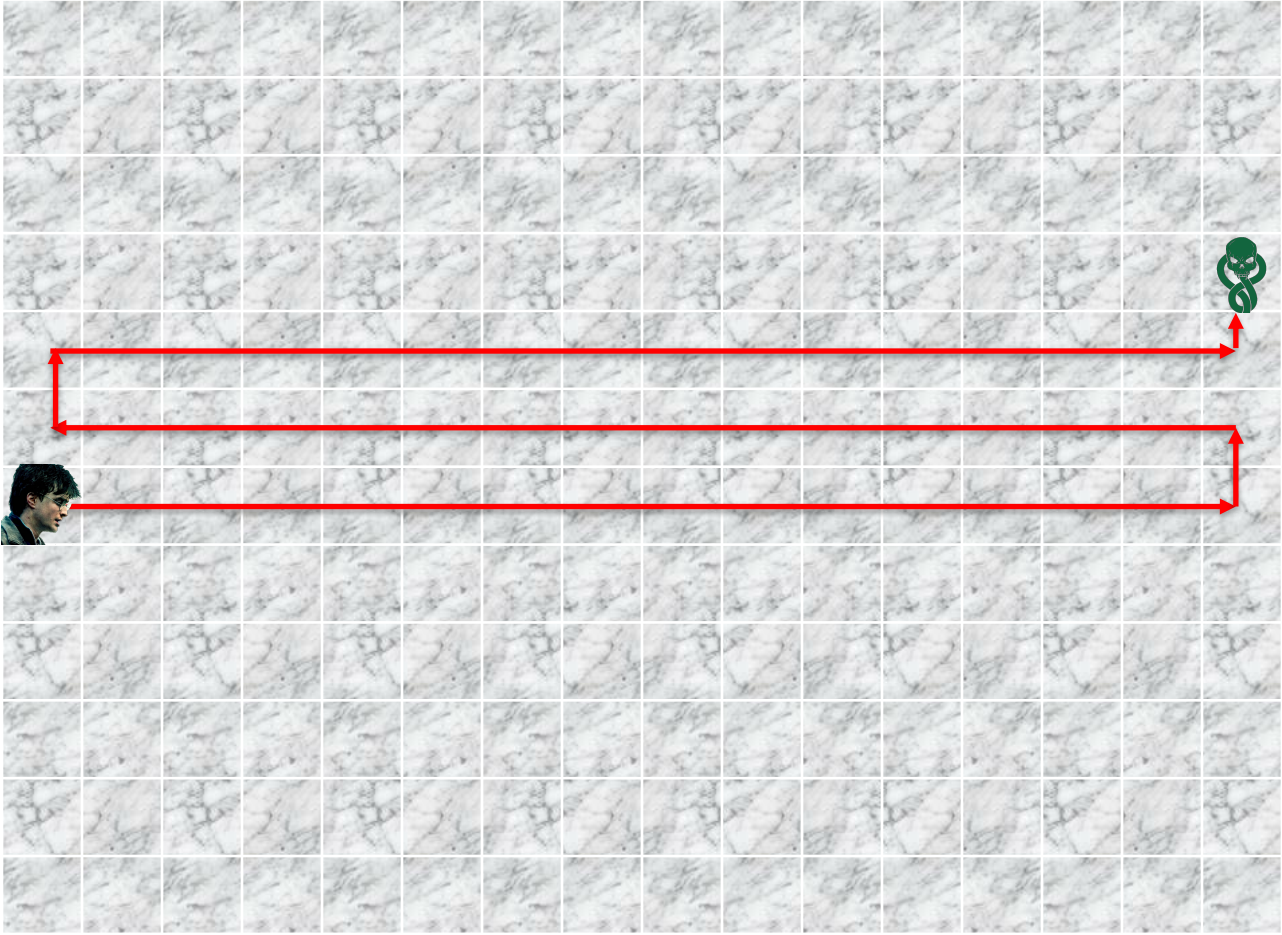
Trouver le chemin le plus court



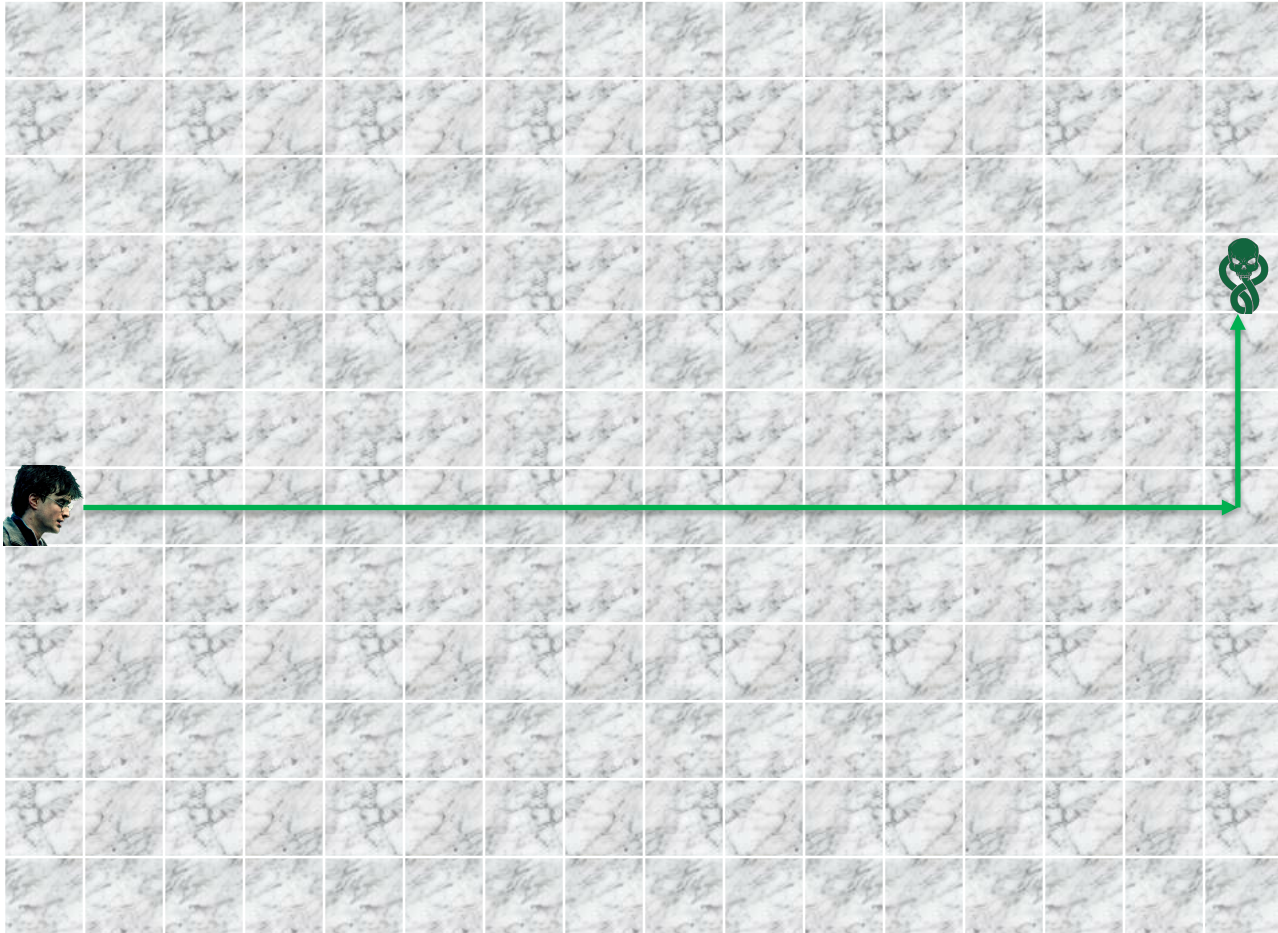
Trouver le chemin le plus court



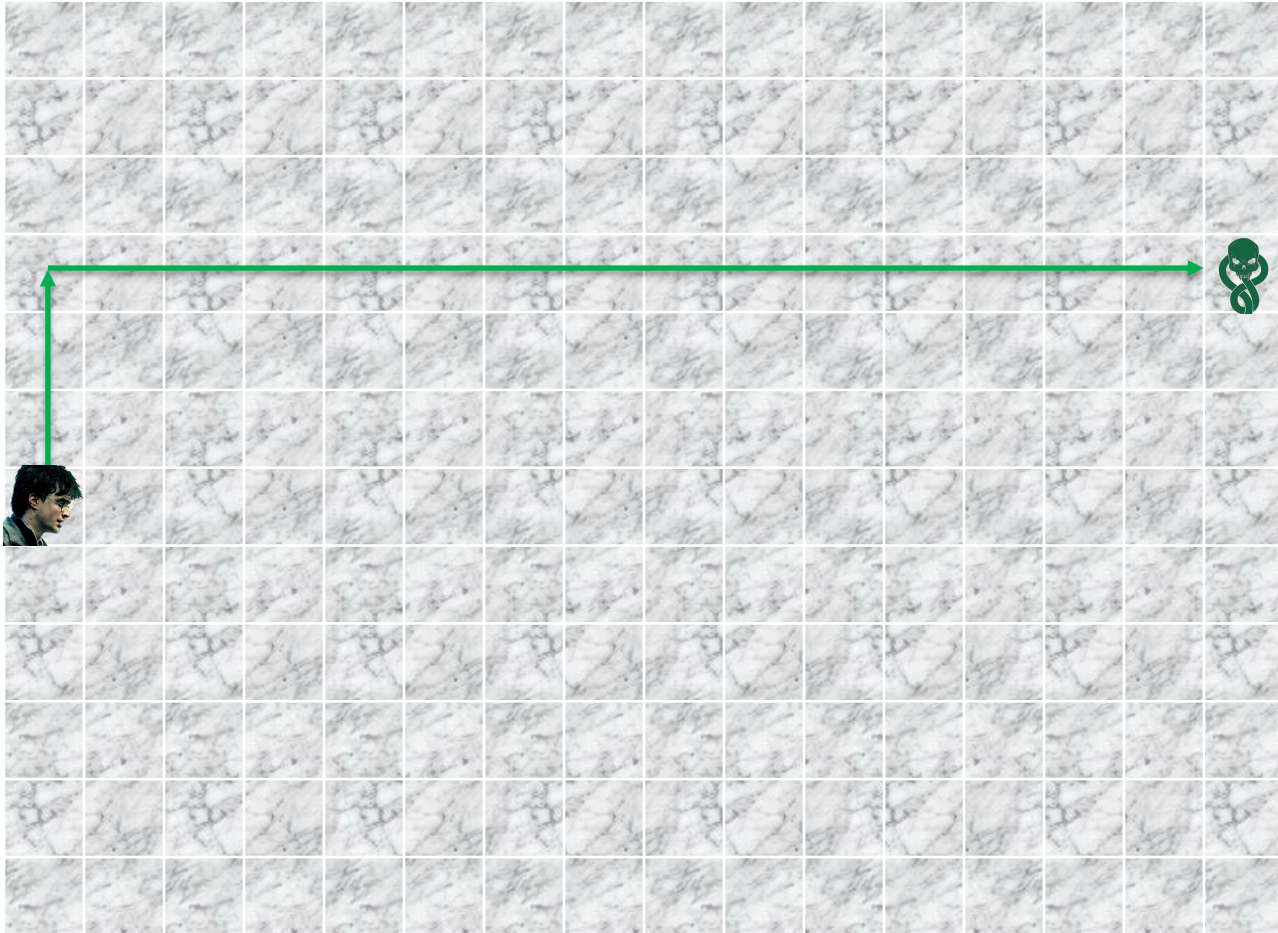
Trouver le chemin le plus court



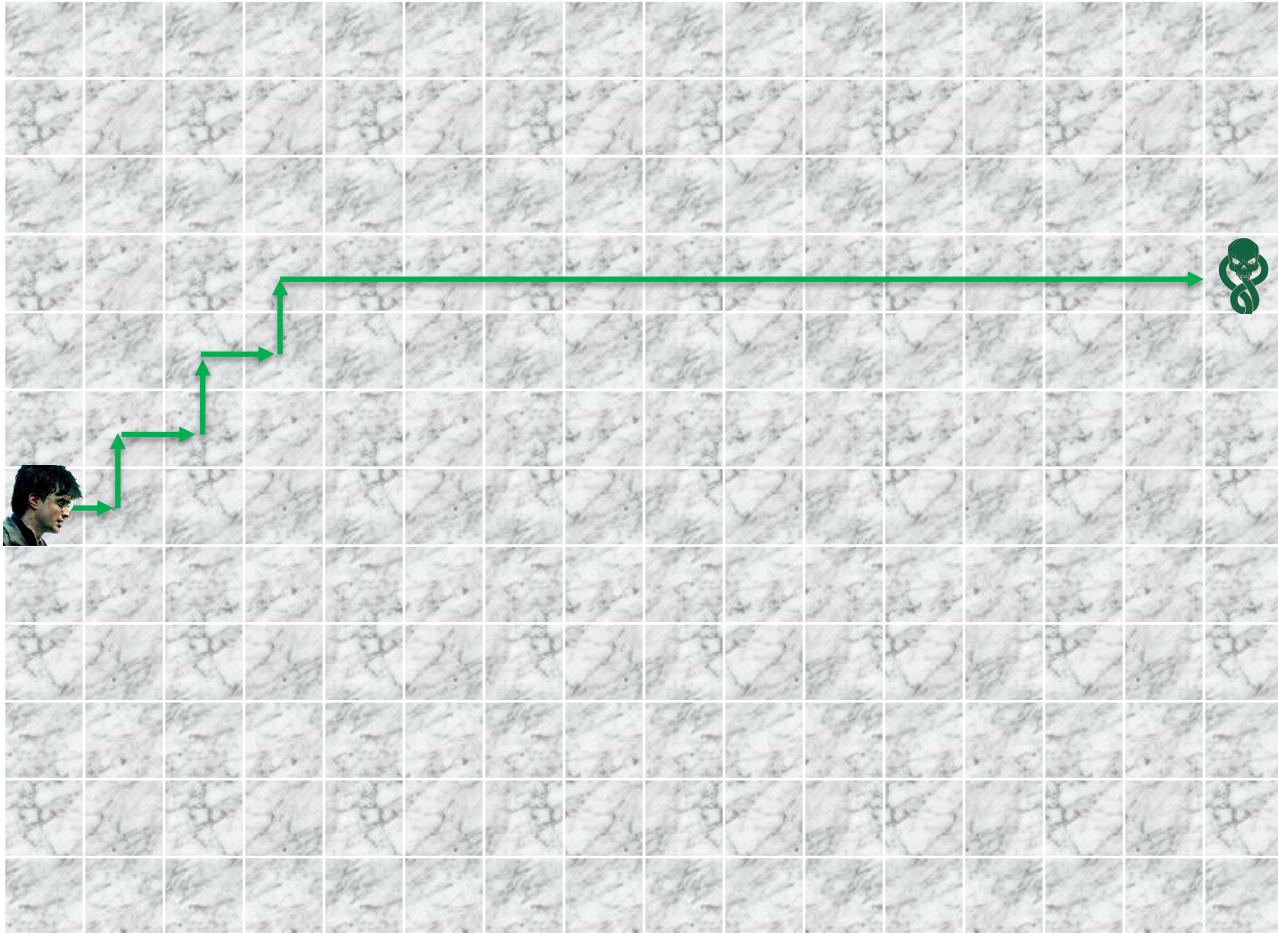
Trouver le chemin le plus court



Trouver le chemin le plus court



Trouver le chemin le plus court



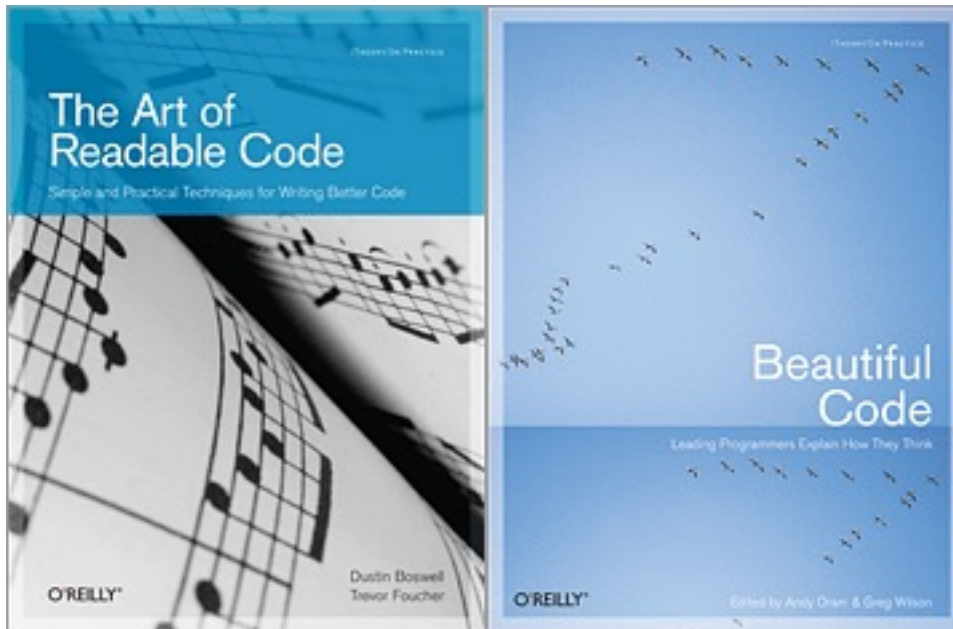
Quel est votre objectif ?

Des programmes

- Ecrits rapidement
- Denses
- Qui utilisent toutes les astuces du langage

Des programmes

- Corrects
 - Le programme doit résoudre correctement le problème posé (même si celui-ci est simple)
- Clairs
 - Le programme doit être lisible et documenté
- Testés
 - Vous devez pouvoir vérifier les limites de votre programme
- Efficaces
 - p.ex. chemin le plus court



Efficacité versus lisibilité

```
while get_direction() != EAST:  
    turn_right()
```

Court
Compact
Compréhensible

```
if get_direction == NORTH:  
    turn_right()  
if get_direction == WEST:  
    turn_right()  
    turn_right()  
if get_direction == SOUTH:  
    turn_left()
```

Un peu "plus efficace"
(moins d'opérations)

mais beaucoup plus long
et moins compréhensible