

Python avancé

Semaine 1

Objectifs

- réaliser un vrai programme Python
- réfléchir à un problème concret
- organiser son code
- faire un premier rendu sous git

Prérequis

- avoir git installé
- et avoir retenu quelques bases du cours git
- avoir python et conda installés
- et avoir retenu quelques bases du primer

Partie I: LE PROJET

Le jeu du serpent

Simplifié

- le serpent est une suite de *blocs*
- à chaque fois qu'il mange un oeuf, il gagne un nouveau bloc
- si il se mord la queue, c'est perdu



A sepia-toned photograph of Harry Potter from the Harry Potter film series. He is wearing round glasses and a plaid shirt, pointing his wand towards the viewer. The background is dark and out of focus.

Setup

on utilise conda¹ et pip comme pour le primer:

```
$ conda create -n pas1 python=3.7  
$ conda activate pas1  
$ conda install pip  
$ pip install pygame
```

¹cf. [Managing Environments](#) dans la doc conda

on utilise conda¹ et pip comme pour le primer:

```
$ conda create -n pas1 python=3.7  
$ conda activate pas1  
$ conda install pip  
$ pip install pygame
```

¹cf. [Managing Environments](#) dans la doc conda

on utilise conda¹ et pip comme pour le primer:

```
$ conda create -n pas1 python=3.7  
$ conda activate pas1  
$ conda install pip  
$ pip install pygame
```

¹cf. [Managing Environments](#) dans la doc conda

enfin on récupère le projet "starter"

le fichier main.py est disponible sur:

<https://github.com/pythonadvanced/2019SI>

ce qui est attendu

1. créer un dépôt git local pour travailler
2. copier ce fichier et faire un commit
3. tout au long du TP, vous ferez des commits pour sauvegarder votre avancée

A dark, moody photograph of a person sitting in a car at night. The person is seen from the side, looking out of the window. The interior of the car is visible, with the dashboard and door panels. The lighting is low, creating a somber atmosphere.

PENSEZ À COMMITER

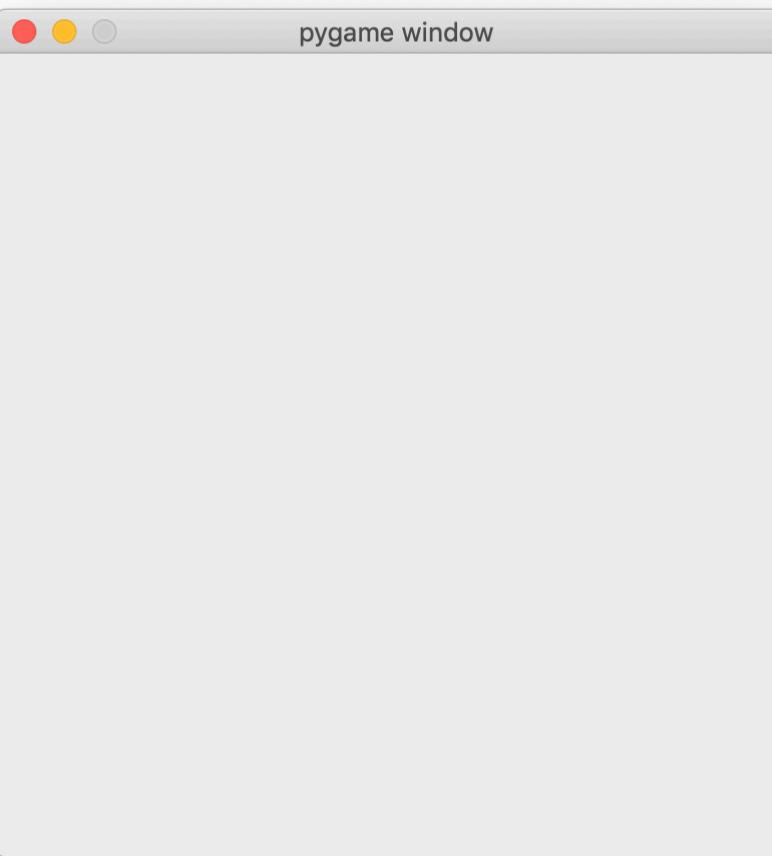
Code de démarrage

```
import sys
import pygame
from pygame.locals import *

# on doit "initialiser" PyGame
pygame.init()
# et définir la taille de la fenêtre (400x400)
screen = pygame.display.set_mode((400, 400))

while True:
    for event in pygame.event.get(KEYDOWN):
        if event.key == K_q: # pygame.locals.K_q
            sys.exit() # quitte le programme
```

```
$ python main.py
```



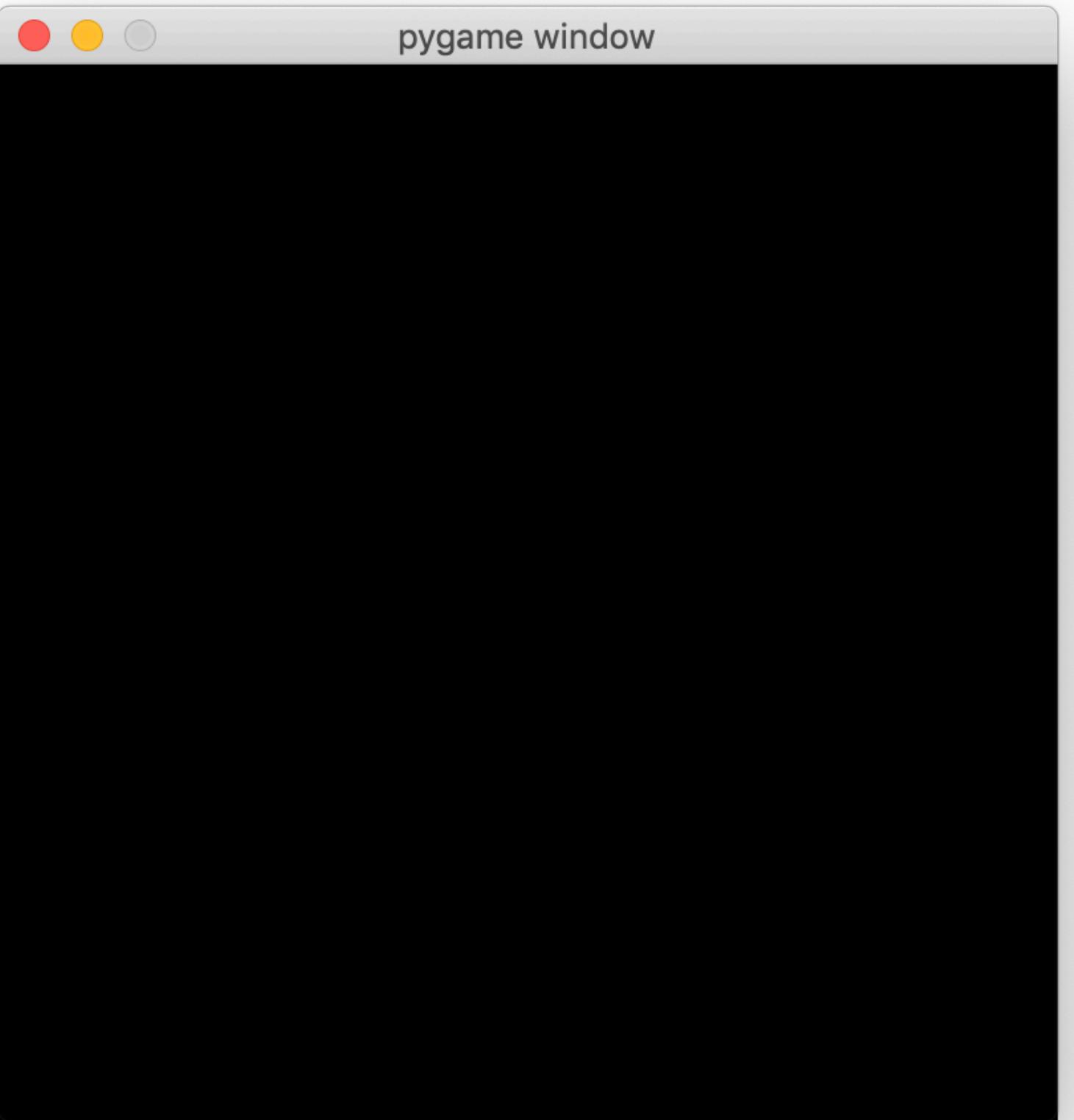
Dessinons

Un fond noir²

```
BLACK = (0, 0, 0)
```

```
white True:
```

```
#...
screen.fill(BLACK)
pygame.display.update()
```



²les couleurs sont en (r, g, b) (rouge - vert - bleu)

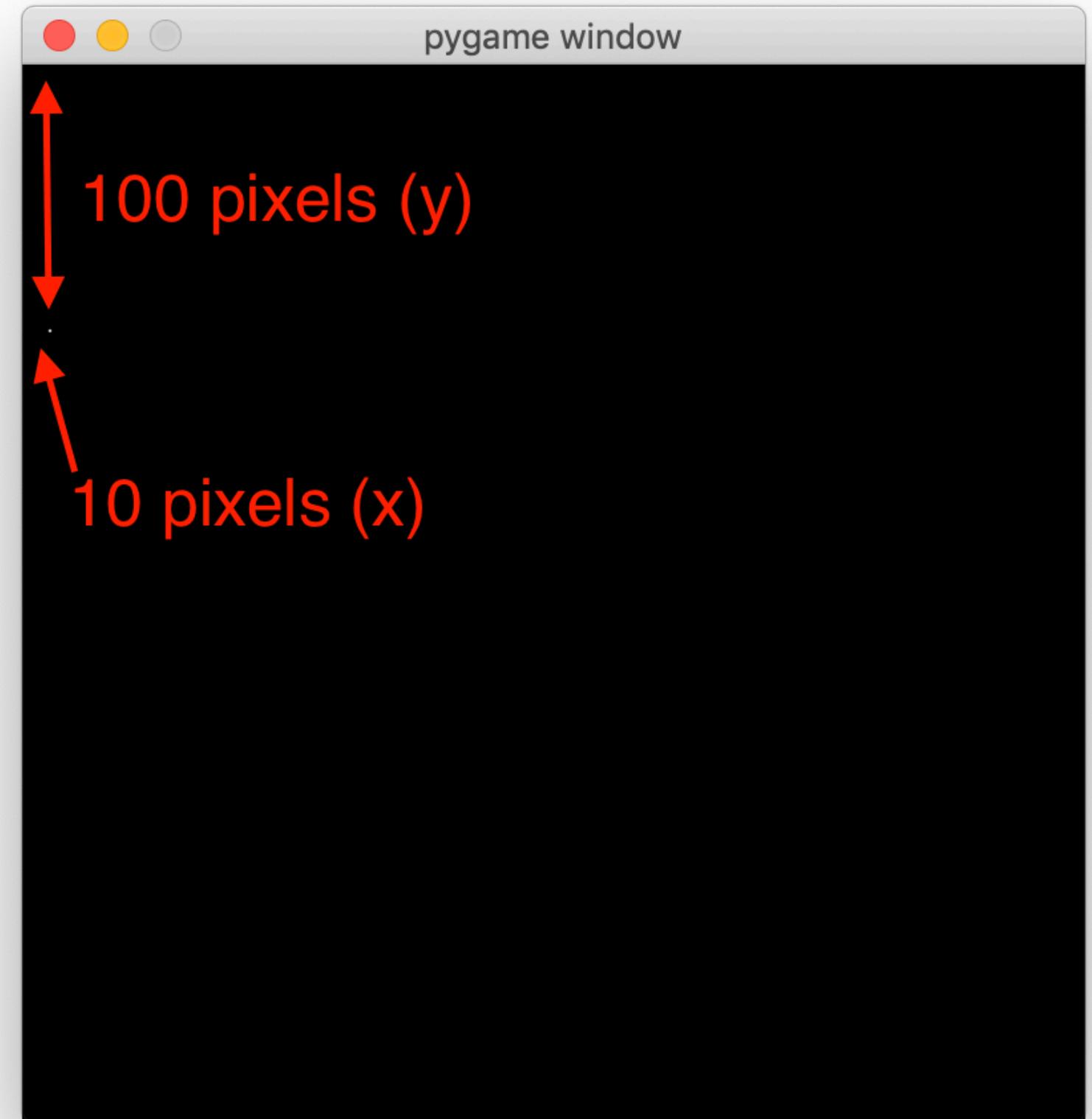
Note importante

- les variables (l’”état”) vont en haut du fichier: ici la constante BLACK
- et les instruction de rendu vont dans la “boucle” qui s’exécute à chaque image: ici screen.fill() et display.update()

Et un pixel blanc

```
WHITE = (255, 255, 255)
```

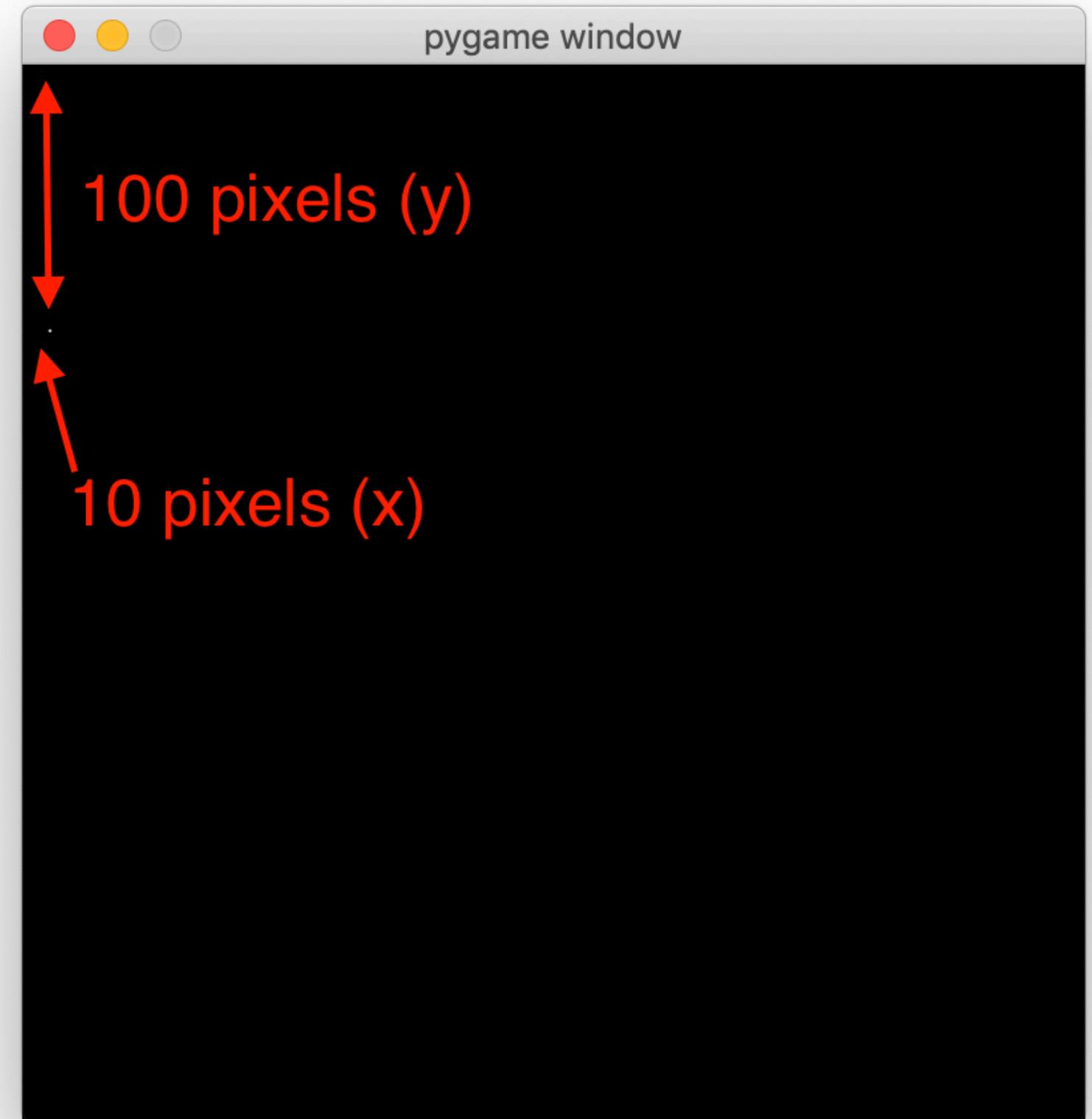
```
white True:  
# ...  
screen.fill(BLACK)  
screen.set_at((10, 100), WHITE)  
pygame.display.update()
```



Et un pixel blanc

```
WHITE = (255, 255, 255)
```

```
white True:  
# ...  
screen.fill(BLACK)  
screen.set_at((10, 100), WHITE)  
pygame.display.update()
```



Exercice: la cellule

- on découpe l'écran en 40 lignes & 40 colonnes
- chaque cellule fait donc 20 pixels de large et de haut

Écrire une fonction `draw_cell(pos, color)` qui reçoit
une position (x, y) ⁴ et remplit le carré de 20×20
correspondant avec `color`.

⁴en coordonnées "plateau" $0 \leq x, y < 20$



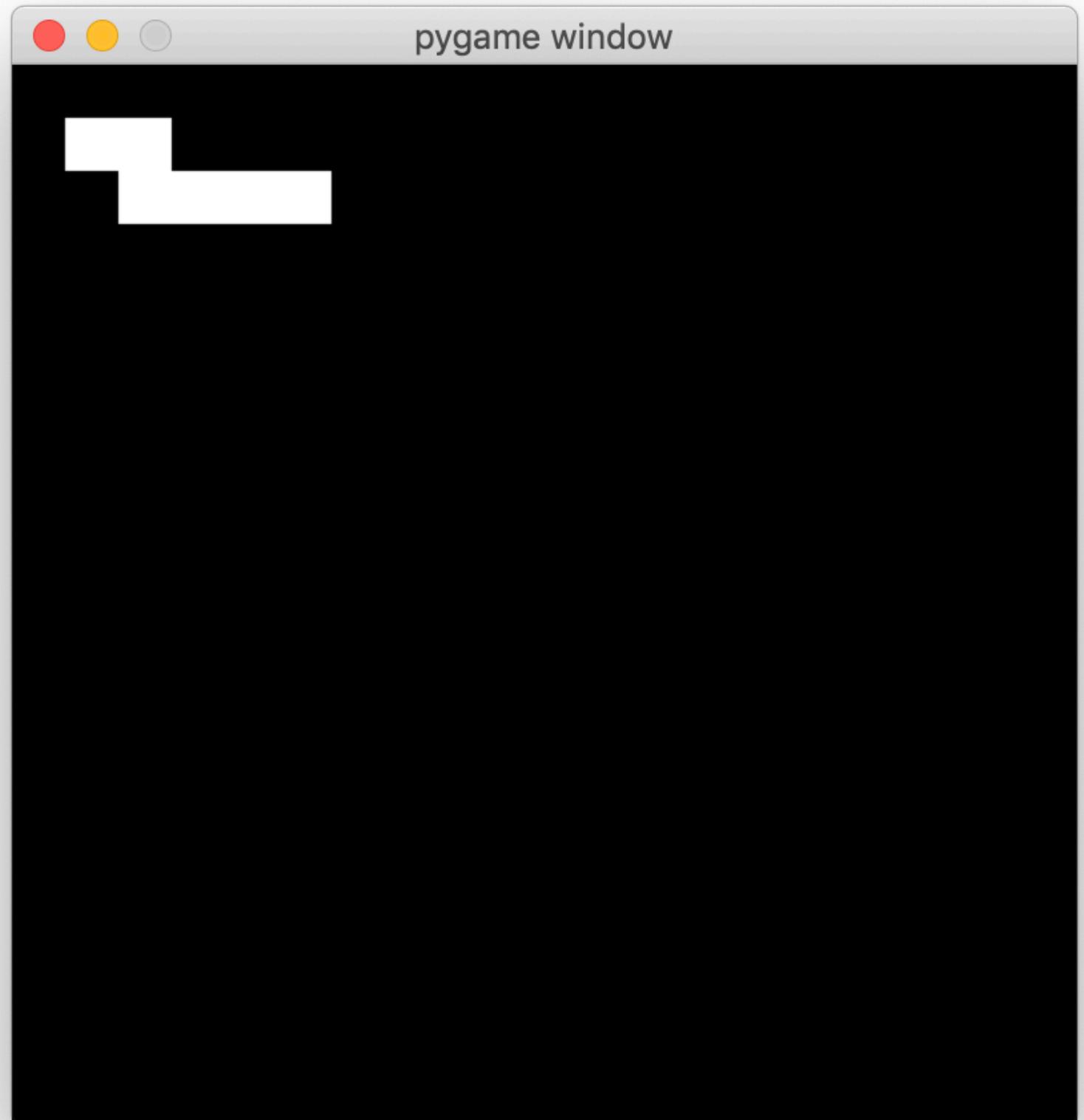
COMMIT

Exercice: le serpent

on définit le serpent comme une suite de positions:

```
snake = [  
    (1, 1), // <- position d'une "cellule" du plateau  
    (2, 1),  
    (2, 2),  
    (3, 2),  
    (4, 2),  
    (5, 2)  
]
```

Tracer la position actuelle du serpent à l'écran





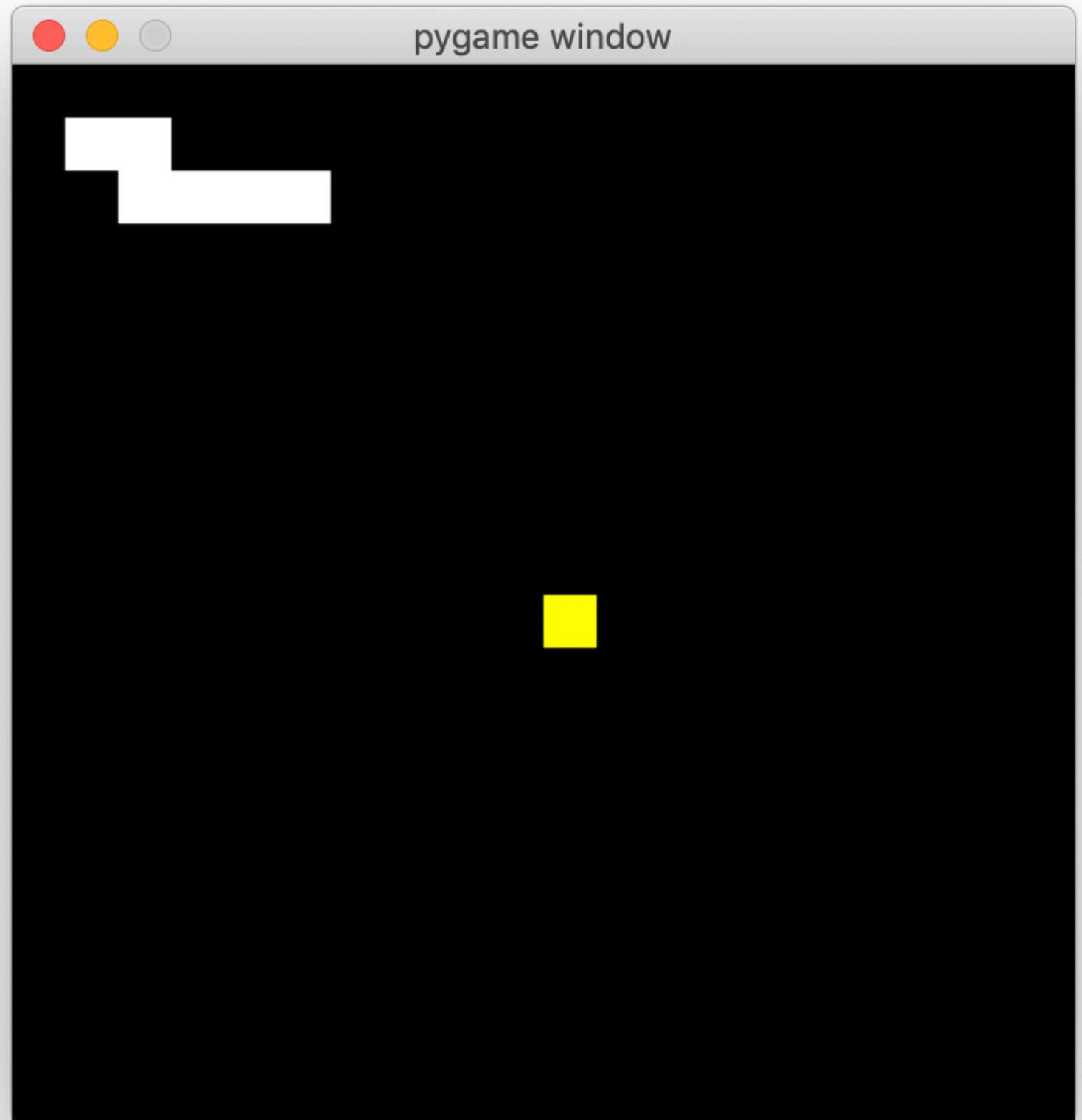
COMMIT

Exrcice: l'oeuf

on définit l'oeuf par une variable
(un état) contenant sa position:

```
egg = (10, 10)
```

afficher l'oeuf à l'écran en jaune





COMMIT

vous avez deviné ?

Animons

La notion de temps

Ajoutons un objet pygame.time.Clock() à notre projet:

```
clock = pygame.time.Clock()
```

```
# ...
```

```
while True:
```

```
    clock.tick(60)
```

```
# ...
```

La notion de temps (suite)

- l'objet `clock` permet de limiter le nombre d'images par secondes
- ici en appelant `clock.tick(60)`, on vérifie qu'au moins 16,66ms se sont écoulées depuis la dernière image, sinon on attend un peu

Exercice: le serpent qui bouge

On souhaite déplacer le serpent.

Pour rappel:

- le serpent bouge par la tête, une case après l'autre
- **BONUS:** quand il arrive au bout de l'écran, il ressort de l'autre côté



Exercice: le serpent qui bouge (suite)

on ajoute deux variables d'état dx et dy

```
// vecteur déplacement du serpent  
dx, dy = 1, 0  
  
while True:  
    #...  
    move_snake() # <- il bouge 1 fois par frame
```

Écrire la fonction move_snake() qui déplace le serpent



COMMIT

Exercice: on ralentit (v1)

le serpent bouge trop vite: modifier le programme pour qu'il ne se déplace que toutes les 200ms

- Utiliser le fait que l'on peut régler nombre d'images par seconde



COMMIT

Exercice: on ralentit (v2)

On veut éviter de limiter le framerate du jeu, nous allons utiliser la valeur de retour de `clock.tick()`:

`while True:`

```
# delay_ms contient le temps écoulé depuis la dernière image (en ms)
delay_ms = clock.tick(60)
```

modifier le programme pour qu'il déplacer le serpent toutes les 200ms, sans modifier le framerate



COMMIT

Événements

contrôler le jeu

`pygame.locals` définit des valeurs de `event.key` pour les flèches directionnelles:

- la flèche du haut: `K_UP`
- la flèche du bas: `K_DOWN`
- la flèche de gauche: `K_LEFT`
- la flèche de droite: `K_RIGHT`

on peut donc détecter ces touches:

```
while True:  
    for event in pygame.event.get(KEYDOWN):  
        if event.key == K_q:  
            sys.exit() # quitte le programme  
    elif event.key = K_UP:  
        # ...
```

Exercice: déplacer le serpent

Compléter le programme pour détecter ces touches et déplacer le serpent en fonction des touches pressées par l'utilisateur

A dark, moody photograph of a person sitting in a car at night, looking out the window. The scene is dimly lit, with the person's face partially obscured by shadow. The word "COMMIT" is overlaid in large, white, sans-serif capital letters.

COMMIT

BONUS: en option

le serpent n'est pas autorisé à revenir sur ses pas, c-à-d.

- repartir vers la gauche quand il va vers la droite
- repartir vers la droit quand il va vers la gauche
- repartir vers le haut quand il va vers le bas
- repartir vers la bas quand il va vers le haut

modifier le programme pour éviter ces cas

Finalisation

(gagner) et perdre

Exercice: l'oeuf aléatoire

créer une fonction `place_egg()` qui crée un oeuf à une position aléatoire du plateau, non occupée par le serpent

- lire la doc de `random.randrange()`
- appeler la méthode en début de programme pour placer le premier oeuf

A dark, moody photograph of a person sitting in a car at night, looking out the window. The scene is dimly lit, with the person's face partially obscured by shadow. The word "COMMIT" is overlaid in large, white, sans-serif capital letters.

COMMIT

Exercice: manger l'oeuf

Modifier le code pour que lorsque le serpent mange un oeuf, il se rallonge de une cellule



COMMIT

Exercice: le serpent se mord la queue

créer une fonction `end_of_game()` qui détecte si le serpent se "mord la queue", c-à-d. si sa tête arrive sur un case déjà occupée par son corps.

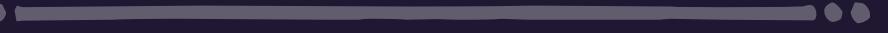


COMMIT

Partie II: get real



Transformer ce code en un vrai projet Python



avec des classes et tout...

Créons une classe

Dans sa version actuelle, notre programme manque d'organisation.

Commençons par embarquer notre projet à l'intérieur d'une classe Game qui encapsule le code actuel.

Exercice

Regrouper tout le code du jeu à l'intérieur d'une classe Game, telle que le démarrage du jeu se fasse ainsi:

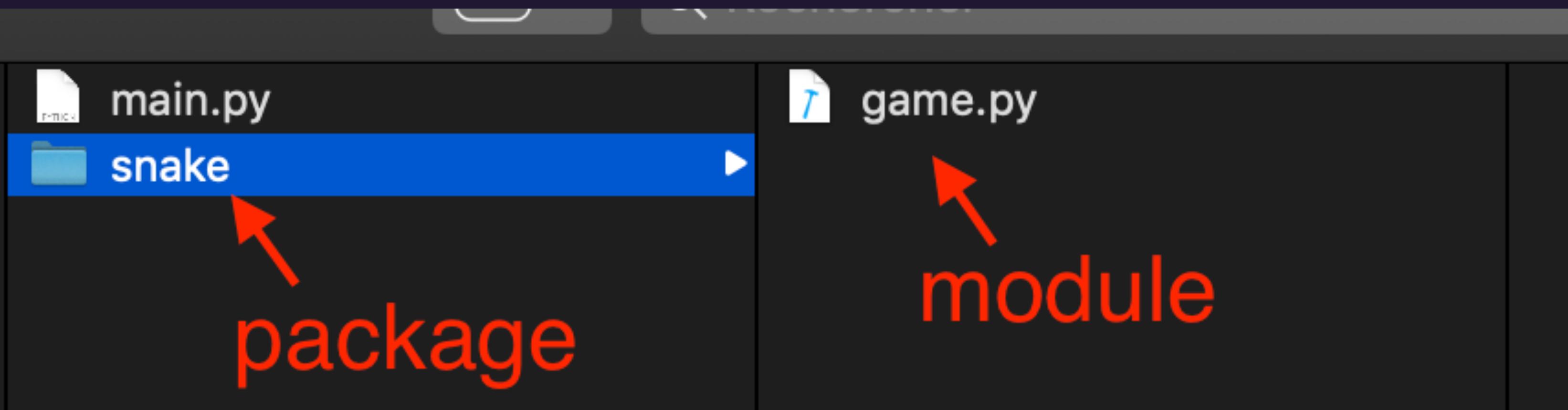
```
game = Game(  
    board_size=(40, 40),  
    cell_size=(20, 20)  
)  
game.start()
```

Exercice: créer un module et un package

Comme nous l'avons vu pendant le Python Primer:

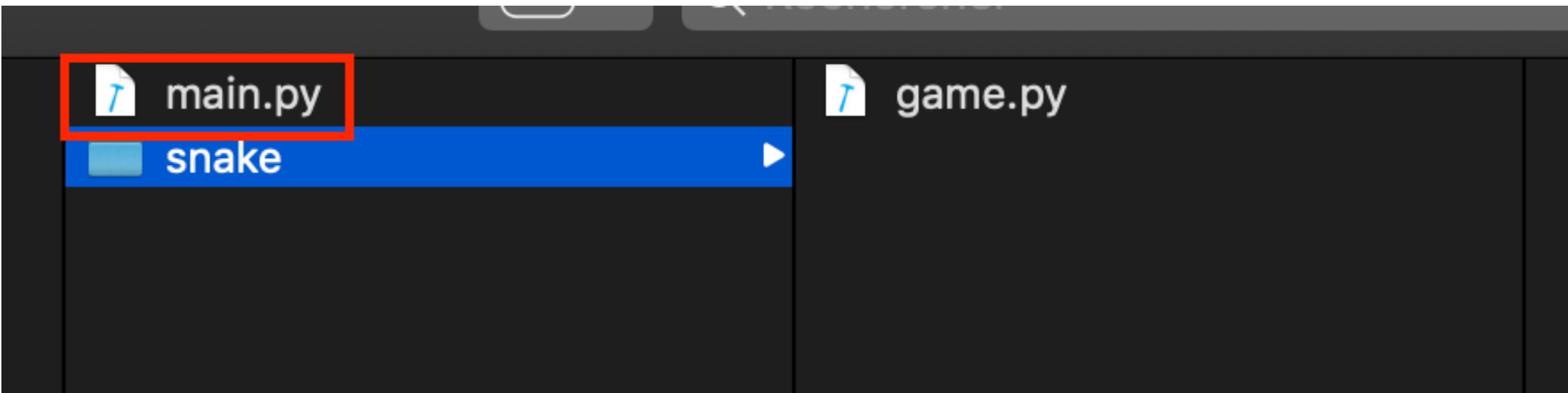
- un module est un fichier .py
- un package est un répertoire contenant des modules

déplaçons notre code dans un package nommé snake et
un module nommé snake.game



Exercice: invoquer notre module

On se propose désormais d'importer notre classe Game depuis notre module game du package snake, le tout depuis un fichier main.py à la racine du projet.



Ajouter des options à notre programme

On souhaite pouvoir ajouter des options à notre programme, afin de pouvoir changer de manière dynamique la taille du plateau:

```
$ python main.py --width 40 --height 40
```

Exercice

- lire la doc du module argparse
- ajouter des options à notre programme:
 - une option --width (ou -W) qui définit la largeur du plateau
 - une option --height (ou -H) qui définit la hauteur du plateau