



BRANCHE	SECTION(S)	ÉPREUVE ÉCRITE
INFORMATIQUE	B	Durée de l'épreuve : 3h Date de l'épreuve : 9 juin 2020

Dans votre répertoire de travail (à définir par chaque lycée), vous trouverez un dossier nommé **EXAMEN\_CB**. Renommez ce dossier par votre code de l'examen (p.ex. : **LXY\_CB1\_07**).

Ce dossier sera appelé dossier de travail dans la suite. Sauvegardez tous vos fichiers dans ce dossier.

Ajouter votre code d'examen sous forme de commentaire tout au début de chaque code-source !

### Question 1 (functions.py, 2 + 6 = 8 points) :

Soit la fonction numérique récursive **fr** au paramètre entier positif **n** illustrée ci-dessous :

Dans votre dossier de travail, créez le programme **functions.py** et encodez-y la fonction **fr** et exécutez-la avec les valeurs suivantes : 0, 5, 521, 525, 12321, 12341

- Quel est, en général, le calcul effectué par cette fonction ?  
Encodez le texte de votre réponse sous forme de commentaire en-dessous de la fonction.
- Encodez une version itérative **fi** de cette fonction.

```
from math import log10
def fr(n):
    if n < 10:
        return True
    else:
        d = 10 ** int(log10(n))
        if n % 10 != n // d:
            return False
        else:
            return fr(n % d // 10)
```

### Question 2 (snake.py, 8+21+23 = 52 points) :

Dans votre dossier de travail, créez le jeu **Snake** (snake.py). Ce jeu est basé sur les classes **Element** et **Snake** et fait appel à la librairie **pygame**. Il consiste à faire bouger (à l'aide du clavier) un serpent vers des éléments de nourriture (en orange) tout en évitant les éléments de poison (en bleu cyan), les bords du terrain de jeu (donc de la fenêtre d'application) et de passer sur lui-même.

Les seules importations permises sont celles de **pygame**, **pygame.locals**, **sys**, et des fonctions **randint** du module **random**, **sqrt** du module **math** et **deepcopy** du module **copy**.

#### I) La classe **Element** : (2+2+2+2 = 8 p.)

La classe **Element** décrit un élément de nourriture, de poison et de corps du serpent. Elle dispose des coordonnées entières de son centre et de ses couleurs de bord et de remplissage.

- Le **constructeur** initialise les attributs sur les valeurs passées aux paramètres respectifs. (2 p.)
- La méthode **draw** dessine sur la toile **pygame** l'élément sous forme de disque de centre (x, y), de rayon 7 pixels et aux couleurs indiquées pour l'intérieur et le bord. L'épaisseur du bord est de 1 pixel. (2 p.)
- La méthode **move** aux paramètres **dx** et **dy** déplace l'élément horizontalement de **dx** pixels et verticalement de **dy** pixels. (2 p.)
- La méthode **has\_touched** au paramètre **other** détecte un contact avec un autre élément (**other**) et retourne le résultat sous forme booléenne. On considère qu'un contact a lieu lorsque la distance entre les deux centres est strictement inférieure à 10 pixels. (2 p.)

Element
x : int
y : int
border_color : Color
fill_color : Color
__init__(x : int, y : int, border_color : Color, fill_color : Color)
draw()
move(dx : int, dy : int)
has_touched(other : Element)

## II) La classe Snake : (4+5+5+2+2+3 = 21 p.)

La classe **Snake** décrit un serpent.

Elle dispose des attributs suivants :

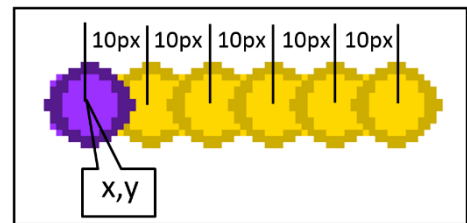
- sa tête (**head**) sous forme d'un élément de couleur pourpre,
- son corps (**body**) sous forme d'une liste d'éléments,
- son énergie (**energy**) (entière),
- sa direction de mouvement (**direction**) qui est l'un des tuplets ci-contre :

tuplet	direction
(1,0)	droite
(-1,0)	gauche
(0,1)	bas
(0,-1)	haut

Snake
head : Element
body : Element[ ]
energy : int
direction : 2-tuple
<code>__init__(x : int, y : int)</code>
draw()
move()
grow()
has_touched(elements : Element[ ])
has_hit_wall(max_x : int, max_y : int)


1) Le **constructeur** effectue les opérations suivantes : (4 p.)

- il crée, aux coordonnées (x, y), la tête de couleurs d'intérieur "**purple1**" et de bord "**purple4**",
- il crée le corps sous forme de 5 éléments de couleurs d'intérieur "**gold1**" et de bord "**gold4**", décalés chacun de 10 pixels vers la droite par rapport à l'élément précédent,
- il initialise l'énergie à 200 unités,
- il fixe la direction de déplacement sur gauche (voir tableau ci-dessus).



2) La méthode **move** fait avancer le serpent de 10 pixels dans la direction mémorisée dans son attribut **direction**. Le mouvement est réalisé comme suit : (5 p.)

- un nouvel élément de corps, aux coordonnées de la tête, est inséré en tête de liste, puis le dernier élément de la liste est supprimé,
- la tête se déplace de 10 pixels dans la direction en question,
- l'énergie est diminuée d'une unité.

3) La méthode **draw** dessine sur la toile pygame, aux coordonnées (5, 5)  la barre d'énergie du serpent sous forme d'un rectangle **jaune** de hauteur 5 pixels et de longueur égale à son énergie. Le restant, allant jusqu'à 200, est de couleur **bleue**.

Ensuite le corps du serpent est dessiné comme illustré ci-dessus et, finalement, sa tête. (5 p.)

4) La méthode **grow** fait augmenter la taille du serpent de 8 éléments. Ceci est réalisé en ajoutant 8 copies du dernier élément en fin de liste. (2 p.)

5) La méthode **has\_hit\_wall** détecte si la distance entre le centre de la tête du serpent et les bords de la surface de jeu (donc de la fenêtre d'application) est strictement inférieure à 7 pixels et retourne le résultat sous forme booléenne. Les paramètres **max\_x** et **max\_y** indiquent la largeur, respectivement la hauteur de la fenêtre d'application. (2 p.)

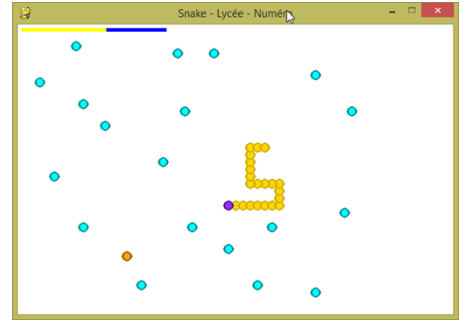
6) La méthode **has\_touched** se sert de la méthode **has\_touched** de la classe **Element** pour détecter si la tête du serpent a touché un des éléments de la liste fournie comme paramètre et retourne le résultat sous forme booléenne. (3 p.)

### III) Le programme principal pygame: (2+2+9+7+3 = 23 p.)

1. Effectuez les initialisations nécessaires pour créer un environnement pygame aux caractéristiques suivantes : (2 p.)
  - la taille de la fenêtre d'application est de 600 x 400 px,
  - l'entête de la fenêtre est "Snake – code d'examen", p. ex. "**Snake - LXY\_CB1\_07**",
  - au début du programme, la surface de dessin est vide et **blanche** et elle est rafraîchie 8 fois par seconde,
  - une variable booléenne **is\_running** indique si le jeu est actif ou non,
  - une variable **game\_color** indique la couleur de l'arrière-fond de la surface de jeu
2. Tant que le jeu n'est pas quitté, la boucle principale réagit correctement aux commandes de l'utilisateur et effectue une mise-à-jour de l'interface du jeu. A la fin du jeu, l'environnement pygame et l'application sont clôturés et quittés correctement. (2 p.)
3. Les commandes gérées sont les suivantes :
  - a) La touche "**n**" crée et active un nouveau jeu en effectuant les opérations suivantes :
    - i. création d'une liste **food** de 8 éléments de nourriture (de type **Element** et de couleurs "**orange1**", resp. "**orange4**") à des coordonnées aléatoires, multiples de 10, mais en gardant une distance minimum de 20 pixels par rapport aux bords de fenêtre, (2 p.)
    - ii. création d'une liste **poison** de 20 éléments de poison (de type **Element** et de couleurs "**cyan1**", resp. "**cyan4**") à des coordonnées aléatoires, multiples de 10, mais en gardant une distance minimum de 20 pixels par rapport aux bords de fenêtre, (2 p.)
    - iii. création d'un nouveau serpent **snake** aux coordonnées aléatoires, multiples de 10, mais en gardant pour les coordonnées de la tête une distance minimum de 100 pixels par rapport aux bords de fenêtre, (2 p.)
    - iv. le jeu est activé et la couleur du jeu est mise sur **blanc**. (1 p.)

Remarque : On accepte que deux éléments (de nourriture, de poison, de serpent) se trouvent éventuellement au même endroit.
  - b) Pendant le jeu **un appui sur une touche de curseur** du clavier est mémorisé par le serpent. (2 p.)
4. La gestion du jeu, s'il est en cours, consiste en les opérations suivantes :
  - déplacer le serpent, (1 p.)
  - vérifier si le serpent a touché le premier élément de nourriture. Dans ce cas, cet élément est supprimé de la liste **food**, le serpent grandit et son énergie augmente de 20 unités, mais sans pouvoir dépasser le maximum de 200 unités. Si le serpent a mangé tous les éléments de nourriture, alors le jeu est fini (désactivé) et la couleur du jeu est mise sur **khaki**, (3 p.)
  - vérifier si le serpent a touché un des éléments de poison, s'il a touché le mur, s'il a touché une partie de son corps (body) ou s'il n'a plus d'énergie. Dans chacun de ces 4 cas, le jeu est fini (désactivé) et la couleur du jeu est mise sur **magenta**,

Remarque : la barre d'énergie peut être touchée sans danger (3 p.)

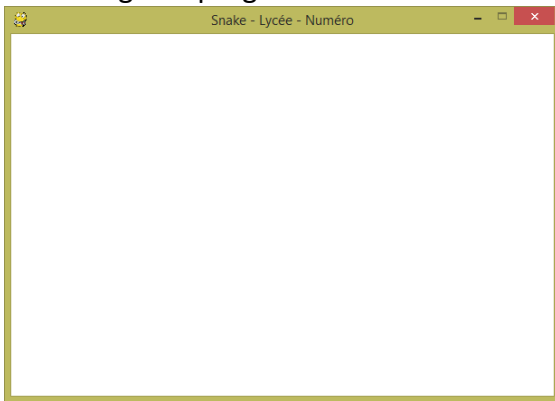


5. Le jeu n'est mis-à-jour que s'il est en cours et consiste en les opérations suivantes : (3 p.)

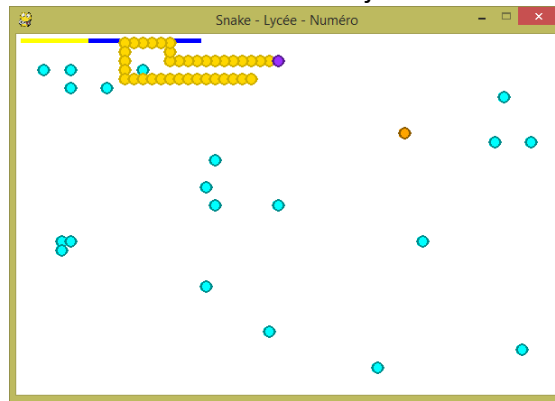
- effacer et remplir la surface de jeu par la couleur de dessin en cours,
  - afficher tous les éléments de poison,
  - afficher le premier élément de nourriture (s'il y en a),
  - afficher le serpent.
- 

Copies d'écrans de différentes étapes du jeu :

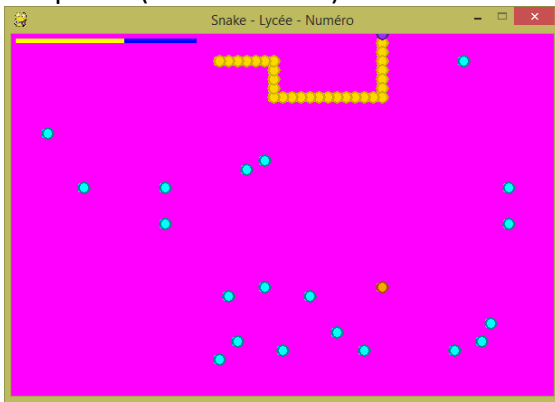
Démarrage du programme :



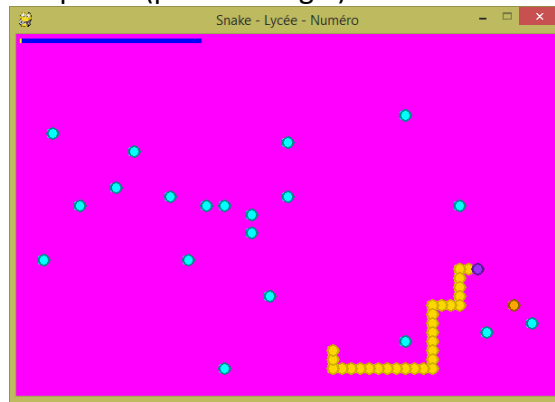
déroulement normal du jeu :



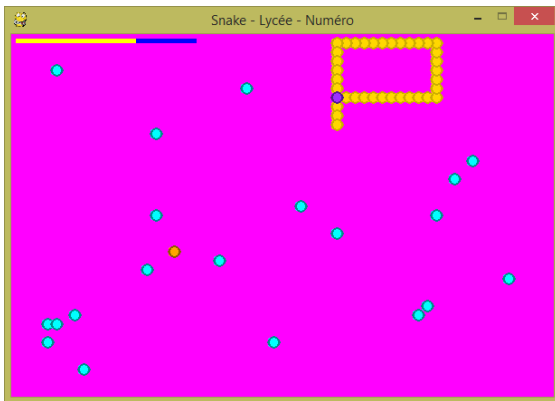
Jeu perdu (touché le mur) :



Jeu perdu (plus d'énergie) :



Jeu perdu (touché son propre corps) :



Jeu gagné :

