But

Le but est d'utiliser une liste chaînée afin d'implémenter le jeu Snake.

La problématique

Un serpent évolue dans une fenêtre graphique dont il ne doit jamais sortir (sinon le jeu s'arrête). Il peut grandir et gagner des points en mangeant des pommes qui apparaissent aléatoirement dans la fenêtre. Il ne doit alors pas s'auto-rentrer dedans (sinon le jeu s'arrête).

Lorsque le serpent avance, le corps suit sa tête. Ainsi le mouvement est entièrement défini par là où passe la tête.

De plus, à chaque tour, lorsque la tête avance chaque partie du corps prend la position de la partie devant. Ainsi seule la tête et la queue du serpent donne l'illusion d'un serpent qui avance.

Il suffit donc de définir le corps du serpent par une liste chaînée dont la tête avance à chaque tour selon sa vitesse horizontale et verticale tandis que la queue est supprimée.

À chaque instant, une pomme verte sera aléatoirement disponible dans la fenêtre. Lorsque la tête du serpent passe dessus, le serpent grandi d'une case.

Les super-pommes (jaunes) en sus de faire grandir le serpent, lui ajoute un bonus de points. Comme les pommes vertes, une seule n'est disponible à chaque instant, mais contrairement aux vertes, il peut y avoir des moments où il n'y a pas de super-pommes sur le plateau.

Les touches directionnelles serviront pour diriger le 1^er serpent.

Exemple / aide

◇ On pourra importer le module `projet03_pygame.py` via la ligne `import projet03_pygame as ppg`.
Ce module contient des couleurs codées en RGB, les dimensions de la fenêtre dans laquelle évoluera le serpent (`DIMX` et `DIMY`), la hauteur prévue pour afficher le score ainsi que la taille d'un pixel.

◇ En `pygame`, l'axe des abscisses est bien orientée mais l'axe des ordonnées est orientée vers le bas.

◇ La boucle principale de la fonction `main() -> None` pourra être composée comme suit :

```
ppg.init()
bconti = True
while bconti :
    ppg.miseAJourActions()
    # ppg.procEvent( s ) # vitesse du serpent mise à jour selon les touches enfoncées
    # votre code
    ppg.wait( 100 - 10*LEVEL )
    if ppg.sortie() : bconti = False
ppg.fin()
```

◇ on pourra générer un point aléatoirement dans la fenêtre ainsi :

```
import random
Point( random.randint(1, ppg.DIMX-1), random.randint(1, ppg.DIMY-1) )
```

Étapes

Par la suite, on ne commentera pas les méthodes (seulement la classe englobante) mais on définira les types de leur entrées et sorties. Ceci dit, on continuera à commenter les fonctions.

a) Dans la section T^le NSI du site [forhan.maths.free.fr], télécharger le fichier `projet03_pygame.py`.

b) Définir une classe `Point` pour laquelle vous définirez outre le constructeur, les méthodes `__repr__() -> str` et `__str__() -> str`.
rq : la méthode `__str__() -> str` sera définie pour que `print( Point(2, 3) )` affiche `"(2,3)"`.

Point
x (int)
y (int)

c) Définir également les méthodes `get_x() -> int` et `get_y() -> int`.
À ce stade, vous pouvez afficher un point via la fonction ***`ppg.affichePt( Point(2,3) )`***

d) Enfin définir la méthode `__eq__() -> bool` et tester votre classe `Point`.

e) Proposer une fonction `testPoint() -> None` qui affiche un point en bleu 1s puis l'efface 1s puis le réaffiche 1s etc.

Pour définir le corps du serpent, on souhaite définir une liste chaînée d'objets `Point`. Pour se faire, on se propose d'utiliser la classe liste chaînée offerte par `python`.
On ajoute au corps, la vitesse horizontale et verticale de la tête du serpent ainsi que le score attaché au serpent.
On définira alors la classe `Snake` ainsi :

Snake
corps (list)
vX (int)
vY (int)
score (int)

f) On souhaite faire commencer le serpent au tiers horizontal et vertical de l'écran avec un déplacement horizontal vers la droite. Compléter le constructeur suivant :

```
def __init__(self, i : int = 1) -> 'Snake' :
    self.corps = [ Point( i*ppg.DIMX//3 , i*ppg.DIMY//3 ) ]
    ...
```

g) Définir la méthode `__len__() -> int`, qui renvoie la longueur du corps du serpent.

Définir la méthode `__str__() -> str`, qui renvoie la chaîne de caractère caractérisant le serpent.

h) Cette chaîne comportera la vitesse de la tête ainsi que les coordonnées de chaque point composant le corps du serpent.
expl avec `(DIMX, DIMY)=(10,10)` :

```
s = Snake()
s.avanceT()
s.avanceT()
print( s ) # affiche :
""" vitesse : [vX=1, vY=0]
corps : (5,3),(4,3),(3,3) """
```

i) Définir les méthodes `get_tete() -> 'Point'` et `get_queue() -> 'Point'`, qui renvoie l'objet point respectivement à la tête du serpent et à la queue du serpent.

j) Définir la méthode `avanceT(self) -> None`, qui ajoute agrandi le serpent depuis la tête selon sa vitesse de déplacement.

À ce niveau, vous pouvez tester votre fonction `__str__() -> str` avec l'exemple donné.

k) Définir la méthode `avanceQ(self) -> None`, qui fait 'avancer' la queue du serpent.

l) Définir les méthodes `get_score() -> int` et `add_score(v : int) -> None`, qui renvoie le score ou ajoute la valeur v au score.

Tester vos méthodes.

m) Définir la fonction `set_vitesse(vitX : int, vitY : int) -> None`, qui modifie la vitesse du serpent.

À ce stade, vous pouvez appeler la fonction `ppg.procEvent( s )`

n) Définir la méthode `__contains__(p : 'Point') -> bool`, qui renvoie `True` si le point `p` fait partie du corps du serpent.

o) Définir la méthode `collision() -> bool`, qui renvoie `True` si le serpent s'auto-percute.

p) Définir la méthode `miam(pom : 'Point') -> bool`, qui renvoie `True` si la tête du serpent vient de manger la pomme `pom`.

À ce stade, votre classe `Snake` est complète.

q) Définir une fonction `inScreen( s : 'Snake') -> bool` qui renvoie `True` si la tête du serpent est dans la fenêtre définie par `ppg.DIMX` et `ppg.DIMY`.

r) Proposer une fonction `main() -> None` qui affiche et fasse bouger le serpent à l'aide des fonctions du module `projet03_pygame`.

À ce stade, vous pouvez afficher votre serpent (sans qu'il grandisse) et le faire bouger dans la fenêtre

Par la suite, modifier la fonction `main()` au fur et à mesure de vos besoins.

s) Définir la fonction `newPomme() -> 'Point'`, qui renvoie un objet 'Point' aléatoirement dans le rectangle de diagonale (1,1)$->(DIMX, DIMY)$.

à ce stade, votre jeu doit être opérationnel. Tester-le pour en être sûr.

t) Coder un menu qui demande si l'utilisateur veut des supers-pommes (de couleur jaunes qui apporteront un bonus +10 sur le score) et à quel niveau il veut jouer (`LEVEL` entre 1 et 9).

Les super-pommes ne s'afficheront que 10s et à chaque tour de boucle, si elle n'existe pas déjà, auront une probabilité de 1% d'apparaître aléatoirement.

u) Approfondissement :

- ◇ Modifier le code du module `projet03_pygame.py` (ajout ou modification d'une fonction `procEvent()` et `score()`) ainsi que votre code afin de permettre à deux joueurs de jouer simultanément :
  - vous pourrez modifier la taille de la fenêtre pygame en ajoutant au module `projet03_pygame.py` une fonction `set_dimensions(dx : int, dy : int) -> None` qui modifiera les variables `DIMX` et `DIMY`,
  - le 2è serpent partira des deux tiers horizontal et vertical de l'écran avec un déplacement horrizontal vers la gauche.
  - il faudra bien gérer la collision entre les serpents,
- ◇ Proposer de sauvegarder les cinq meilleurs scores dans un fichier `projet03_scores.csv` de la forme

nom,score
Méduse,9999
boa,1000
python,50

- ◇ Proposer un menu() qui ne demande aucun entrée dans le shell mais uniquement de taper sur des touches du clavier qui seront reconnues par une fonction similaire à `procEvent()`.

Rendu

À la fin du temps imparti, chaque groupe rendra 2 travaux :

a) un compte-rendu (au maximum 2 pages) contenant un résumé de votre projet, les jeux de tests effectués ainsi que leurs résultats.

b) le fichier contenant le code `python` sera envoyé sous format numérique via l'ENT