Un premier jeu avec Pyxel

Doc. 2,5

Objectif

En construisant pas à pas un petit jeu avec la bibliothèque Pyxel, vous allez réinvestir les fonctions, les instructions conditionnelles et les boucles dans un vrai programme, et découvrir la différence entre variables locales et variables globales.

Dans le jeu final, le joueur est un carré vert posé sur un sol. Il se déplace avec les flèches du clavier et doit attraper des cibles jaunes qui apparaissent au hasard. Chaque cible attrapée rapporte un point ; la partie est gagnée à 10 points.

Le programme de départ est fourni dans l'activité Capytale. Lorsqu'une question commence par « Sans exécuter le programme », il faut écrire sa réponse avant de lancer le programme, puis vérifier.

§1. Découverte de Pyxel

Voici le programme de départ :

```python
import pyxel

LARGEUR = 160
HAUTEUR = 120

def update():
    pass

def draw():
    pyxel.cls(0)
    pyxel.text(50, 60, "Bonjour NSI !", 7)

pyxel.init(LARGEUR, HAUTEUR, title="Mon premier jeu")
pyxel.run(update, draw)
```

Un programme Pyxel contient deux fonctions particulières. La fonction `update()` fait évoluer l'état du jeu (position des personnages, score…) ; la fonction `draw()` dessine cet état à l'écran. L'instruction `pyxel.run(update, draw)` lance le jeu : à partir de ce moment, Pyxel appelle `update()` puis `draw()` à chaque nouvelle image, jusqu'à la fermeture de la fenêtre.

Q1. Lancer le programme, puis modifier successivement les trois nombres de l'instruction `pyxel.text(50, 60, "Bonjour NSI !", 7)`.

1. Que représente chacun de ces nombres ?
2. Où se trouve l'origine du repère ? Dans quel sens augmentent les coordonnées x et y ?

Q2. Chaque caractère occupe 4 pixels de large et 6 pixels de haut. Calculer les coordonnées à donner à `pyxel.text` pour que le message « Bonjour NSI ! » apparaisse dans le coin inférieur droit de la fenêtre, puis vérifier.

Q3. Le programme définit les fonctions `update` et `draw`, mais on ne trouve nulle part l'instruction `draw()`. Pourtant le texte s'affiche.

1. Qui appelle la fonction `draw` ?
2. Pourquoi écrit-on `pyxel.run(update, draw)` et non `pyxel.run(update(), draw())` ?

Q4. Remplacer l'instruction `pyxel.text(...)` par `pyxel.text(50, 60, str(pyxel.frame_count), 7)`. La variable `pyxel.frame_count` contient le nombre d'images affichées depuis le lancement du programme.

À l'aide d'un chronomètre, estimer le nombre de fois où la fonction `draw` est appelée en une seconde.

Q5. Sans exécuter le programme, prévoir ce qui se passe si l'on supprime l'instruction `pyxel.cls(0)`. Vérifier, puis expliquer le rôle de `pyxel.cls`.

§2. Le joueur

Pour que le programme reste lisible et facile à modifier, les valeurs numériques ne sont pas écrites directement dans les instructions. Elles sont rangées au début du programme dans des variables au nom explicite. Ajouter, sous `HAUTEUR` :

```python
TAILLE_JOUEUR = 8
COULEUR_JOUEUR = 11


x_joueur = 76
y_joueur = 100
```

L'instruction `pyxel.rect(x, y, l, h, col)` dessine un rectangle plein de largeur `l`, de hauteur `h` et de couleur `col`, dont le coin supérieur gauche a pour coordonnées `(x, y)`.

Q6. Écrire une fonction `dessiner_joueur()`, sans paramètre, qui dessine le joueur : un carré de côté `TAILLE_JOUEUR`, de couleur `COULEUR_JOUEUR`, dont le coin supérieur gauche a pour coordonnées `(x_joueur, y_joueur)`. Remplacer ensuite, dans `draw()`, l'affichage du texte par un appel à cette fonction.

Q7. La valeur 76 place le joueur au milieu de l'écran, dans le sens horizontal.

1. Pourquoi n'a-t-on pas choisi 80, qui est la moitié de 160 ?
2. Remplacer 76 par une expression qui utilise `LARGEUR` et `TAILLE_JOUEUR`.
3. Pourquoi utiliser l'opérateur `//` plutôt que `/` dans cette expression ?

Q8. Remplacer `LARGEUR = 160` par `LARGEUR = 200` et relancer le programme.

1. Qu'est-ce qui a changé sans qu'il soit nécessaire de modifier une autre ligne ?
2. Que faudrait-il faire si le nombre 160 avait été écrit partout dans le programme ?

Constantes

Une variable dont la valeur ne change pas pendant l'exécution du programme est appelée une **constante**. Par convention, en Python, son nom s'écrit en majuscules (`LARGEUR`, `TAILLE_JOUEUR`). Python n'interdit pas de la modifier : c'est une convention entre programmeurs. Regrouper les constantes au début du programme permet de changer une valeur en un seul endroit.

Q9. Dans `draw()`, remplacer `dessiner_joueur()` par `dessiner_joueur`, sans les parenthèses. Que se passe-t-il ? Faire le lien avec l'instruction `pyxel.run(update, draw)` et formuler la différence entre **désigner** une fonction et **appeler** une fonction.

§3. Déplacer le joueur : variables locales et globales

L'instruction `pyxel.btn(touche)` renvoie `True` tant que la touche est enfoncée, `False` sinon. Les flèches du clavier sont désignées par `pyxel.KEY_LEFT`, `pyxel.KEY_RIGHT`, `pyxel.KEY_UP` et `pyxel.KEY_DOWN`.

Ajouter la constante `VITESSE = 2` ainsi que la fonction suivante, qui contient volontairement une erreur, puis appeler cette fonction dans `update()` (à la place de `pass`).

```python
def deplacer_joueur():
    if pyxel.btn(pyxel.KEY_LEFT):
        x_joueur = x_joueur - VITESSE
```

Q10. Lancer le programme, attendre quelques secondes, puis appuyer sur ←.

1. Quel message d'erreur obtient-on ? Quelle ligne est en cause ?
2. Pourquoi l'erreur n'apparaît-elle pas dès le lancement du programme ?

Dans une fonction, une variable qui reçoit une valeur par une affectation est considérée par Python comme une variable **locale** : elle n'existe que pendant l'exécution de la fonction et n'a aucun lien avec la variable de même nom définie au début du programme. Ici, pour calculer `x_joueur - VITESSE`, Python cherche la variable locale `x_joueur`, qui n'a encore reçu aucune valeur : d'où l'erreur.

Pour modifier une variable définie au début du programme, appelée variable **globale**, il faut l'annoncer au début de la fonction avec le mot-clé `global`.

Variables locales et variables globales

- Une variable définie en dehors de toute fonction est une variable **globale**. Toute fonction peut lire sa valeur.
- Une variable qui reçoit une valeur à l'intérieur d'une fonction est une variable **locale** à cette fonction.
- Pour qu'une fonction puisse modifier une variable globale, il faut écrire `global nom_de_la_variable` au début du corps de la fonction.

Q11. Ajouter la ligne `global x_joueur, y_joueur` au début de `deplacer_joueur()`, puis compléter cette fonction pour que le joueur se déplace dans les quatre directions.

Q12. La fonction `dessiner_joueur()` utilise `x_joueur` et `y_joueur`, et la fonction `deplacer_joueur()` utilise `VITESSE`, sans aucune déclaration `global`. Pourquoi cela ne provoque-t-il pas d'erreur ?

Q13.
1. Maintenir les touches → et ↑ enfoncées en même temps. Qu'observe-t-on ?
2. Un élève propose de remplacer les quatre `if` par une structure `if … elif … elif … elif ….`. Sans exécuter le programme, prévoir ce qui se passerait avec les mêmes touches, puis vérifier.

Q14. Sans exécuter le programme, calculer le nombre de pixels parcourus en une seconde lorsque la touche → reste enfoncée, puis le temps nécessaire pour traverser l'écran de gauche à droite. Vérifier avec un chronomètre.

Q15. Pourquoi l'appel à `deplacer_joueur()` est-il placé dans `update()` et non dans `draw()` ?

§4. Rester dans l'écran

Déplacer le joueur jusqu'au bord de l'écran : il peut en sortir. Il faut donc n'autoriser un déplacement que si le joueur reste entièrement visible.

Q16. Compléter les conditions suivantes, en n'utilisant que des variables (aucun nombre sauf 0) :

```python
if pyxel.btn(pyxel.KEY_LEFT) and ................:
if pyxel.btn(pyxel.KEY_RIGHT) and ................:
if pyxel.btn(pyxel.KEY_UP) and ................:
if pyxel.btn(pyxel.KEY_DOWN) and ................:
```

Expliquer pourquoi la condition sur le bord droit doit tenir compte de la taille du joueur, alors que celle sur le bord gauche n'en tient pas compte.

Q17. Remplacer `VITESSE = 2` par `VITESSE = 3`.

1. Sans exécuter le programme, écrire les valeurs successives de `x_joueur` lorsque le joueur part de 76 et que la touche ⟵ reste enfoncée. Le joueur reste-t-il dans l'écran ?
2. Modifier les quatre conditions pour que le joueur ne sorte jamais de l'écran, quelle que soit la valeur de `VITESSE`. Revenir ensuite à `VITESSE = 2`.

§5. La cible : une fonction avec des paramètres

L'instruction `pyxel.circ(x, y, r, col)` dessine un disque de centre `(x, y)`, de rayon `r` et de couleur `col`. Ajouter les constantes et les variables suivantes :

```python
RAYON_CIBLE = 4
COULEUR_CIBLE = 10

x_cible = 80
y_cible = 30
```

puis la fonction ci-dessous, et l'instruction `dessiner_cible(x_cible, y_cible)` dans `draw()`.

```python
def dessiner_cible(x, y):
    pyxel.circ(x, y, RAYON_CIBLE, COULEUR_CIBLE)
```

Q18. Dans la définition `def dessiner_cible(x, y):` et dans l'appel `dessiner_cible(x_cible, y_cible)` :

1. identifier les paramètres et les arguments ;
2. indiquer la valeur de `x` et celle de `y` pendant l'exécution de cet appel.

Q19. Sans exécuter le programme, prévoir le résultat de chacun des appels suivants, placés dans `draw()`, puis vérifier.

1. `dessiner_cible(20, 20)` suivi de `dessiner_cible(140, 90)` ;
2. `dessiner_cible(y_cible, x_cible)` ;
3. `dessiner_cible(20)`.

Q20. Ajouter à la fonction `dessiner_cible` un troisième paramètre `couleur`, puis modifier l'appel dans `draw()` pour que la cible soit toujours dessinée avec la couleur `COULEUR_CIBLE`. Vérifier en affichant une deuxième cible d'une autre couleur, puis la supprimer.

§6. Le sol : une boucle for

Ajouter les constantes et la fonction suivantes, puis appeler `dessiner_sol()` dans `draw()`.

```python
Y_SOL = HAUTEUR - 4
EPAISSEUR_SOL = 2
LARGEUR_DALLE = 6
PAS_DALLES = 8
COULEUR_SOL = 3

def dessiner_sol():
    for x in range(0, LARGEUR, PAS_DALLES):
```

```
        pyxel.rect(x, Y_SOL, LARGEUR_DALLE, EPAISSEUR_SOL, COULEUR_SOL)
```

Q21. Sans exécuter le programme :

1. combien de dalles sont dessinées ?
2. quelles sont les abscisses de la première et de la dernière dalle ?
3. quelle est la largeur de l'espace entre deux dalles voisines ?

Q22. Sans exécuter le programme, prévoir l'aspect du sol dans chacun des cas suivants, puis vérifier :

1. `LARGEUR_DALLE = 8` ;
2. `LARGEUR_DALLE = 6` et `PAS_DALLES = 4`.

Q23. Réécrire la fonction `dessiner_sol()` avec une boucle `while` (voir Doc. 2,4). Laquelle des deux versions est la plus adaptée ? Pourquoi ?

Q24. Dans `draw()`, placer l'appel à `dessiner_sol()` avant `pyxel.cls(0)`. Qu'observe-t-on ? Pourquoi ?

Q25. Le joueur peut descendre dans le sol. Modifier la condition sur la touche ↓ pour qu'il s'arrête au-dessus du sol, et la position de départ `y_joueur` pour qu'il soit posé sur le sol au lancement du jeu.

§7. Distance entre deux points : une fonction qui renvoie une valeur

Pour savoir si le joueur attrape la cible, il faut mesurer la distance qui les sépare. La distance entre les points de coordonnées (x_1, y_1) et (x_2, y_2) vaut :

$$d = \sqrt{(x_2 - x_1)^2 + (y_2 - y_1)^2}$$

La racine carrée s'obtient avec la fonction `sqrt` du module `math`, à importer au début du programme : `from math import sqrt`.

Q26. Écrire une fonction `distance(x1, y1, x2, y2)` qui renvoie la distance entre les points (x_1, y_1) et (x_2, y_2).

Sans exécuter le programme, prévoir la valeur de `distance(0, 0, 3, 4)` et son type. Vérifier en ajoutant temporairement `print(distance(0, 0, 3, 4))` juste après la définition de la fonction.

Q27. Un élève a écrit une fonction qui affiche la distance au lieu de la renvoyer :

```python
def distance_affichee(x1, y1, x2, y2):
    print(sqrt((x2 - x1) ** 2 + (y2 - y1) ** 2))
```

1. Sans exécuter le programme, prévoir le résultat de `print(distance_affichee(0, 0, 3, 4) + 1)`, puis vérifier.
2. Pourquoi cette fonction ne peut-elle pas servir dans le jeu ?

§8. La cible est-elle attrapée ? Une fonction qui renvoie un booléen

On considère la fonction suivante :

```python
def cible_attrapee():
    d = distance(x_joueur + 4, y_joueur + 4, x_cible, y_cible)
    return d < 8
```

Q28.
1. Pourquoi ajoute-t-on 4 aux coordonnées du joueur ?
2. Justifier la valeur 8, en s'appuyant sur un schéma.
3. Réécrire la fonction en remplaçant les nombres 4 et 8 par des expressions qui utilisent les constantes `TAILLE_JOUEUR` et `RAYON_CIBLE`.

Q29.
1. Quelles valeurs la fonction `cible_attrapee()` peut-elle renvoyer ?

2. Un élève a écrit la fin de la fonction ainsi :

```python
if d < 8:
    return True
else:
    return False
```

Les deux versions sont-elles équivalentes ? Laquelle préférer ?

Q30. Dans `draw()`, afficher le texte « Touché ! » en haut de l'écran lorsque le joueur touche la cible. Tester.

§9. Faire apparaître une nouvelle cible

L'instruction `pyxel.rndi(a, b)` renvoie un entier choisi au hasard entre `a` et `b`, bornes comprises. Un disque de centre `(x, y)` et de rayon `r` occupe les pixels d'abscisse `x - r` à `x + r` ; les pixels de l'écran ont des abscisses comprises entre 0 et `LARGEUR - 1`.

Q31.
1. Déterminer les valeurs minimale et maximale de `x_cible` pour que la cible soit entièrement visible. Faire de même pour `y_cible`, en imposant que la cible reste au-dessus du sol.
2. Écrire une fonction `nouvelle_cible()` qui donne à la cible une position choisie au hasard, entièrement visible.
3. Dans `update()`, appeler `nouvelle_cible()` lorsque la cible est attrapée. Tester.

Q32. Il peut arriver que la nouvelle cible apparaisse directement sur le joueur.

1. Que se passe-t-il dans ce cas ?
2. Pour l'éviter, il faut tirer une nouvelle position tant que la cible touche le joueur. Quelle boucle faut-il utiliser ? Pourquoi ?
3. Renommer la fonction de la question précédente en `tirer_position_cible()`, puis écrire une nouvelle fonction `nouvelle_cible()` qui l'utilise.

§10. Le score

Ajouter la variable `score = 0` au début du programme.

Q33.
1. Pourquoi le nom `score` est-il écrit en minuscules, alors que `LARGEUR` est écrit en majuscules ?
2. Modifier `update()` pour que le score augmente de 1 à chaque cible attrapée. Que faut-il ajouter au début de la fonction ?

Q34. Remplacer l'affichage de « Touché ! » par celui du score, à l'aide de l'instruction `pyxel.text(5, 5, "Score : " + str(score), 7)`.

1. Pourquoi faut-il écrire `str(score)` ?
2. Réécrire cette instruction à l'aide d'une f-string (voir Doc. 2,2).
3. Faut-il ajouter `global score` dans `draw()` ?

§11. Gagner la partie

Ajouter la constante `SCORE_GAGNANT = 10` et modifier `update()` pour que le joueur ne puisse plus bouger une fois la partie gagnée :

```python
def update():
    global score

    if score < SCORE_GAGNANT:
        deplacer_joueur()
```

```
        if cible_attrapee():
            score = score + 1
            nouvelle_cible()
```

Q35. Pour tester rapidement la fin de partie, quelle ligne suffit-il de modifier ?

Q36. Un élève a écrit la fonction `draw()` suivante pour afficher « BRAVO ! » au centre de l'écran en fin de partie :

```python
def draw():
    if score < SCORE_GAGNANT:
        pyxel.cls(0)
        dessiner_sol()
        dessiner_cible(x_cible, y_cible, COULEUR_CIBLE)
        dessiner_joueur()
        pyxel.text(5, 5, f"Score : {score}", 7)
    else:
        pyxel.text(LARGEUR // 2, HAUTEUR // 2, "BRAVO !", 7)
```

1. Sans exécuter le programme, prévoir les deux défauts de l'affichage en fin de partie. Vérifier.
2. Corriger la fonction. Pour centrer le texte, utiliser les constantes `pyxel.FONT_WIDTH` et `pyxel.FONT_HEIGHT`, qui valent la largeur (4) et la hauteur (6) d'un caractère, ainsi que la fonction `len`.

§12. Quitter et rejouer

L'instruction `pyxel.btnp(touche)` renvoie `True` uniquement au moment où la touche vient d'être enfoncée, et non pendant tout le temps où elle reste enfoncée comme `pyxel.btn`. L'instruction `pyxel.quit()` ferme le jeu.

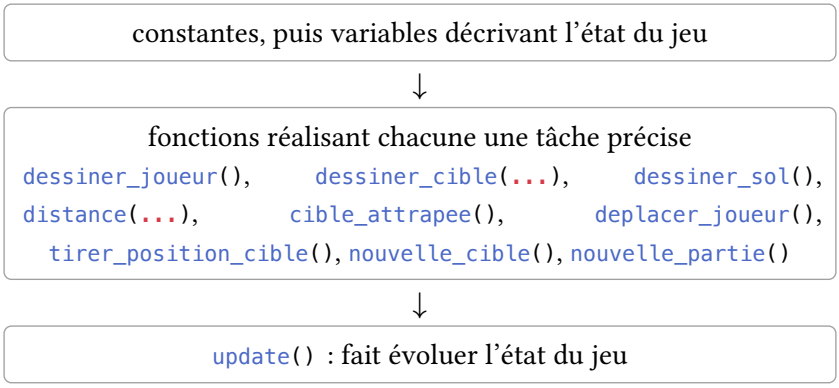
Q37.
1. Pourquoi utilise-t-on `pyxel.btn` pour déplacer le joueur, et non `pyxel.btnp` ?
2. Modifier `update()` pour que la touche Q ferme le jeu à tout moment, y compris après la victoire.

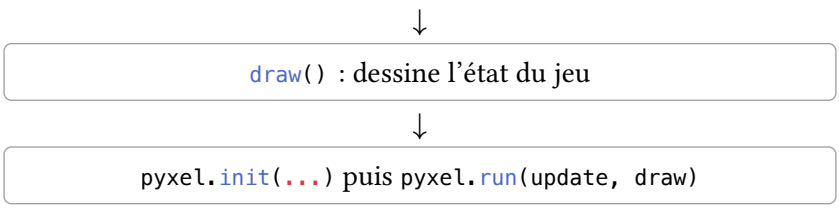
Q38. Une fois la partie gagnée, un appui sur R doit lancer une nouvelle partie : le joueur revient à sa position de départ, le score revient à 0 et une nouvelle cible apparaît.

1. Pour ne pas écrire deux fois la position de départ, créer deux constantes `X_DEPART` et `Y_DEPART`, et les utiliser pour initialiser `x_joueur` et `y_joueur`.
2. Écrire une fonction `nouvelle_partie()`. Quelles variables doivent être déclarées `global` ?
3. Appeler cette fonction dans `update()` au bon endroit.

§13. Le programme obtenu

Le programme complet est organisé ainsi :





Q39. Dans votre programme, trouver un exemple de chacune des notions suivantes et le recopier.

Notion	Exemple dans votre programme
Constante	
Définition d'une fonction	
Appel d'une fonction	
Fonction avec paramètres	
Fonction renvoyant une valeur	
Fonction renvoyant un booléen	
Instruction conditionnelle	
Boucle `for`	
Boucle `while`	
Fonction qui modifie une variable globale	

§14. Défis

Les défis sont classés par difficulté croissante.

Q40. **Défi A : cible multicolore.** La couleur de la cible change à chaque point marqué.

Q41. **Défi B : difficulté croissante.** À partir de 5 points, le joueur se déplace plus vite. `VITESSE` peut-elle rester une constante ?

Q42. **Défi C : temps limité.** La partie dure 30 secondes. Le temps restant est affiché à l'écran ; lorsqu'il atteint 0, la partie est perdue.

Q43. **Défi D : deux cibles.** Une deuxième cible, d'une autre couleur, rapporte 3 points.

§15. Bilan

À l'issue de cette activité, je sais…

- distinguer la définition d'une fonction de son appel, et désigner une fonction sans l'appeler ;
- utiliser des constantes plutôt que des valeurs numériques écrites dans les instructions ;
- expliquer quand une variable est locale ou globale, et utiliser `global` à bon escient ;
- écrire une condition qui tient compte des dimensions d'un objet ;
- distinguer paramètres et arguments ;
- choisir entre une boucle `for` et une boucle `while` ;
- distinguer une fonction qui renvoie une valeur d'une fonction qui l'affiche ;
- écrire une fonction qui renvoie un booléen et l'utiliser comme condition ;
- utiliser quelques instructions de Pyxel (`text`, `rect`, `circ`, `btn`, `btnp`, `rndi`).