Tir sur cible

Utilise Python pour dessiner une cible et marquer des points en la frappant avec des flèches

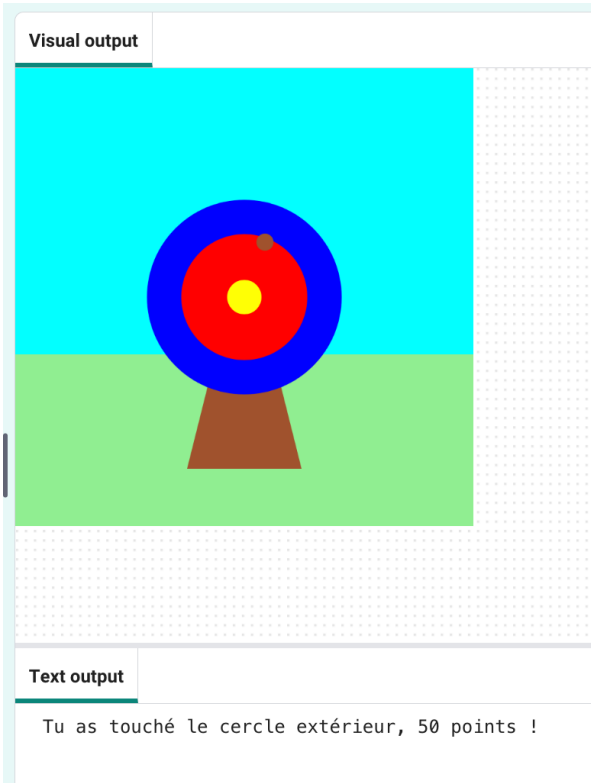
Étape 1 Ce que tu vas faire

Réponds à notre enquête (https://form.raspberrypi.org/f/code-editor-feedback) pour nous aider à améliorer notre Code Editor !

Utilise Python, avec la bibliothèque graphique `p5`, pour dessiner une cible et marquer des points en l'atteignant avec des flèches.

Tu vas devoir :

- Personnaliser ton jeu avec des **couleurs RVB**
- Utiliser les **instructions conditionnelles** (`if`, `elif`, `else`) pour prendre des décisions
- Positionner les formes avec les coordonnées **x, y**



Les plus anciennes traces de **tir à l'arc** proviennent de la grotte de Sibudu au KwaZulu-Natal, en Afrique du Sud. Des restes de pointes de flèches en pierre et en os ont été trouvés, qui datent d'il y a entre 60 000 et 70 000 ans.

Étape 2 Créer un arrière-plan

Ton jeu a besoin d'un arrière-plan coloré.

Ouvrir le projet de démarrage

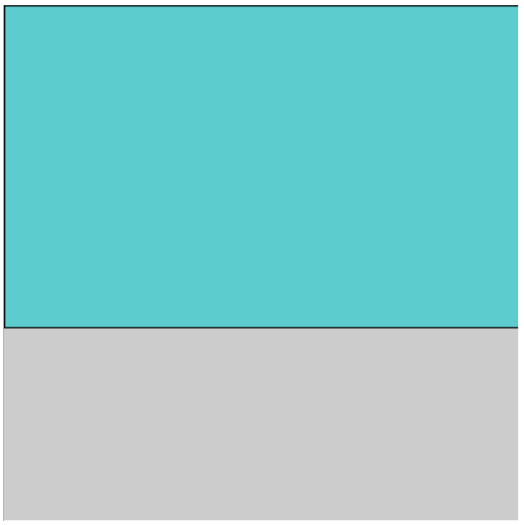
Ouvre le **projet de démarrage Tir sur cible** (https://editor.raspberrypi.org/fr-FR/projects/target-practice-starter). Le Code Editor s'ouvrira dans un autre onglet du navigateur.

Si tu as un compte Raspberry Pi, tu peux cliquer sur le bouton **Enregistrer** pour enregistrer une copie dans tes **Projets**.

Modifier le ciel

Le projet de démarrage contient du code déjà écrit pour toi.

Clique sur **« Run »** pour voir un rectangle rempli de bleu dessiné à partir de x=`0`, y=`0` (le haut de l'écran). Ce rectangle de `400` x `250` pixels représente le ciel.



Astuce : les coordonnées commencent à partir de (x=0, y=0) dans le coin supérieur gauche. Cela peut être différent des autres systèmes de coordonnées que tu as utilisés.

Le ciel a été dessiné avec une bordure noire (trait).

Pour désactiver le trait pour toutes les formes, ajoute `no_stroke()` à la fonction `setup` :

main.py — setup()

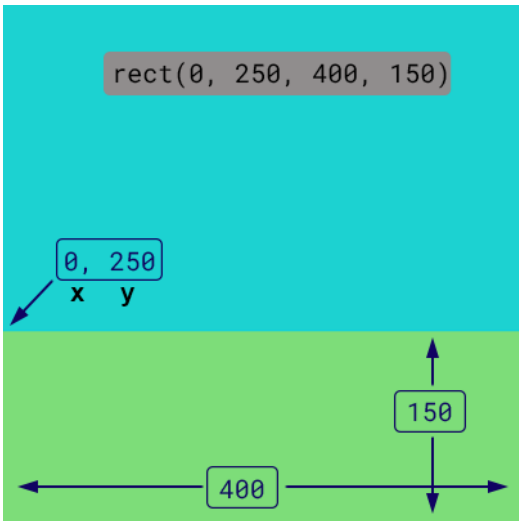
```
def setup():
# Configurer ton jeu ici

    size(400, 400)  # largeur et hauteur de l'écran
    no_stroke()
```

Exécute à nouveau ton projet pour vérifier 👀 que la bordure (trait) a disparue.

Astuce : 💡 tu devras appuyer sur **Stop** pour arrêter ton programme, cela fera réapparaître le bouton **Run**.

Dessiner l'herbe

Ajoute du code pour dessiner un rectangle vert en bas de l'écran.



main.py — draw()

```
def draw():
# Choses à faire dans chaque frame

    fill('cyan')  # Définis la couleur de remplissage du ciel sur cyan
    rect(0, 0, 400, 250)  # Dessine un rectangle pour le ciel avec ces valeurs pour x, y, largeur, hauteur
    fill('lightgreen')  # Définis la couleur de remplissage de l'herbe sur vert clair
    rect(0, 250, 400, 150)  # Dessine un rectangle pour l'herbe avec ces valeurs pour x, y, largeur,
hauteur
```

Astuce : 💡 nous avons ajouté des commentaires à notre code, comme `# Définir la couleur de remplissage du ciel en cyan`, pour t'indiquer ce qu'il fait. Tu n'as pas besoin d'ajouter des commentaires à ton code, mais ils peuvent être utiles pour te rappeler ce que font les lignes de code.

Test : exécute à nouveau ton projet pour voir l'arrière-plan terminé.

Sauvegarde ton projet

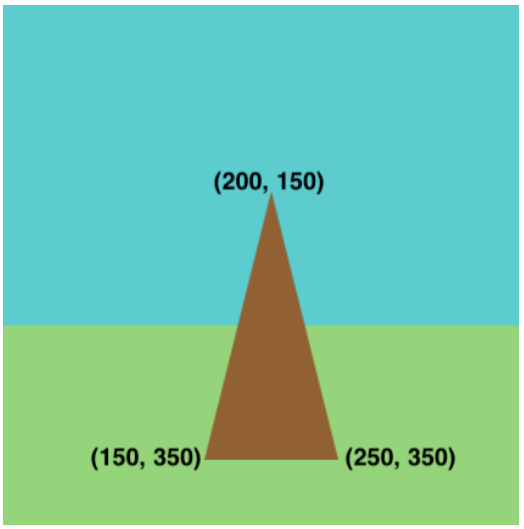
Étape 3 Dessiner ta cible

Ton jeu a besoin d'une cible pour tirer des flèches.

Dessiner un support triangulaire

Définis la couleur de remplissage sur `sienna` (marron).

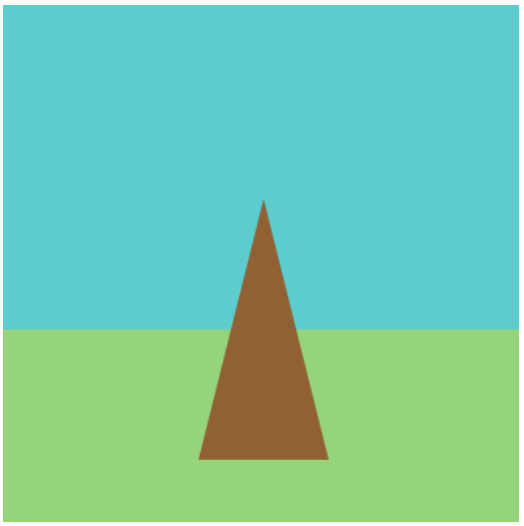
Dessine un triangle en utilisant les coordonnées x et y de chacun des angles.



main.py - draw()

```
fill('lightgreen')  # Définis la couleur de remplissage de l'herbe sur vert clair
rect(0, 250, 400, 150)  # Dessine un rectangle pour l'herbe avec ces valeurs pour x, y, largeur, hauteur
fill('sienna')  # Couleur marron
triangle(150, 350, 200, 150, 250, 350)  # Dessine un triangle pour le support de la cible
```

Test : 🔄 exécute ton code pour voir le support de ta cible :

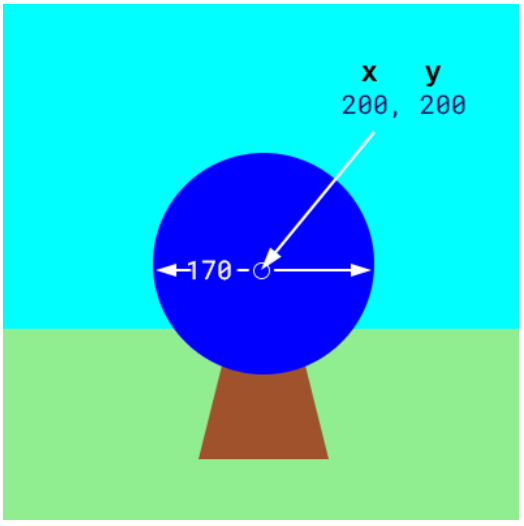


Dessiner les cibles

La plus grande partie de la cible est un **cercle** bleu.

Définis la couleur de remplissage sur `blue`.

Dessine un cercle avec des coordonnées x et y pour son centre et une largeur.



main.py - draw()

```
fill('sienna')  # Couleur marron
triangle(150, 350, 200, 150, 250, 350)  # Dessine un triangle pour le support de la cible
fill('blue')  # Définis la couleur de remplissage sur blue
circle(200, 200, 170)  # Dessine le cercle extérieur
```

Test : exécute ton code pour voir le premier grand cercle bleu.

Le cercle bleu a été dessiné après le support donc il est devant.

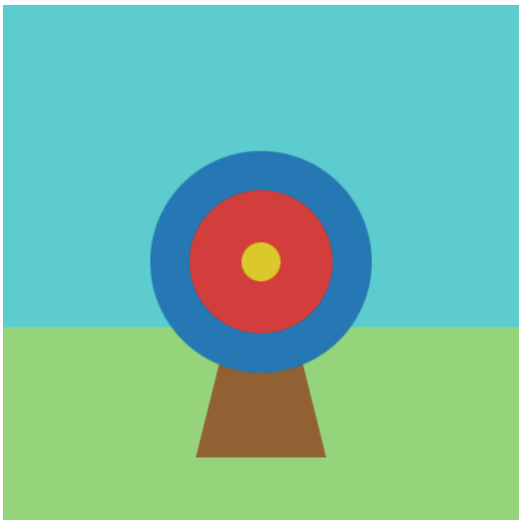
La cible est constituée de cercles de tailles différentes ayant les mêmes coordonnées centrales (200, 200).

Ajoute des cercles de couleur pour les parties intérieure et centrale de la cible.

main.py - draw()

```
fill('sienna')  # Couleur marron
triangle(150, 350, 200, 150, 250, 350)  # Dessine un triangle pour le support de la cible
fill('blue')  # Définis la couleur de remplissage sur blue
circle(200, 200, 170)  # Dessine le cercle extérieur
fill('red')  # Définis la couleur du remplissage du cercle sur rouge
circle(200, 200, 110)  # Dessine le cercle intérieur en utilisant x, y, largeur
fill('yellow')  # Définis la couleur du remplissage du cercle sur jaune
circle(200, 200, 30)  # Dessine le cercle du milieu en utilisant x, y, largeur
```

Test : exécute ton projet pour voir la cible avec trois cercles colorés.



Choisir : 💭 modifie l'une des couleurs en utilisant un nom de couleur différent. Tu peux trouver une liste de tous les noms de couleurs disponibles sur **W3 Schools** (https://www.w3schools.com/colors/colors_names.asp).

Exemple de code utilisant différentes couleurs

main.py - draw()

```
def draw():
# Choses à faire dans chaque frame

    fill('BlueViolet')
    rect(0, 0, 400, 250)  # Ciel
    fill('DeepSkyBlue')
    rect(0, 250, 400, 150)  # Sol
    fill('FireBrick')
    triangle(150, 350, 200, 150, 250, 350)  # Support
    fill('LemonChiffon')
    circle(200, 200, 170)  # Cercle extérieur
    fill('DeepPink')
    circle(200, 200, 110)  # Cercle intérieur
    fill('BlueViolet')
    circle(200, 200, 30)  # Cercle du milieu
```

Sauvegarde ton projet

Étape 4 Tirer la flèche

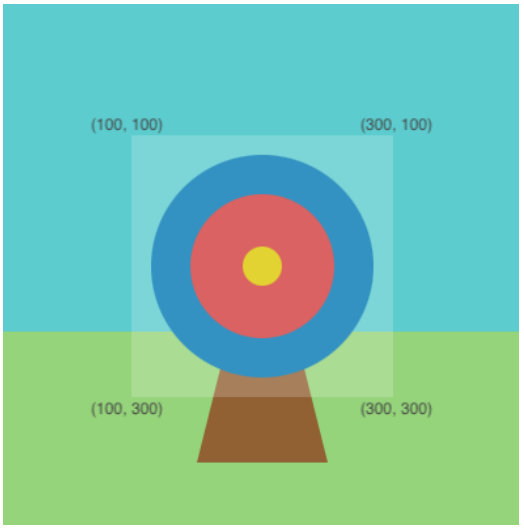
Lorsque tu cliques ou appuies, une flèche est tirée à la position d'un cercle cible en mouvement.

Dessiner un cercle cible à chaque frame

> Les ordinateurs créent l'effet de mouvement en montrant de nombreuses images l'une après l'autre. Chaque image est appelée une **frame**.

Définis ta fonction `tire_fleche()` sous le commentaire **# La fonction tire_fleche vient ici**.

Ajoute du code pour dessiner au hasard un cercle marron dans une zone cible :



main.py — shoot_arrow()

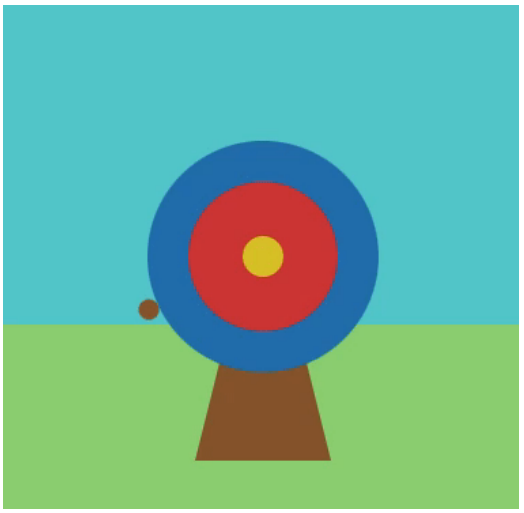
```
# La fonction tire_fleche vient ici
def tire_fleche():
    fleche_x = randint(100, 300)
    fleche_y = randint(100, 300)
    touche_couleur = get(fleche_x, fleche_y) #Enregistrer la couleur avant de dessiner la flèche
    ellipse(fleche_x, fleche_y , 15, 15)
```

Va dans la fonction `draw` et appelle ta nouvelle fonction `tire_fleche`.

main.py — draw()

```
fill('yellow')  # Définis la couleur du remplissage du cercle sur jaune
circle(200, 200, 30)  # Dessine le cercle du milieu en utilisant x, y, largeur
tire_fleche()
```

Test : 🔄 exécute ton code et vois la flèche apparaître dans une position aléatoire à chaque frame.



L'arrière-plan et la cible seront dessinés sur l'ancienne flèche. Cela signifie que tu ne vois qu'une seule flèche à la fois.

Obtenir la couleur touchée par la flèche

La fonction `get()` renvoie la couleur d'un pixel.

> Un **pixel**, abréviation d'élément d'image, est un point coloré unique dans une image. Les images sont composées de beaucoup de pixels colorés.

Ajoute une **variable globale** appelée `touche_couleur` qui peut être utilisée dans tout ton code.

Ajoute du code pour `obtenir` la couleur du pixel au centre de la flèche et stocke-le dans la variable `touche_couleur` . Afin de comparer les couleurs, nous devons utiliser le code hexadécimal. Cela peut être fait avec la chaîne `.hex` .

main.py — shoot_arrow()

```
# La fonction tire_fleche vient ici
def tire_fleche():
    global couleur_touche #Peut être utilisé dans d'autres fonctions
    fleche_x = randint(100, 300)
    fleche_y = randint(100, 300)
    couleur_touche = get(fleche_x, fleche_y) #Enregistrer la couleur avant de dessiner la flèche
    ellipse(fleche_x, fleche_y, 15, 15)
```

Astuce : 💡 le code pour `obtenir` la couleur doit être **avant** le code pour dessiner le `cercle` sinon tu enregistreras toujours la couleur bois de la flèche !

Imprimer la couleur lorsque la souris est pressée

La bibliothèque `p5` « écoute » certains événements, l'un d'eux est la pression du bouton de la souris. Lorsqu'elle détecte que le bouton a été pressé, elle exécute le code qui lui a été donné dans la fonction `mouse_pressed`.

Définis ta fonction `mouse_pressed()` sous le commentaire **# La fonction mouse_pressed vient ici**.

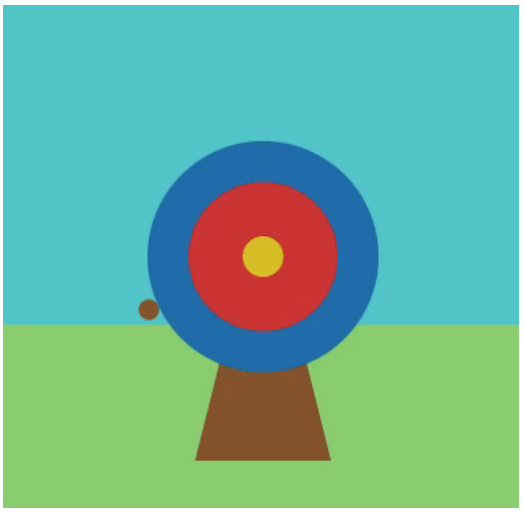
Ajoute du code pour imprimer l'emoji cible 🎯 lorsque tu cliques sur la souris.

main.py - mouse_pressed()

```
5  # La fonction mouse_pressed vient ici
6  def mouse_pressed():
7      print('🎯')
```

Test : 🔄 exécute ton projet.

Le projet imprime 🎯 à chaque fois que la flèche est redessinée.



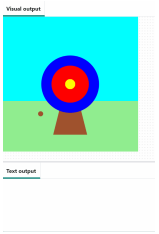
Débogage : si tu vois un message indiquant que `touche_couleur` n'est pas défini, reviens à `tire_fleche()` et vérifie que tu as bien la ligne `global touche_couleur`.

Débogage : vérifie très attentivement la ligne `print` pour les virgules et les parenthèses.

Sauvegarde ton projet

Étape 5 Marquer des points

Ton jeu ajoutera des scores en fonction de l'endroit où la flèche frappe.



Nous utilisons des **conditions** tout le temps pour prendre des décisions. On pourrait dire « si le crayon est émoussé, alors taille-le ». De même, les conditions "if" nous permettent d'écrire du code qui fait quelque chose de différent selon qu'une condition est vraie ou fausse.

Afficher les scores

Supprime ❌ la ligne de code `print('🎯')`.

main.py

```
# La fonction mouse_pressed vient ici
def mouse_pressed():
```

Affiche un message **if** la `touche_couleur` est égale à la couleur du cercle `extérieur` (bleu) 🎯.

Remarque 👀 que le code utilise deux signes égal `==` pour signifier **égal à**.

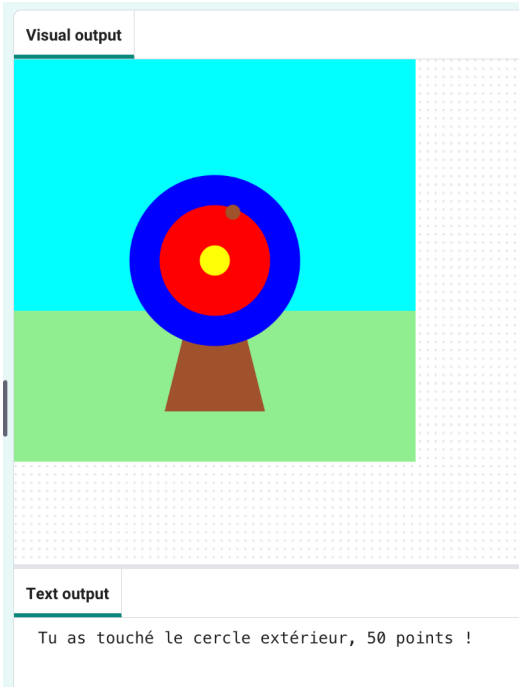
main.py - mouse_pressed()

```
# La fonction mouse_pressed vient ici
def mouse_pressed():
    if touche_couleur == Color('blue').hex: #Comme les fonctions, les instructions "if" sont indentées
        print('tu as touché le cercle extérieur, 50 points !')
```

Astuce : 💡 si tu as modifié la couleur de ton cercle extérieur alors tu devras remplacer `'blue'` par le nom de couleur que tu as choisi.

Test : 🔄 exécute ton projet. Essaie de tirer la flèche sur le cercle extérieur bleu pour voir le message.

Astuce : `frame_rate()`, dans `setup()`, contrôle la vitesse à laquelle ton jeu dessine. S'il va trop vite, règle-le sur un nombre inférieur.



Débogage : 🐞 vérifie que tu as utilisé l'orthographe américaine de 'Color' (sans 'u') et que 'Color' est en majuscule.

Debogage : 🐞 assure-toi que ton code correspond exactement et que tu as indenté le code à l'intérieur de ta déclaration `if`.

Débogage : 🐞 assure-toi d'avoir entré le nom de couleur correct que tu as utilisé pour **ton** cercle extérieur.

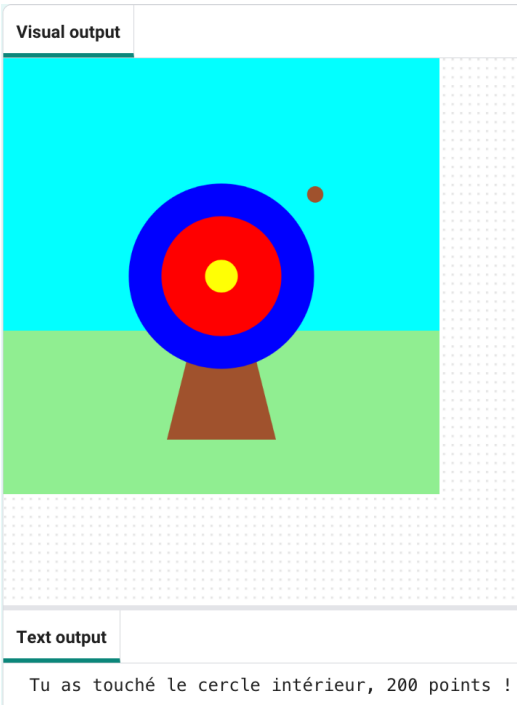
`elif` (else - if) peut être utilisé pour ajouter des conditions supplémentaires à ta déclaration `if`. Elles seront lues de haut en bas. Dès qu'une condition **True** est trouvée, elle sera traitée. Toutes les conditions restantes seront ignorées.

Marque des points si la flèche atterrit sur les cercles `interieur` ou `centre` 🎯 :

main.py - mouse_pressed()

```
6  def mouse_pressed():
7      if touche_couleur == Color('blue').hex:
8          print('Tu as touché le cercle extérieur, 50 points !')
9      elif touche_couleur == Color('red').hex:
10         print('Tu as touché le cercle intérieur, 200 points !')
11     elif touche_couleur == Color('yellow').hex:
12         print('Tu as touché le centre, 500 points !')
```

Test : 🔄 exécute ton projet. Essaie de tirer la flèche sur les cercles intérieurs et intermédiaires pour voir leurs messages.



Débogage : 🐞 vérifie que ton indentation correspond à l'exemple.

Débogage : 🐞 si tu vois un message indiquant que `touche_couleur` n'est pas « défini », reviens à `draw()` et vérifie que la ligne déclaration `touche_couleur` comme une variable globale.

Débogage : 🐞 assure-toi d'avoir entré le nom de couleur correct pour **tes** cercles.

Débogage : 🐞 assure-toi d'avoir utilisé la chaîne `.hex` pour **tes** couleurs de cercle.

Manquer la cible

Il te reste une décision à prendre : que se passe-t-il si la flèche n'atterrit sur aucun des cercles cibles ? ❌

Pour faire cette dernière vérification, tu utilises `else`.

Ajoute du code à `print` un message `else` si aucune des déclarations `if` et `elif` n'ont été remplies.

main.py

```
6  def mouse_pressed():
7      if touche_couleur == Color('blue').hex:
8          print('Tu as touché le cercle extérieur, 50 points !')
9      elif touche_couleur == Color('red').hex:
10         print('Tu as touché le cercle intérieur, 200 points !')
11     elif touche_couleur == Color('yellow').hex:
12         print('Tu as touché le centre, 500 points !')
13     else:
14        print('Tu as loupé la cible ! Aucun point !')
```

Test : 🔄 exécute ton projet. Tire la flèche dans l'herbe ou dans le ciel pour voir le message manqué.

Choisir : 💭 modifie le nombre de points marqués pour les différentes couleurs.

Sauvegarde ton projet

Améliorer ton projet

Personnalise et ajoute plus à ton projet. Peut-être pourrais-tu changer le niveau de difficulté ou ajouter plus de cercles à ta cible.

Tu pourrais :

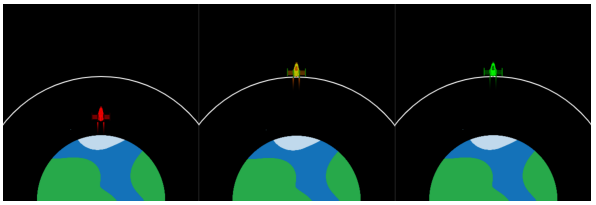
- Ajouter un `quatrième` et `cinquième` cercle, dans de nouvelles couleurs, qui marquent différents nombres de points en fonction de leur position 🟠🟣
- Mettre des emojis dans tes messages imprimés (**voici une liste d'emojis** (https://unicode.org/emoji/charts/full-emoji-list.html) que tu peux copier)
- Rendre le jeu plus facile ou plus difficile en modifiant la valeur `frame_rate(2)` 👈
- Utiliser `input()` pour demander à l'utilisateur à quel niveau de difficulté il veut jouer 🥷

Le projet terminé

Tu peux voir le **projet terminé ici** (https://editor.raspberrypi.org/fr-FR/projects/target-practice-solution).

Et ensuite ?

Si tu suis le parcours **Introduction à Python** (https://projects.raspberrypi.org/fr-FR/raspberrypi/python-intro) , tu peux passer au projet **Lancement de fusée** (https://projects.raspberrypi.org/fr-FR/projects/rocket-launch). Dans ce projet, tu réaliseras une animation interactive d'une fusée lançant un satellite en orbite.



Si tu veux t'amuser davantage en explorant Python, tu peux essayer n'importe lequel de **ces projets** (https://projects.raspberrypi.org/fr-FR/projects?software%5B%5D=python).

Ce projet a été traduit par des bénévoles:

Michel Arnols
Jonathan Vannieuwkerke

Grâce aux bénévoles, nous pouvons donner aux gens du monde entier la chance d'apprendre dans leur propre langue. Vous pouvez nous aider à atteindre plus de personnes en vous portant volontaire pour la traduction - plus d'informations sur **rpf.io/translate** (https://rpf.io/translate).

Publié par **Raspberry Pi Foundation** (https://www.raspberrypi.org) Sous un **Creative Commons license** (https://creativecommons.org/licenses/by-sa/4.0/).
Voir le projet et la licence sur GitHub (https://github.com/RaspberryPiLearning/target-practice)