Découvrir Python avec ARIEL Cube

De 9 à 99 ans — Version découverte 1.0.0 — Association ARIEL-IA

Ce carnet accompagne un projet gratuit pour jouer avec un cube, comprendre son fonctionnement et découvrir le code Python. Aucun savoir en programmation n'est nécessaire au départ. Pour un enfant, un adulte disponible peut aider à préparer l'ordinateur et partager les premières découvertes.

Deux portes ouvertes

Pour jouer tout de suite, la version Windows portable ouvre ARIEL Cube sans installer Python. Elle propose un cube 3D, la saisie d'un vrai cube et une résolution animée par couches.

Pour apprendre et modifier le code, le parcours contient quatorze fichiers numérotés. Il commence par un carré et progresse vers les faces, les rotations, la 3D et l'accompagnement du vrai cube. Les trois dernières étapes utilisent des outils communs fournis dans le même dossier.

Apprendre à son rythme

Chaque étape présente une mission, une notion, une modification à essayer et un résultat à vérifier. Tu peux suivre les versions prêtes ou construire une copie personnelle avec quelqu'un. Les outils de géométrie et de résolution peuvent rester des composants préparés jusqu'au jour où tu voudras les explorer.

Le rituel est simple : deviner, changer une chose, enregistrer, relancer et observer. Une erreur donne une information. Tu peux toujours revenir au fichier d'origine.

Pour les parents et les personnes qui accompagnent

Il n'est pas nécessaire d'avoir toutes les réponses. Demande ce qui pourrait se passer avant un essai, laisse le clavier à la personne qui apprend et garde les séances à une durée qui lui convient. Le plaisir de chercher ensemble compte davantage que le nombre d'étapes terminées.

Les sources et ce carnet sont partagés sous licence MIT. Les composants tiers conservent leurs crédits et leurs licences. Les calculs et les saisies de l'application restent sur l'ordinateur ; aucun compte n'est demandé.

Jouer sans installer Python

Cette page concerne le paquet Windows compilé. Pour modifier les programmes, passe à la page suivante et télécharge également le parcours pédagogique.

Ouvrir le paquet portable

- Télécharge ARIEL_Cube_Windows_1.0.0.zip et choisis Extraire tout dans l'explorateur Windows.
- Ouvre le dossier ARIELCube extrait et double-clique ARIELCube.exe. Garde l'exe et le dossier _internal ensemble.
- À l'accueil, choisis Jouer avec le cube, Résoudre mon vrai cube ou Découvrir le code. Le bouton Accueil permet de changer d'atelier.

Faire les premiers gestes

Dans le jeu, les boutons tournent les faces. Maintiens le bouton gauche de la souris sur le cube et glisse pour changer de vue ; la molette règle le zoom. Mélanger prépare un défi, Annuler défait ton dernier coup et Remettre à zéro retrouve le cube résolu.

Dans l'atelier du vrai cube, recopie les six faces, vérifie-les et demande le guide. Les repères de saisie et les étapes de résolution sont détaillés plus loin dans ce carnet.

Conserver les couleurs

L'application publique enregistre les saisies dans le profil utilisateur. Sous Windows : %LOCALAPPDATA%\ARIEL-IA\ARIELCube. Utilise Enregistrer la saisie puis Charger ma saisie. Avant une pause dans le guide, reviens à la saisie, compare l'état au vrai cube et enregistre. Le jeu libre et l'historique du guide ne sont pas sauvegardés.

Préparer son atelier Python

Pour programmer, télécharge et extrais le parcours complet. L'application Windows seule permet de jouer ; les fichiers sources permettent de comprendre et de modifier le fonctionnement.

Installer les outils une première fois

L'environnement de référence utilise Python 3.12 et Pygame 2.6.1. Pour ce paquet, utilise Python 3.10 à 3.13. Télécharge Python uniquement depuis https://www.python.org/downloads/windows/ et conserve IDLE, l'éditeur fourni avec Python.

Dans le dossier extrait, ouvre parcours, puis etapes. Dans la barre d'adresse de l'explorateur Windows, tape cmd et appuie sur Entrée : un terminal s'ouvre dans ce dossier. Exécute ces commandes, une ligne à la fois.

```
py -m pip install pygame==2.6.1
py rubiks_etape_01.py
```

Si py n'est pas reconnu mais que python fonctionne, remplace py par python. Si aucun des deux ne fonctionne, vérifie l'installation de Python avec la personne qui t'accompagne. La première installation de Pygame utilise Internet ; les exercices fonctionnent ensuite localement.

Ouvrir et modifier un fichier

Lance IDLE, choisis File puis Open et ouvre rubiks_etape_01.py. Fais une copie avec Save As, par exemple mon_cube_01.py. Modifie une seule valeur, enregistre, puis utilise Run et Run Module, ou F5. Ferme l'ancienne fenêtre avant de relancer.

Tu peux aussi garder ton éditeur de code habituel et lancer ta copie depuis le terminal avec py mon_cube_01.py. Le nom du fichier doit correspondre exactement à celui que tu as enregistré.

Retrouver ses repères

Les fichiers 01 à 11 fonctionnent chacun seuls. Les étapes 12 à 14 partagent atelier_cube_reel.py, resolution_cube.py, rubiks_etape_11.py et le dossier rubik : garde-les ensemble. Les relevés du dossier pour_aller_plus_loin détaillent les changements pour une construction accompagnée.

Les sources sont également utilisables avec Python et Pygame sous Linux ou macOS. Les commandes peuvent commencer par python3. Ce téléchargement ne contient pas d'exécutable macOS ou Linux.

Pour te repérer, le numéro de l'étape dans le carnet est toujours le numéro du fichier. La carte suivante donne l'ordre complet. Tu peux faire une pause et reprendre au même fichier.

La carte de ton aventure

Les fichiers sont dans etapes. Le numéro du carnet correspond au numéro du fichier. Garde tout le dossier : les trois derniers fichiers utilisent des outils communs. Pour changer d'étape, remplace le numéro dans la commande.

```
py rubiks_etape_02.py
```

Étape et fichier	Ce que tu découvres
01 rubiks_etape_01.py	Une fenêtre et un carré
02 rubiks_etape_02.py	Une face de neuf cases
03 rubiks_etape_03.py	Les six faces dépliées
04 rubiks_etape_04.py	La mémoire des couleurs
05 rubiks_etape_05.py	Le premier mouvement réel
06 rubiks_etape_06.py	Les commandes et le sens inverse
07 rubiks_etape_07.py	Mélanger et compter
08 rubiks_etape_08.py	Personnaliser et annuler
09 rubiks_etape_09.py	Observer le cube en 3D
10 rubiks_etape_10.py	Animer les rotations en 3D
11 rubiks_etape_11.py	Jouer avec mon cube en 3D
12 rubiks_etape_12.py	Recopier mon vrai cube
13 rubiks_etape_13.py	Vérifier un cube réalisable
14 rubiks_etape_14.py	Résoudre mon vrai cube par couches

Les lettres du cube

Les noms internationaux viennent de l'anglais. Ils restent identiques au clavier et sur les boutons.

U signifie Up ou haut et commence en blanc. D signifie Down ou bas et commence en jaune. F signifie Front ou devant et commence en vert. B signifie Back ou derrière et commence en bleu. L signifie Left ou gauche et commence en orange. R signifie Right ou droite et commence en rouge.

À plat, U est au-dessus de F, D en dessous ; au milieu, L F R B. En 3D, on tourne la vue autour du cube. Les noms des faces restent fixes : F désigne toujours la face dont le centre est vert, même si tu la regardes depuis derrière.

Un tour horaire se lit en regardant directement la face choisie depuis l'extérieur du cube. Les boutons avec une apostrophe font le mouvement inverse.

Étape 01 La fenêtre et le carré

Fichier repère : rubiks_etape_01.py

Mission : faire apparaître un carré, puis lui donner la couleur et la taille de ton choix. Commence par observer avant de modifier les valeurs.

Repère ces réglages en haut du fichier

```
COULEUR_CARRE = (255, 140, 0)
TAILLE_CARRE = 120
POSITION_X = 240
POSITION_Y = 160
```

Une variable est un nom qui garde une valeur. TAILLE_CARRE garde le nombre 120. Python distingue les majuscules et les minuscules : recopie les noms exactement.

À toi de coder

- Remplace la couleur par (0, 200, 255). Chaque nombre dose le rouge, le vert ou le bleu, de 0 à 255.
- Enregistre et relance. Puis passe la taille à 60 et compare.
- Augmente POSITION_X de 40. Devine d'abord de quel côté le carré partira.

Ce que tu viens de comprendre

Le point de départ de l'écran est en haut à gauche. X augmente vers la droite. Y augmente vers le bas. La position désigne le coin supérieur gauche du carré, pas son centre.

La boucle while en_marche recommence tant que la fenêtre est ouverte. Elle écoute les événements, dessine l'image et l'affiche. Le clic sur la croix met en_marche à False : la boucle s'arrête.

Mon défi libre

Fais un carré violet de 90 pixels et place-le près du coin supérieur gauche. Quelle valeur de X et de Y choisis-tu ?

Je vérifie mon résultat

Une fenêtre s'ouvre. Mon carré a changé. La croix ferme le programme. Si le carré disparaît, je vérifie qu'il n'est pas en dehors de la fenêtre de 600 × 440 pixels.

Réponse possible au défi : couleur (150, 60, 210), taille 90, X = 20 et Y = 20. Il existe beaucoup d'autres bonnes couleurs !

Étape 02 Les neuf cases

Fichier repère : rubiks_etape_02.py

Mission : dessiner trois lignes de trois carrés sans écrire neuf fois la même instruction.

La nouveauté à construire

Dans une copie de l'étape 01, remplace le dessin du carré unique par deux boucles. La version 02 montre aussi comment ranger le programme dans une fonction main. Pour débuter seul, ouvre la version 02 et cherche ces lignes dans main.

```
for ligne in range(3):
    for colonne in range(3):
        x = 185 + colonne * TAILLE_CASE
        y = 105 + ligne * TAILLE_CASE
        pygame.draw.rect(fenetre, COULEUR,
            (x, y, TAILLE_CASE - ECART,
             TAILLE_CASE - ECART))
```

range(3) fournit 0, puis 1, puis 2. Pour chaque ligne, Python parcourt trois colonnes. Cela fait 3 × 3 dessins. Les espaces en début de ligne montrent quelles instructions appartiennent à quelle boucle.

À toi de coder

- Dans le fichier 02, remplace COULEUR par (150, 60, 210), puis relance.
- Passe ECART de 5 à 10 : les séparations grandissent. Garde ECART plus petit que TAILLE_CASE.
- Pour une expérience, remplace les deux range(3) du dessin par range(2). Combien de cases attends-tu ? Remets ensuite 3.

Ce que tu viens de comprendre

Une boucle répète un travail. Les valeurs ligne et colonne changent à chaque passage ; les calculs donnent donc une nouvelle position au carré.

main est le nom d'une fonction qui regroupe le programme. Tout en bas, main() lui demande de démarrer. Tu peux utiliser cette organisation sans apprendre chaque détail dès aujourd'hui.

Mon défi libre

Déplace toute la face vers la gauche : quels nombres dois-tu modifier pour déplacer les neuf cases ensemble ?

Je vérifie mon résultat

Avec les réglages de départ, je compte neuf cases et des séparations visibles. Avec deux range(2), j'obtiens quatre cases. Pour déplacer la face, je modifie 185 dans le calcul de x et 105 dans celui de y.

Étape 03 Les six faces

Fichier repère : rubiks_etape_03.py

Mission : afficher 54 cases en organisant six faces sur un patron.

La nouveauté à construire

Garde les deux boucles de l'étape 02. Entoure-les d'une boucle sur les faces. Le dictionnaire COULEURS associe une lettre à une couleur ; POSITIONS associe une lettre à une position sur le patron.

```
for face in POSITIONS:
    x, y = origine(face)
    for ligne in range(3):
        for colonne in range(3):
            couleur = COULEURS[face]
```

Le dessin de chaque case vient ensuite, comme à l'étape 02. Le fichier 03 contient le bloc complet. La fonction origine convertit la position dans le patron en coordonnées sur l'écran.

À toi de coder

- Trouve COULEURS en haut du fichier. Remplace les trois nombres associés à F par (0, 210, 180).
- Lance : seule la face F doit changer de couleur. Les lettres permettent au programme de choisir la bonne couleur.
- Lis POSITIONS : F est en (1, 1), U en (1, 0). Pourquoi U est-elle dessinée plus haut ?

Ce que tu viens de comprendre

Un dictionnaire ressemble à un carnet de correspondances : à chaque nom correspond une information. COULEURS["F"] retrouve la couleur de F. Il ne faut pas confondre cette position dans le patron avec un numéro de pixel.

Nous gardons ce patron fixe pour apprendre les mouvements. Les faces éloignées sur le dessin peuvent se toucher lorsque le patron est replié en cube.

Mon défi libre

Dessine le même patron sur papier. Écris U D F B L R sur les faces. Sans découper le papier, imagine quelles faces touchent F.

Je vérifie mon résultat

Six faces sont visibles, avec neuf cases chacune. F touche U, D, L et R. B est la face opposée à F. J'ai modifié la palette sans déplacer les faces.

Pour les étapes suivantes, garde six couleurs bien différentes : on suivra plus facilement les déplacements.

Étape 04 La mémoire des couleurs

Fichier repère : rubiks_etape_04.py

Mission : cliquer une case et voir sa couleur changer. Ce laboratoire sert à comprendre la mémoire du programme. Il peut créer des couleurs impossibles sur un vrai Rubik's Cube.

La nouveauté à construire

Jusqu'ici, toutes les cases d'une face avaient la même couleur. Nous créons maintenant une mémoire cube avec trois lignes de trois lettres par face. Dans main, cherche cube = creer_cube().

```
cube["F"][0][0] = "R"
```

Cette instruction met la lettre R dans la case en haut à gauche de F. Dans le dessin, COULEURS[cube[face][ligne][colonne]] choisit désormais la couleur mémorisée pour chaque case.

À toi de coder

- Ajoute la ligne ci-dessus juste après cube = creer_cube(), dans main. Garde les mêmes quatre espaces au début de la ligne.
- Enregistre et relance : le coin supérieur gauche de F est rouge. Clique ailleurs pour explorer la palette.
- Appuie sur Suppr pour refaire un cube uniforme. Ton changement écrit dans le code réapparaîtra au prochain lancement. Retire cette ligne pour retrouver le départ habituel.

Ce que tu viens de comprendre

Les listes commencent à l'indice 0. Sur une face, les lignes et les colonnes portent les numéros 0, 1 et 2. cube["F"][2][1] désigne donc la case du milieu sur la ligne du bas de F.

Le programme reçoit un événement de souris. collidepoint vérifie si le clic tombe dans le rectangle d'une case. Il change la mémoire ; au prochain dessin, la nouvelle couleur apparaît.

Mon défi libre

Sans cliquer, fais apparaître une case bleue au milieu de la dernière ligne de F. Quelle instruction ajouterais-tu au démarrage ?

Je vérifie mon résultat

Réponse : cube["F"][2][1] = "B". Le clic modifie une seule case. Suppr remet les couleurs de départ. Fermer la fenêtre ne sauvegarde pas les changements faits à la souris.

À partir de l'étape 05, on retire la peinture libre et on repart d'un cube résolu : seules de vraies rotations changeront les couleurs.

Étape 05 Le premier mouvement réel

Fichier repère : rubiks_etape_05.py

Mission : faire tourner F et observer aussi les quatre bandes qui la touchent.

La nouveauté à construire

Ouvre le fichier 05. Il apporte la fonction tourner et la table VOISINS. Pour la construction accompagnée, reprends ces deux blocs depuis le fichier 05 ; retire le traitement du clic qui peignait les cases dans le fichier 04. Le nouveau clic déclenche un mouvement entier.

```
tourner(cube, "F")
```

Cette seule ligne demande un quart de tour horaire de F. Dans tourner, on fait deux travaux : tourner les neuf cases de la face, puis échanger les bandes de ses quatre voisines.

Une rotation sur papier

Dessine une grille de neuf cases et écris a, b, c sur la première ligne, d, e, f sur la deuxième, puis g, h, i sur la dernière. Après un quart de tour horaire, les lignes deviennent g, d, a puis h, e, b puis i, f, c.

Le centre reste au centre. Le coin a arrive en haut à droite. Les bandes voisines changent aussi : après F, le bas de U arrive sur le bord gauche de R. Regarde un vrai cube si tu en as un.

À toi de coder

- Lance la version 05 et clique F une fois. Note les bandes qui ont changé.
- Clique encore trois fois : tu dois retrouver exactement le départ.
- Pour commencer avec F déjà tourné, ajoute tourner(cube, "F") juste après cube = creer_cube() dans main. Relance, observe, puis retire cette ligne.

Ce que tu viens de comprendre

Une fonction peut cacher une longue recette derrière un nom court. VOISINS est une carte des cases qui échangent leur contenu. Chaque groupe contient quatre cases : la première va dans la deuxième, la deuxième dans la troisième, et la dernière revient dans la première.

Cette table est la partie la plus délicate. Tu peux l’utiliser comme une pièce de moteur déjà préparée et comprendre d’abord un seul groupe. Tu n’as pas besoin de tout recopier ou de tout retenir.

Je vérifie mon résultat

F quatre fois ramène les six faces au départ. Le milieu de chaque face garde sa couleur. Le bouton F agit aussi sur les bandes de U, R, D et L ; B reste inchangée.

Étape 06 Les commandes et le sens inverse

Fichier repère : rubiks_etape_06.py

Mission : commander chaque face avec un bouton ou une lettre du clavier.

La nouveauté à construire

Le fichier 06 étend la création des boutons de F aux six lettres UDFBLR. Chaque face reçoit aussi un bouton avec une apostrophe. Le clavier transforme une touche en commande et l'envoie à la même fonction tourner.

```
tourner(cube, "F")
tourner(cube, "F", inverse=True)
```

La deuxième ligne défait la première. Dans notre moteur, un quart de tour inverse est réalisé par trois quarts de tour horaires. True veut dire vrai ; False veut dire faux.

À toi de coder

- Clique R, puis R' : le cube doit revenir à son état précédent.
- Clique dans la fenêtre pour lui donner le clavier. Appuie sur U, puis sur Maj + U. Tu dois retrouver le départ.
- Dans main, cherche enumerate("UDFBLR"). Remplace ce texte par "FRULDB". Les boutons changent d'ordre ; leurs mouvements restent identiques. Garde les six lettres, une seule fois chacune.

Ce que tu viens de comprendre

Un bouton contient un rectangle, un texte et une commande. La commande est une paire comme ("R", False) ou ("R", True). L'interface choisit la commande ; le moteur exécute le mouvement.

Pour suivre le sens horaire, imagine ton regard placé face à la face choisie. Le bas du patron n'est pas une caméra qui regarderait directement la face D : c'est simplement une représentation dépliée.

Mon défi libre

Depuis un cube résolu, joue F, puis R, puis U. Trouve comment tout défaire sans fermer la fenêtre. Indice : commence par le dernier mouvement.

Je vérifie mon résultat

Il faut jouer U', puis R', puis F'. On inverse le sens des mouvements ET leur ordre. Les touches et les boutons donnent le même résultat.

Ferme et relance le fichier pour repartir d'un cube résolu. Les boutons de remise à zéro arrivent à l'étape suivante. Il n'y a pas encore de compteur dans cette version.

Étape 07 Mélanger et compter

Fichier repère : rubiks_etape_07.py

Mission : préparer un mélange avec de vrais mouvements, puis compter tes essais.

La nouveauté à construire

Le fichier 07 ajoute import random, un compteur, deux boutons et la détection du cube résolu. Le mélange part du cube résolu et effectue vingt mouvements tirés au hasard. Il évite de choisir deux fois de suite la même face.

```
compteur = 0
# Après un mouvement joué par toi :
compteur += 1
```

+= 1 augmente la valeur d'une unité. Un clic compte pour un quart de tour, même dans le sens inverse. Les mouvements du mélange ne sont pas comptés dans tes coups.

À toi de coder

- Au départ, joue F : le compteur devient 1 et le cube est en cours. Joue F' : le compteur devient 2 et le cube est résolu.
- Clique Mélanger, ou appuie sur Espace. Le compteur repart à zéro. Clique Remettre à zéro, ou appuie sur Suppr, pour retrouver le cube résolu.
- Cherche for coup in range(20): dans le mélange. Remplace 20 par 3 pour commencer par un petit défi. Enregistre et relance.

Ce que tu viens de comprendre

Un mélange légal est une suite de mouvements permis. Choisir au hasard la couleur de chaque case produirait souvent un cube impossible à résoudre. Notre programme fait donc tourner le cube au lieu de le repeindre.

est_resolu vérifie que toutes les cases ont retrouvé la lettre de leur face. Les centres restent fixes. Le message « Cube résolu ! » apparaît aussi au démarrage, puisque le cube commence résolu.

Mon défi libre

Sans utiliser Mélanger, fais seulement deux mouvements dont tu te souviens. Retrouve le cube résolu en appliquant leurs inverses dans le bon ordre. Recommence avec trois mouvements.

Je vérifie mon résultat

Après un mélange, je vois un état obtenu par des mouvements réels et zéro coup joué. Après Suppr, je vois six faces uniformes et zéro coup. Un mélange est aléatoire : il peut être facile, et ne garantit pas une difficulté précise.

Le programme ne résout pas le cube automatiquement. Apprendre une méthode de résolution pourra devenir une autre aventure.

Étape 08 Ma version personnelle

Fichier repère : rubiks_etape_08.py

Mission : donner un nom à ton application et essayer sans crainte grâce au bouton Annuler un coup.

Les réglages qui t'appartiennent

```
TITRE = "Mon laboratoire de cubes"
LONGUEUR_MELANGE = 20
AFFICHER_LETTRES = True
```

Ces réglages se trouvent avant main. TITRE change le grand texte dans la fenêtre. LONGUEUR_MELANGE choisit le nombre de mouvements aléatoires. Les lettres sur les cases aident à suivre les couleurs et se désactivent avec False.

À toi de coder

- Fais une copie du fichier sous le nom mon_cube.py. Change TITRE en gardant les guillemets. Lance ta copie avec py mon_cube.py.
- Choisis LONGUEUR_MELANGE = 3 pour un petit défi. Garde un entier positif, par exemple entre 1 et 50.
- Passe AFFICHER_LETTRES à False. Compare, puis choisis ton affichage préféré.

La nouveauté à comprendre

historique est une liste qui retient tes coups depuis le dernier mélange ou la dernière remise à zéro. append ajoute un coup. pop reprend le dernier. Pour l'annuler, le programme tourne la même face dans l'autre sens, puis retire un coup au compteur.

```
face, inverse = historique.pop()
tourner(cube, face, not inverse)
```

not inverse change True en False ou False en True. Le programme vérifie d'abord que la liste contient un coup : inutile de reprendre quelque chose dans une boîte vide !

Je vérifie mon résultat

Depuis le départ, je joue F, puis R. Deux clics sur Annuler un coup ramènent le cube résolu et le compteur à zéro. Un troisième clic ne casse rien. La touche Retour arrière a le même effet que le bouton.

Mélanger ou remettre à zéro efface l'historique. Annuler ne défait donc pas le mélange. Fermer la fenêtre oublie la partie ; tes changements dans le code restent enregistrés.

Mon défi libre

Présente ta version à quelqu'un. Montre un mouvement et son inverse. Explique une ligne que tu comprends. Puis demande à cette personne de prédire ce qui arrivera après quatre tours de F.

Étape 09 Observer le cube en 3D

Fichier repère : rubiks_etape_09.py

Mission : regarder le cube de tous les côtés. Pygame suffit : tu n’as aucun nouveau logiciel à installer si les étapes précédentes fonctionnent.

Un nouvel affichage pour la même mémoire

Les couleurs sont toujours dans cube. Le dessin utilise désormais 27 petits cubes. Trois coordonnées placent chaque petit cube dans l’espace : X à droite, Y vers le haut et Z vers le devant. La caméra transforme ces positions en points sur ton écran.

Le moteur graphique est fourni comme une boîte à outils. Commence par main et Camera ; la formule de rotation pourra attendre ta curiosité.

À toi de jouer et de coder

- Maintiens le bouton gauche sur le cube et déplace la souris. Retrouve la face bleue, puis la jaune. La molette rapproche ou éloigne la vue.
- Clique Recentrer la vue, ou appuie sur Début. Le cube revient à son angle de départ. Le cube lui-même n’a pas été mélangé.
- Dans les réglages du fichier, change TITRE. Essaie ensuite TAILLE_AUTOCOLLANT = 0.65 : les séparations sombres deviennent plus larges. Garde cette valeur entre 0.5 et 0.9.

Je vérifie mon résultat

Je peux voir les six couleurs en tournant autour. Les faces restent uniformes : à cette étape, seule la vue change. Tab ou Afficher les lettres donne les lettres des couleurs. Au départ F est devant, R à droite et U en haut.

Étape 10 Les tranches prennent vie

Fichier repère : rubiks_etape_10.py

Mission : faire tourner une tranche et suivre le voyage des couleurs. La même fonction tourner que dans la version à plat met à jour la mémoire du cube.

La nouveauté à construire

Le fichier 10 apporte Partie, une file de commandes et une animation. Pour la construction accompagnée, compare les fichiers 09 et 10 : la caméra et le dessin restent utilisables, tandis que Partie prend en charge les mouvements. Tu peux aussi ouvrir directement le fichier 10.

```
DUREE_TOUR = 0.45
```

Cette durée est exprimée en secondes. À chaque image, le programme avance un peu dans le quart de tour. Les neuf petits cubes de la tranche bougent ensemble. Les couleurs mémorisées changent lorsque le mouvement est terminé.

À toi de coder

- Clique F. Regarde les bandes voisines se déplacer pendant l'animation. Clique F trois autres fois pour revenir au départ.
- Remplace DUREE_TOUR par 1.2, enregistre et relance. Le mouvement devient assez lent pour suivre une case du regard. En Python, les nombres décimaux utilisent un point.
- Clique R, puis R' pour vérifier le retour. Essaie aussi les lettres du clavier ; Maj donne le sens inverse.

Une file de mouvements

Si tu cliques plusieurs fois pendant une rotation, le programme garde les commandes dans une file. Il termine le mouvement courant, puis prend le suivant. La vue peut bouger pendant ce travail : observe la tranche depuis un autre angle.

```
fraction = temps / duree
```

L'idée de fraction est simple : 0 au départ, 1 à l'arrivée. Une petite formule adoucit le départ et l'arrêt pour éviter un mouvement brutal. L'extrait explique le principe ; dans le fichier, ces valeurs appartiennent à animation.

Mon défi libre

Depuis le départ, joue F, R, U, puis U', R', F'. Observe si le cube revient à son état initial, même lorsque tu changes la vue pendant la séquence.

Je vérifie mon résultat

Une tranche entière tourne. Les autres petits cubes restent à leur place. Il n'y a pas de saut de couleur à la fin. Le mélange et l'annulation arrivent à l'étape 11 ; pour repartir du départ ici, ferme et relance le programme.

Étape 11 Mon cube jouable en 3D

Fichier repère : rubiks_etape_11.py

Mission : réunir la vue 3D, les animations et les commandes de jeu de ton cube. Fais une copie mon_cube_3d.py pour garder ta version personnelle.

Les réglages à personnaliser

```
TITRE = "Mon cube en 3D"
DUREE_TOUR = 0.45
LONGUEUR_MELANGE = 20
```

Le fichier contient déjà ces trois réglages. Change le titre, choisis 1.0 seconde pour un mouvement lent et commence avec un mélange de trois mouvements. Enregistre et relance ta copie.

Mes commandes

- Souris maintenue sur le cube : tourner la vue. Molette : zoomer. Début : recentrer la caméra. Ces actions ne comptent pas comme des coups.
- U D F B L R ou boutons : tourner les faces. Maj + lettre ou bouton avec apostrophe : sens inverse.
- Mélanger ou Espace : repartir d'un cube résolu et regarder une suite de vrais mouvements. Le mélange est animé plus vite que tes mouvements et ne compte pas dans tes coups.
- Annuler un coup ou Retour arrière : défaire ton dernier mouvement. Les commandes en attente sont traitées dans l'ordre, puis l'annulation s'applique.
- Remettre à zéro ou Suppr : arrêter le mouvement courant, effacer les commandes en attente et retrouver immédiatement le cube résolu.

À toi de vérifier

Depuis le départ, joue F puis R. Attends la fin des animations. Le compteur indique 2. Annule deux fois : le cube est résolu et le compteur indique 0. Mélange ensuite et joue un coup, puis annule-le : tu retrouves le mélange précédent.

Le mélange efface l'historique. Annuler ne remonte donc pas avant ce mélange. Remettre à zéro efface aussi l'historique. Comme dans les versions à plat, fermer la fenêtre oublie la partie ; les changements de code restent dans ton fichier.

Mon défi de présentation

Tourne la vue jusqu'à regarder le cube depuis derrière. Appuie sur F : quelle face doit bouger ? Son centre est toujours vert. Les lettres désignent des faces fixes du cube, pas la face momentanément devant tes yeux.

Présente ta version à un proche. Montre le réglage dont tu es le plus fier et explique une chose que tu as découverte. La suite de l'aventure pourra venir de ton idée !

Préparer la saisie du vrai cube

Avant les étapes 12 à 14, prends ton cube 3 × 3. Tu vas recopier ses couleurs, sans le démonter et sans tourner une tranche pendant la saisie. Tu peux tourner le cube entier pour regarder ses faces.

Commence par les centres

Sur un cube classique, place le centre blanc en haut et le centre vert devant toi. Le centre jaune est dessous, le bleu derrière, l'orange à gauche et le rouge à droite. Si ton cube a d'autres centres à ces positions, corrige d'abord les six centres dans le logiciel. Les noms U D F B L R désignent les positions que tu as choisies.

Un centre ne change pas de face quand on tourne les tranches. Il sert de nom à sa face. Les coins ont trois couleurs ; les arêtes en ont deux. Tu recopies les petits carrés colorés, pas des cubes entiers.

Dans quel sens regarder chaque face

Sélectionne une face en cliquant son nom sur le patron, ou en appuyant sur sa lettre. Le logiciel affiche la couleur du centre, celle du centre voisin à placer en haut et celle du voisin à garder à droite. Ces deux repères empêchent de recopier une face tournée de 90 degrés.

Face	Centre devant tes yeux	Centre voisin en haut	Centre voisin à droite
U	Blanc	Bleu	Rouge
D	Jaune	Vert	Rouge
F	Vert	Blanc	Rouge
B	Bleu	Blanc	Orange
L	Orange	Blanc	Vert
R	Rouge	Blanc	Bleu

Cette table correspond aux centres classiques. Si tu les as changés dans le logiciel, suis les couleurs affichées par celui-ci. Regarde chaque face directement de devant : ne retourne pas les lignes comme dans un miroir.

Un premier essai rassurant

Commence avec un cube résolu, recopie-le et vérifie que le logiciel le reconnaît. Fais ensuite un seul mouvement sur le vrai cube et recommence la saisie. Quand tu es à l'aise avec les repères, essaie un mélange plus grand.

Si tu as bougé une tranche pendant la copie, reprends les faces concernées. Une saisie valide signifie qu'un cube pourrait avoir ces couleurs ; il faut encore comparer les six faces pour vérifier que ce sont bien celles de TON cube.

Étape 12 Recopier mon vrai cube

Fichier à lancer : rubiks_etape_12.py

Mission : saisir les couleurs du cube que tu tiens dans les mains. Garde le dossier etapes complet : cet atelier utilise plusieurs fichiers qui travaillent ensemble.

À toi de saisir

- Clique une couleur dans la palette. Clique ensuite les cases correspondantes sur le patron. Les points marquent les centres ; les points d'interrogation sont les cases à renseigner.
- Pour changer de face sans peindre une case, clique son nom U D F B L R, ou utilise sa lettre au clavier. Lis les repères du haut et de la droite avant de recopier les neuf cases.
- Si tu hésites sur une case, choisis Case inconnue. Les compteurs indiquent ce qui reste à saisir et le total de chaque couleur.
- Vérifier la saisie contrôle ici les centres, les cases manquantes et les neuf exemplaires de chaque couleur. Le contrôle des pièces physiques arrive à l'étape 13.
- Observer en 3D permet de comparer la saisie au vrai cube. Corriger la saisie ramène au patron. Effacer les cases recommence avec les centres préremplis.

Garder mon travail

Enregistrer la saisie écrit mon_cube_reel.json à côté des fichiers Python. Tu peux fermer le programme, puis utiliser Charger ma saisie. Un nouvel enregistrement remplace ce même fichier ; fais-en une copie si tu veux garder plusieurs cubes.

À toi de coder

Ouvre atelier_cube_reel.py et cherche TAILLE_SAISIE = 39. Dans une copie complète du dossier, essaie 37 : les cases deviennent légèrement plus petites. Garde une taille raisonnable pour ne pas superposer les faces.

```
TAILLE_SAISIE = 39
```

Les fichiers 12, 13 et 14 ouvrent le même atelier avec des fonctions progressives. Les règles de saisie et le dessin sont rangés dans atelier_cube_reel.py ; le calcul est dans resolution_cube.py.

Je vérifie mon résultat

Il ne reste aucune case inconnue. Chaque couleur compte neuf cases et les six centres sont différents. Je peux fermer, rouvrir et recharger ma saisie. À cette étape, neuf cases par couleur ne prouvent pas encore que le cube est réalisable.

Étape 13 Vérifier un cube réalisable

Fichier à lancer : rubiks_etape_13.py

Mission : vérifier que les couleurs pourraient réellement provenir d'un cube manipulé par des rotations normales.

Pourquoi neuf cases par couleur ne suffisent pas

Imagine que tu échanges seulement les deux autocollants d'une arête. Le compte de chaque couleur reste bon, mais aucune suite de rotations normales ne peut produire cet état. Le logiciel doit donc vérifier aussi les pièces et leurs orientations.

À toi de vérifier

- Clique Charger un exemple pour obtenir un mélange légal. Clique Vérifier la saisie : le logiciel doit l'accepter.
- Charge ensuite ta propre saisie. Compare les repères, puis vérifie. Si un message signale un problème, commence par relire les faces copiées dans le mauvais sens.
- Essaie Observer en 3D et compare les six faces. Un cube accepté peut être différent du tien si tu as recopié un autre état légal ; la comparaison reste nécessaire.

Les contrôles du logiciel

Le programme cherche les douze arêtes et les huit coins, vérifie qu'aucune pièce ne manque ou n'apparaît deux fois, puis contrôle les orientations. Il détecte notamment une arête retournée seule, un coin tourné seul ou un échange isolé de deux pièces.

Ces messages indiquent une incohérence de la saisie avec les rotations normales. Ils peuvent aussi concerner un cube démonté puis remonté ou une pièce tournée à la main. Inutile de forcer une pièce : demande à un adulte de regarder avec toi.

À toi de coder

Dans resolution_cube.py, repère NOMS. Ce dictionnaire relie les lettres utilisées pour les couleurs à leurs noms. Puis cherche normaliser et verifier_cube : deux fonctions réalisent deux niveaux de contrôle différents. Ton défi est d'expliquer pourquoi le second a besoin du premier.

```
normal, centres = verifier_cube(saisie)
```

Cette ligne explique l'idée : une saisie acceptée donne un cube dont les centres servent de repère. Tu peux utiliser le contrôleur préparé sans apprendre aujourd'hui les calculs sur les coins et les arêtes.

Je vérifie mon résultat

Un exemple généré par le programme est accepté. Une case oubliée est signalée. Si ma saisie est acceptée et correspond aux six faces du vrai cube, je suis prêt pour l'étape 14.

Étape 14 Résoudre mon vrai cube

Fichier à lancer : rubiks_etape_14.py

Mission : suivre une solution avec ton vrai cube, à ton rythme. Saisis ou recharge ses couleurs, vérifie-les, puis clique Résoudre mon vrai cube. Le calcul se fait sur ton ordinateur, sans Internet.

Les sept jalons de la résolution

Le guide construit la croix blanche, complète la première couche avec les T sur les côtés, puis la deuxième couche. Il prépare ensuite la croix de la dernière face, place les derniers coins, les oriente et termine les dernières arêtes. Cet ordre peut différer de celui d'un autre tutoriel.

Un coup à la fois

- Lis Prochain coup. La lettre indique la face ; le logiciel donne aussi la couleur de son centre. Un tour horaire se lit en regardant directement cette face depuis l'extérieur du cube.
- Clique Montrer le coup et regarde l'animation. Le texte À reproduire rappelle le mouvement montré. Fais le même quart de tour sur le vrai cube, puis demande le suivant.
- Une apostrophe signifie le sens inverse. Un demi-tour est présenté comme deux quarts de tour, pour garder le même geste à chaque démonstration.
- Revenir un coup montre le mouvement inverse et recule dans le guide. Pour garder les deux cubes identiques, reproduis aussi ce mouvement sur le vrai cube.

Observer une étape entière

Lire cette étape anime les mouvements jusqu'au prochain jalon, puis s'arrête. C'est utile pour observer. Pause après ce coup termine le mouvement en cours et arrête la lecture. Pour manipuler le vrai cube, commence plutôt coup par coup.

À chaque jalon, compare le vrai cube à celui de l'écran. Des parties déjà alignées peuvent bouger pendant une séquence avant de retrouver leur place à la fin. Tu peux tourner la vue à la souris sans modifier la solution.

Si je me suis trompé de mouvement

Attends la fin de l'animation et clique Corriger la saisie. Le patron repart de l'état actuellement montré par le logiciel. Corrige-le pour correspondre à l'état actuel du vrai cube, vérifie, puis demande une nouvelle solution. Le logiciel ne voit pas tes mains et ne peut pas détecter seul un mouvement différent.

Je vérifie mon résultat

À la fin du guide, les six faces du vrai cube doivent être uniformes comme à l'écran. La méthode privilégie les couches ; elle ne cherche pas la solution la plus courte. Une longue résolution peut se répartir en plusieurs séances en enregistrant la saisie corrigée de l'état courant.

Quand quelque chose coince

Une erreur est une information. Relis les dernières lignes du message, repère le fichier et le numéro de ligne, puis compare avec le fichier numéroté d'origine. Tu peux aussi revenir à ta dernière copie qui fonctionnait.

Le terminal ne trouve pas le fichier

Vérifie le dossier ouvert. Dans le dossier etapes, tu dois voir les quatorze fichiers numérotés. Utilise le nom complet, par exemple py rubiks_etape_06.py. Après extraction, ne lance pas les fichiers directement depuis le ZIP.

Le message parle de pygame

Si tu lis No module named pygame, installe-le avec le même lanceur que pour démarrer le programme : py -m pip install pygame, puis py rubiks_etape_01.py. Si plusieurs Python sont installés, demande à un adulte de choisir le bon.

La fenêtre apparaît puis disparaît

Lance depuis le terminal pour conserver le message d'erreur. Si tu utilises IDLE, ouvre le fichier puis utilise Run > Run Module, ou F5. Modifier le texte du fichier ne change pas une fenêtre déjà ouverte : enregistre, ferme et relance.

Le message parle des espaces ou de la syntaxe

IndentationError concerne les espaces en début de ligne. Utilise quatre espaces par niveau et évite de mélanger tabulations et espaces. SyntaxError peut signaler un guillemet, une parenthèse ou les deux points oubliés après for, if ou def. Regarde aussi la ligne précédente.

Le clavier ne répond pas

Dans les versions 06 à 11, clique dans la fenêtre du cube et utilise U D F B L R ; Maj donne le sens inverse. Dans l'atelier du vrai cube, les lettres sélectionnent les faces à recopier. Utilise les boutons du guide pour montrer les mouvements.

Mes réglages cachent une partie du cube

Aux étapes 03 à 08, la fenêtre est prévue pour TAILLE_CASE = 44. De trop grandes cases débordent ou touchent les boutons. Reviens à 44. Pour changer fortement la taille du cube, il faut aussi repenser la fenêtre et ses positions.

Je veux garder mes découvertes

Enregistre tes fichiers Python personnels. La partie en cours et les couleurs cliquées ne sont pas sauvegardées automatiquement. Note dans un cahier ce que tu as changé, ce que tu attendais et ce que tu as observé.

La 3D ne se comporte pas comme prévu

Commence le glissement sur la zone du cube et utilise Recentrer la vue si tu te perds. Les noms des faces restent fixes ; les fichiers 09 à 11 n'exigent aucune installation supplémentaire après Pygame.

Si une étape devient fatigante, garde ta copie et fais une pause. Tu peux reprendre au même fichier demain ; le cube saura attendre.

Repères pour le binôme adulte

Le parcours propose quatorze étapes. Les onze premières sont autonomes ; les trois dernières partagent un atelier de saisie et de résolution fourni dans le dossier. L'étape 01 est conservée telle qu'elle a été testée. Les suivantes organisent le code dans main et ajoutent les notions par paliers. Une étape n'est pas une durée imposée : les rotations peuvent demander plusieurs séances.

Construire à partir de la version précédente

Travaillez sur une copie personnelle. Repérez les nouveautés décrites dans le carnet, puis comparez les deux fichiers. Le dossier pour_aller_plus_loin contient les treize relevés complets changements_01_vers_02.txt à changements_13_vers_14.txt : les lignes précédées de - sont retirées, celles précédées de + sont ajoutées. Les relevés sont destinés à l'adulte, pas à être recopiés comme programmes.

Les courts extraits du carnet expliquent les idées ; ils ne reproduisent pas tous les changements nécessaires. Pour une reconstruction exacte, les versions numérotées et les relevés servent de références complètes. L'enfant peut aussi explorer directement chaque version prête et modifier sa copie.

Les passages qui méritent du temps

Étape 02 : les deux boucles et leur indentation. Étape 04 : distinguer données et dessin, puis expliquer les indices 0 à 2. Étape 05 : commencer par une rotation sur papier ; fournir VOISINS comme un composant préparé. Étape 06 : relier les commandes au moteur. Étape 08 : découvrir une liste utilisée comme historique.

Convention des mouvements

Un mouvement est horaire vu de l'extérieur, directement face à la face concernée. La table VOISINS contient trois cycles de quatre cases par face. La grille de la face tourne séparément ; ses deux cycles de coins et d'arêtes portent à vingt le nombre de cases déplacées. Le centre ne bouge pas.

Vérifications effectuées

Les versions 01 à 11 ont été compilées et exécutées avec Pygame en affichage virtuel. Les moteurs des étapes 05 à 08 ont passé 96 vérifications au total avec les interfaces : comparaison des mouvements avec un modèle géométrique indépendant, quatre tours identiques, mouvement et inverse, couleurs et centres conservés, mille mouvements suivis de leurs inverses, et commandes de mélange, remise à zéro et annulation.

La 3D a passé 63 contrôles supplémentaires, dont 36 raccords entre la géométrie en fin d'animation et les mouvements logiques. L'affichage a été inspecté. Vos essais restent nécessaires pour le clavier et le confort de lecture. Les versions 01 à 11 sont locales et sans solveur ni sauvegarde de partie ; les étapes 12 à 14 ajoutent une saisie enregistrable et un solveur local. Les mélanges sont pédagogiques, sans protocole de compétition.

Les formules de perspective et de rotation peuvent rester des outils préparés. Le but est qu'il puisse expliquer une modification et en prévoir un effet ; la reconstruction intégrale du rendu 3D n'est pas un prérequis pour jouer et apprendre.

Repères du solveur pour le binôme adulte

L’extension travaille sur un cube 3 × 3 à six couleurs. La saisie est manuelle : il n’y a pas de reconnaissance par caméra. La validation vérifie la cohérence physique de l’état, puis le guide fournit une séquence par couches avec des pauses aux jalons.

Les fichiers communs des étapes 12 à 14

Les fichiers rubiks_etape_12.py, rubiks_etape_13.py et rubiks_etape_14.py ouvrent les niveaux successifs du même atelier. atelier_cube_reel.py gère la saisie et la lecture guidée. resolution_cube.py contrôle l’état et adapte le solveur au moteur 3D de rubiks_etape_11.py. Le dossier rubik contient le moteur libre fourni avec sa licence. Il faut conserver cet ensemble dans le même dossier ; Pygame reste la seule installation nécessaire.

Origine et méthode

Le moteur de résolution est rubik-cube 0.0.2 de Paul Glass, distribué sous licence MIT. Sa source Python est fournie sans modification dans rubik ; notre adaptateur traduit ses mouvements vers les six faces fixes de l’application. Les rotations de repère et de tranches centrales sont converties pour ne proposer à l’enfant que des quarts de tour U D F B L R et leurs inverses.

Source du projet : https://github.com/pglass/cube. La notice de licence complète est conservée dans etapes/rubik/LICENSE.txt, avec les indications d’origine dans ORIGINE.txt.

Vérifications de cette extension

150 mélanges de 1 à 79 mouvements ont été résolus, avec vérification de la croix blanche alignée, de la première couche, de la deuxième couche, de la croix opposée, des états de fin de phase et du cube final. Les essais ont aussi couvert des centres de couleurs personnalisés, les états impossibles et la sauvegarde. 30 contrôles supplémentaires ont vérifié les ateliers, les commandes, le retour en arrière et l’arrêt à un jalon. Les écrans de saisie et de guide ont été inspectés.

Accompagner sans surcharger

La solution n’est pas minimale : les essais ont parfois demandé plus de 200 quarts de tour. Pour une première séance, utiliser un cube résolu, puis un petit mélange. La lecture automatique sert d’abord à observer ; la manipulation réelle se fait plus sereinement coup par coup. Le logiciel ne détecte pas les gestes réels : une erreur demande de corriger la saisie de l’état courant et de recalculer.

La sauvegarde conserve les couleurs saisies, pas la position dans la lecture ni l’historique du guide. Pour interrompre une résolution, attendre la fin du mouvement, revenir à la saisie, vérifier l’accord avec le vrai cube et enregistrer. À la reprise, un nouveau guide est calculé depuis cet état.