Mémo Détaillé : Python Turtle

Ce guide approfondi explore les fonctionnalités du module `turtle` de Python, des commandes de base aux concepts plus avancés, pour vous permettre de maîtriser le dessin et l'animation graphique.

Page 1 : Les Fondamentaux et le Contrôle du Crayon

---

1. Introduction et Configuration

Le module `turtle` est une manière ludique d'introduire la programmation en Python. Il fournit un environnement de dessin simple où une "tortue" (le curseur) se déplace sur un canevas 2D (l'écran).

Pour commencer, il faut toujours importer le module :

Python
```
import turtle
```

Configuration de l'environnement de travail :

- **L'Écran (Screen) :** C'est la fenêtre dans laquelle la tortue dessine. Il est recommandé de le créer et de le configurer au début.
- **La Tortue (Turtle) :** C'est l'objet qui se déplace et dessine. Vous pouvez en créer plusieurs.

Python
```
# Initialisation de l'environnement
screen = turtle.Screen()
screen.title("Mon Premier Dessin Turtle")
screen.bgcolor("lightcyan")

# Création d'une instance de tortue
t = turtle.Turtle()
```

---

2. Mouvements de la Tortue

Ce sont les commandes de base pour déplacer la tortue sur le canevas.

Commande	Alias	Description Détaillée
`turtle.forward(distance)`	`fd()`	Fait avancer la tortue dans la direction où elle pointe, sur une `distance` en pixels.
`turtle.backward(distance)`	`bk()` / `back()`	Fait reculer la tortue, à l'opposé de sa direction actuelle.
`turtle.right(angle)`	`rt()`	Pivote la tortue sur elle-même de `angle` degrés vers la droite (sens horaire).
`turtle.left(angle)`	`lt()`	Pivote la tortue sur elle-même de `angle` degrés vers la gauche (sens anti-horaire).
`turtle.goto(x, y)`	`setpos()` / `setposition()`	Déplace la tortue à une position absolue `(x, y)` sur le canevas. Le centre est `(0, 0)`.

Commande	Alias	Description Détaillée
`turtle.setx(x)`		Déplace la tortue horizontalement à la coordonnée `x`, sans changer `y`.
`turtle.sety(y)`		Déplace la tortue verticalement à la coordonnée `y`, sans changer `x`.
`turtle.setheading(angle)`	`seth()`	Oriente la tortue dans une direction absolue. 0° vers l'Est, 90° vers le Nord, 180° vers l'Ouest, 270° vers le Sud.
`turtle.home()`		Déplace la tortue à l'origine `(0, 0)` et la réoriente vers l'Est (0°).

---

3. Contrôle du Crayon

Ces commandes gèrent si la tortue dessine ou non, ainsi que l'apparence du trait.

Commande	Alias	Description Détaillée
`turtle.pendown()`	`pd()` / `down()`	Abaisse le crayon. La tortue laissera une trace lors de ses prochains mouvements. C'est l'état par défaut.
`turtle.penup()`	`pu()` / `up()`	Soulève le crayon. La tortue peut se déplacer sans dessiner, utile pour créer des formes discontinues.
`turtle.pensize(width)`	`width()`	Définit l'épaisseur du trait en pixels. Un entier positif est attendu.
`turtle.isdown()`		Retourne `True` si le crayon est abaissé, `False` sinon.

---

4. Couleurs et Remplissage

La gestion des couleurs est essentielle pour rendre les dessins vivants.

Commande	Description Détaillée
`turtle.pencolor(color)`	Définit la couleur du trait. `color` peut être une chaîne (ex: `"red"`, `"#FF5733"`) ou un tuple RGB.
`turtle.fillcolor(color)`	Définit la couleur de remplissage pour les formes.
`turtle.color(p_color, f_color)`	Raccourci pour définir la couleur du trait (`p_color`) et du remplissage (`f_color`) en une seule commande.
`turtle.begin_fill()`	Marque le point de départ d'une forme qui sera remplie. Doit être appelé avant de dessiner la forme.

Commande	Description Détaillée
`turtle.end_fill()`	Termine la capture de la forme et la remplit avec la couleur définie par `fillcolor()`.

Modes de couleur :

Par défaut, Turtle utilise des chaînes de caractères pour les couleurs. Pour utiliser des valeurs RGB, il faut changer le mode de couleur de l'écran.

Python
```
screen.colormode(255)  # Permet d'utiliser des tuples RGB (R, G, B)
t.pencolor((255, 0, 128)) # Une couleur rose
```

Page 2 : État de la Tortue et Commandes Avancées

---

5. Gestion de l'État et de l'Apparence de la Tortue

Contrôlez l'apparence de la tortue et obtenez des informations sur son état actuel.

Commande	Alias	Description Détaillée
`turtle.shape(name)`		Change l'apparence du curseur. Formes disponibles : `"arrow"`, `"turtle"`, `"circle"`, `"square"`, `"triangle"`, `"classic"`.
`turtle.speed(speed)`		Définit la vitesse de l'animation. `0` est la plus rapide (pas d'animation), `1` (lent) à `10` (rapide) sont des vitesses progressives.
`turtle.hideturtle()`	`ht()`	Rend la tortue invisible. Utile pour un rendu final propre.
`turtle.showturtle()`	`st()`	Rend la tortue visible.
`turtle.isvisible()`		Retourne `True` si la tortue est visible, `False` sinon.
`turtle.position()`	`pos()`	Retourne les coordonnées actuelles `(x, y)` de la tortue.
`turtle.heading()`		Retourne l'angle actuel (orientation) de la tortue en degrés.
`turtle.xcor()` / `turtle.ycor()`		Retournent respectivement la coordonnée `x` ou `y` de la tortue.

---

6. Dessiner des Formes Complexes

En plus des lignes, Turtle peut dessiner des formes géométriques.

Commande	Description Détaillée
`turtle.circle(radius, extent, steps)`	Dessine un arc. `radius` est le rayon du cercle. `extent` est un angle qui définit la portion du cercle à dessiner. `steps` permet de dessiner un polygone qui approxime le cercle.
`turtle.dot(size, color)`	Dessine un point plein. `size` est le diamètre en pixels. `color` est optionnel.
`turtle.stamp()`	"Tamponne" une copie de la forme de la tortue sur le canevas à sa position actuelle. Retourne un ID pour pouvoir effacer ce tampon plus tard.
`turtle.clearstamp(stamp_id)`	Efface un tampon spécifique en utilisant l'ID retourné par `stamp()`.
`turtle.clearstamps()`	Efface tous les tampons de la tortue.

Exemple de `circle()` :

Python

```
# Dessine un demi-cercle de rayon 80
t.circle(80, 180)

# Dessine un hexagone avec la fonction circle
t.circle(60, steps=6)
```

---

7. Commandes de la Fenêtre (Screen)

L'objet `Screen` a ses propres méthodes pour contrôler l'environnement global.

Commande	Description Détaillée
`screen.setup(width, height)`	Définit la taille de la fenêtre Turtle en pixels.
`screen.setworldcoordinates(llx, lly, urx, ury)`	Permet de définir un système de coordonnées personnalisé. `llx`, `lly` sont les coordonnées du coin inférieur gauche et `urx`, `ury` celles du coin supérieur droit.
`screen.tracer(n, delay)`	Active/désactive l'animation. `tracer(0)` désactive les mises à jour, utile pour des dessins complexes afin d'accélérer le processus. Il faut ensuite appeler `screen.update()` pour afficher le résultat.
`screen.update()`	Met à jour le dessin à l'écran lorsque le `tracer` est désactivé.
`screen.listen()`	Permet à la fenêtre d'écouter les événements clavier ou souris.
`screen.onkey(fun, key)`	Associe une fonction `fun` à l'appui sur une touche `key` (ex: "space", "Up", "a").

Commande	Description Détaillée
`screen.onclick(fun)`	Associe une fonction `fun` à un clic de souris sur le canevas.
`turtle.done()` / `turtle.exitonclick()`	`done()` maintient la fenêtre ouverte. `exitonclick()` attend un clic pour fermer la fenêtre. Essentiel à la fin d'un script.

Page 3 : Interactivité, Utilitaires et Exemples

---

8. Gestion des Événements (Interactivité)

Rendez vos programmes Turtle interactifs en réagissant aux actions de l'utilisateur.

Exemple de contrôle au clavier :

Python

```python
def move_forward():
    t.forward(10)

def turn_left():
    t.left(10)

def turn_right():
    t.right(10)

screen.listen() # La fenêtre écoute les événements
# Associe la fonction move_forward à la touche "Up" (flèche haut)
screen.onkey(move_forward, "Up")
screen.onkey(turn_left, "Left")
screen.onkey(turn_right, "Right")
```

Exemple de clic de souris :

Python

```python
def draw_on_click(x, y):
    # La fonction reçoit les coordonnées du clic
    t.goto(x, y)

screen.onclick(draw_on_click)
```

---

9. Commandes Utilitaires et de Réinitialisation

Commande	Description Détaillée
`turtle.clear()`	Efface tous les dessins faits par la tortue, mais la laisse à sa position et état actuels.
`turtle.reset()`	Réinitialisation complète. Efface les dessins, ramène la tortue à l'origine (0,0), et réinitialise toutes ses propriétés (couleur, épaisseur, etc.) à leurs valeurs par défaut.
`turtle.undo()`	Annule la dernière action de la tortue.

Commande	Description Détaillée
`turtle.write(arg, move, align, font)`	Écrit du texte sur le canevas à la position de la tortue. `arg` est le texte à écrire. `move` (bool) indique si la tortue doit bouger. `align` peut être "left", "center" ou "right". `font` est un tuple (nom_police, taille, style).

Exemple avec `write()`:

Python
```
t.penup()
t.goto(0, 150)
t.pendown()
t.write("Bonjour Turtle!", align="center", font=("Arial", 24, "bold"))
```

---

10. Exemple de Script Complet : Spirale Colorée

Ce script combine plusieurs des concepts vus pour créer un dessin plus complexe.

Python
```
import turtle
import random

# Configuration
screen = turtle.Screen()
screen.bgcolor("black")
screen.colormode(255) # Active le mode RGB

t = turtle.Turtle()
t.speed(0) # Vitesse maximale
t.hideturtle()
t.pensize(2)

# Dessin de la spirale
for i in range(300):
    # Choisit une couleur RGB aléatoire
    r = random.randint(0, 255)
    g = random.randint(0, 255)
    b = random.randint(0, 255)
    t.pencolor((r, g, b))

    t.forward(i) # La longueur du segment augmente à chaque itération
    t.left(59)   # Angle de rotation (expérimentez avec cette valeur!)

# Finalisation
turtle.done()
```