Schritt-für-Schritt Tutorial: Snake mit Python & pygame

Ziel: Du baust ein vollständiges Snake-Spiel in Python mit `pygame`. Das Tutorial führt dich von den Grundlagen (Fenster + Spiel-Loop) bis zu Kollisionsprüfung, Punktestand und Erweiterungen. Versuche jede Aufgabe zuerst selbst — die *Lösung* steht im Anhang, nur zum Nachschauen.

Voraussetzungen

- Python 3.9 oder neuer installiert.
- `pygame` installieren: `pip install pygame` (oder `python -m pip install pygame`).
- Ein Ordner für das Projekt, z. B. `snake_game`. Erstelle darin `snake.py`.

Kurzer Plan (was du schrittweise baust)

1. Fenster + Spiel-Loop
2. Gitter/Block-Größe definieren
3. Schlange zeichnen (Liste von Segmenten)
4. Bewegung & Eingaben
5. Futter (Food) zufällig platzieren
6. Essen / Wachsen
7. Kollisionen (Wand & Eigenkörper)
8. Punktestand & Game-Over-Bildschirm
9. Extra-Aufgaben (Pause, Sound, Highscore)

Wichtige Konzepte (kurz)

- **Block-Grid:** Verwende eine feste Block-Größe (z. B. 20 px). Positionen sind Vielfache davon — so bleiben Snake-Segmente und Futter immer auf einem sauberen Raster.
- **Snake als Liste:** `snake = [[x1,y1], [x2,y2], ...]` — Kopf ist das letzte Element oder erste, je nach Implementierung.
- **Bewegung:** Ändere pro Frame nur die Kopf-Koordinate um `±BLOCK` in x oder y.
- **Keine 180°-Wende:** Verhindere, dass die Schlange direkt umkehrt (sonst kollidiert sie sofort mit sich selbst).

Schritt 1 — Fenster + Spiel-Loop

Ziel: Öffne ein Fenster, fülle den Hintergrund, und sorge dafür, dass das Fenster beim Schließen sauber beendet wird.

```python
import pygame
pygame.init()

WIDTH, HEIGHT = 600, 400
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption('Snake')

running = True
while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

    screen.fill((50,153,213))
    pygame.display.update()

pygame.quit()
```

Aufgabe: Starte das Programm. Ändere die Fenstergröße und die Hintergrundfarbe. Wenn das klappt — weiter zu Schritt 2.

Schritt 2 — Grid und Konstanten

Ziel: Definiere `BLOCK` (Größe eines Segments) und nutze nur Vielfache davon für Positionen.

```python
BLOCK = 20  # Größe eines Snake-Segments (px)
FPS = 10    # Grundgeschwindigkeit (Frames pro Sekunde)
```

Hinweis: Wenn du `random.randrange(0, WIDTH, BLOCK)` benutzt, erzeugst du automatisch koordinaten, die auf dem Grid liegen.

Schritt 3 — Schlange zeichnen

Ziel: Implementiere `draw_snake()` und zeichne die anfängliche Schlange (1 Segment).

```python
def draw_snake(snake_list):
    for x, y in snake_list:
        pygame.draw.rect(screen, (0,255,0), (x, y, BLOCK, BLOCK))

# Beispielinitialisierung
snake = [[WIDTH//2, HEIGHT//2]]
```

Aufgabe: Zeichne das Segment in der Mitte des Bildschirms.

Schritt 4 — Bewegung & Eingaben

Ziel: Reagiere auf Pfeiltasten und bewege die Schlange in Block-Schritten. Verhindere 180°-Wenden.

Wichtige Logik:

- `dx, dy` geben die Bewegungsrichtung an (z. B. `dx = BLOCK, dy = 0` beim Rechts-Drücken).
- Beim Drücken von LEFT prüfe vorher `dx != BLOCK` — so lässt du eine direkte Umkehr nicht zu.

Muster:

```
x = WIDTH // 2
y = HEIGHT // 2
dx = 0
dy = 0

for event in pygame.event.get():
    if event.type == pygame.KEYDOWN:
        if event.key == pygame.K_LEFT and dx != BLOCK:
            dx = -BLOCK; dy = 0
        elif event.key == pygame.K_RIGHT and dx != -BLOCK:
            dx = BLOCK; dy = 0
        elif event.key == pygame.K_UP and dy != BLOCK:
            dx = 0; dy = -BLOCK
        elif event.key == pygame.K_DOWN and dy != -BLOCK:
            dx = 0; dy = BLOCK

# pro Frame:
x += dx
y += dy
```

Aufgabe: Lasse die Schlange sich bewegen. Achte darauf, dass sie sich nicht diagonal bewegt (immer genau eine Achse pro Movement).

Schritt 5 — Futter (Food)

Ziel: Erzeuge Essen an zufälliger, grid-aligned Position.

```
import random

def random_food():
    fx = random.randrange(0, WIDTH, BLOCK)
    fy = random.randrange(0, HEIGHT, BLOCK)
    return fx, fy
```

```
food_x, food_y = random_food()
# zeichnen:
pygame.draw.rect(screen, (0,0,0), (food_x, food_y, BLOCK, BLOCK))
```

Aufgabe: Zeichne das Futter und überprüfe, ob `food_x, food_y` immer innerhalb des Fensters liegen.

Schritt 6 — Essen & Wachsen

Ziel: Wenn der Kopf die Futter-Koordinaten erreicht, wachse die Schlange und erzeuge neues Futter.

Pattern:

- `snake.append([x, y])` für neue Kopfposition
- Wenn `len(snake) > length`: `del snake[0]` — so bewegt sich die Schlange, ohne zu wachsen.
- Bei Essen: `length += 1` und `food = random_food()`

```
# Kopfposition check
if x == food_x and y == food_y:
    length += 1
    food_x, food_y = random_food()
```

Aufgabe: Implementiere das Wachstum und erhöhe optionally den Score.

Schritt 7 — Kollisionen (Wand & Eigenkörper)

Ziel: Spiel beenden, wenn Kopf außerhalb des Fensters ist oder in einen Körperteil läuft.

- Wand: `if x < 0 or x >= WIDTH or y < 0 or y >= HEIGHT: game_over`.
- Eigenkörper: vergleiche Kopf mit allen Segmenten außer dem letzten.

```
for segment in snake[:-1]:
    if segment == [x, y]:
        game_over = True
```

Game-Over-Bildschirm: Zeige Text an und lass den Spieler `Q` zum Beenden und `C` zum Neustarten drücken.

Schritt 8 — Score & UI

Ziel: Zeige Punktestand oben links an.

```
font = pygame.font.SysFont('consolas', 24)
text = font.render(f"Score: {score}", True, (0,0,0))
screen.blit(text, (10, 10))
```

Aufgabe: Aktualisiere `score` bei jedem gegessenen Futter und zeige ihn an.

Schritt 9 — Zusammensetzen & Ablaufübersicht

1. initialisiere Variablen (Position, snake list, length, food, score)
2. im Loop: Event-Handling → Bewegung → Kollisionen prüfen → Zeichenoperationen → `pygame.display.update()` → `clock.tick(speed)`
3. Beim Game Over: zeige Text & warte auf Benutzeraktion (Q/C)

Test-Checkpoint: Wenn dein Programm diese Schritte korrekt ausführt, hast du ein spielbares Snake.

Erweiterungen (Challenge-Tasks)

- **Geschwindigkeit erhöhen**: alle 5 Punkte `speed += 1`.
- **Pause**: Taste `P` toggelt Pause.
- **Highscore speichern**: schreibe höchsten Score in eine Textdatei.
- **Hindernisse**: generiere Blöcke, die tödlich sind.
- **Mobiler Touch-Support**: mit Keras? (komplizierter)
- **Verbesserte Grafik**: benutze Bilder statt Rechtecken, Animationen.

Für jede Erweiterung gebe ich dir gerne kleine, schrittweise Hinweise.

Troubleshooting (häufige Fehler)

- `ModuleNotFoundError: No module named 'pygame'` → `pip install pygame`.
- Fenster hängt/keine Events verarbeitet → prüfe, ob du `pygame.event.get()` in jedem Frame aufrufst.
- Snake "springt" nicht auf Grid → verwende nur Vielfache von `BLOCK`.

Anhang: komplette Referenzimplementierung (nur anschauen, wenn du stecken bleibst)

```
import pygame
import random
import sys

pygame.init()

# === Einstellungen ===
```

```
WIDTH, HEIGHT = 600, 400
BLOCK = 20
FPS = 10

WHITE = (255,255,255)
BLACK = (0,0,0)
RED = (213,50,80)
GREEN = (0,255,0)
BLUE = (50,153,213)

screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption('Snake')
clock = pygame.time.Clock()

font_style = pygame.font.SysFont('consolas', 20)
score_font = pygame.font.SysFont('consolas', 25)


def draw_snake(snake_list):
    for x,y in snake_list:
        pygame.draw.rect(screen, GREEN, (x,y,BLOCK,BLOCK))


def show_score(score):
    value = score_font.render("Score: " + str(score), True, BLACK)
    screen.blit(value, [0,0])


def random_food():
    x = random.randrange(0, WIDTH, BLOCK)
    y = random.randrange(0, HEIGHT, BLOCK)
    return x,y


def message(msg, color, pos=None):
    if pos is None:
        pos = (WIDTH//6, HEIGHT//3)
    mesg = font_style.render(msg, True, color)
    screen.blit(mesg, pos)


def game_loop():
    game_over = False
    game_close = False

    x = WIDTH//2
    y = HEIGHT//2
    dx = 0
    dy = 0

    snake_list = []
```

```
    snake_length = 1

    food_x, food_y = random_food()
    score = 0
    speed = FPS

    while not game_over:
        while game_close:
            screen.fill(BLUE)
            message("Game Over! Drücke C zum Neustart oder Q zum Beenden",
RED)
            show_score(score)
            pygame.display.update()

            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    game_over = True
                    game_close = False
                if event.type == pygame.KEYDOWN:
                    if event.key == pygame.K_q:
                        game_over = True
                        game_close = False
                    elif event.key == pygame.K_c:
                        game_loop()  # Neustart

        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                game_over = True
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_LEFT and dx != BLOCK:
                    dx = -BLOCK; dy = 0
                elif event.key == pygame.K_RIGHT and dx != -BLOCK:
                    dx = BLOCK; dy = 0
                elif event.key == pygame.K_UP and dy != BLOCK:
                    dx = 0; dy = -BLOCK
                elif event.key == pygame.K_DOWN and dy != -BLOCK:
                    dx = 0; dy = BLOCK

        x += dx
        y += dy

        if x < 0 or x >= WIDTH or y < 0 or y >= HEIGHT:
            game_close = True

        screen.fill(BLUE)
        pygame.draw.rect(screen, BLACK, (food_x, food_y, BLOCK, BLOCK))

        snake_head = [x,y]
        snake_list.append(snake_head)
        if len(snake_list) > snake_length:
            del snake_list[0]
```

```
        for segment in snake_list[:-1]:
            if segment == snake_head:
                game_close = True

        draw_snake(snake_list)
        show_score(score)
        pygame.display.update()

        if x == food_x and y == food_y:
            food_x, food_y = random_food()
            snake_length += 1
            score += 1
            if score % 5 == 0:
                speed += 2

        clock.tick(speed)

    pygame.quit()
    sys.exit()

if __name__ == '__main__':
    game_loop()
```

Wenn du möchtest, begleite ich dich Schritt für Schritt: du implementierst Schritt 1, schickst mir den Code/Fehlermeldungen oder schreibst "Fertig Schritt 1" und ich helfe beim nächsten Schritt. Wie willst du fortfahren?