

**ĐẠI HỌC QUỐC GIA TP. HỒ CHÍ MINH**  
**TRƯỜNG ĐẠI HỌC CÔNG NGHỆ THÔNG TIN**



**BÁO CÁO BÀI TẬP MÔN KỸ NĂNG NGHỀ NGHIỆP**  
**BÀI TẬP: SS004.8 - Hướng Dẫn Làm Quen Với Git**

**Giảng viên hướng dẫn: Nguyễn Văn Toàn**

**Nhóm: 36**

**Sinh viên thực hiện:**

**Trần Cao Anh Kiệt – 24520914**

**Nguyễn Quốc Thắng – 24521609**

**Trần Trí Thiện – 24521670**

**Lớp: SS004.Q12**

Tp. Hồ Chí Minh, Ngày 11 Tháng 16 năm 2025

# Mục lục

1. Giới Thiệu Tổng Quan.....	1
2. Cấu trúc chương trình.....	1
- snake.py (Lớp Snake):.....	1
- food.py (Lớp Food): .....	1
- scoreboard.py (Lớp Scoreboard): .....	1
- main.py (File điều khiển chính): .....	2
3. Phân tích các chức năng và logic chính .....	2
3.1. Quản lý màn hình và tốc độ Game.....	2
3.2. Logic di chuyển của Rắn .....	2
3.3. Xử lý va chạm .....	2
3.4. Hệ thống điểm cao (High Score) .....	3
4. Kết luận.....	3

## 1. Giới Thiệu Tổng Quan

Dự án này thực hiện việc xây dựng lại trò chơi "Rắn Săn Mồi" (Snake Game) cổ điển. Mục tiêu là tạo ra một trò chơi hoàn chỉnh, trong đó người chơi điều khiển một con rắn di chuyển trên màn hình để ăn "mồi". Mỗi khi ăn mồi, con rắn sẽ dài ra. Trò chơi kết thúc khi con rắn đâm vào tường hoặc tự đâm vào chính thân mình. Trò chơi được phát triển bằng ngôn ngữ Python và sử dụng thư viện Turtle để xử lý toàn bộ phần đồ họa và giao diện.

## 2. Cấu trúc chương trình

Dự án được thiết kế theo mô hình Lập trình Hướng đối tượng (OOP), giúp mã nguồn trở nên rõ ràng, dễ quản lý và bảo trì. Cấu trúc được chia thành 4 file chính:

- **snake.py (Lớp Snake):**

- + Quản lý toàn bộ hoạt động của con rắn.

- + Lưu trữ các "đốt" (segments) của con rắn trong một danh sách (self.segments).

- + Điều khiển logic di chuyển: các đốt sau sẽ di chuyển đến vị trí của đốt ngay trước nó, và đốt đầu tiên (head) di chuyển về phía trước.

- + Cung cấp các phương thức để điều khiển hướng (up, down, left, right) và ngăn chặn rắn quay đầu 180 độ.

- + Phương thức extend() để thêm một đốt mới vào đuôi khi rắn ăn mồi.

- + Phương thức reset() để tái tạo con rắn về trạng thái ban đầu.

- **food.py (Lớp Food):**

- + Kế thừa từ lớp Turtle của thư viện Turtle.

- + Đại diện cho "mồi" mà con rắn phải ăn.

- + Có hình dạng là một hình tròn nhỏ màu xanh.

- + Phương thức refresh() để di chuyển mồi đến một vị trí ngẫu nhiên mới trên màn hình sau khi bị ăn.

- **scoreboard.py (Lớp Scoreboard):**

- + Kế thừa từ lớp Turtle.

- + Quản lý và hiển thị điểm số (score) của người chơi.

- + Đọc và lưu trữ điểm cao nhất (high score) bằng cách sử dụng file data.txt.

- + Phương thức `increase_score()` để tăng điểm và cập nhật hiển thị.
- + Phương thức `reset()`: So sánh điểm hiện tại với điểm cao nhất. Nếu vượt qua, cập nhật `high_score` và ghi vào file `data.txt` trước khi reset điểm về 0.
- **main.py (File điều khiển chính):**
  - + Đây là file "trung tâm" kết nối tất cả các lớp lại với nhau.
  - + Khởi tạo màn hình chơi game (Screen) với kích thước 600x600, màu nền đen.
  - + Khởi tạo các đối tượng từ 3 lớp: snake, food, và scoreboard.
  - + Sử dụng `screen.listen()` và `screen.onkey()` để lắng nghe sự kiện nhấn phím (Mũi tên Lên, Xuống, Trái, Phải) và gọi các phương thức tương ứng của snake.
  - + Chứa vòng lặp chính của game (game loop), nơi toàn bộ logic của trò chơi diễn ra

### 3. Phân tích các chức năng và logic chính

#### 3.1. Quản lý màn hình và tốc độ Game

- Trong `main.py`, hai lệnh quan trọng nhất để game chạy mượt là:
  - + **`screen.tracer(0)`**: Lệnh này tắt chế độ tự động cập nhật (vẽ lại) màn hình của Turtle.
  - + **`screen.update()`**: Đặt bên trong vòng lặp `while`, lệnh này cho phép chúng ta kiểm soát chính xác thời điểm vẽ lại màn hình.
- Kết hợp với `time.sleep(0.1)`, chúng ta tạo ra một tốc độ khung hình (frame rate) ổn định, giúp con rắn di chuyển từng bước một thay vì di chuyển tức thời.

#### 3.2. Logic di chuyển của Rắn

- **Đây là phần logic cốt lõi trong `snake.py` (phương thức `move()`):**
  - + Vòng lặp `for` chạy ngược từ đuôi rắn (`len(self.segments) - 1`) về đến đốt thứ hai (ngay sau đầu).
  - + Mỗi đốt sẽ di chuyển đến chính xác vị trí (tọa độ x, y) của đốt ngay phía trước nó.
  - + Sau khi tất cả các đốt sau đã "bắt kịp" đốt trước, đốt đầu tiên (`self.head`) sẽ di chuyển về phía trước một khoảng `MOVE_DISTANCE`.
- Điều này tạo ra ảo ảnh một con rắn di chuyển mượt mà như một thể thống nhất.

#### 3.3. Xử lý va chạm

Logic phát hiện va chạm được xử lý hoàn toàn trong `main.py`:

#### **- Va chạm với Môi:**

+ if snake.head.distance(food) < 15:

+ Sử dụng phương thức distance() của Turtle để đo khoảng cách giữa đầu rắn và môi.

+ Nếu va chạm: Môi được làm mới (food.refresh()), rắn dài ra (snake.extend()), và điểm tăng (scoreboard.increase\_score()).

#### **- Va chạm với Tường:**

+ if snake.head.xcor() > 280 or ... < -280:

+ Kiểm tra xem tọa độ x hoặc y của đầu rắn có vượt quá ranh giới màn hình (600x600, với lề là 280) hay không.

+ Nếu va chạm: Game được reset (scoreboard.reset(), snake.reset()).

#### **- Va chạm với Đuôi (Tự cắn):**

+ for segment in snake.segments[1:]: (Sử dụng slicing [1:] để bỏ qua việc kiểm tra va chạm với chính nó - đầu rắn).

+ if snake.head.distance(segment) < 10:

+ Kiểm tra khoảng cách từ đầu rắn đến từng đốt còn lại trong thân nó.

+ Nếu va chạm: Game được reset.

### **3.4. Hệ thống điểm cao (High Score)**

- Một tính năng nâng cao là lưu điểm cao. Lớp Scoreboard xử lý việc này:

+ Khi khởi tạo (\_\_init\_\_): Mở file data.txt, đọc giá trị bên trong và gán cho self.high\_score.

+ Khi kết thúc game (reset): So sánh self.score với self.high\_score. Nếu điểm mới cao hơn, mở file data.txt ở chế độ "write" (mode="w") và ghi đè giá trị mới vào.

## **4. Kết luận**

Dự án đã thành công trong việc xây dựng một phiên bản hoàn chỉnh và có thể chơi được của trò Snake Game. Việc áp dụng Lập trình Hướng đối tượng đã giúp phân tách rõ ràng các nhiệm vụ: Snake lo di chuyển, Food lo vị trí môi, Scoreboard lo điểm số, và main.py điều phối tất cả.

Các chức năng chính như di chuyển, ăn môi, phát hiện va chạm (với tường và thân), và hệ thống điểm cao bền vững (lưu vào file) đều hoạt động chính xác. Đây là một dự án tuyệt vời để thực hành các khái niệm OOP và logic xử lý game trong Python.