PART 1: Importing Libraries

```
import pygame
import sys
import random
```

🔍 What's going on here?

- `import` means "bring in" some tools that doesn't have by default. We need them to build our game.
- `pygame` is a special library made just for making games. It lets us draw things, detect keyboard presses, and update the screen.
- `sys` is a built-in library that helps us close the game properly when it's over.
- `random` gives us tools to get random numbers. We use it to place the food in a random spot on the screen.

PART 2: Initial Setup (Getting the game ready to start)

```
pygame.init()

WIDTH, HEIGHT = 800, 600
CELL_SIZE = 30
screen = pygame.display.set_mode((WIDTH, HEIGHT))
pygame.display.set_caption("Simple Snake Game")
```

🔍 What's going on here?

- `pygame.init()` is like flipping a switch to "turn on" Pygame so we can use it.
- `WIDTH` and `HEIGHT` set how big our game window is in pixels. (800 wide, 600 tall)

- `CELL_SIZE` tells us how big each part of the snake and the food should be.
- `screen = pygame.display.set_mode(...)` tells Pygame to create a window with the width and height we just picked.
- `pygame.display.set_caption(...)` sets the name of the window that pops up.

💡 Hint: A pixel is a tiny dot on the screen. The more pixels, the bigger the window.

PART 3: Defining Colors

```
WHITE = (255, 255, 255)
GREEN = (0, 200, 100)
RED = (255, 0, 0)
BLACK = (0, 0, 0)
```

🔍 What's going on here?

- These lines define the **colors** we'll use in our game.
- Each color is made using an **RGB value**: Red, Green, and Blue.
    - `(255, 0, 0)` means 255 red, 0 green, 0 blue → makes red.
    - `(0, 0, 0)` means no color → black.

💡 Hint: This is like mixing paint. High numbers make the color stronger.

PART 4: Game Tools – Clock and Font

```
clock = pygame.time.Clock()
font = pygame.font.SysFont(None, 36)
```

🔍 What's going on here?

- `clock` helps us control how fast the game updates (we'll use it later).
- `font` is how we'll show text (like the score or "Game Over") on the screen.
- `None` means we'll use the default font.
- `36` is the font size.

> 💡 Hint: Computers run VERY fast. If we didn't slow the game down with a clock, the snake would zoom off the screen instantly.

PART 5: Snake and Direction

```
snake = [pygame.Rect(100, 100, CELL_SIZE, CELL_SIZE)]
direction = pygame.K_RIGHT
```

🔍 What's going on here?

- We are creating the snake as a list with one starting block.
- `pygame.Rect(x, y, width, height)` creates a rectangle (which we'll use to draw the snake).
- The snake starts at position `(100, 100)` and is 30x30 pixels.
- `direction` tells us which way the snake is currently going. It starts going RIGHT.

> 💡 Hint: Each part of the snake is like a square box that we will move around.

PART 6: Making the First Food

```
food = pygame.Rect(
    random.randint(0, WIDTH // CELL_SIZE - 1) * CELL_SIZE,
    random.randint(0, HEIGHT // CELL_SIZE - 1) * CELL_SIZE,
    CELL_SIZE, CELL_SIZE
)
```

🔍 What's going on here?

- This line places the food somewhere **random** on the screen.
- `random.randint(...)` gives a random number between two values.
- We divide the screen size by the cell size to make sure the food fits nicely in a grid.
- We multiply by `CELL_SIZE` so the food lands exactly on one of the snake's cells.

  💡 Hint: Without this, food might land between squares and look weird.

PART 7: Starting the Score

```
score = 0
```

🔍 What's going on here?

- We create a variable called `score` to keep track of how many pieces of food the snake eats.
- We start the score at 0.

PART 8: Function to Move the Snake

```
def move_snake():
    head = snake[0].copy()
    if direction == pygame.K_LEFT:
        head.x -= CELL_SIZE
    elif direction == pygame.K_RIGHT:
        head.x += CELL_SIZE
    elif direction == pygame.K_UP:
        head.y -= CELL_SIZE
    elif direction == pygame.K_DOWN:
        head.y += CELL_SIZE
    return head
```

🔍 What's going on here?

- This **function** moves the snake's head in the correct direction.
- We use `.copy()` so we don't change the original head until we check for crashes.
- Then we move it left, right, up, or down depending on the current `direction`.

> 💡 Hint: `snake[0]` is always the head of the snake.

PART 9: Function for Game Over

```
def game_over():
    text = font.render("Game Over!", True, RED)
    screen.blit(text, (WIDTH // 2 - 80, HEIGHT // 2))
    pygame.display.update()
    pygame.time.wait(2000)
    pygame.quit()
    sys.exit()
```

🔍 What's going on here?

- This function ends the game and shows “Game Over!” on the screen.
- `font.render()` makes a picture of the text.
- `blit()` puts the picture on the screen.
- `pygame.quit()` shuts down Pygame.
- `sys.exit()` exits the whole program.

PART 10: The Main Game Loop

```
while True:
    screen.fill(WHITE)

    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            pygame.quit()
            sys.exit()

    keys = pygame.key.get_pressed()
    if keys[pygame.K_LEFT] and direction != pygame.K_RIGHT:
        direction = pygame.K_LEFT
    elif keys[pygame.K_RIGHT] and direction != pygame.K_LEFT:
        direction = pygame.K_RIGHT
    elif keys[pygame.K_UP] and direction != pygame.K_DOWN:
        direction = pygame.K_UP
    elif keys[pygame.K_DOWN] and direction != pygame.K_UP:
        direction = pygame.K_DOWN
```

🔍 What's going on here?

- `while True:` keeps the game running forever, until we quit.
- `screen.fill(WHITE)` clears the screen so we can draw new stuff.
- `pygame.event.get()` checks for things like clicking the X button to close the window.
- `pygame.key.get_pressed()` checks which keys are being pressed.
- We only let the snake turn left if it's not already going right, etc., to stop it from crashing into itself.

> 💡 Hint: This is like the game's "heartbeat." It checks for inputs, moves stuff, draws stuff, and repeats.

PART 11: Moving and Growing the Snake

```
   new_head = move_snake()

    if (new_head.left < 0 or new_head.right > WIDTH or
        new_head.top < 0 or new_head.bottom > HEIGHT or
        new_head.collidelist(snake) != -1):
        game_over()

    snake.insert(0, new_head)

    if new_head.colliderect(food):
        score += 1
        food.x = random.randint(0, WIDTH // CELL_SIZE - 1) * CELL_SIZE
        food.y = random.randint(0, HEIGHT // CELL_SIZE - 1) *
CELL_SIZE
    else:
        snake.pop()
```

🔍 What's going on here?

- We create the new head and move the snake.
- We check if the new head hit the wall or hit the snake’s own body (`collidelist` checks all parts).
- If yes, we call `game_over()`.
- `snake.insert(0, new_head)` adds the new head at the start of the list.
- If the snake eats the food (collision with food), we increase the score and move the food.
- If not, we remove the last part of the snake so it doesn’t grow.

💡 Hint: This is what makes the snake move like it's slithering.

PART 12: Drawing Everything on Screen

```
for segment in snake:
    pygame.draw.rect(screen, GREEN, segment)

pygame.draw.rect(screen, RED, food)

score_text = font.render(f"Score: {score}", True, BLACK)
screen.blit(score_text, (10, 10))

pygame.display.update()
clock.tick(10)
```

🔍 What's going on here?

- We draw each part of the snake using `pygame.draw.rect(...)`.
- Then we draw the food in red.
- We create a text image that shows the score and draw it on the screen.

- `pygame.display.update()` actually shows all the things we just drew.

- `clock.tick(10)` means the game will update 10 times per second. This slows things down to a playable speed.