

Task 1:

Question : implement the snake game where the player controls a snake that grows longer as it eats food. The game ends if the snake collides with itself or the edge of the screen (using python)

Answer:

```
import turtle
```

```
import time
```

```
import random
```

```
delay = 0.1
```

```
# Score
```

```
score = 0
```

```
high_score = 0
```

```
# Set up the screen
```

```
wn = turtle.Screen()
```

```
wn.title("Snake Game by @TokyoEdTech")
```

```
wn.bgcolor("green")
```

```
wn.setup(width=600, height=600)
```

```
wn.tracer(0) # Turns off the screen updates
```

```
# Snake head
```

```
head = turtle.Turtle()
```

```
head.speed(0)
```

```
head.shape("square")
```

```
head.color("black")
```

```
head.penup()
```

```
head.goto(0,0)
```

```
head.direction = "stop"
```

```
# Snake food
```

```
food = turtle.Turtle()
food.speed(0)
food.shape("circle")
food.color("red")
food.penup()
food.goto(0,100)
```

```
segments = []
```

```
# Pen
```

```
pen = turtle.Turtle()
pen.speed(0)
pen.shape("square")
pen.color("white")
pen.penup()
pen.hideturtle()
pen.goto(0, 260)
pen.write("Score: 0 High Score: 0", align="center", font=("Courier", 24, "normal"))
```

```
# Functions
```

```
def go_up():
```

```
    if head.direction != "down":
        head.direction = "up"
```

```
def go_down():
```

```
    if head.direction != "up":
        head.direction = "down"
```

```
def go_left():
```

```
    if head.direction != "right":
        head.direction = "left"
```

```
def go_right():
    if head.direction != "left":
        head.direction = "right"

def move():
    if head.direction == "up":
        y = head.ycor()
        head.sety(y + 20)

    if head.direction == "down":
        y = head.ycor()
        head.sety(y - 20)

    if head.direction == "left":
        x = head.xcor()
        head.setx(x - 20)

    if head.direction == "right":
        x = head.xcor()
        head.setx(x + 20)

# Keyboard bindings
wn.listen()
wn.onkeypress(go_up, "w")
wn.onkeypress(go_down, "s")
wn.onkeypress(go_left, "a")
wn.onkeypress(go_right, "d")

# Main game loop
while True:
    wn.update()
```

```
# Check for a collision with the border
```

```
if head.xcor()>290 or head.xcor()<-290 or head.ycor()>290 or head.ycor()<-290:
```

```
    time.sleep(1)
```

```
    head.goto(0,0)
```

```
    head.direction = "stop"
```

```
# Hide the segments
```

```
for segment in segments:
```

```
    segment.goto(1000, 1000)
```

```
# Clear the segments list
```

```
segments.clear()
```

```
# Reset the score
```

```
score = 0
```

```
# Reset the delay
```

```
delay = 0.1
```

```
pen.clear()
```

```
pen.write("Score: {} High Score: {}".format(score, high_score), align="center", font=("Courier", 24, "normal"))
```

```
# Check for a collision with the food
```

```
if head.distance(food) < 20:
```

```
    # Move the food to a random spot
```

```
    x = random.randint(-290, 290)
```

```
    y = random.randint(-290, 290)
```

```
    food.goto(x,y)
```

```
# Add a segment
```

```
new_segment = turtle.Turtle()
```

```
new_segment.speed(0)
new_segment.shape("square")
new_segment.color("grey")
new_segment.penup()
segments.append(new_segment)
```

```
# Shorten the delay
delay -= 0.001
```

```
# Increase the score
score += 10
```

```
if score > high_score:
    high_score = score
```

```
pen.clear()
pen.write("Score: {} High Score: {}".format(score, high_score), align="center", font=("Courier", 24, "normal"))
```

```
# Move the end segments first in reverse order
```

```
for index in range(len(segments)-1, 0, -1):
    x = segments[index-1].xcor()
    y = segments[index-1].ycor()
    segments[index].goto(x, y)
```

```
# Move segment 0 to where the head is
```

```
if len(segments) > 0:
    x = head.xcor()
    y = head.ycor()
    segments[0].goto(x,y)
```

```
move()
```

```
# Check for head collision with the body segments
for segment in segments:
    if segment.distance(head) < 20:
        time.sleep(1)
        head.goto(0,0)
        head.direction = "stop"

# Hide the segments
for segment in segments:
    segment.goto(1000, 1000)

# Clear the segments list
segments.clear()

# Reset the score
score = 0

# Reset the delay
delay = 0.1

# Update the score display
pen.clear()
pen.write("Score: {} High Score: {}".format(score, high_score), align="center", font=("Courier", 24, "normal"))
time.sleep(delay)
wn.mainloop()
```

OUTPUT:

Score : 25

