sandworm_turtle.py

```
import turtle
import time
import random
delay = 0.1

# Scores
score = 0
high_score = 0

# Screen setup
wn = turtle.Screen()
wn.title("The Sandworm on Mars")
wn.bgcolor("tan")
wn.setup(width=600, height=600)
wn.tracer(0) # Turns off the screen updates ///

# Snake head
head = turtle.Turtle()
head.speed(0)
head.shape("triangle")
head.color("black")
head.left(90)
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 = []

# Font and Writing
pen = turtle.Turtle()
pen.speed(0)
pen.shape("square")
pen.color("Forest Green")
pen.penup()
pen.hideturtle()
pen.goto(0, 260)
pen.write("Current Score: 0  High Score: 0", align="center", font=("Courier New", 25, "bold"))


# Direction of the snake
def go_up():
    if head.direction != "down":
        if head.direction == "right":
            head.left(90)
        elif head.direction == "left":
            head.right(90)
        head.direction = "up"


def go_down():
    if head.direction != "up":
        if head.direction == "right":
            head.right(90)
        elif head.direction =="left":
            head.left(90)
        head.direction = "down"


def go_left():
    if head.direction != "right":
        if head.direction == "up":
            head.left(90)
        else:
            head.right(90)
        head.direction = "left"


def go_right():
    if head.direction != "left":
        if head.direction == "down":
            head.left(90)
        else:
            head.right(90)
        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 buttons
wn.listen()
wn.onkeypress(go_up, "Up")
wn.onkeypress(go_down, "Down")
wn.onkeypress(go_left, "Left")
wn.onkeypress(go_right, "Right")

colours = ["white", "black", "white", "black"]
cou = 0

#Main Loop
```

```
while True:
    wn.update()

    # 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)

        # Reset
        segments.clear()
        score = 0
        delay = 0.1

        pen.clear()
        pen.write("Current Score: {}  High Score: {}".format(score, high_score), align="center", font=("Courier New", 25, "bold"))

        # Food collision
    if head.distance(food) < 20:
        # Food movement
        x = random.randint(-290, 290)
        y = random.randint(-290, 290)
        food.goto(x, y)

        # Addings segments/sqaures
        new_segment = turtle.Turtle()
        new_segment.speed(0)
        new_segment.shape("square")
        #Color Switch Effect
        new_segment.color(colours[cou])
        if cou == 3:
            cou = 0
        else:
            cou = cou + 1
        new_segment.penup()
        segments.append(new_segment)


        # To clear 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 New", 25, "bold"))

        # 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()

    # Stops game once head collides with body
    for segment in segments:
        if segment.distance(head) < 20:
            time.sleep(1)
            head.goto(0, 0)
            head.direction = "stop"
            #Added to hide segments
            for segment in segments:
                segment.goto(1000, 1000)

            # Reset segments, score and delay
            segments.clear()
            score = 0
            delay = 0.1

            # Reset the score display
            pen.clear()
            pen.write("Score: {}  High Score: {}".format(score, high_score), align="center",
                      font=("Courier New", 25, "bold"))

    time.sleep(delay)

wn.mainloop() #//
```