COVER PAGE

**Disclaimer:**

In this mordern era, Science and technolegies are indispensable factors , More over coding has become an invavitable factors for transcending industries and disciplines.By understanting the various, Control structures, functions and Data type student will gain better understanting amoung the words of development.

## A diagram of a computer Description automatically generated**Code contents**

else:  
 if len(self.food) < 5:  
 self.food.append(self.generate\_food())  
 self.window.after(100, self.update)

def bind\_keys(self):  
 self.window.bind("<Up>", self.up\_key)  
 self.window.bind("<Down>", self.down\_key)  
 self.window.bind("<Left>", self.left\_key)  
 self.window.bind("<Right>", self.right\_key)

def draw\_food(self):  
 self.canvas.delete("food")  
 for x, y in self.food:

Basics of Coding

Gravitational Wave Simulator

# Introduction

The Snake game,  provides a fun and engaging way to learn fundamental programming concepts. Through this assignment, I build a snake game in Basic, focusing on core skills like loops, function,if-else conditional statements by basic proto-typic representation. By implementing these elements, I am not only create a playable game but also gain a solid understanding of programming fundamentals.

# Project concept

This project aims to develop a classic Snake game using Pycharm . In the game, the player controls a snake that moves around the screen, trying to eat feed. The snake grows longer with each Red dots while eaten. The game will ends at the time when the snake hits the boarders by it-self or by gamer mistake The player will interfaces the basic of games that results the a moving snake in ‘Green colour ‘ it can freely move vertically and horizontally eqaully, In addition to that our game also displays food for snake as in the format of ‘RED’ dots .Also it have point system which scores ‘one’ point for one food.

More over with a ‘white’ background. Further, if the gamer fail by hitting the wall it will show ‘RESTART’ option in ‘RED’

For the better results , The program will handle user input for direction changes, updates the position of the snake to avoids the bugs and not to collide each other.

By completing this project I am able to understant much more about the progarmmings and codes including user input, game loop logic,and functions .

# Design concept

This design is very simple and confortable to the user , It also have pleasant background and RGB colour to maintain good emotion.

Figure 1 - UML Diagram of my project

# Technical implementation

The design concept represent concept ideas to develop the with pygame

import tkinter as tk  
import random

This code imports two libraries commonly used in Python for building Graphical User Interfaces (GUIs) and working with random numbers:

self.canvas = tk.Canvas(self.window, width=500, height=500, bg="white")  
self.canvas.pack()

This code is used to genarate the canvas with size of game with width and height and background colour.

self.snake = [(250, 250), (240, 250), (230, 250)]  
self.food = [self.generate\_food() for \_ in range(5)]  
self.direction = "Right"  
self.score = 0

Tuples use parentheses () to group values.And Inside the loop, self.generate\_food() is called 5 times, and the resulting positions (tuples) are added to the self.foodlist. And also it can track the direction of snake.

x = random.randint(0, 49) \* 10  
y = random.randint(0, 49) \* 10

x represents a random horizontal position within a grid of 50 cells (0 to 49).

y represents a random vertical position within the same grid.

Each cell has a width and height of 10 units (due to the multiplication by 10).

if (self.snake[0][0] < 0 or self.snake[0][0] > 490 or  
 self.snake[0][1] < 0 or self.snake[0][1] > 490 or  
 self.snake[0] in self.snake[1:]):  
 self.game\_over()

This codes demostrate the cammand that to avoid wall collition and self collition . This code ensures the game ends when the snake collides with either the walls or its body. These are very common in snake games. In addition it is also an if-else statement.

def restart\_game(self):  
 self.canvas.delete("all")  
 self.snake = [(250, 250), (240, 250), (230, 250)]  
 self.food = [self.generate\_food() for \_ in range(5)]  
 self.direction = "Right"  
 self.score = 0  
 self.score\_label.config(text="Score: 0")  
 self.draw\_snake()  
 self.draw\_food()  
 self.update()

This is the final part , the restart\_game function provides a chance for the player to replay the game after fails . It always resets all game elements to their ace state and resume the game loop, offering a noval start .

if \_\_name\_\_ == "\_\_main\_\_":  
 game = SnakeGame()

# In glance , this code will ensures Snake game begin as properly when you run the script directly (e.g., by double-clicking the .py file).

# Github link

<https://github.com/wmbm/organigramm_tracker>

**MAKE SURE IT’S A PUBLIC REPOSITORY**

1. Sign up for a Github account
2. Create a repository for your project
3. Upload your code to the repository
4. Create/update a simple README.md file in your repository to describe your project

# Conclusion

The code of the Snake was extremely difficult with many errors are pop uping . Many systems had to be written numerous ways before a final working solution was found. however, even the final version as vertical movement causes the snake to change dirction . There were also issues with the food – snake collision detection. While the final version resulted in a snake that could eat food, the movement glitch caused the food to cause further size , More over I used diferent while loops, If\_else , and function to result the better concept ideas to conquer to maintain beautifull emotions for the users there by there can ease the difficult situation in their daily life.