Introduction to Computing & Programming

C-CS111

***Fall 2023***

***[snake and ladder]***

Team Members

|  |  |
| --- | --- |
| Student Name | Student ID |
| Omar Mohamed | 23-101149 |
| Maroska Osama | 23-101209 |
| Youssef Ihab | 23-101138 |
| Hamdy El Saeed | 23-101232 |

Table of contents

1. Abstract
2. Introduction
3. Motivation
4. System design

* System modules and functions
* Hierarchy chart

1. Description
2. Team’s workload distribution
3. User manual
4. Testing and validation
5. Challenges and conclusion

References

The abstract:

This report outlines how a computer software system was designed and developed using the programming language Python. Our program provides two modes for the user the first mode is player vs. another player and the other mode is player vs. computer. In addition, our program has two modes the easy and hard one, which provides a better experience for the user. Also, there are many extra features included in our game such as the drop-down menu, which includes four buttons to exit, replay, start a new game, and even save the game; also in our program, there is a progress bar, that keeps track of the users to show their advancements that make the user very excited to finish the game by reaching the 100th square on the board or even when they climb a ladder the progress bar will be filled with colour, whereas when the player steps on a snake the progress bar will be emptied.

The introduction:

Snake and Ladder is a classic board game that traces its origins back to ancient India. Also known as Moksha Patam, the game has transcended cultures and time, becoming a beloved pastime worldwide. It combines luck and strategy as players navigate through a numbered grid, aiming to reach the final square while facing the twists of climbing ladders and the setbacks of descending snakes. The game's simplicity and element of chance make it a timeless source of entertainment for players of all ages.

What made us choose the snake and ladder?

1. it reminds us of our childhood as no child doesn’t play snake and ladder in this world. Secondly, it has many benefits for the children such as:
2. Emotion regulation: Children learn to control their emotions through experiencing the highs and lows of sliding their game piece down the back of the snake, which moves them further away from the goal, or the thrill of climbing the ladder, which moves them past another player and closer to the goal.
3. Losing: It's critical that kids learn that nobody ever wins all the time. Young children are unable to express their emotions. When things don't go their way, talking about how losing makes them feel helps them deal with their feelings.
4. Interact with others: Speaking and listening are both necessary for understanding the game's rules.
5. Observing the rules: Snakes & Ladders has few rules, which makes it simple to learn how to play. However, by adhering to the rules when their character falls off the snake, kids can learn that even though things don't always go as planned, they can still advance and even win.

the hierarchy chart:

A diagram of a diagram

Description automatically generated

The functions used in our program:

Def start\_new\_game(): this function allows the user to create a new game and start the game.

Def show\_custom\_message(): this function

Def Coin\_movement(): this function manages the coin's movement through the board.

Def diceimg(): this function makes the dice image in the board.

Def input\_player(): this function allows the user to enter the player’s name.

Def resolve\_path(file): this function allows the program to search for a specific file inside the folder.

Def welcome\_message(): this function prints a welcome message for the user.

Def switch\_turns(): this function allows the user to swap the turns.

Def optionmenu\_callback(): this function allows the user to choose some options from the drop-down menu.

Def rules(): it allows the user to read the instructions of the game.

Def player\_turn(): it allows the user to play its turn.

Def button\_window(): it allows the user to create a new window.

Def rematch(): this function allows the user to play with the same player again.

The description of our code contains the explanation when:

1. 1 player vs. 1 player (easy and hard)
2. 1 player vs. pc (easy and hard)
3. The main program

The description of 2 players' mode:

|  |  |
| --- | --- |
| The code of importing the libraries: | The explanation: |
| from customtkinter import \*  from tkinter import \*  from CTkMessagebox import CTkMessagebox  from PIL import Image  import time  import random  import os | The code begins by importing necessary modules:  customtkinter: A custom implementation of Tkinter.  tkinter: The standard Python interface to the Tk GUI toolkit.  CTkMessagebox: A custom message box for the game.  PIL: The Python Imaging Library, used for image processing.  time: Provides various time-related functions.  random: Provides functions to generate random numbers.  os: Provides a way of interacting with the operating system |

|  |  |
| --- | --- |
| The code of defining the global variables: | The explanation: |
| State1 = NORMAL  State2 = NORMAL  turn1 = 0  turn2 = 0  Counter1 = 0  Counter2 = 0 | Global variables are initialized to manage the game state, player turns, and counters for each player. |

|  |  |
| --- | --- |
| The code for Tkinter window initialization: | The explanation: |
| root = CTk()  root.title("Snake and Ladder game")  root.iconbitmap(resolve\_path("\Icon\SnakeLadder.ico"))  root.geometry("1200x800") | This part of the code creates using Tkinter the title, the dimensions and also the icon of the game. |

|  |  |
| --- | --- |
| The code for displaying photos: | The explanation : |
| Background = CTkImage(light\_image=Image.open(resolve\_path("/Backgrounds/Background1.png")), size=(1370, 700))  Lab11 = CTkLabel(root, image=Background, text=None, width=1370, height=700).place(x=0, y=0)  GameBoard = CTkImage(light\_image=Image.open(resolve\_path("/Board/GameBoardE.png")), size=(600, 600))  Lab = CTkLabel(root, image=GameBoard, text=None, width=600, height=600).place(x=100, y=50) | This part of the code explains setting up the background of the game and also the game board to be displayed in the window of Tkinter with its size and co-ordinates |

|  |  |
| --- | --- |
| The code for input user name: | The explanation: |
| msg1 = CTkInputDialog(title="Player 1 name", text="Enter the name of the first player")  Player1 = msg1.get\_input()  msg2 = CTkInputDialog(title="Player 2 name", text="Enter the name of the second player")  Player2 = msg2.get\_input() | This part of the code allows the two players to input their names. |

|  |  |
| --- | --- |
| The code for displaying names and the scores: | The explanation: |
| Player1Name = CTkLabel(root, text=Player1, font=("Arial", 20), fg\_color="orange", text\_color="white", anchor="center", width=100, height=50).place(x=800, y=100)  Player2Name = CTkLabel(root, text=Player2, font=("Arial", 20), fg\_color="black", text\_color="white", anchor="center", width=100, height=50).place(x=1100, y=100)  Player1Score = CTkLabel(root, text=Counter1, font=("Arial", 20), fg\_color="orange", text\_color="white", width=100, height=50).place(x=800, y=200)  Player2Score = CTkLabel(root, text=Counter2, font=("Arial", 20), fg\_color="black", text\_color="white", width=100, height=50).place(x=1100, y=200) | This part of the code aims to display the names of the players and the scores that they get. |

|  |  |
| --- | --- |
| The code for the progress bar: | The explanation: |
| progressbar1 = CTkProgressBar(root, orientation="horizontal", width=100, height=12, fg\_color="white", progress\_color="orange", border\_width=2).place(x=800, y=300)  progressbar2 = CTkProgressBar(root, orientation="horizontal", width=100, height=12, fg\_color="white", progress\_color="black", border\_width=2).place(x=1100, y=300) | The target of that code is to create the progress bar in the Tkinter window that keeps track of the player's movement. |

|  |  |
| --- | --- |
| The code for the game loop and the coin movements: | The explanation: |
| while coin1\_position < 100 and coin2\_position < 100:  # Game logic for player 1  if turn1 == 1 and coin1\_position < 100:  coin1\_position = coin1\_position + Position\_Diff  x1 = PositionDict[coin1\_position][0]  y1 = PositionDict[coin1\_position][1]  player1\_coin.place(x=x1, y=y1)  # Game logic for player 2  elif turn2 == 1 and coin2\_position < 100:  coin2\_position = coin2\_position - Position\_Diff  x2 = PositionDict[coin2\_position][0]  y2 = PositionDict[coin2\_position][1]  player2\_coin.place(x=x2, y=y2)  # Updating progress bars  progressbar1.set(coin1\_position / 100)  progressbar2.set(coin2\_position / 100) | The main game loop where the coins of Player 1 and Player 2 move based on their turns and positions. Progress bars are updated accordingly to the numbers that they get. |

|  |  |
| --- | --- |
| The code of the actions after coin movement: | The Explanation: |
| if coin1\_position == 100 or coin2\_position == 100:  EndMessage = CTkMessagebox(title='Game Over!', text=f'{winner} wins!', image=resolve\_path("/Icon/YouWon.ico"),  icon\_size=(200, 250), bg\_color="white", fg\_color="white", width=300, height=250,  option\_1='Exit', option\_2='New Game', option\_3='Rematch')  response = EndMessage.get()  if response == 'Rematch':  rematch()  elif response == 'New Game':  start\_new\_game()  elif response == 'Exit':  root.destroy() | The aim of that part is to display a message box when one of the players reaches position 100, then the player will be asked to choose a rematch, start a new game, or exit. |

|  |  |
| --- | --- |
| The code for the main loop: | The explanation: |
| root.mainloop() | This function allows the main loop in the game to start and for the user to srart playing the game. |

|  |  |
| --- | --- |
| The difference between the easy and the hard mode | |
| Easy : | Hard: |
| The dictionaries of the snakes and ladders :  snakes = {17:13, 52:29,77:58,62:22,57:40,88:18,95:51,97:79,}  ladders = {3:21,8:30,28:84,58:77,75:86,80:100,90:91, }  easier game board | snakes = {68:2,59:18, 46:15,98:13,52:11,44:22,64:24,69:33,  95:37,83:39,  87:50,92:51,94:71,48:9 }  ladders = {5:23, 62:96,43:77,8:26,10:49,  19:38,28:53,21:61,35:47,47:76,  36:57,54:88,67:86,70:91 80:99 }  Harder game board |

The description of 1 player vs. pc

|  |  |
| --- | --- |
| The code of importing the libraries: | The explanation: |
| from customtkinter import \*  from tkinter import \*  from CTkMessagebox import CTkMessagebox  from PIL import Image  import time  import random  import os | The code begins by importing the necessary modules:  customtkinter: A custom implementation of Tkinter.  tkinter: The standard Python interface to the Tk GUI toolkit.  CTkMessagebox: A custom message box for the game.  PIL: The Python Imaging Library, used for image processing.  time: Provides various time-related functions.  random: Provides functions to generate random numbers.  os: Provides a way of interacting with the operating system |

|  |  |
| --- | --- |
| The code of the global variables: | The explanation: |
| turn1 = 0  turn2 = 0  sixcount1 = 0 | The target of that part of the code is to keep track of the turns and to count the number of consecutive 6 for player1 |

|  |  |
| --- | --- |
| The code of the gui initializing: | The explanation: |
| State1 = NORMAL  set\_window\_scaling(1.5)  root = CTk()  # ... (Window configurations, player name input, background and game board initialization) | The purpose of it initialize the Tkinter window and set up various configurations such as window scaling, title, icon, and dimensions |

|  |  |
| --- | --- |
| The code for the esc button: | The explanation: |
| root.bind('<Escape>', lambda event: button\_window(event)) | This part describes how did we bind the escape of the button to the button window. |

|  |  |
| --- | --- |
| The code for the main loop: | The explanation: |
| root.mainloop() | This function allows the main loop in the game to start and for the user to srart playing the game. |

|  |  |
| --- | --- |
| The difference between the easy and the hard mode | |
| Easy : | Hard: |
| The dictionaries of the Snakes and ladders :  snakes = {17:13, 52:29,77:58,62:22,57:40,88:18,95:51,97:79,}  ladders = {3:21,8:30,28:84,58:77,75:86,80:100,90:91, }  easier game board | snakes = {68:2,59:18, 46:15,98:13,52:11,44:22,64:24,69:33,  95:37,83:39,  87:50,92:51,94:71,48:9 }  ladders = {5:23, 62:96,43:77,8:26,10:49,  19:38,28:53,21:61,35:47,47:76,  36:57,54:88,67:86,70:91 80:99 }  Harder game board |

The main program:

|  |  |
| --- | --- |
| The code for importing the statements: | The explanation: |
| from customtkinter import \*  from subprocess import call  import os  from CTkMessagebox import CTkMessagebox | This part of the code imports the libraries of the Gui |

|  |  |
| --- | --- |
| The code for initializing the gui: | The explanation: |
| app = CTk()  app.geometry("1200x800")  app.title("Snake and Ladder game")  app.iconbitmap(resolve\_path("\\Icon\\SnakeLadder.ico"))  app.attributes('-fullscreen', True) | This part of the code targets to initialize the Tkinter window, the size of the window, and the title beside the fullscreen mode. |

|  |  |
| --- | --- |
| The code for some buttons: | The explanation: |
| play\_button = CTkButton(app, text="Play Game", font=("Arial", 20), fg\_color=("green", "lime"), bg\_color=("white", "white"), corner\_radius=10, command=play)  play\_button.place(relx=0.5, rely=0.3, anchor=CENTER)  about\_button = CTkButton(app, text="About Us", font=("Arial", 20), fg\_color=("blue", "dodgerblue"), bg\_color=("white", "white"), corner\_radius=10, command=about)  about\_button.place(relx=0.5, rely=0.5, anchor=CENTER)  exit\_button = CTkButton(app, text="Exit", font=("Arial", 20), fg\_color=("red", "tomato"), bg\_color=("white", "white"), corner\_radius=10, command=exit)  exit\_button.place(relx=0.5, rely=0.7, anchor=CENTER) | That part of the code aims to create a start the game button, about us bottom, and the exit button. With their setting its dimensions and the colour of each button. |

The workload distribution:

|  |  |
| --- | --- |
| Omar Mohamed: | Youssef Ihab: |
| Coin movement  Progress bars  Message boxes | Saving and loading games  Dice functions  Player turns |
| Hamdy El Saeed: | Maroska Osama: |
| Buttons  Drop menu with different functions  Position dictionary | Snakes and ladders  Design and user interface  User inputs |

The user manual:

There are many extra features the program provides for the user that creates a better experience for him such as:

* The drop-down menu.
* Different game modes: (player vs. player) and (player vs. pc).
* Different difficulty levels: (easy and hard).
* Background in the game.
* Progress bar.
* Output massage with stickers with the random function when climbing the ladder and when sliding down the snake.
* The player can’t roll the dice twice except if he gets 6.
* If the player rolls the dice and gets six, he will be asked to roll it again, and if he gets six the third time he won’t play.
* At any time, the user can exit the game, replay, or even start a new game.

How to use our system?

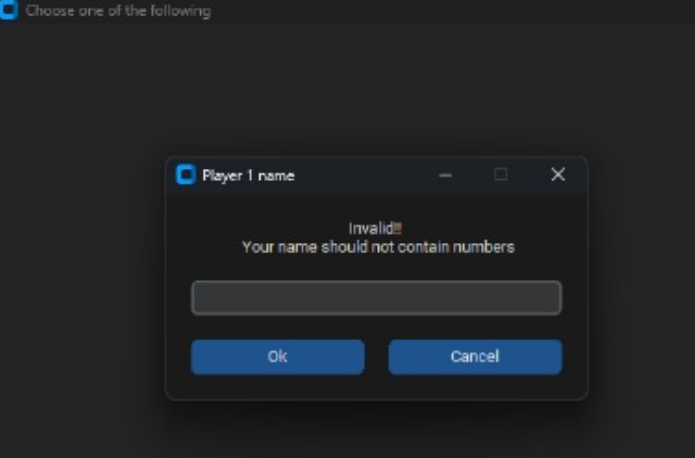
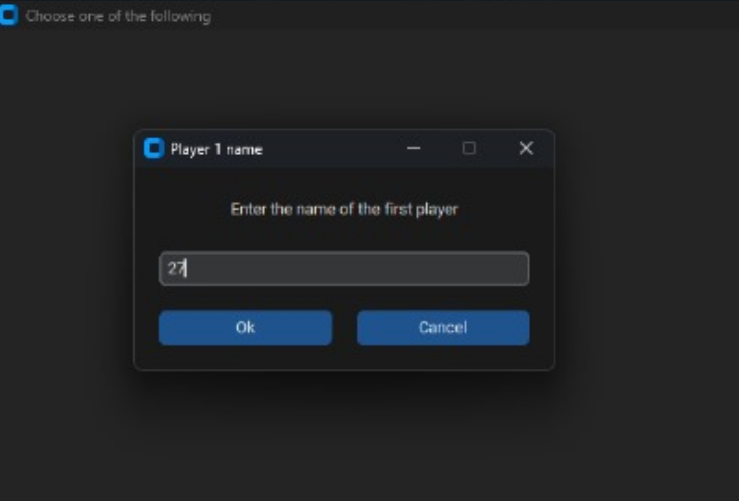
* 1. After selecting a piece of equipment, each player positions it on the initial square.
* 2. Roll a die to see who goes first. Go-first is the player with the highest rolling number.
* 3. The player advances their piece to the spaces indicated by rolling the dice during their turn.
* 4. If a player touches down on a square that has a ladder at its base, their piece ascends to the top of the ladder.
* 5. A player's piece descends to the snake's base if they land on a square with a snake at its head.
* 6. When a piece of a different player lands on a square that another player has already taken up, that player's piece is knocked back to the beginning square.
* 7. The winner of the game is the first person to reach the 100th tile.
* 8. A player's piece stays in its current location if they roll a number that would move it past the 100th square.
* 9. The piece that was moved last stays on the square if a player lands on a spot already occupied by that player's other piece.

There are some things that the user should use to avoid the error such as:

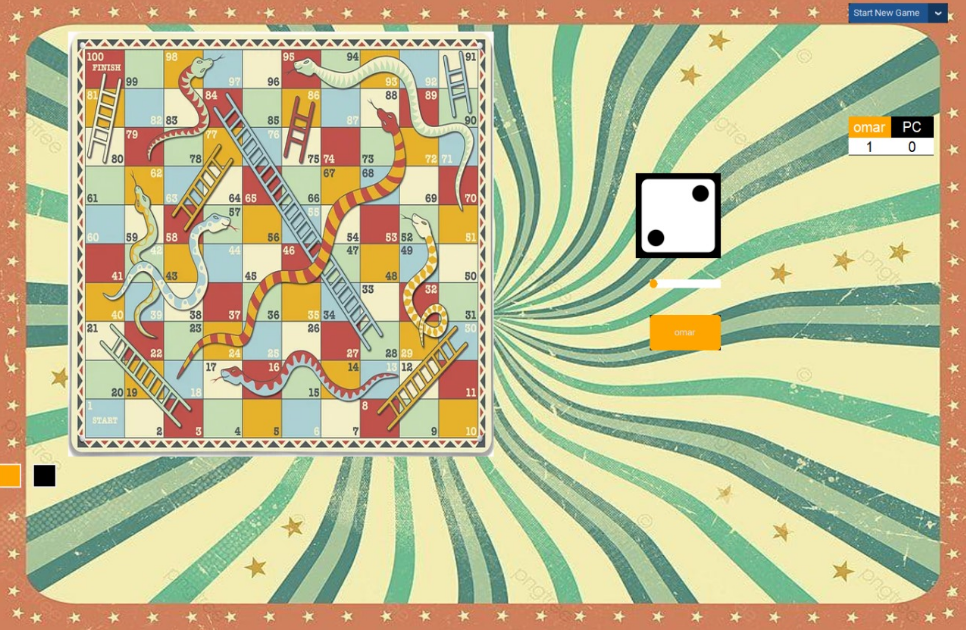
When inputting the name if the user enters an integer an invalid message will be displayed and the user will be asked to enter his name and when finally the player enter the name in letters, the game will proceed .

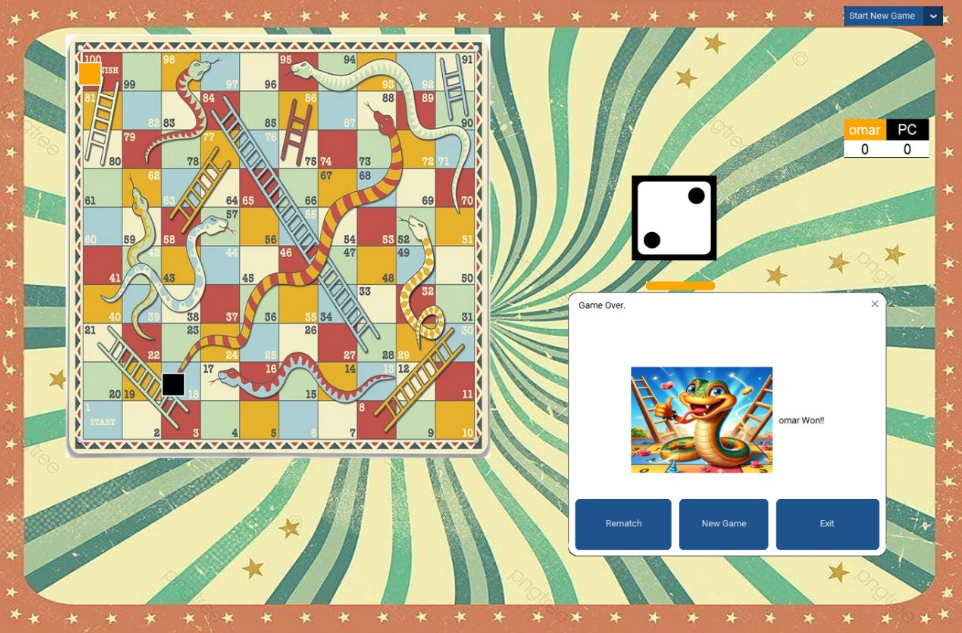
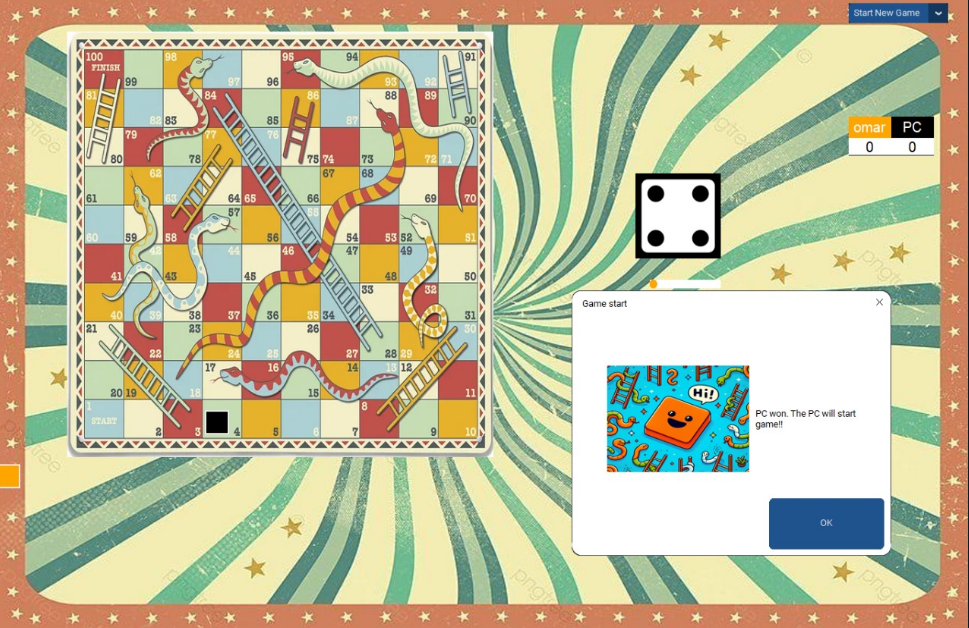
Testing and validation:

If the name input anything but characters it shows invalid input and prompts the user for a valid input.



A functioning user-friendly game with message boxes when climbing a ladder, getting bitten by a snake, or winning, allowing the user to know how the game is progressing.





The conclusion:

Our system is very successful, as it contains every required rule and function with no errors or bugs. It’s simple and user-friendly allowing everyone to enjoy playing our snake and ladder game with different modes and difficulty levels.

From that project, we have gained new skills such as:

Self-learning and depending on ourselves to solve our problems, teamwork, time management, communication, and most importantly working under stress while being burnt out.

Some of the challenges we have faced:

Not finding enough reliable sources for our project and it was difficult and took plenty of time to fix the bugs and error.

The references:

<https://www.bing.com/videos/riverview/relatedvideo?&q=python+snake+and+ladder+game&&mid=3FACE7964B21134480D33FACE7964B21134480D3&&FORM=VRDGAR>

stack over flow:

<https://stackoverflow.com/>

[snake and ladder game using python tamil | paramapatham tamil | Python (youtube.com)](https://www.youtube.com/watch?v=-IR4VFEr-oE)

Create gui python and tkinter:

[Create Graphical User Interfaces With Python And TKinter (youtube.com)](https://www.youtube.com/watch?v=yQSEXcf6s2I&list=PLCC34OHNcOtoC6GglhF3ncJ5rLwQrLGnV&index=1)

Creating buttons with tkinter:

[Creating Buttons With Tkinter - Python Tkinter GUI Tutorial #3 (youtube.com)](https://www.youtube.com/watch?v=yuuDJ3-EdNQ&list=PLCC34OHNcOtoC6GglhF3ncJ5rLwQrLGnV&index=3)

Using Icons, Images, and Exit Buttons:

[Using Icons, Images, and Exit Buttons - Python Tkinter GUI Tutorial #8 (youtube.com)](https://www.youtube.com/watch?v=NoTM8JciWaQ&list=PLCC34OHNcOtoC6GglhF3ncJ5rLwQrLGnV&index=8)

Creating message boxing by tkinter:

[Message Boxes with TKinter - Python Tkinter GUI Tutorial #13 (youtube.com)](https://www.youtube.com/watch?v=S3AaSwpb5GE&list=PLCC34OHNcOtoC6GglhF3ncJ5rLwQrLGnV&index=13)

Creating new windows using tkinter:

[Create New Windows in tKinter - Python Tkinter GUI Tutorial #14 (youtube.com)](https://www.youtube.com/watch?v=qC3FYdpJI5Y&list=PLCC34OHNcOtoC6GglhF3ncJ5rLwQrLGnV&index=14)

Creating the drop-down menu:

[Dropdown Menus With TKinter - Python Tkinter GUI Tutorial #18 (youtube.com)](https://www.youtube.com/watch?v=3E_fK5hCUnI&list=PLCC34OHNcOtoC6GglhF3ncJ5rLwQrLGnV&index=18)