

CSC 1024 PROGRAMMING PRINCIPLES

PROGRAMMING PROJECT: MASTER MIND GAME

LEE JIA QIAN

19117613

14th July 2021

Presentation Video Web Link:

<https://youtu.be/IdHpuJf76OM>

```
mirror_mod = modifier_ob.  
set mirror object to mirror.  
mirror_mod.mirror_object  
operation == "MIRROR_X":  
mirror_mod.use_x = True  
mirror_mod.use_y = False  
mirror_mod.use_z = False  
operation == "MIRROR_Y":  
mirror_mod.use_x = False  
mirror_mod.use_y = True  
mirror_mod.use_z = False  
operation == "MIRROR_Z":  
mirror_mod.use_x = False  
mirror_mod.use_y = False  
mirror_mod.use_z = True  
  
selection at the end -add  
mirror_ob.select= 1  
modifier_ob.select=1  
context.scene.objects.active  
("Selected" + str(modifier_ob.  
mirror_ob.select = 0  
= bpy.context.selected_object  
data.objects[one.name].select  
  
print("please select exactly  
  
-- OPERATOR CLASSES --  
  
types.Operator):  
X mirror to the selected  
object.mirror_mirror_x"  
mirror X"  
  
context):  
context.active_object is not
```

Introduction

About

The following programming project is the final assessment for CSC1024

Objective

To create a MASTER MIND board game using Python programming language

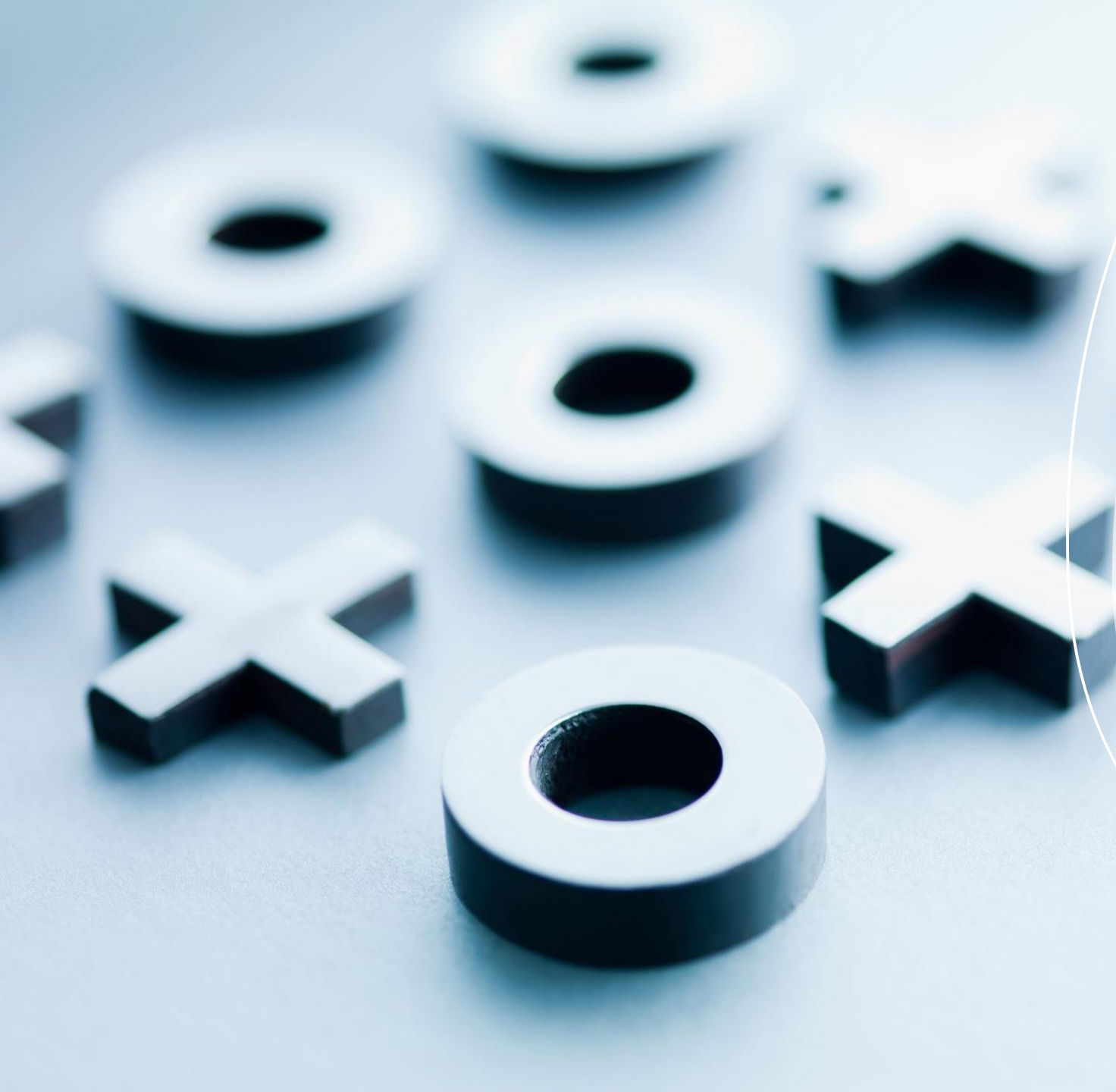
What is the MASTER MIND GAME?

MASTER MIND is a board game that a person must decipher a four colours code in the correct order

There are two players which one is a codemaster (who make the code) and the guesser (who decipher the code)

The player wins when the player successfully guessed the exact colour code set by the codemaster within given attempts





What is the different from the traditional MASTER MIND board game?

There are a total of 6 colours

The computer will randomly
generate 4 colours code
instead of a real player

A Mouse and keyboard are
used to play the game



Flowchart

A flowchart is a graphic representation of a procedure, system or computer algorithm

They are extensively utilized in a variety of areas to documents, analyse, plan, develop and explain often complicated processes in clear and simple diagrams.

Rectangles, ovals, diamonds and perhaps other forms are used to indicate the kind of step in flowcharts, along with connecting arrows to describe flow and sequence.

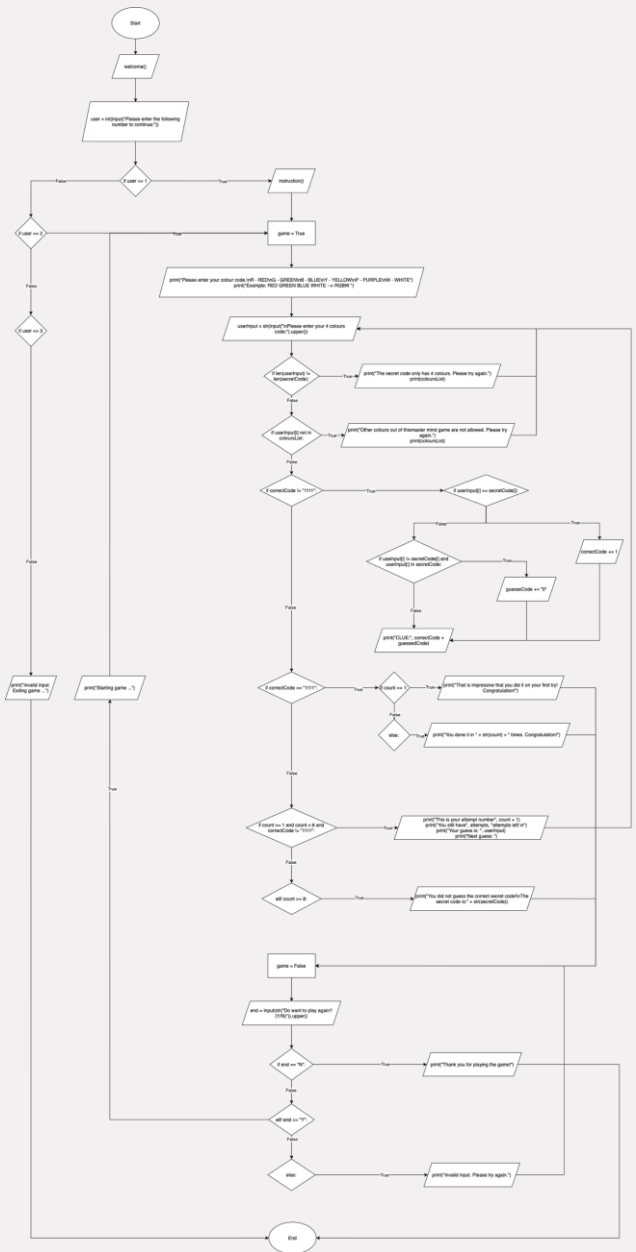


Figure 1: Flowchart for current MASTER MIND program

Interactive User Menu

Using user input command, assist the users in navigating the software and allowing them to make decision with the program.

```
3  # Welcome menu
4  def welcome():
5      print()
6      print(" ----- ")
7      print("          MASTER MIND GAME ")
8      print(" ----- ")
9      print("Welcome to the MASTER MIND GAME")
10     print("[1] Instruction")
11     print("[2] Start game")
12     print("[3] Exit game")
13
14  # Instruction menu
15  def instruction():
16      print()
17      print("Computer will randomly choose 4 colours from the list and arrange in a order.\n")
18      + "There are a total of 6 colours to choose from: (RED - R, GREEN - G, BLUE - B, YELLOW - Y, PURPLE - P, WHITE - W)\n"
19      + "You must guess the 4 correct colours in the correct order to win the game.\n"
20      + "After entering the colour code, computer will show player the clue:\n"
21      + "0 shown means colour is correct with incorrect position.\n"
22      + "1 shown means colour is in the correct position.\n"
23      + "You are given 8 attempts only.\n"
24      + "ENJOY! Goodluck out there.")
25
26  # Welcome selection menu
```

Figure 2: Welcome and instruction menus

```

26 # Welcome selection menu
27 welcome()
28 user = int(input("Please enter the following number to continue: "))
29 if user == 1:
30     instruction()
31     game = True
32 elif user == 2:
33     game = True
34 elif user == 3:
35     print("Exiting game ...")
36     game = False
37 else:
38     print("Invalid input. Exiting game...")
39     game = False
40

```

Figure 3: User input selection on welcome menu

```

49
50 # Game
51 while game == True:
52     # Initialize
53     correctCode = ""
54     guessedCode = ""
55     print()
56     print("Please enter your colour code.\nR - RED\nG - GREEN\nB - BLUE\nY - YELLOW\nP - PURPLE\nW - WHITE")
57     print("Example: RED GREEN BLUE WHITE --> RGBW ")
58     # User input colours code
59     userInput = str(input("\nPlease enter your 4 colours code: ").upper())
60

```

Figure 4: User input for guessing colour code

```
103 # Replay or end the game
104 while game == False:
105     end = input(str("Do want to play again? (Y/N)")).upper()
106     count = 0
107     if end == "N":
108         print("Thank you for playing the game!")
109         break
110     elif end == "Y":
111         game = True
112         print()
113         print("Starting game... ")
114         print()
115     else:
116         print("Invalid input. Please try again.")
117         game == False
118
```

Figure 5: User input selection to replay game or end the game

All of the above code is intended to prevent users from entering incorrect commands into the application, which would result in problems.

Lists

A list is a container that is both ordered and changeable. It is one of the most popular data structures.

List can include items of various kinds as shown in figure 6 as well as duplicated components.

```
40
41 # Colors list
42 coloursList = ["R", "G", "B", "Y", "P", "W"]
43 # Secret code
44 secretCode = random.sample(coloursList, 4)
45 print(secretCode)
46 # Initialize
```

Figure 6: List for colours used in this program

Randomized Values

The only Python module imported in this program is the “random” module.

The random module is only used once, but it is essential to the program since it generates a random colour code when the program is restarted or terminated.

```
41 # Colors list
42 coloursList = ["R", "G", "B", "Y", "P", "W"]
43 # Secret code
44 secretCode = random.sample(coloursList, 4)
45 print(secretCode)
```

Figure 7: Generate randomized colour code

Decision Making

Decision making perhaps the most essential element of the coding in this program.

In this software, decision making is used to verify the number of correct and incorrect places in the user guess order. The program will not operate properly if decision making is not implemented

```

73     # Check user input colour code
74     if correctCode != "1111":
75         for i in range(4):
76             if userInput[i] == secretCode[i]:
77                 correctCode += "1"
78             if userInput[i] != secretCode[i] and userInput[i] in secretCode:
79                 guessedCode += "0"
80         print()
81         print("CLUE: ", correctCode + guessedCode)
82         count += 1
83         attempts -= 1
84
85     if correctCode == "1111":
86         if count == 1:
87             print("That is impressive that you did it on your first try! Congratulation!")
88         else:
89             print("You done it in " + str(count) + " times. Congratulation!")
90         game = False
91
92     # Check number of user input
93     if count >= 1 and count < 8 and correctCode != "1111":
94         print("This is your attempt number", count + 1)
95         print("You still have", attempts, "attempts left.")
96         print("Your guess is: ", [userInput])
97         print()
98         print("Next guess: ")
99     elif count >= 8:
100         print("You did not guess the correct secret code!\n\nThe secret code is: " + str(secretCode))
101         game = False
102

```

Figure 8: Checking for user guess colour code sequence for number of correct and incorrect positions

```

26     # Welcome selection menu
27     welcome()
28     user = int(input("Please enter the following number to continue: "))
29     if user == 1:
30         instruction()
31         game = True
32     elif user == 2:
33         game = True
34     elif user == 3:
35         print("Exiting game ...")
36         game = False
37     else:
38         print("Invalid input. Exiting game...")
39         game = False
40

```

Figure 9: Prevent user from entering invalid commands using decision making



Calculating several outcomes for a program with many choices.



Work in close collaboration with iterative processes.



When creating a program, it helps with logical issue solving.



It assists the software in identifying the error and debugging through decision making.



Creates numerous cases to handle different circumstances, preventing program errors.

**Additional
applications for
computational
problem solving
decision
making:**

Iterative Process

Iterative process is available in the user menu in this program and utilized to loop the code guessing.

In this program, iterative processes are critical to ensuring a seamless user experience with no unexpected program terminations.

Iterative process such as while loop that allows to repeat the process when the conditions was met

```

26 # Welcome selection menu
27 welcome()
28 user = int(input("Please enter the following number to continue: "))
29 if user == 1:
30     instruction()
31     game = True
32 elif user == 2:
33     game = True
34 elif user == 3:
35     print("Exiting game ...")
36     game = False
37 else:
38     print("Invalid input. Exiting game...")
39     game = False
40
50 # Game
51 while game == True:
52     # Initialize
53     correctCode = ""
54     guessedCode = ""
55     print()
56     print("Please enter your colour code.\nR - RED\nG - GREEN\nB - BLUE\nY - YELLOW\nP - PURPLE\nW - WHITE")
57     print("Example: RED GREEN BLUE WHITE --> RGBW ")
58     # User input colours code
59     userInput = str(input("\nPlease enter your 4 colours code: ").upper())
60
61     # Check length of user input
62     if len(userInput) != len(secretCode):
63         print("The secret code only has 4 colours. Please try again.")
64         print(coloursList)
65         continue
66     # Check user input from colourList
67     for i in range(1):
68         if userInput[i] not in coloursList:
69             print("Other colours out of this master mind game are not allowed. Please try again")
70             print(coloursList)
71             continue
72

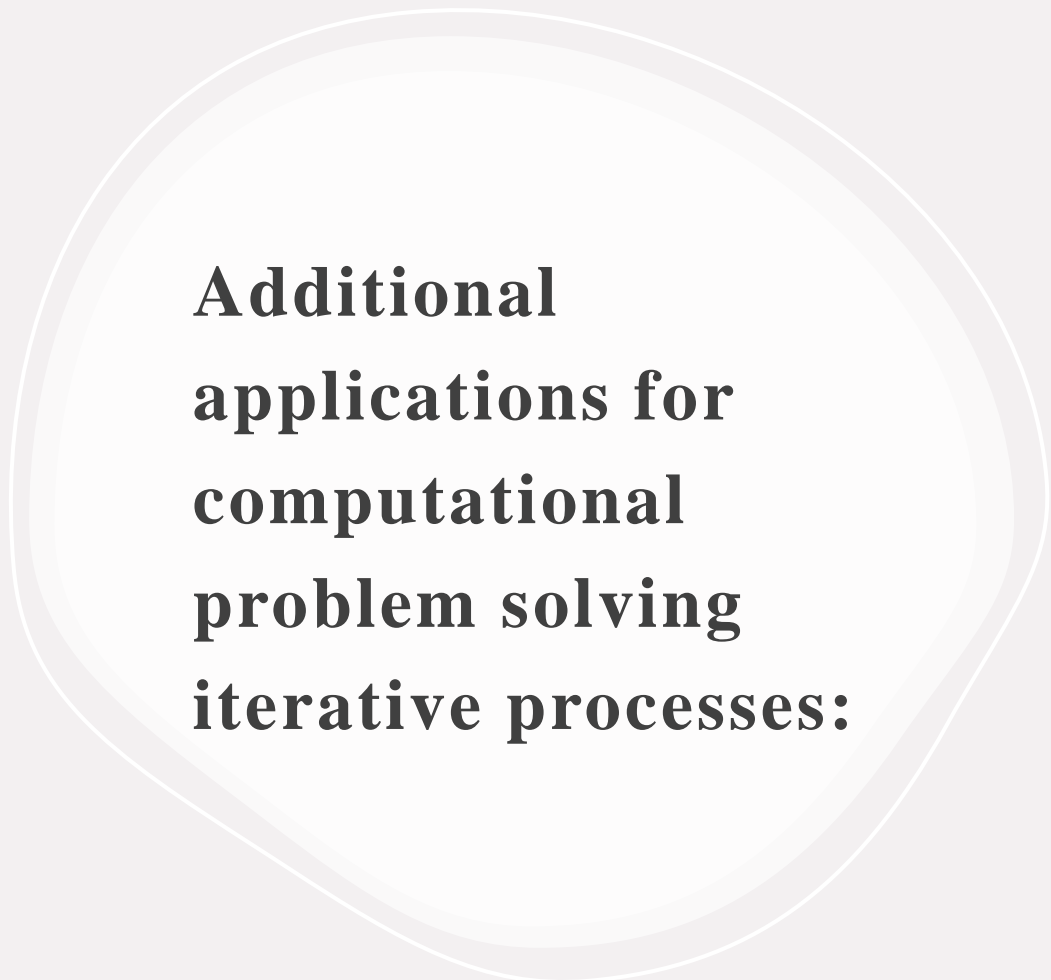
```

Figure 10: The program will keep running while the game is True

It is also necessary to define cases in this program to break the loop if it has to be broken. The “break” functionality may be used to do this.

```
103     # Replay or end the game
104     while game == False:
105         end = input(str("Do want to play again? (Y/N)")).upper()
106         count = 0
107         if end == "N":
108             print("Thank you for playing the game!")
109             break
110         elif end == "Y":
111             game = True
112             print()
113             print("Starting game... ")
114             print()
115         else:
116             print("Invalid input. Please try again.")
117             game == False
118
```

Figure 11: When the program ended the game will equal to False and prompt user whether they wanted to repeat or exit the program



Additional applications for computational problem solving iterative processes:

- Decreases risk, increases efficiency and tackle challenges in a more flexible and dynamic way by utilising an iterative processes.
- Used to avoid the program from being terminated unexpectedly as a result of user input.
- When a program necessitates repetition.
- To develop a more sophisticated and thorough program, it collaborates closely with decision making statements.

User-define Function

User-defined functions are utilized in this program to divide the code into distinct primary purposes or functions. Each functions serves a specific role in the game and is defined in such a way that it may be readily accessed inside the program.

A user-defined function can assist to simplify program code by allowing multiple “screens” to be called up at any moment, making the experience more user friendly.

```

3  # Welcome menu
4  def welcome():
5      print()
6      print(" ----- ")
7      print("          MASTER MIND GAME ")
8      print(" ----- ")
9      print("Welcome to the MASTER MIND GAME")
10     print("[1] Instruction")
11     print("[2] Start game")
12     print("[3] Exit game")
13
14 # Instruction menu
15 def instruction():
16     print()
17     print("Computer will randomly choose 4 colours from the list and arrange in a order.\n")
18     + "There are a total of 6 colours to choose from: (RED - R, GREEN - G, BLUE - B, YELLOW - Y, PURPLE - P, WHITE - W)\n"
19     + "You must guess the 4 correct colours in the correct order to win the game.\n"
20     + "After entering the colour code, computer will show player the clue:\n"
21     + "0 shown means colour is correct with incorrect position.\n"
22     + "1 shown means colour is in the correct position.\n"
23     + "You are given 8 attempts only.\n"
24     + "ENJOY! Goodluck out there.")
25
26 # Welcome selection menu
27 welcome()
28 user = int(input("Please enter the following number to continue: "))
29 if user == 1:
30     instruction()
31     game = True
32 elif user == 2:
33     game = True
34 elif user == 3:
35     print("Exiting game ...")
36     game = False
37 else:
38     print("Invalid input. Exiting game...")
39     game = False
40

```

Figure 12: The functions for welcome and instruction menus screen

Conclusion

Finally, I am extremely pleased with the outcome of my project on MASTER MIND GAME. I learned how to build a completely functional program while also improving my coding skills and knowledge. Thanks to this project, I now have a better understanding of concepts like if else statements, iterative process and functions.



Future Improvements

I want to develop the MASTER MIND GAME in the future by adding graphic aspects, colours and sound effect to the application. This may be accomplished by importing other modules besides random. For the board as well as the pins, which will display the number of wrong and right placements of colours, display different colours text and sound effect when entered input.

```
print("Thank You")
```

