CSC1024 PROGRAMMING PRINCIPLES

PROGRAMMING PROJECT: A MASTER MIND COMPUTER GAME

15 July 2022

Lester Koon Zhy Min
— 20068813 —

https://youtu.be/WJ11J12xauw

Mastermind game

PROGRAMMING & PROBLEM-SOLVING TECHNIQUES

User-Defined Functions: -

a block of code which only runs when it is called, used for repeated task so the same code does not need to be written again

interface ():
displays the main interface of the game to the user when called

```
# A function created to print the main interface for the master mind game
def interface():
    print()
    print("                        M A S T E R   M I N D                      ")
    print("                         ==================                       ")
    print()
    print("\nWelcome to the Master Mind Game.\nThe CPU will randomly generate a list 4 fruits.",
          "\nTo win this game, you would need to guess each fruit from the list correctly.")
    print()
    print("                               R U L E S !                          ")
    print("                                ========                            ")
    print()
    print("\n1. Please guess 1 fruit at a time.")
    print()
    print("\n2. To win, all 4 fruits and their position must be correct.")
    print()
    print("\n3. The list of available fruits are: \n   -- apple, orange, grape and melon --")
    print()
    print("                                G A M E {}!                          ".format(number_of_games))
    print("                                  ======                            ")
```

display_score (correct_guess, correct_fruit):
displays how many fruits correct and in the correct position, and how many fruits are correct but in the wrong position based on the user's guesses

```
# A function to display the score of the user's guess to the randomized list
def display_score(correct_guess, correct_fruit):
    print("Correct fruit in the right position:", correct_guess)
    print("Correct fruit but wrong position:", correct_fruit)
```

List: -

linear data structure that are used to store multiple items in a single variable, each item in the list are indexed, changeable, and repeatable

A total of 5 list are used throughout the program

```
FruitList = ['apple', 'orange', 'grape', 'melon'] #List of fruits that can be choosen by both the cpu and user
CPUList = []  # List of the randomised fruits that the user needs to guess
UserList = []  # List of fruits guessed by the user
CheckerList1 = []  # Lists to check the user's list with the cpu's list
CheckerList2 = []
```

- FruitList [] : List of valid fruits that can be used in the program
- CPUList [] : List used to store the returned elements from the method random.choices ()
- UserList [] : List used to store the user's input
- CheckerList1 [] & CheckerList2 [] : List used for decision making to check the number of correct guesses made by the user

```
while len(UserList) != 4:
    guess = str(input("Choice of fruit:").lower())
    if guess not in FruitList:
        print(
            "\nInvalid Input! \nPlease enter only apple, orange, grape or melon. \n")
    else:
        UserList.append(guess)
```

Example 1:
the append () method is used to add the user's input into the UserList

```
while j < len(CheckerList1):
    k = 0
    while k < len(CheckerList2):
        if CheckerList1[j] == CheckerList2[k]:
            del CheckerList2[k]
            correct_fruit += 1
            break
        else:
            k += 1
    j += 1
```

Example 2:
the del keyword is used to delete the specified index from CheckerList2

Random choice from a list: -

segment of code use to generate a random list of items

random module implements pseudo-random number generators for various distribution

```
# the random module is imported to generate a random selection
import random

# variables used throughout the program
FruitList = ['apple', 'orange', 'grape', 'melon']
CPUList = []  # List of the randomised fruits that the user needs to guess

# Generates a randomized list of fruits with repetition allowed from the fruits in the FruitList
CPUList = random.choices(FruitList, k=4)
```

A list containing the fruits used throught the game

specified size of the returned list (optional parameter)

```
CPUList = random.choices(FruitList, k=4)
```

Randomised list of fruits, which the users need to guess

choices ():
method used to return multiple random elements from a specified sequence with replacement

specified sequence where the random elements are chosen from (required parameter)

User Input: -

segment of code prompt the user for their guesses and store the valid guesses into a list

The lower () method is used to return the user's input in lower case, thus the user's input would not be case sensitive.

A while loop is used is used to prompt the user to enter their guesses until the users enter 4 valid guesses.

```
print("\nPlease enter fruits from the list above only.")
# Loop to get the user's guesses
while len(UserList) != 4:
    # Returns the user's input in lower case
    guess = str(input("Choice of fruit:").lower())
    if guess not in FruitList:  # Catches wrong inputs or input formats
        print(
            "\nInvalid Input! \nPlease enter only apple, orange, grape or melon. \n")
    else:
        UserList.append(guess)  # Adds valid input into the user's list
```

An if...else statement is used to validate the user's guesses. If the guess is not a fruit from the FruitList or is in the wrong data format, the input is invalid and the program will display a error message and prompt the user to enter a new guess. If the user guess is valid, it will be add to UserList using the append () method.

User input prompt and error message:

```
Please enter fruits from the list above only.
Choice of fruit:
```

```
Please enter fruits from the list above only.
Choice of fruit:Peach

Invalid Input!
Please enter only apple, orange, grape or melon.

Choice of fruit:
```

Display data: -

Displays their guesses in a list to the user

```
print("\nPlease enter fruits from the list above only.")
# Loop to get the user's guesses
while len(UserList) != 4:
    # Returns the user's input in lower case
    guess = str(input("Choice of fruit:").lower())
    if guess not in FruitList:  # Catches wrong inputs or input formats
        print(
            "\nInvalid Input! \nPlease enter only apple, orange, grape or melon. \n")
    else:
        UserList.append(guess)  # Adds valid input into the user's list

print("\nYour guesses were:", UserList, "\n")
```

This will only be displayed once the user enters 4 valid inputs.

```
Your guesses were: ['apple', 'apple', 'melon', 'orange']
```

Displays the scores of the user's guesses

```
# Statement when the user guess all the fruits and their position correctly
if UserList == CPUList:
    if attempts == 1:…
    else:…
# Statement to display the score of each guess.
# It also empties the user's list and checker lists, which allows the player to attempt to guess the fruits again.
else:
    display_score(correct_guess, correct_fruit)
    UserList = []
    CheckerList1 = []
    CheckerList2 = []
    print("\nWrong Guesss, please try again.")
```

This will only be displayed if the user's guesses is incorrect. (When UserList != CPUList)

```
Correct fruit in the right position: 2
Correct fruit but wrong position: 1

Wrong Guesss, please try again.
```

Logic used to calculate the score: -

segment of code used to calculate the score of the user's guesses

```
# Variables used to calculate the score of each guess
correct_guess = 0 #Both fruit and position are correct
correct_fruit = 0 #The fruit is correct but its in the wrong position
i = 0
j = 0
k = 0

# Loop to calculate the guesses with the correct fruit and position
while i < len(UserList):
    if UserList[i] == CPUList[i]:
        correct_guess += 1
    else:
        CheckerList1.append(UserList[i])
        CheckerList2.append(CPUList[i])
    i += 1

# Loop to calculate the guesses with the correct fruit abut the wrong position
while j < len(CheckerList1):
    k = 0
    while k < len(CheckerList2):
        if CheckerList1[j] == CheckerList2[k]:
            del CheckerList2 [k]
            correct_fruit += 1
            break
        else:
            k += 1
    j += 1
```

This segment of code uses a combination of while loops, if...else statements, logical operators and lists operations to calculate the score.
The logical flow of this segment is as follows:-

To calculate the guesses with the correct fruit and correct position;

1. A while loop is used to ensure that all elements (fruits) of the lists are checked with regards to their index (position).
2. The fruits of the UserList (user's guess) and the CPUList (randomised list) with the same index are checked with each other, like pairs. If the fruit pair is matching, then the value of correct_guess will increase. However, if the fruit pair is different, the fruit from UserList and CPUList will be added into CheckerList1 and CheckerList2 respectively.
3. The loop will end once all fruits are checked.

To calculate the guesses with the correct fruit but wrong position;

1. A while loop is used to ensure that all fruits which did not match previously are checked (CheckerList1 and CheckerList2).
2. A nested while loop is then used to ensure that the each fruit guessed by the user from CheckerList1 is checked with every randomised fruit from the CheckerList2.
3. If a fruit from CheckerList1 matches with any fruit in CheckerList2, the value of correct_fruit will increase. The fruit matched in CheckerList2 will then be deleted to prevent a false score, as the user are able to enter a fruit repeatedly.
4. The loops will end once all guesses from CheckerList1 are checked

Example of how the scores are calculated: -

```
CPUList = ['apple', 'grape', 'grape', 'melon']      UserList = ['apple', 'melon', 'melon', 'grape']
```

- Logic to check if the fruit and its position is correct -

	UserList	CPUList
0	apple	apple
1	melon	grape
2	melon	grape
3	grape	melon

- **correct_guess** = 0
- **correct_fruit** = 0

all the fruits with the same index from UserList and CPUList are checked with one another as pair, starting with index [0] to index [3]

If fruits are equal:-

	UserList	CPUList
0	apple	apple
1	melon	grape
2	melon	grape
3	grape	melon

- **correct_guess** = 1 +1
- **correct_fruit** = 0

since the fruit from UserList and CPUList are equal, both the fruits are 'apple', the correct_guess score is incremented by 1

If fruits are not equal:-

	UserList	CPUList
0	apple	apple
1	melon	grape
2	melon	grape
3	grape	melon

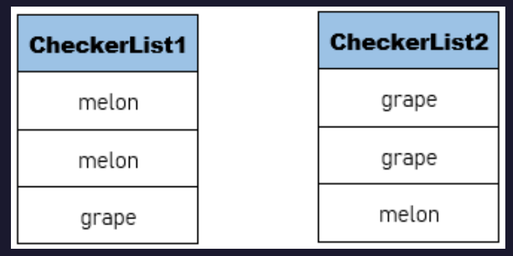
- **correct_guess** = 1
- **correct_fruit** = 0

since the fruits are not equal, the fruits from UserList and CPUList will be added into CheckerList1 and CheckerList2 respectively

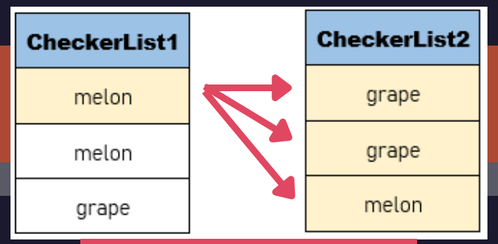
CheckerList1	CheckerList2
melon	grape
melon	grape

2

- Logic to check if the fruit position is correct -

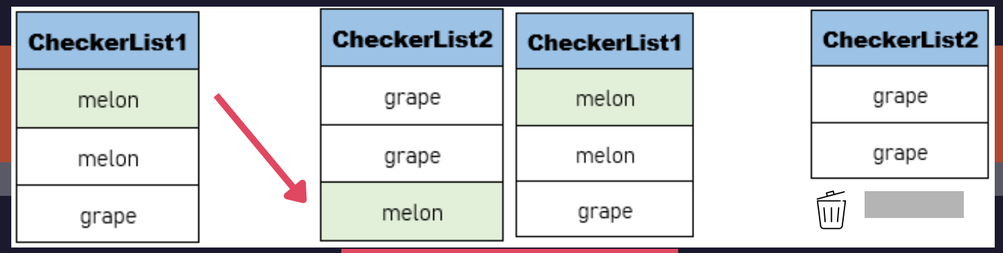


results of the lists once the previous logical loop has ended



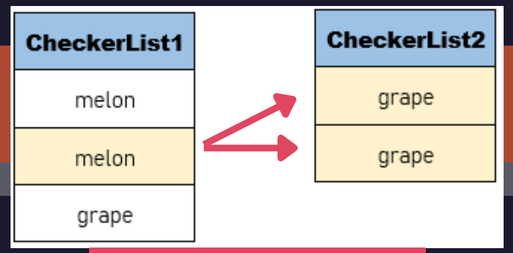
- **correct_guess** = 1
- **correct_fruit** = 0

each fruit from CheckerList1 is checked with every fruit from CheckerList2



- **correct_guess** = 1
- **correct_fruit** = 1 +1

if the fruit from CheckerList1 matches with any fruit from CheckerList2, the correct_fruit score is incremented by 1. The matched fruit from CheckerList2 is then deleted



- **correct_guess** = 1
- **correct_fruit** = 1

since the matched fruit was deleted from CheckerList2, the next 'melon' from CheckerList1 does not match with any fruit from CheckerList2


THANK YOU