# Guess the day ver4

|  |  |
| --- | --- |
| ver0 | original description |
| ver1 | finding class, attribute and method candidates |
| ver2 | Refining the public interface, refining attributes, methods, classes |
| ver3 | Defining more methods and attrivutes |
| ver4 | Tkinter implementation, class inheritance and composition |

## Original description

Select-the-day, the program shows that day’s weekday and three other randomly picked week days, asks a user to select which day is today, checks the answer and saves the information to data storage accessible to the personnel of the service home. If the user picked the wrong day, the program asks again giving a gentle hint e.g. today is the first day of the week. And if the user guesses wrong the second time, the program shows a nice greeting and sets the game as FAILED to the data storage. If the user answered correctly the second time the game is saved as FOLLOW, and if the answer was right the first time, the game is saved as PASSED. The personnel checks the saved games to verify if the resident has problems in keeping up with the time.

## 1st iteration, check nouns.

**Select-the-day, the program shows that day’s weekday and three others randomly picked week days, asks a user to select which day is today, checks the answer and saves the information to data storage accessible to the personnel of the service home. If the user picked the wrong day, the program asks again giving a gentle hint e.g. today is the first day of the week. And if the user** **guesses wrong the second time, the program shows a nice greeting and sets the game as FAILED to the data storage. If the user answered correctly the second time the game is saved as FOLLOW, and if the answer was right the first time, the game is saved as PASSED. The personnel checks the saved games to verify if the resident has problems in keeping up with the time.**

|  |  |  |  |
| --- | --- | --- | --- |
| noun | basic form, synonym, duplicate value, notes | value or collections | possible class, object, attribute |
| **Select-the-day** |  | value | **Possible class:**  game’s title, game class, knows the rules of the game and keeps track of the game’s data |
| **weekdays** | weekday | value | **Possible attribute:**  Stores weekdays |
| **user** |  | type instance | **Possible attribute:**  Stores user instance |
| **today** |  | value | **Possible attribute:**  Stores current weekday |
| **answer** |  | Type string | **Possible attribute:**  Keeps value of user’s answers |
| **gentle hint** |  | type list, value | **Possible attribute:**  Stores hints user are given in different states of the game |
| **state** |  | Type string | **Possible attribute:** Keeps count of user attempts |
| **options** |  | type list | **Possible attribute:**  Stores days user can select from |
| **Tries** | Try | type integer | **Possible attribute:**  Keeps count of user tries to answer correctly |

## 2nd iteration, check verbs

**Select-the-day, the program shows that day’s weekday and three other randomly picked week days, asks a user to select which day is today, checks the answer and saves the information to data storage accessible to the personnel of the service home. If the user picked the wrong day, the program asks again giving a gentle hint e.g. today is the first day of the week. And if the user** **guesses wrong the second time, the program shows a nice greeting and sets the game as FAILED to the data storage. If the user answered correctly the second time the game is saved as FOLLOW, and if the answer was right the first time, the game is saved as PASSED. The personnel checks the saved games to verify if the resident has problems in keeping up with the time.**

|  |  |  |
| --- | --- | --- |
| verb | subject – object (who does, who is the target) | possible action, function, method |
| **checks** | Game’s action | **Possible method:**  Check is user answer correct, return True if it is, Else False |
| **saves** | Games’s action | **Possible method:**  Saves user’s answer to be accessed by service home personnel |
| **set state** | Games’s action | **Possible method:**  Sets game state according to user answer which is later saved in data storage |
| **random generate** | Games’s action | **Possible method:**  Randomly select three other weekdays + add correct weekday into mix |
| **quit** | Games’s action | **Possible method:**  Quits the game |
| **restart** | Games’s action | **Possible method:**  Restart the game |
| **set hint** | Games’s action | **Possible method:**  Check user answer and gives hint based on today’s weekday |

3rd iteration, refining the logic, attributes and methods for ver2

@ver3 description

Select the day game asks user to select from 4 different options which weekday is today. If user answers incorrectly the first time, user is given gentle hint which weekday is today. If user answers wrong the second time, user is given nice greeting and state is saved as “FAILED”. If user answers correctly on one of the times, he gets message “PASSED” and state is saved as “PASSED”. Authorized staff will be able to access answer state of particular user in later time.

## 4th iteration, Tkinter implementation, class inheritance and composition for ver3

@ver4 description

Select the day game asks user to select from 4 different options which weekday is today. If user answers incorrectly the first time, user is given gentle hint which weekday is today. If user answers wrong the second time, user is given nice greeting and state is saved as “FAILED”. If user answers correctly on one of the times, he gets message “PASSED” and state is saved as “PASSED”. Authorized staff will be able to access answer state of particular user in later time.

### Attributes that are removed with @ver4:

answer, tries, state, options,

### Attributes that are added with @ver4:

days, attempt, status

### Methods that are removed with @ver4

Set\_hint, random\_generate, set\_state, generate\_options

### Methods that are added with @ver4

Pick, hint, show

1. class GuessTheDay
   1. Attributes
      1. Title
         1. Stores title of the game
      2. Weekdays
         1. Stores all weekdays.
      3. Today
         1. Stores today’s weekday
      4. Days
         1. Stores the randomly generated days to show
      5. Status
         1. Stores three statuses for the patient follow up
      6. Hints
         1. Stores hints user are given in different states of the game.
      7. Attempt
         1. Keeps count of user attempts.
   2. Methods
      1. Check
         1. Check is user answer correct, return True if it is, Else False
      2. pick
         1. Generates three wrong days
      3. hint
         1. Gives a hint based on today’s weekday.
      4. show
         1. Shows 4 weekdays (3 wrong days + today)
   3. Reserve topic
   4. Reserve topic
2. class Player
   1. Attributes
      1. player\_name
         1. Stores the player’s name
      2. player\_points
         1. Stores the points of the player
      3. player\_count
         1. Stores the amount of players generated by the player class
      4. player\_id
         1. Stores an id for the player instance
3. class GamePlay
   1. Attributes
      1. game\_play
         1. Runs the game
      2. Restart
         1. Restarts the game

### Graphical user interface, text, application, chat or text message Description automatically generated

### Generated UML class diagram (<https://app.genmymodel.com/>) @ver4

### Code generated from the UML diagram @ver4

class Player(object):

def \_\_init\_\_(self):

self.player\_name = ""

self.player\_points = 0

self.player\_count = 0

self.player\_id = 0

# Start of user code -> properties/constructors for Player class

# End of user code

# Start of user code -> methods for Player class

# End of user code

class GamePlay(object):

pass

# Start of user code -> properties/constructors for GamePlay class

# End of user code

def game\_play(self):

# Start of user code protected zone for game\_play function body

raise NotImplementedError

# End of user code

def restart(self):

# Start of user code protected zone for restart function body

raise NotImplementedError

# End of user code

# Start of user code -> methods for GamePlay class

# End of user code

class GuessTheDay(object):

def \_\_init\_\_(self):

self.title = ""

self.weekdays = None

self.today = None

self.days = None

self.attempt = 0

self.\_status = None

self.hints = None

# Start of user code -> properties/constructors for GuessTheDay class

# End of user code

def check(self):

# Start of user code protected zone for check function body

raise NotImplementedError

# End of user code

def pick(self):

# Start of user code protected zone for pick function body

raise NotImplementedError

# End of user code

def hint(self):

# Start of user code protected zone for hint function body

raise NotImplementedError

# End of user code

def show(self):

# Start of user code protected zone for show function body

raise NotImplementedError

# End of user code

# Start of user code -> methods for GuessTheDay class

# End of user code

# Start of user code -> functions/methods for GuessTheDay package

# End of user code

Completed code @ver 4

# File : guesstheday.py

# Authors : Sebastian Sopola, Uras Ayanoglu, Jerry Karkainen

# Description: This game is memory game. It is played to check if user gets day's weekday correct.

# Import necessary libaries

from tkinter import messagebox

import tkinter as tk

import tkinter.ttk as ttk

from datetime import datetime

import random

from minigames.game\_components import GamePlay, Player

# This class handles game interaction

class GuessTheDay(tk.Frame, GamePlay):

# Establishing parameters

def \_\_init\_\_(self,parent):

self.parent = parent

parent.update()

self.width = parent.winfo\_width()

self.height = parent.winfo\_height()

'''initializes Window's attributes'''

super().\_\_init\_\_(master=parent)

self.title = "Guess The Day"

self.weekdays = ["Monday", "Tuesday", "Wednesday", "Thursday", "Friday", "Saturday", "Sunday"]

self.today = datetime.now().weekday()

self.days = []

self.attempt = 0

self.\_status = ("passed", "failed", "follow")

self.hints = [f"Yesterday was the {self.today}. day of the week", f"Today is the {self.today + 1}. day of the week", f"8 days later, it will be {self.weekdays[(self.today +1)]}"]

game\_label = ttk.Label(self, text=self.title, font=("Helvetica", 40))

game\_label.grid(row=0, column=0, columnspan=5, sticky=tk.NSEW)

# Buttons for the days of the week

day\_options = self.pick()

day1\_button = ttk.Button(self, text=day\_options[0], command=lambda: self.game\_play(day\_options[0]))

day1\_button.grid(row=1, column=1, pady=3, sticky=tk.NSEW)

day2\_button = ttk.Button(self, text=day\_options[1], command=lambda: self.game\_play(day\_options[1]))

day2\_button.grid(row=1, column=2, pady=3, sticky=tk.NSEW)

day3\_button = ttk.Button(self, text=day\_options[2], command=lambda: self.game\_play(day\_options[2]))

day3\_button.grid(row=1, column=3, pady=3, sticky=tk.NSEW)

day4\_button = ttk.Button(self, text=day\_options[3], command=lambda: self.game\_play(day\_options[3]))

day4\_button.grid(row=1, column=4, pady=3, sticky=tk.NSEW)

# Button for closing the game

close\_button = ttk.Button(self, text='Quit', command=self.\_\_close)

close\_button.grid(row=5, column=4, columnspan=5, sticky=tk.NSEW)

# Start of user code -> properties/constructors for Guess\_The\_day class

def pick(self):

# randomly generate three wrong weekdays

if not self.days:

while len(self.days) <= 2:

wrong\_day = random.randint(0,6)

if wrong\_day != self.today and self.weekdays[wrong\_day] not in self.days:

self.days.append(self.weekdays[wrong\_day])

self.days.insert(random.randint(0,3), self.weekdays[self.today])

return self.days

def check(self, answer):

# Check user answer

if answer == self.weekdays[self.today]:

print("True") # This print statement is for testing purposes only to check if the function works

return True

else:

self.attempt += 1

print("False") # This print statement is for testing purposes only to check if the function works

return False

def hint(self):

print(random.choice(self.hints))

def show(self):

# Show 4 weekdays to choose one

for i in range(4):

print(f"{i}. {self.days[i]}")

def restart(self):

# restart the game

self.attempt = 0

self.days = []

def \_\_close(self):

'''asking if closing is intended'''

if messagebox.askyesno("Close", "Do you want to close the Guess the Day game?"):

self.parent.destroy()

def game\_play(self, guess):

self.attempt += 1

if self.attempt < 2:

if guess == self.weekdays[self.today]:

print(self.\_status[0]) # This should be added to the database to be followed up by nursing home personnel

ttk.tkinter.messagebox.showinfo(title="Correct!", message="You guessed the day corret!\nThanks for playing.\nHave a nice day!")

self.parent.destroy()

else:

print(self.attempt)

self.hint()

ttk.tkinter.messagebox.showinfo(title="hint", message=random.choice(self.hints))

elif self.attempt == 2 and guess == self.weekdays[self.today]:

ttk.tkinter.messagebox.showinfo(title="Correct!", message="You guessed the day corret!\nThanks for playing.\nHave a nice day!")

self.parent.destroy()

else:

ttk.tkinter.messagebox.showinfo(title="Good Bye!", message="Thanks for playing.\nHave a nice day!")

self.parent.destroy()

# End of user code

if \_\_name\_\_ == "\_\_main\_\_":

app = GuessTheDay()

app.mainloop()

# Start of user code -> functions/methods for memorygame package

# End of user code