A

Report On

"Wordle Game Using Python"

Under the subject

Programming With Python(22616)

Submitted by

Sr. No	**Enrollment No**	**Name of the student**
1	2000100150	Miss. Pranali Dilip Bhosale
2	2000100170	Miss. Shreya Indrajeet Hande
3	2000100209	Miss. Priti Shashikant Suryawanshi

Under the guidance of

Mr. Sunil P. Emekar

Department of Computer Engineering

Government Polytechnic, Karad.

Maharashtra State Board of Technical Education

Certificate of Completion

Of Micro-Project Assessment at the end of the Diploma program

(By respective Head of the Department and Head of the Institute)

This is to certify that

Roll No.: 2203	Ms. Pranali Dilip Bhosale.	Enrollment no.:	2000100150
Roll No.: 2222	Ms. Shreya Indrajeet Hande.	Enrollment no.:	2000100170
Roll No.: 2258	Ms. Priti Shashikant Suryawanshi.	Enrollment no.:	2000100209

has successfully completed the **"WORDLE GAME Using Python"** Micro-project as in the enclosed 'Portfolio' for the course **Programming With Python (22616)** during his tenure of completing the Diploma program in Computer Engineering from **Government Polytechnic, Karad (Code:0010)**

Signature
Course Faculty

Signature
Head of the Department

1. RATIONALE :

Solving puzzles is a way to relax and pass the time after a long day. It is also beneficial to the mind.

And even better – there are correlations between puzzle-solving and increased problem-solving skills.

Wordle is a new word puzzle game that challenges its players to guess a five-letter word in six tries.

2. AIM AND BENEFITS :

1. To build and improve the programming logic.
2. To study the Python language.
3. To make GUI based game using Python.
4. To improve problem solving skills.

3. COURSE OUTCOMES :

a. Display message on screen using Python script on IDLE.

b. Develop python program to demonstrate use of python operators

c. Perform operations on data structures in python.

d. Develop functions for given problem.

e. Design classes for given problem.

f. Handle Exceptions.

4. LITERATURE REVIEW:

We have referenced some websites which have already implemented this Wordle game using python and its tools. But some of them provided console based websites , and they do not have provided the proper GUI for the game, they do not provided animations and event handlings also.

1. https://realpython.com/python-wordle-clone/.
2. https://www.freecodecamp.org/news/building-a-wordle-game/
3. https://www.practicepython.org/blog/2022/02/12/wordle.html

5. ACTUAL PROCEDURE FOLLOWED :

Sr. No.	Details of Activity	Planned Start Date	Completed Finish Date	Name of responsible Team Members
1.	Selection Of Topic	5/3/2023	5/3/2023	Whole Team
2.	Proposal Preparation	8/3/2023	10/3/2023	Pranali Bhosale
3.	Designing of the pages	15/3/2023	20/3/2023	Whole Team
4.	Coding(Implementation)	29/3/2023	5/4/2023	Whole Team
5.	Preparing final project report	6/4/2023	10/4/2023	Pranali Bhosale
6.	Presentation	15/4/2023	15/4/2023	Whole Team
7.	Submission of report	17/4/2023	17/4/2023	Whole Team

6. ACTUAL RESOURCES USED :

Sr No.	Name of Resource	Specification	Quantity
1	Laptop	HP Pavilion ryzen5, 8gb RAM, 512 SSD, Windows 10	1
2	Software Used	Python IDLE	1
4.	Internet	4G, Google, YouTube And Web Browser.	–
5.	Book	Programming with Python	1

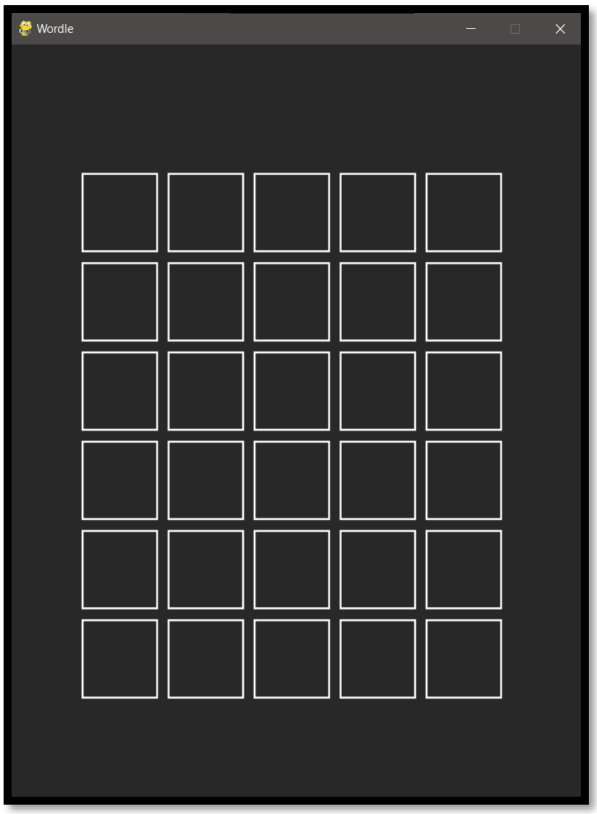
7. OUTPUT OF THE PROJECT :

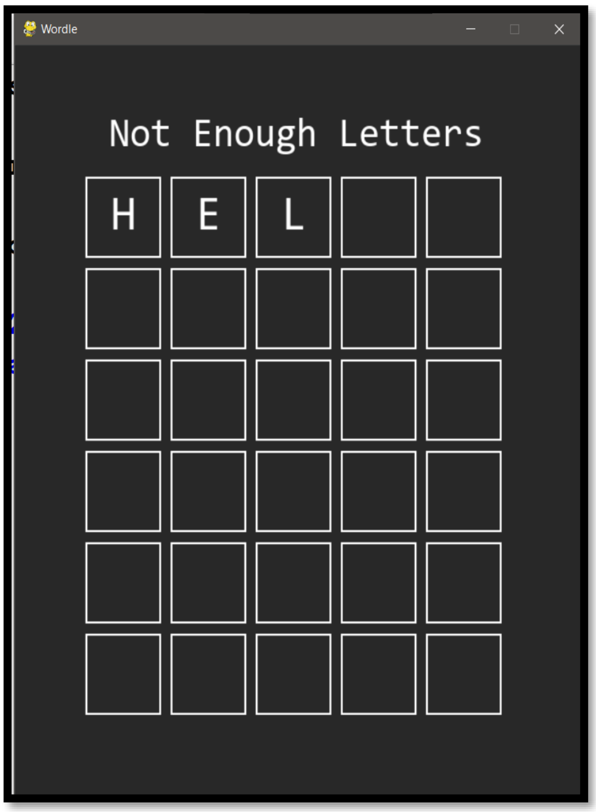
- **SOURCE CODE:**

```
import random
import pygame
from settings import *
from sprites import *
class Game:
    def __init__(self):
        pygame.init()
        self.screen = pygame.display.set_mode((WIDTH, HEIGHT))
        pygame.display.set_caption(title)
        self.clock = pygame.time.Clock()
        self.create_word_list()
        self.letters_text = UIElement(100, 70, "Not Enough Letters", WHITE)
    def create_word_list(self):
        with open("words.txt", "r") as file:
            self.words_list = file.read().splitlines()
    def new(self):
        self.word = random.choice(self.words_list).upper()
        print(self.word)
        self.text = ""
        self.current_row = 0
        self.tiles = []
        self.create_tiles()
        self.flip = True
        self.not_enough_letters = False
        self.timer = 0
    def create_tiles(self):
        for row in range(6):
            self.tiles.append([])
            for col in range(5):
                self.tiles[row].append(Tile((col * (TILESIZE + GAPSIZE)) +
MARGIN_X, (row * (TILESIZE + GAPSIZE)) + MARGIN_Y))
    def run(self):
        self.playing = True
        while self.playing:
            self.clock.tick(FPS)
            self.events()
            self.update()
            self.draw()
    def update(self):
        self.add_letter()
    def add_letter(self):
        # empty all the letter in the current row
        for tile in self.tiles[self.current_row]:
            tile.letter = ""
        # add the letters typed to the current row
        for i, letter in enumerate(self.text):
            self.tiles[self.current_row][i].letter = letter
            self.tiles[self.current_row][i].create_font()
    def draw_tiles(self):
        for row in self.tiles:
            for tile in row:
                tile.draw(self.screen)
    def draw(self):
        self.screen.fill(BGCOLOUR)
        # display the not enough letters text
        if self.not_enough_letters:
            self.timer += 1
            self.letters_text.fade_in()
            if self.timer > 90:
                self.not_enough_letters = False
                self.timer = 0
        else:
            self.letters_text.fade_out()
        self.letters_text.draw(self.screen)
        self.draw_tiles()
        pygame.display.flip()
    def row_animation(self):
        # row shaking if not enough letters is inputted
        self.not_enough_letters = True
        start_pos = self.tiles[0][0].x
        amount_move = 4
        move = 3
        screen_copy = self.screen.copy()
        screen_copy.fill(BGCOLOUR)
        for row in self.tiles:
            for tile in row:
                if row != self.tiles[self.current_row]:
                    tile.draw(screen_copy)
        while True:
            while self.tiles[self.current_row][0].x < start_pos + amount_move:
                self.screen.blit(screen_copy, (0, 0))
                for tile in self.tiles[self.current_row]:
                    tile.x += move
                    tile.draw(self.screen)
                self.clock.tick(FPS)
                pygame.display.flip()
            while self.tiles[self.current_row][0].x > start_pos - amount_move:
                self.screen.blit(screen_copy, (0, 0))
                for tile in self.tiles[self.current_row]:
                    tile.x -= move
                    tile.draw(self.screen)
                self.clock.tick(FPS)
                pygame.display.flip()
            amount_move -= 2
            if amount_move < 0:
                break
    def box_animation(self):
        # tile scale animation for every letter inserted
        for tile in self.tiles[self.current_row]:
            if tile.letter == "":
                screen_copy = self.screen.copy()
                for start, end, step in ((0, 6, 1), (0, -6, -1)):
                    for size in range(start, end, 2*step):
                        self.screen.blit(screen_copy, (0, 0))
                        tile.x -= size
                        tile.y -= size
                        tile.width += size * 2
                        tile.height += size * 2
                        surface = pygame.Surface((tile.width, tile.height))
                        surface.fill(BGCOLOUR)
                        self.screen.blit(surface, (tile.x, tile.y))
                        tile.draw(self.screen)
                        pygame.display.flip()
                        self.clock.tick(FPS)
                    self.add_letter()
                break
    def reveal_animation(self, tile, colour):
        # reveal colours animation when user input the whole word
        screen_copy = self.screen.copy()
        while True:
            surface = pygame.Surface((tile.width + 5, tile.height + 5))
```

```
            surface.fill(BGCOLOUR)
            screen_copy.blit(surface, (tile.x, tile.y))
            self.screen.blit(screen_copy, (0, 0))
            if self.flip:
                tile.y += 6
                tile.height -= 12
                tile.font_y += 4
                tile.font_height = max(tile.font_height - 8, 0)
            else:
                tile.colour = colour
                tile.y -= 6
                tile.height += 12
                tile.font_y -= 4
                tile.font_height = min(tile.font_height + 8, tile.font_size)
            if tile.font_height == 0:
                self.flip = False
            tile.draw(self.screen)
            pygame.display.update()
            self.clock.tick(FPS)
            if tile.font_height == tile.font_size:
                self.flip = True
                break
    def check_letters(self):
        # algorithm to check if the letters inputted correspond to any of the letters
in the actual word
        copy_word = [x for x in self.word]
        for i, user_letter in enumerate(self.text):
            colour = LIGHTGREY
            for j, letter in enumerate(copy_word):
                if user_letter == letter:
                    colour = YELLOW
                    if i == j:
                        colour = GREEN
                    copy_word[j] = ""
                    break
            # reveal animation
            self.reveal_animation(self.tiles[self.current_row][i], colour)
    def events(self):
        for event in pygame.event.get():
            if event.type == pygame.QUIT:
                pygame.quit()
                quit(0)
            if event.type == pygame.KEYDOWN:
                if event.key == pygame.K_RETURN:
                    if len(self.text) == 5:
                        # check all letters
                        self.check_letters()
                        # if the text is correct or the player has used all his turns
                        if self.text == self.word or self.current_row + 1 == 6:
                            # player lose, lose message is sent
                            if self.text != self.word:
                                self.end_screen_text = UIElement(100, 700, f"THE
WORD WAS: {self.word}", WHITE)
                            # player win, send win message
                            else:
                                self.end_screen_text = UIElement(100, 700, "YOU
GUESSED RIGHT", WHITE)
                            # restart the game
                            self.playing = False
                            self.end_screen()
                            break
                        self.current_row += 1
                        self.text = ""
                    else:  # row animation, not enough letters message
                        self.row_animation()
                elif event.key == pygame.K_BACKSPACE:
                    self.text = self.text[:-1]
                else:
                    if len(self.text) < 5 and event.unicode.isalpha():
                        self.text += event.unicode.upper()
                        self.box_animation()
    def end_screen(self):
        play_again = UIElement(85, 750, "PRESS ENTER TO PLAY AGAIN", WHITE, 30)
        while True:
            for event in pygame.event.get():
                if event.type == pygame.QUIT:
                    pygame.quit()
                    quit(0)
                if event.type == pygame.KEYDOWN:
                    if event.key == pygame.K_RETURN:
                        return
            self.screen.fill(BGCOLOUR)
            self.draw_tiles()
            self.end_screen_text.fade_in()
            self.end_screen_text.draw(self.screen)
            play_again.fade_in()
            play_again.draw(self.screen)
            pygame.display.flip()
game = Game()
while True:
    game.new()
game.run()
```

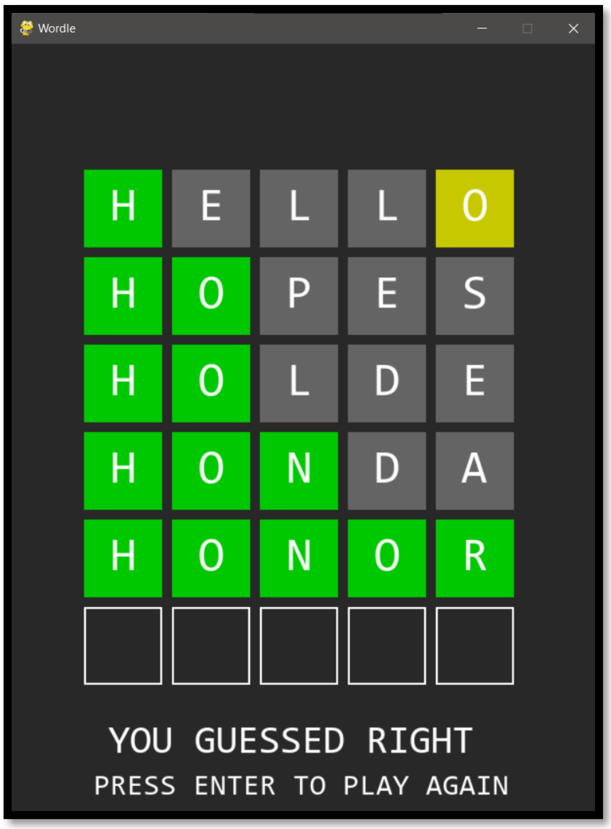
- **OUTPUT:**





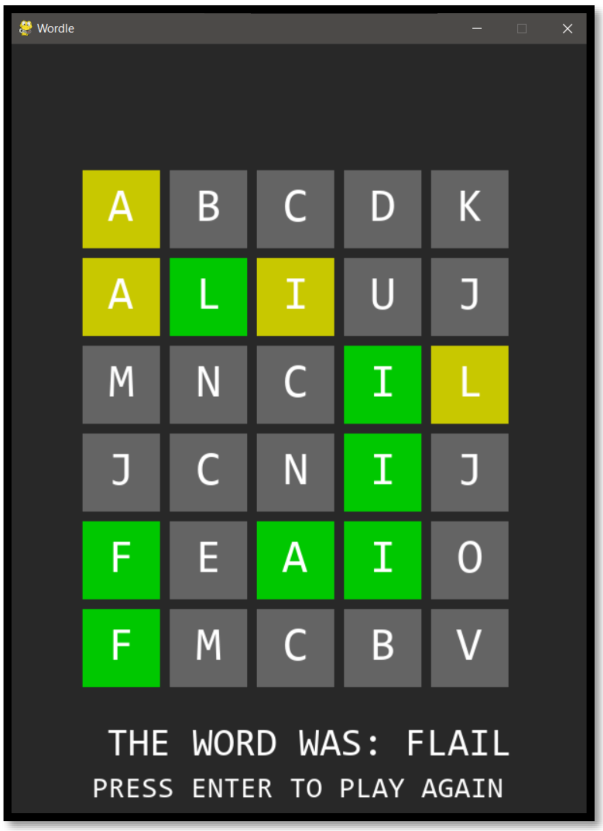
```
Not Enough Letters

H E L
```



```
H E L L O
H O P E S
H O L D E
H O N D A
H O N O R

YOU GUESSED RIGHT
PRESS ENTER TO PLAY AGAIN
```



```
A B C D K
A L I U J
M N C I L
J C N I J
F E A I O
F M C B V

THE WORD WAS: FLAIL
PRESS ENTER TO PLAY AGAIN
```

8. SKILL DEVELOPED OUT OF THIS MICRPROJECT:

After Implementing this micro-project we have learnt :

1. To make attractive GUI based project using python.
2. Working with data stored in text file.
3. Learned to apply different kinds of logic in programming.
4. Learned to make games using python language.
5. Efficient communication skills.
6. Working as a team.
7. Developing leadership qualities.

9. APPLICATIONS OF THE PROJECT :

1. Wordle is the most intriguing word puzzle around, and it has become the favorite among word lovers.
2. It increases our problem solving skills and vocabulary also.
3. Anyone can play the game Wordle, and it is free to play.