AMERICAN INTERNATIONAL UNIVERSITY-BANGLADESH

Faculty of Science and Technology

Report

Assignment Title:	Word Polygon			
Assignment No:	01		Date of Submission:	26 April 2025
Course Title:	Programming in Python			
Course Code:	CSC4162		Section:	A
Semester:	Spring	2024-25	Course Teacher:	Dr. Abdus Salam

Declaration and Statement of Authorship:

1. I/we hold a copy of this Assignment/Case-Study, which can be produced if the original is lost/damaged.
2. This Assignment/Case-Study is my/our original work and no part of it has been copied from any other student's work or from any other source except where due acknowledgement is made.
3. No part of this Assignment/Case-Study has been written for me/us by any other person except where such collaboration has been authorized by the concerned teacher and is clearly acknowledged in the assignment.
4. I/we have not previously submitted or currently submitting this work for any other course/unit.
5. This work may be reproduced, communicated, compared and archived for the purpose of detecting plagiarism.
6. I/we give permission for a copy of my/our marked work to be retained by the faculty for review and comparison, including review by external examiners.
7. I/we understand that Plagiarism is the presentation of the work, idea or creation of another person as though it is your own. It is a formofcheatingandisaveryseriousacademicoffencethatmayleadtoexpulsionfromtheUniversity. Plagiarized material can be drawn from, and presented in, written, graphic and visual form, including electronic data, and oral presentations. Plagiarism occurs when the origin of their arterial used is not appropriately cited.
8. I/we also understand that enabling plagiarism is the act of assisting or allowing another person to plagiarize or to copy my/our work.

* *Student(s) must complete all details except the faculty use part.*
** Please submit all assignments to your course teacher or the office of the concerned teacher.

Group Name/No.: 09

No	Name	ID	Program	Signature
1	Nafisa Anjum Moon	21-45803-3	BSc [CSE]	Nafisa
2	Tahiyat Ahmed	21-45703-3	BSc [CSE]	Tahiyat
3	Md. Shariful Islam	21-45701-3	BSc [CSE]	Shariful
4	Tafsirul Islam Shafin	22-47325-1	BSc [CSE]	Tafsirul

Faculty use only

FACULTY COMMENTS	**Marks Obtained**	
	Total Marks	

PROJECT OVERVIEW

The Word Polygon game is a word puzzle application built using Python and Tkinter. The game challenges players to form valid words from a set of seven letters arranged in a honeycomb hexagon layout, with the constraint that each word must include a mandatory letter. It includes features such as real-time validation, timer-based gameplay, a scoring system, and interactive graphics. The application is modular, with each feature implemented as a separate function to ensure clarity and maintainability.

FEATURES

Feature 1: Word List Loading

This feature loads a list of valid words from a file called `wordlist.txt`, ensuring that only words with a length of three or more characters are included.

```
def load_wordlist():
with open("wordlist.txt", "r") as f:
return [word.strip().lower() for word in f if len(word.strip()) >=
3]
```

Feature 2: Random Letter Generation

The game generates a random 7-letter word from the word list and extracts its letters. One of the letters is randomly selected to be mandatory. These letters are then shuffled.

```
def generate_letter_set(wordlist):
long_words = [w for w in wordlist if len(w) == 7]
base_word = random.choice(long_words)
letters = list(set(base_word.upper()))
while len(letters) < 7:
letters.append(random.choice('ABCDEFGHIJKLMNOPQRSTUVWXYZ'))
random.shuffle(letters)
return letters, random.choice(letters)
```

Feature 3: Word Validation

This function checks whether a word is valid according to the game rules: at least 3 letters, contains the mandatory letter, uses only the given letters, is found in the dictionary, and hasn't already been found.

```
def is_valid(word):
word = word.lower()
if len(word) < 3:
return False
if MANDATORY_LETTER.lower() not in word:
return False
if any(ch not in [l.lower() for l in LETTERS] for ch in word):
return False
if word not in VALID_WORDS:
return False
if word in found_words:return False
return True
```

Feature 4: Word Submission

This function is triggered when the user submits a word. It checks if the word is valid and updates the score and list of found words.

```
def check_word():
word = ent_word.get()
ent_word.delete(0, tk.END)
if is_valid(word):
found_words.append(word)
listbox.insert(tk.END, word)
lbl_score.config(text=f"Score: {len(found_words)}")
```

Feature 5: Game Reset

This function resets the game state, including the timer, score, and letter set. It allows the user to restart the game.

```
def reset_game():
found_words.clear()
listbox.delete(0, tk.END)
LETTERS, MANDATORY_LETTER = generate_letter_set(VALID_WORDS)
draw_hexagons()
lbl_score.config(text="Score: 0")
ent_word.config(state='normal')
btn_submit.config(state='normal')
time_left = 60
lbl_timer.config(text=f"Time: {time_left}s")
update_timer()
```

Feature 6: Timer Management

The game has a countdown timer that disables user input when it reaches zero. It also displays potential valid words the player could have made.

```
def update_timer():
global time_left
if time_left > 0:
time_left -= 1
lbl_timer.config(text=f"Time: {time_left}s")
root.after(1000, update_timer)
else:
ent_word.config(state='disabled')
btn_submit.config(state='disabled')
# Show possible words
```

Feature 7: Honeycomb Hexagon Drawing

This function arranges the seven letters in a hexagonal layout on a Tkinter canvas. The mandatory letter is highlighted.

```
def draw_hexagons():
canvas.delete("all")
offsets = [...]
for i, (dx, dy) in enumerate(offsets):
draw_single_hex(x, y, LETTERS[i], LETTERS[i] == MANDATORY_LETTER)
```

Feature 8: Individual Hexagon Drawing

Each hexagon is drawn with specific coordinates, size, and color depending on whether it's the mandatory letter.

```
def draw_single_hex(x, y, letter, is_mandatory):
size = 40
points = [...] # Calculate vertices
color = "#FF5722" if is_mandatory else "#546E7A"
canvas.create_polygon(points, fill=color, outline="black")
canvas.create_text(x, y, text=letter)
```

Feature 9: Clickable Letters

Each letter hexagon is interactive. Clicking a hexagon adds that letter to the word entry box.

```
def click_letter(letter):
ent_word.insert(tk.END, letter)
```

Feature 10: Scoring and Word List

The score and words found are dynamically updated on the screen as the player interacts with the game.

```
lbl_score.config(text=f"Score: {len(found_words)}")
listbox.insert(tk.END, word)
```