# Crossword Puzzle Game

Welcome to the Crossword Puzzle Game! This simple yet engaging game combines the challenge of crossword puzzles with the fun of interactive gameplay.

## Features

Dynamic Grid: A 15x15 grid is generated with random clues related to software engineering. The blocked cells create the classic crossword structure.

Random Clues: Each time you start the game, a set of 5 random clues is chosen from a pool of 15. You need to fill in the corresponding answers in the grid.

Clue Numbers: Clue numbers are randomly placed on the grid cells where the words start. These numbers are uneditable and give you a hint about where to begin.

Check answers: After filling in the grid, click the "Check Answers" button. A new window will open, displaying the correct answers.

## How to Play

Run the game using the provided Python script.

Enter the words corresponding to the displayed clues in the grid cells.

Click "Check Answers" to check your answers.

Have fun solving software engineering-themed crossword puzzles!

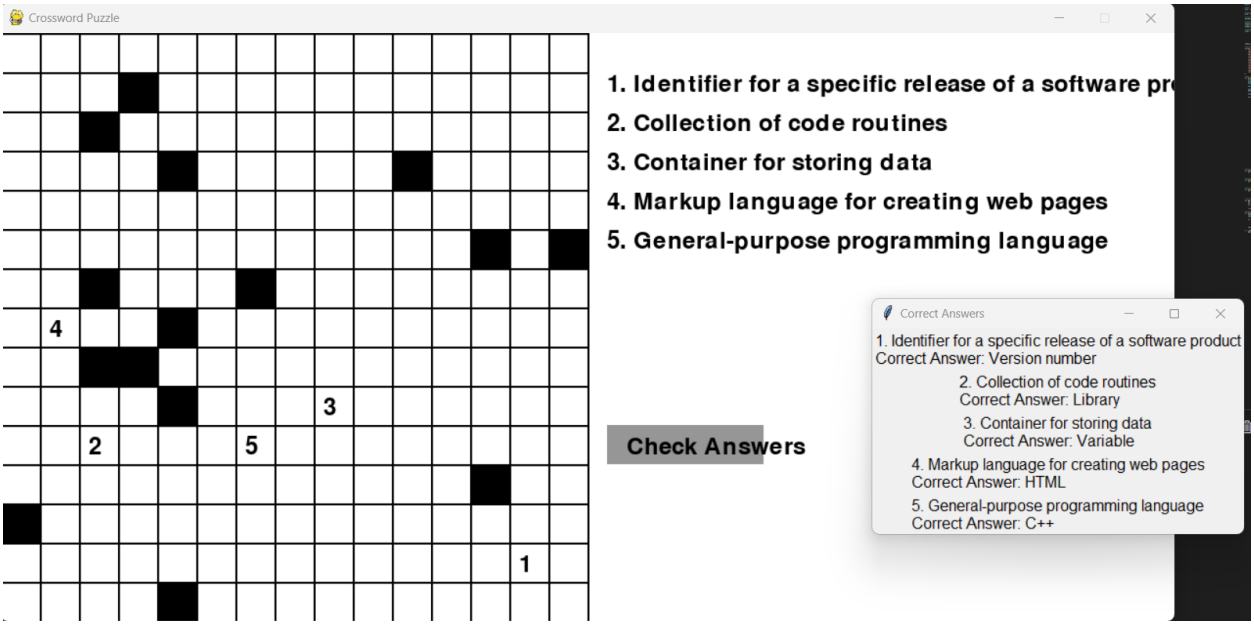
```python
import pygame
import tkinter as tk
from tkinter import messagebox
import random
import sys


# Initialize Pygame
pygame.init()
pygame.font.init()

# Constants
WINDOW_WIDTH, WINDOW_HEIGHT = 1200, 600
GRID_SIZE = 15
CELL_SIZE = 40
GRID_WIDTH = GRID_SIZE * CELL_SIZE
CLUES_WIDTH = WINDOW_WIDTH - GRID_WIDTH

# Colors
WHITE = (255, 255, 255)
BLACK = (0, 0, 0)

# Font
font = pygame.font.Font(None, 36)

# Initialize grid
grid = [[' ' for _ in range(GRID_SIZE)] for _ in range(GRID_SIZE)]

# Initialize blocked cells
blocked_cells = random.sample([(i, j) for i in range(GRID_SIZE) for j in range(GRID_SIZE)], 15)

# Generate random clue positions
valid_clue_positions = [(i, j) for i in range(GRID_SIZE) for j in range(GRID_SIZE) if (i, j) not in blocked_cells]
random.shuffle(valid_clue_positions)
random_clue_positions = valid_clue_positions[:5]

# Generate random clues
clues = ["A popular programming language", "Object-oriented programming language",
```

```python
clues = ["A popular programming language", "Object-oriented programming language",
         "General-purpose programming language", "Markup language for creating web pages",
         "Style sheet language for web development", "Scripting language for web development",
         "Organized collection of data", "Step-by-step procedure or formula",
         "Finding and fixing errors in code", "Container for storing data",
         "Reusable piece of code", "Point of interaction with software or hardware",
         "Foundation for developing software", "Identifier for a specific release of a software product",
         "Collection of code routines"]
random_clues = random.sample(clues, 5)

# Correct answers
correct_answers = {
    "A popular programming language": "Python",
    "Object-oriented programming language": "Java",
    "General-purpose programming language": "C++",
    "Markup language for creating web pages": "HTML",
    "Style sheet language for web development": "CSS",
    "Scripting language for web development": "JavaScript",
    "Organized collection of data": "Database",
    "Step-by-step procedure or formula": "Algorithm",
    "Finding and fixing errors in code": "Debugging",
    "Container for storing data": "Variable",
    "Reusable piece of code": "Function",
    "Point of interaction with software or hardware": "Interface",
    "Foundation for developing software": "Framework",
    "Identifier for a specific release of a software product": "Version number",
    "Collection of code routines": "Library"
}

# Main game loop
def main():
    global grid
    screen = pygame.display.set_mode((WINDOW_WIDTH, WINDOW_HEIGHT))
    pygame.display.set_caption("Crossword Puzzle")

    clock = pygame.time.Clock()
```

```
        screen.fill(WHITE)

        # Draw grid
        for i in range(GRID_SIZE + 1):
            pygame.draw.line(screen, BLACK, (0, i * CELL_SIZE), (GRID_WIDTH, i * CELL_SIZE), 2)
            pygame.draw.line(screen, BLACK, (i * CELL_SIZE, 0), (i * CELL_SIZE, GRID_WIDTH), 2)

        # Draw clues
        draw_clues(screen)

        # Draw blocked cells
        for i, j in blocked_cells:
            pygame.draw.rect(screen, BLACK, (j * CELL_SIZE, i * CELL_SIZE, CELL_SIZE, CELL_SIZE))

        # Draw clue numbers
        draw_clue_numbers(screen)

        # Draw grid letters
        draw_grid_letters(screen)

        # Draw check answers button
        pygame.draw.rect(screen, (150, 150, 150), check_button_rect)
        text = font.render("Check Answers", True, BLACK)
        screen.blit(text, (GRID_WIDTH + 40, 410))

        pygame.display.flip()
        clock.tick(30)

# Function to draw the clues
def draw_clues(screen):
    for i, clue in enumerate(random_clues, 1):
        text = font.render(f"{i}. {clue}", True, BLACK)
        screen.blit(text, (GRID_WIDTH + 20, i * 40))

# Function to draw the clue numbers
def draw_clue_numbers(screen):
```

```
# Function to draw the clue numbers
def draw_clue_numbers(screen):
    for clue_num, (i, j) in enumerate(random_clue_positions, 1):
        text = font.render(str(clue_num), True, BLACK)
        screen.blit(text, (j * CELL_SIZE + CELL_SIZE // 2 - 10, i * CELL_SIZE + CELL_SIZE // 2 - 10))

# Function to draw grid letters
def draw_grid_letters(screen):
    for i in range(GRID_SIZE):
        for j in range(GRID_SIZE):
            text = font.render(grid[i][j], True, BLACK)
            screen.blit(text, (j * CELL_SIZE + CELL_SIZE // 2 - 10, i * CELL_SIZE + CELL_SIZE // 2 - 10))

# Function to get the cell that was clicked
def get_clicked_cell(pos):
    row = pos[1] // CELL_SIZE
    col = pos[0] // CELL_SIZE
    if (row, col) not in blocked_cells:
        return row, col
    return None

# Function to display correct answers in Tkinter window
def display_answers():
    root = tk.Tk()
    root.title("Correct Answers")

    for i, clue in enumerate(random_clues, 1):
        correct_answer = correct_answers.get(clue, "Answer not available")
        label = tk.Label(root, text=f"{i}. {clue}\nCorrect Answer: {correct_answer}", anchor="w", justify="left", font=('Arial', 12))
        label.pack()

    root.mainloop()

if __name__ == "__main__":
    main()

```

OUTPUT


Crossword Puzzle
1. Identifier for a specific release of a software pr
2. Collection of code routines
3. Container for storing data
4. Markup language for creating web pages
5. General-purpose programming language
Check Answers
Correct Answers
1. Identifier for a specific release of a software product
Correct Answer: Version number
2. Collection of code routines
Correct Answer: Library
3. Container for storing data
Correct Answer: Variable
4. Markup language for creating web pages
Correct Answer: HTML
5. General-purpose programming language
Correct Answer: C++