# Bomb Defuser Game - Complete Guide

A Python debugging game where players defuse bombs by fixing code errors. Perfect for learning Python programming and debugging skills!

## 🚀 Installation & Setup

### Prerequisites

- Python 3.8 or higher
- pip (Python package manager)

### Step 1: Install Dependencies

```bash
pip install -r requirements.txt
```

Required packages:

- `PyQt5==5.15.10` - GUI framework
- `PyQt5-svg==5.15.6` - SVG support for graphics
- `Pygments==2.17.2` - Syntax highlighting
- `PyInstaller==6.3.0` - For creating executable (optional)

### Step 2: Test Installation

Run the test script to verify all components work:

```bash
python test_game.py
```

You should see:

```
✓ ALL TESTS PASSED - Game should work correctly!
```

### Step 3: Launch the Game

```bash
```

```
python bomb_defuser.py
```

## 🎮 How to Play

1. **Start Screen**: Click "🚀 START GAME" to begin
2. **Game Interface**:
   - **Left Panel**: Code editor with broken Python code
   - **Right Panel**: Bomb with 4 colored wires and countdown timer
3. **Your Mission**: Fix the broken code to defuse the bomb before time runs out!
4. **Submit Code**: Click "🚀 TEST CODE" to validate your solution
5. **Success**: All 4 wires get cut when you fix all bugs correctly
6. **Failure**: Timer expires = bomb explodes! Use "🔄 RESTART LEVEL" to try again

### Game Features

- **10 Progressive Levels**: From basic syntax to advanced statistics
- **Hint System**: Hints appear when 80% of timer has elapsed
- **Error Feedback**: Shows specific errors when code fails validation
- **Wire Cutting Animation**: Visual feedback for successful solutions

## 📚 Level Guide & Solutions

### Level 1: Quadratic Discriminant

**Difficulty**: Beginner | **Timer**: 60 seconds

**Problem**: Calculate discriminant of ax$^2$ + bx + c = 0

| Line | Broken Code | Issue | Correct Code |
|---|---|---|---|
| 2 | `def calculate_discriminant(a, b, c)` | Missing colon | `def calculate_discriminant(a, b, c):` |

### Level 2: Linear Equation Solver

**Difficulty**: Beginner | **Timer**: 90 seconds

**Problem**: Solve linear equation ax + b = 0

| Line | Broken Code | Issue | Correct Code |
|---|---|---|---|
| 9 | `print(f"{a}x + {b} = 0 => x = {solution}"` | Missing closing parenthesis | `print(f"{a}x + {b} = 0 => x = {solution}")` |

### Level 3: Prime Number Checker

**Difficulty**: Beginner | **Timer**: 120 seconds

**Problem**: Check if a number is prime

| Line | Broken Code | Issue | Correct Code |
|---|---|---|---|
| 5 | `for i in range(2, int(n**0.5) + 1)` | Missing colon | `for i in range(2, int(n**0.5) + 1):` |

### Level 4: Greatest Common Divisor

**Difficulty**: Intermediate | **Timer**: 180 seconds

**Problem**: Find GCD using Euclidean algorithm

| Line | Broken Code | Issue | Correct Code |
|---|---|---|---|
| - | No actual errors | Testing validation | Code is already correct |

*Note: This level tests the validation system with correct code*

### Level 5: Fibonacci Sequence

**Difficulty**: Intermediate | **Timer**: 240 seconds

**Problem**: Generate Fibonacci numbers

| Line | Broken Code | Issue | Correct Code |
|---|---|---|---|
| 12 | `fib_seq = [next_fib]` | Replacing list instead of appending | `fib_seq.append(next_fib)` |

### Level 6: Statistical Calculations

**Difficulty**: Intermediate | **Timer**: 300 seconds

**Problem**: Calculate mean, median, and variance

| Line | Broken Code | Issue | Correct Code |
|---|---|---|---|
| 16 | `n == len(sorted_nums)` | Equality operator instead of assignment | `n = len(sorted_nums)` |
| 23 | `return sorted_nums[n//2 + 1]` | Wrong median index for odd length | `return sorted_nums[n//2]` |
| 32 | `return sum(squared_diffs) / (len(numbers) - 1)` | Sample variance instead of population | `return sum(squared_diffs) / len(numbers)` |

## Level 7: Matrix Operations

**Difficulty**: Advanced | **Timer**: 375 seconds

**Problem**: Perform matrix multiplication and transpose

| Line | Broken Code | Issue | Correct Code |
|---|---|---|---|
| 11 | `C = [[0 for _ in range(rows_A)] for _ in range(cols_B)]` | Wrong result matrix dimensions | `C = [[0 for _ in range(cols_B)] for _ in range(rows_A)]` |
| 16 | `C[j][i] += A[i][k] * B[k][j]` | Swapped indices in assignment | `C[i][j] += A[i][k] * B[k][j]` |
| 25 | `transposed = [[0 for _ in range(cols)] for _ in range(rows)]` | Wrong transpose dimensions | `transposed = [[0 for _ in range(rows)] for _ in range(cols)]` |
| 29 | `transposed[i][j] = matrix[j][i]` | Swapped transpose indices | `transposed[j][i] = matrix[i][j]` |

## Level 8: Standard Deviation Calculator

**Difficulty**: Advanced | **Timer**: 450 seconds

**Problem**: Calculate population standard deviation and Z-scores

| Line | Broken Code | Issue | Correct Code |
|---|---|---|---|
| 6 | `variance = sum(squared_diffs) / (len(numbers) - 1)` | Sample variance instead of population | `variance = sum(squared_diffs) / len(numbers)` |
| 23 | `z_scores = [(x - mean) / variance for x in numbers]` | Using variance instead of std_dev | `z_scores = [(x - mean) / std_dev for x in numbers]` |
| 49 | `print(f"  Z-scores: {[round(z, 2) for z in z_scores}")` | Missing closing bracket | `print(f"  Z-scores: {[round(z, 2) for z in z_scores]}")` |
| Missing | No division by zero check | Missing defensive programming | Add `if std_dev == 0: return [0] * len(numbers)` |

## Level 9: Correlation Coefficient

**Difficulty**: Advanced | **Timer**: 525 seconds

**Problem**: Calculate Pearson correlation coefficient

| Line | Broken Code | Issue | Correct Code |
|---|---|---|---|
| 21 | `numerator = sum((x - mean_x) * y - mean_y for x, y in zip(x_values, y_values))` | Missing parentheses in calculation | `numerator = sum((x - mean_x) * (y - mean_y) for x, y in zip(x_values, y_values))` |
| 26 | `denominator = (sum_sq_x + sum_sq_y)**0.5` | Wrong correlation formula | `denominator = (sum_sq_x * sum_sq_y)**0.5` |
| 73 | `print(f"  Correlation: {[r:.4f if r else 'None'}")` | Wrong bracket type | `print(f"  Correlation: {r:.4f if r else 'None'}")` |

## Level 10: Advanced Statistics Suite

**Difficulty**: Expert | **Timer**: 600 seconds

**Problem**: Complete statistical analysis class with multiple bugs

| Line | Broken Code | Issue | Correct Code |
|---|---|---|---|
| 15 | `self.mean = sum(self.data) / self.n` | Division by zero risk | `self.mean = self.calculate_mean() if self.n > 0 else 0` |
| 19 | `self.std_dev = math.sqrt(self.variance)` | Using potentially invalid variance | `self.std_dev = math.sqrt(self.variance) if self.variance >= 0 else 0` |
| 31 | `divisor = self.n if sample else self.n - 1` | Inverted sample/population logic | `divisor = self.n - 1 if sample else self.n` |
| 42-45 | `python<br/>for x in self.data:<br/> for _ in range(1): # unnecessary<br/> result += ((x - self.mean) / self.std_dev) ** 3` | Unnecessary nested loop in skewness | `python<br/>cubed_diffs = [((x - self.mean) / self.std_dev) ** 3 for x in self.data]<br/>return sum(cubed_diffs) / self.n` |
| 54-57 | `python<br/>for x in self.data:<br/> for _ in range(2): # double counting!<br/> result += ((x - self.mean) / self.std_dev) ** 4` | Nested loop double-counts values | `python<br/>fourth_diffs = [((x - self.mean) / self.std_dev) ** 4 for x in self.data]<br/>return (sum(fourth_diffs) / self.n) - 3` |
| 75 | `return sorted_data[lower_index] * weight + sorted_data[upper_index] * (1 - weight)` | Reversed interpolation weights | `return sorted_data[lower_index] * (1 - weight) + sorted_data[upper_index] * weight` |
| 82 | `if self.n < 2:` missing in confidence_interval | No edge case handling | `if self.n < 2 or self.std_dev == 0: return (self.mean, self.mean)` |
| 22 | `return sum(self.data) / self.n` in calculate_mean | Division by zero risk | `return sum(self.data) / self.n if self.n > 0 else 0` |
| 28 | `if divisor == 0:` check placement | Wrong condition handling | `if self.n < 2: return 0` (before divisor calculation) |
| 37 | Missing `if self.std_dev == 0 or self.n < 2:` in skewness | No edge case handling | Add `if self.std_dev == 0 or self.n < 2: return 0` |
| 48 | Missing `if self.std_dev == 0 or self.n < 2:` in kurtosis | No edge case handling | Add `if self.std_dev == 0 or self.n < 2: return 0` |
| 64-66 | Missing single-element handling in percentile | Index out of bounds risk | Add `if self.n == 1: return sorted_data[0]` |

## 🎯 Tips for Success

### General Debugging Strategy

1. **Read the Error Messages**: They often point to the exact problem
2. **Check Syntax First**: Missing colons, parentheses, brackets
3. **Understand the Algorithm**: Know what the code should accomplish
4. **Test Edge Cases**: Consider empty lists, single elements, zero values
5. **Use Print Statements**: Add debugging output to understand data flow

### Common Python Bugs by Category

**Syntax Errors** (Levels 1-3):

- Missing colons after function definitions and loops
- Missing closing parentheses or brackets
- Typos in keywords

**Logic Errors** (Levels 4-6):

- Wrong operators (`=` vs `==`)
- Incorrect index calculations
- Off-by-one errors in loops

**Mathematical Errors** (Levels 7-10):

- Sample vs population formulas
- Swapped variables in calculations
- Missing edge case handling

### Level-Specific Hints

**Level 6**: Remember that median calculation differs for odd vs even length arrays **Level 7**: Pay attention to matrix dimension ordering (rows × columns) **Level 8**: Population variance divides by N, sample variance by N-1 **Level 9**: Correlation requires proper parentheses in the numerator calculation **Level 10**: Class-based code needs defensive programming for edge cases

## 🛠️ Troubleshooting

### Common Installation Issues

**PyQt5 Installation Problems**:

```bash
# Try installing with specific version
pip install PyQt5==5.15.10

# On Ubuntu/Debian, might need system packages
sudo apt-get install python3-pyqt5
```

**Import Errors**:

- Make sure you're in the correct directory with all game files
- Check that Python can find all modules: `python -c "import PyQt5; print('PyQt5 OK')"`

### Game Issues

**Game Won't Start**:

1. Run `python test_game.py` to identify the problem
2. Check that all required files are present
3. Verify Python version: `python --version` (needs 3.8+)

**Code Editor Not Working**:

- Try clicking in the editor area to focus
- Make sure you're typing valid Python syntax
- Use the restart button if editor becomes unresponsive

**Timer Issues**:

- Hints appear at 80% of countdown time
- Timer is automatically doubled from original specs for better gameplay

## 🏆 Completion Certificate

Congratulations on defusing all 10 bombs! You've mastered:

- ✅ Python syntax debugging
- ✅ Algorithm implementation
- ✅ Mathematical programming
- ✅ Statistical calculations
- ✅ Object-oriented programming

- ✅ Edge case handling

Share your achievement: "I successfully completed all 10 levels of the Bomb Defuser Python debugging game! 🎉 "

---

**Game Version**: 1.0
**Created for**: Coding Club Educational Presentations
**GitHub**: [Repository Link]

*Happy debugging!* 💻 💣