# Gabby Strevay

# Nakia Patton

# Rae Yoshioka

# Coding Style Guide: Backyard Hunt Project

# 1. Naming Conventions

• **Variables and Functions**: Use descriptive, lowercase names with underscores (snake\_case). Examples: total\_score, calculate\_average. • **Classes**: Use first word lowercase and rest CapitalizedWords (CamelCase). Example: friendshipScoreCalculator.

# 2. Indentation and Spacing

* Use **4 spaces per indentation level**. Avoid mixing tabs and spaces.
* Keep lines under **79 characters** to improve readability.
* Separate functions and class definitions with **two blank lines**; inside functions, separate logical sections with a single blank line.

# 3. Comments and Documentation

* **Docstrings**: Include **a docstring** at the start of each function and class. Briefly explain what it does, the parameters, and the return value. def calculate\_score(name1, name2):

"""Calculate friendship score based on names.

Args:

name1 (str): First name. name2 (str): Second name.

Returns:

int: Friendship score.

"""

* **Inline Comments**: Use comments sparingly to clarify complex code, but avoid obvious comments. Example:

# Convert names to lowercase for case-insensitive

comparison name1 = name1.lower()

# 4. Code Structure and Organization

* **Imports**: Place all imports at the beginning of the file, in the order of standard libraries, third-party libraries, and local modules.
* **Functions**: Write modular functions that perform single tasks. Avoid overly long or complex functions.

# 5. Functions

* **Placement**: Define all functions at the top of the program or in a separate script/module if organizing larger projects. If calling functions within the same script, place the main code that calls these functions at the end, after all function definitions.
* **Single Responsibility**: Each function should perform a single, clear task. Avoid making functions overly complex.

# 6. Error Handling

• Anticipate possible errors and handle them gracefully (e.g., input validation).

1. **Consistency** 
   * Consistency is key. If working in a group, ensure everyone follows the same naming, formatting, and commenting conventions.
   * Adhere to either this guide or a chosen external style guide (e.g., **PEP 8** for Python, **Google C++ Style Guide** for C++), and maintain these conventions throughout the project.
2. **File Organization**

• Name files descriptively and avoid spaces in filenames. Use underscores instead (e.g., friendship\_score\_calculator.py).

* + Include a comment at the top of each file with the author(s) and a brief description of the file's purpose.

# 9. Programming Best Practices

* Avoid hardcoding values; use constants where appropriate.
* Test your code incrementally. Regularly review and refine code to improve readability and efficiency.
* Ensure variable and function names reflect their purpose to enhance clarity for future readers