Python Library Style Guide

1. File Header

- Each Python file in the library **must** begin with the following header:

```
 #! /usr/bin/env python
# -*- coding: utf-8 -*-

##############################################################################
##  -- PhyNetPy --
##  Library for the Development and use of Phylogenetic Network Methods
##
##  Copyright 2025 Mark Kessler, Luay Nakhleh.
##  All rights reserved.
##
##  See "LICENSE.txt" for terms and conditions of usage.
##
##  If you use this work or any portion thereof in published work,
##  please cite it as:
##
##     Mark Kessler, Luay Nakhleh. 2025.
##
##############################################################################
```

2. Function and Method Signatures

- Each function/method signature **must** include type hints for all arguments and the return type.

3. Documentation

- Each function and class **must** have a docstring in the following format:

```
 """
<method description>

Raises:
    SomeTypeOfError: <why the error might be raised> (only if applicable)

Args:
    param1 (param1 type): <param1 description>.
    param2 (param2 type): <param2 description>.

Returns:
    returntype: <return type description>.
"""
```

Example:

```
def add_numbers(a: int, b: int) -> int:
    """
    Adds two numbers and returns the result.

    Args:
        a (int): The first number.
        b (int): The second number.

    Returns:
        int: The sum of the two numbers.
    """
    return a + b
```

4. Code Commenting

- Functions/methods **must** be sufficiently commented.
- If a function contains more than **5 lines of code**, it **must** include directives explaining the logic.

5. Type Hinting

- It is **recommended** but **not required** to type hint simple variables such as `int`, `str`, `float`, or `bool`.
- It is **required** to type hint complex variables that use generics, such as:
  - `list[str]`
  - `dict[tuple[int, str], float]`

6. Line Length

- Each line of code **must not** exceed **80 characters**.

7. Line Breaking and Alignment

- If a line exceeds 80 characters:
  - Function signatures and similar structures should align each subsequent line **with the opening parenthesis**.

  Example:

```
def example_function(param1: str,
                     param2: int,
                     param3: list[str]
) -> dict[str, int]:
    """Example function description."""
    pass
```

8. Imports

- Imports should usually be on **separate lines**:

  Correct:

```
import os
import sys
```

  Wrong:

```
import sys, os
```

- It's acceptable to import multiple names from a module in a single line:

  Correct:

```
from subprocess import Popen, PIPE
```

- Imports should always be placed at the **top of the file**, just after any module comments and docstrings, and before module globals and constants.
- Imports should be grouped in the following order:
  1. **Standard library imports**
  2. **Related third-party imports**
  3. **Local application/library-specific imports**
- There should be a **blank line** between each group of imports.

9. Naming Conventions

- **Variable Names**: Should be **lowercased** and use **snake_case**.

  Correct:

```
user_name = "Alice"
total_count = 42
```

- **Constants**: Should be in **ALL CAPS**.

  Correct:

```
MAX_CONNECTIONS = 100
API_KEY = "abc123"
```

- **Class Names**: Should use **CamelCase** with each word capitalized and no underscores.

  Correct:

```
class UserProfile:
    pass
```

10. General Conventions

- In all other general cases, the **PEP 8 style guide** should be followed.
- This includes, but is not limited to:
    - Proper indentation (4 spaces per indentation level).
    - Avoiding extraneous whitespace.
    - Using meaningful variable names.
    - Limiting the use of inline comments.
    - Ensuring consistent formatting for readability and maintainability.
- When in doubt, refer to the PEP 8 documentation for best practices.