The **Google Python Style Guide** is a widely accepted set of conventions that guide Python code formatting, documentation, and style. It promotes readability and consistency, making code easier to understand and maintain. Here's a summary of key aspects of the guide, along with examples:

1. **Naming Conventions**

- **Variable names**: Use `snake_case` for variable names.
- **Function names**: Use `snake_case` for function names.
- **Class names**: Use `PascalCase` (or `CamelCase`) for class names.
- **Constants**: Use `ALL_CAPS` for constants.

Example:

```
MAX_COUNT = 10  # Constant

def get_user_data(user_id):  # Function
    user_name = "John"  # Variable
    return user_name

class UserProfile:  # Class
    pass
```

2. **Indentation and Line Length**

- Use **4 spaces** per indentation level.
- Limit all lines to **80 characters**.

Example:

```
def process_data(data_list):
    for item in data_list:
        if item is None:
            continue
        print(item)
```

3. **Docstrings**

- Use **triple double quotes (`"""`)** for docstrings.
- The **first line** of a docstring should be a short description.
- If more explanation is needed, include additional details after a blank line.

Example:

```
def add_numbers(a, b):
    """Add two numbers and return the result.

    Args:
        a (int): First number.
        b (int): Second number.

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

4. **Imports**

- Group imports into three sections:
    1. **Standard library imports**.
    2. **Related third-party imports**.
    3. **Local application imports**.
- Each section should be separated by a blank line.

Example:

```
import os  # Standard library
import requests  # Third-party

from myproject.models import User  # Local module
```

5. **Whitespace**

- Avoid extraneous whitespace:
    - Inside parentheses, brackets, or braces.
    - Before a comma, colon, or semicolon.
    - At the end of a line.
- Use a single space around binary operators (`=`, `+`, etc.).

Example:

```
x = (1 + 2) * (3 + 4)
```

6. **Comments**

- **Inline comments** should be used sparingly and begin with `#`, followed by a space.

- **Block comments** should be indented at the same level as the code they refer to.

Example:

```
# This is a block comment explaining the following code
result = add_numbers(5, 7)  # Inline comment explaining this line
```

7. **Exceptions**

- Use specific exceptions rather than a generic `except` clause.
- Always include an error message when raising exceptions.

Example:

```
try:
    result = 1 / 0
except ZeroDivisionError as e:
    print(f"Error: {e}")
```

8. **Type Hints**

- Google Python style encourages the use of **type hints** for better code clarity and static analysis.

Example:

```
def add(a: int, b: int) -> int:
    """Adds two integers."""
    return a + b
```

9. **Comprehensions**

- Use **list comprehensions** or **generator expressions** where appropriate, but avoid overly complex comprehensions that reduce readability.

Example:

```
squares = [x * x for x in range(10)]
```

10. **Trailing Commas**

- Trailing commas should be included when the closing container is on a separate line from the last element, for better diffs.

Example:

```
my_list = [
    1,
    2,
    3,
]
```

By following the Google Python Style Guide, developers can write cleaner, more maintainable code that is easier for others to read and contribute to.

Reference Used: https://code.google.com/archive/p/soc/wikis/PythonStyleGuide.wiki (https://code.google.com/archive/p/soc/wikis/PythonStyleGuide.wiki)