Coding Standard For Project

Python Naming Conventions Guideline

(Based on PEP 8 — the Official Python Style Guide)

1 General Rules

- Use **descriptive, meaningful names**.
- Avoid **abbreviations** unless widely known (url, id, db are fine).
- Use **lowercase letters**, **underscores**, and **capitalization** consistently.
- Names should **reflect purpose**, not implementation details.

2 Naming Styles Overview

Element	Convention	Example	Notes
Variables	snake_case	total_amount, user_name	Use lowercase words separated by underscores.
Functions	snake_case	calculate_total() , get_data()	Use verbs to describe actions.

Classes	PascalCase	UserProfile, DataProcessor	Each word capitalized, no underscores.
Constants	ALL_CAPS	MAX_RETRIES, DEFAULT_PATH	Use for values that should not change.
Modules / Files	lowercase_with_underscores	data_loader.py, config_parser.py	Keep short and descriptive.
Packages / Directories	lowercase	utilities, core	Avoid special characters.
Private Members	_leading_underscore	_helper_function, _internal_var	Indicates internal use only.
Strongly Private (Name Mangling)	__double_leading_underscore	__password	Used inside classes to avoid accidental override.
Global Variables	snake_case	global_cache, app_version	Use only when necessary.
Instance Variables	snake_case	self.user_name	Same rule as normal variables.

Class Variables	snake_case	counter = 0	Declared inside class but outside methods.

3️⃣ Special Naming Rules

- **Avoid single-letter names** (l, O, I) — easily confused with digits.
- **Booleans:** start with is_, has_, or can_ → is_active, has_error.
- **Exceptions:** class names should end with Error → FileNotFoundError.
- **Functions returning bool:** use question-like phrasing → is_valid(), has_access().

4️⃣ Pros & Cons

✅ Pros	❌ Cons
Makes code consistent and professional	Can feel strict for quick prototypes
Improves readability and collaboration	Requires discipline across teams
Easier debugging and maintenance	May take time to learn naming rules

5 Quick Example

```
class StudentRecord:
    MAX_SCORE = 100

    def __init__(self, name, student_id):
        self.name = name
        self.student_id = student_id

    def calculate_grade(self, score):
        """Return grade based on score."""
        if score >= 90:
            return "A"
        elif score >= 80:
            return "B"
        else:
            return "C"
```