Programming Project Guidelines

AnGp211 - Major project programming

IPSA – M.AMIR MOAZAMI / 2024

1

DEVELOPMENT LIFECYCLE

The software development cycle

2

MVC

Model-View-Controller

3

UML DIAGRAMS

Designing the architecture of a project

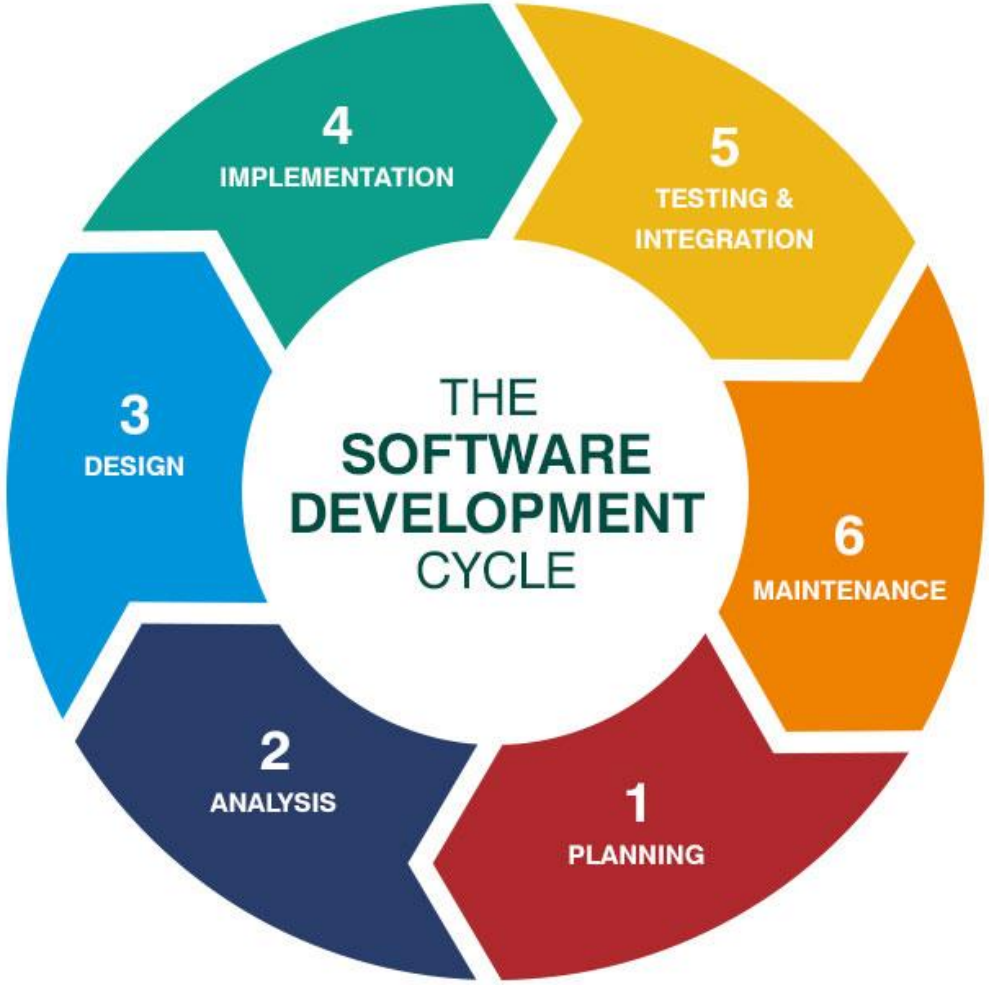
4

CODING RULES

PEP 8 – Style Guide for Python Code

DEVELOPMENT LIFECYCLE

Software development life cycle (SDLC)
is a structured process that is used to design, develop, and test good-quality software.



DEVELOPMENT STAGES

PLANNING

- Define objectives
- Identify resources
- Establish timelines

REQUIREMENT ANALYSIS

- Gather requirements : Clarify what the project should accomplish, including inputs, outputs, and any constraints.
- Break down the project: Divide the project into smaller, manageable tasks.

DESIGN

- Design the architecture: Plan how the program will be structured (e.g., classes, functions, and modules). If a user interface is involved, sketch or design it.
- Select technologies: Decide on tools, libraries, and frameworks (like Python, Tkinter, or others) to use.

DEVELOPMENT STAGES

IMPLEMENTATION (DEVELOPMENT PHASE)

- Set up the environment: Install Python, libraries.
- Start coding based on the planned structure.
- Ensure all parts of the code are well-commented for future reference.

TESTING & DEBUGGING

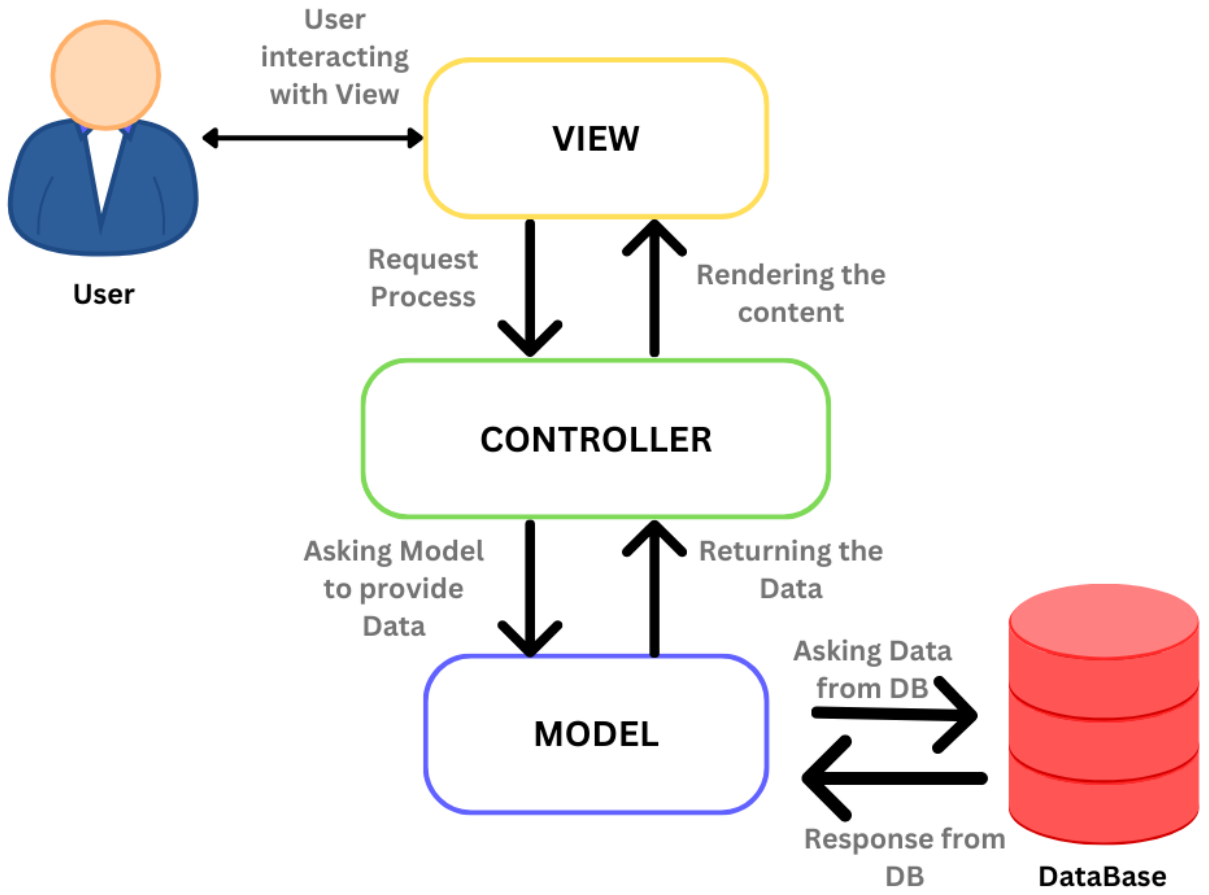
- Perform system testing
- Identify and fix bugs

DEPLOYMENT AND MAINTENANCE

- Provide ongoing support
- Perform updates and improvements
- Monitor performance

SOFTWARE ARCHITECTURE : MVC

MVC(Model-View-Controller) architecture is a fundamental design pattern in software development, separating an application into Model, View, and Controller components.



SOFTWARE ARCHITECTURE : MVC

- **What Is MVC?**

MVC is a software architecture approach, helps to separate the logic of the program, making it easier to develop, debug, and maintain.

It divides the responsibilities of the system into three distinct parts:

- **Model:** The model holds the state information of the system.
- **View:** The view presents the model information to the user.
- **Controller:** The controller makes sure that user commands are executed correctly, modifying the appropriate model objects, and updating the view objects.

MVC : MODEL

What Goes in the Model ?

- A model in MVC represents the data. A model deals with getting data from or writing data into storage such as a database or file.
- The model may also contain the logic to validate the data to ensure data integrity.
- The model must not depend on the view and controller. In other words, you can reuse the model in other non-Tkinter applications such as web and mobile apps.

MVC : VIEW

What Goes in the View ?

- The view is how the model is presented and interacted with by the user.
- A view is the user interface that represents the data in the model.
- The view doesn’t directly communicate with the model, it communicates with the controller directly.
- In Tkinter applications, the view is the root window that consists of widgets.

MVC : CONTROLLER

What Goes in the Controller ?

- A controller acts as the intermediary between the views and models.
- Handles the user input and updates both the Model (data) and the View (display).
- For example, if users click the save button on the view, the controller routes the “save” action to the model to save the data into a database and notify the view to display a message.

SOFTWARE ARCHITECTURE : UML

UML (Unified modeling language)

is a standardized general-purpose visual modeling language in the field of Software Engineering. It is used for specifying, visualizing, constructing, and documenting the primary artifacts of the software system.

SOFTWARE ARCHITECTURE : UML

In development projects, using UML diagrams and activity diagrams can help developers better visualize and structure their systems.

These are essential for:

- Designing the architecture of a project before coding.
- Communicating how different components of the project interact.
- Organizing the workflow and understanding relationships between classes, objects, and interactions.

UML : CLASS DIAGRAM

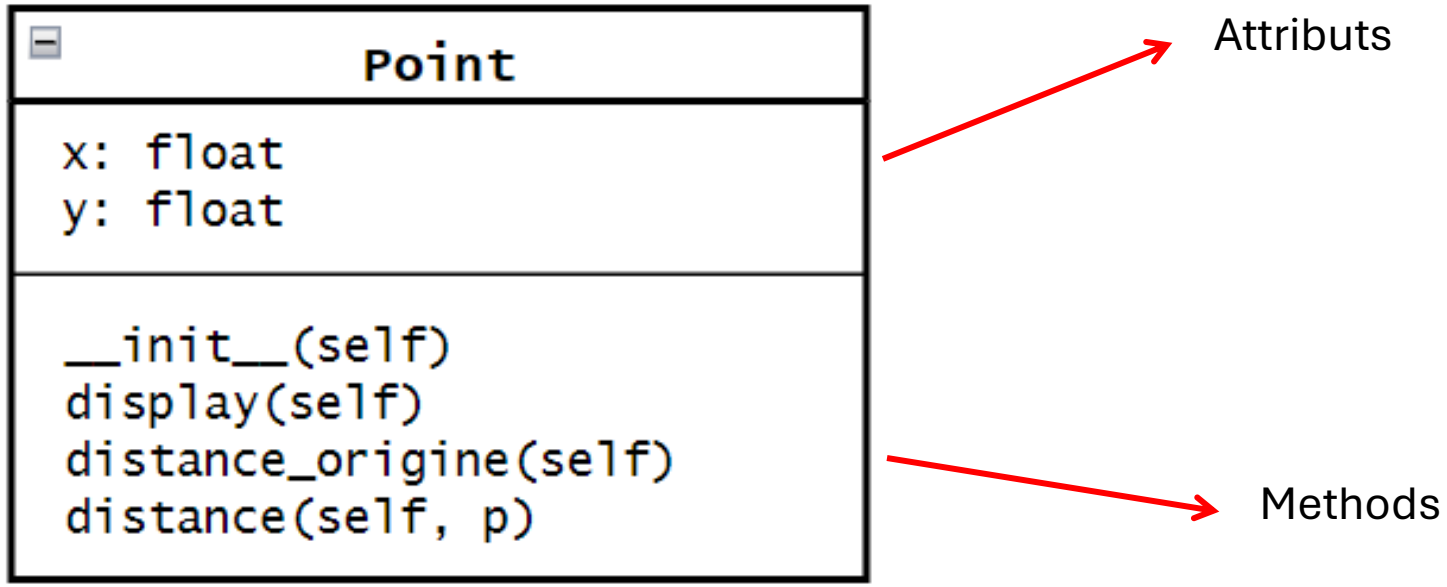
The UML Class diagram is a graphical notation used to construct and visualize object-oriented systems.

A class diagram in the Unified Modeling Language (UML) is a type of static structure diagram that describes the structure of a system by showing the system's:

- Classes
- Attributes
- operations (or methods)
- the relationships among objects

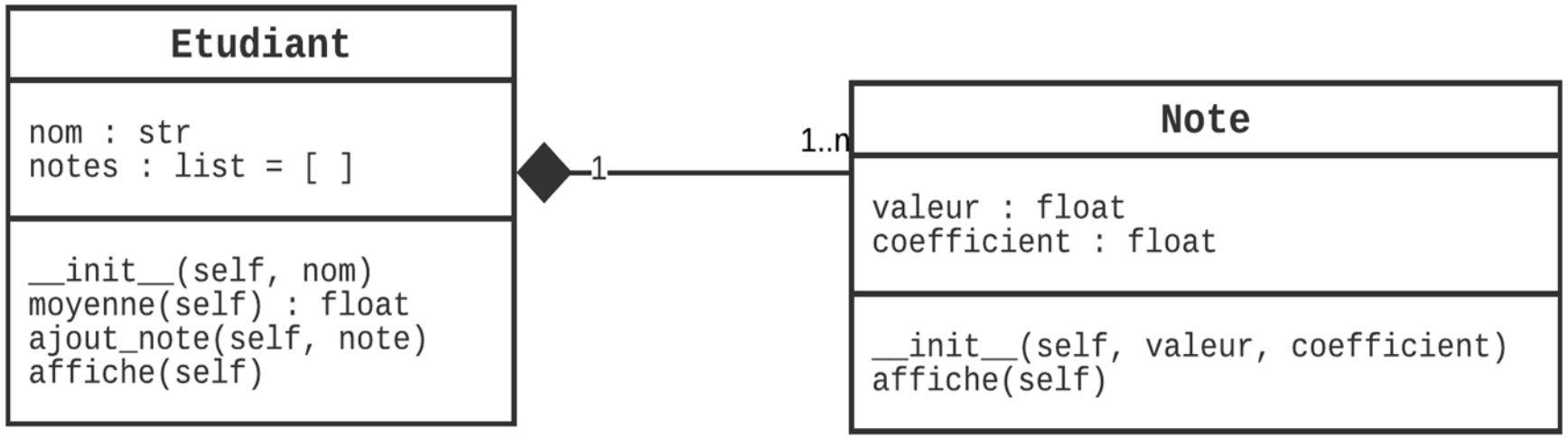
UML : CLASS DIAGRAM

- UML example: class Point



UML : CLASS DIAGRAM

- UML example : composition



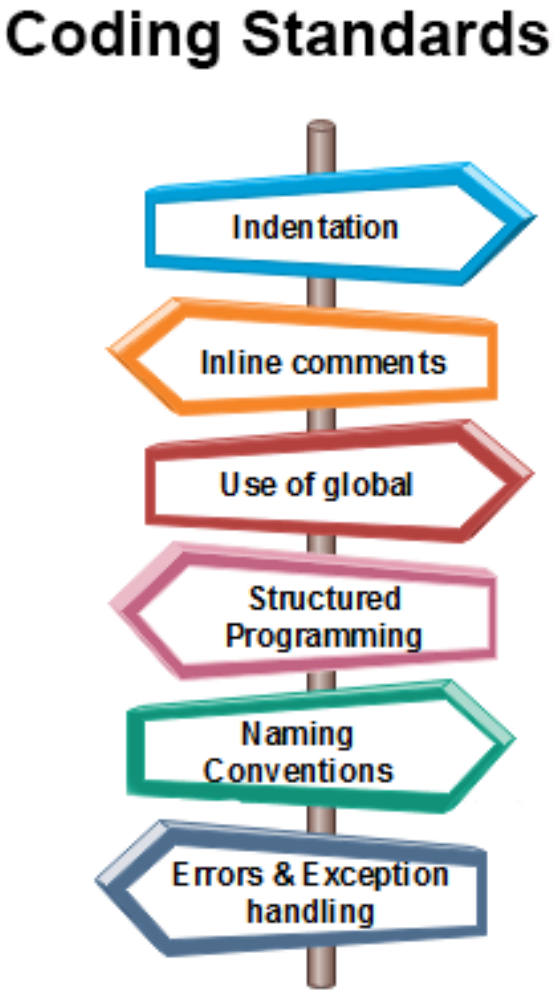
- Attributes `notes` of the class `Etudiant` is composed of `1-n` object `Note`

CODING RULES

Why we do we use coding rules :

By following these best practices, you contribute to the creation of high-quality software that is easier to understand, maintain, and extend.

Following these standards helps ensure consistency across projects and make code more understandable for others and for future.



CODING RULES : PEP 8

Indentation

The preferred method of indentation is spaces, the 4 spaces indentation is accepted and accurate, but still, most people prefer tab indentation. Please keep in mind not to mix both spaces and tabs for indentation.

Line Length:

Your source code should not contain lines that are longer than 80 characters long. If you have a line that is longer than 80 character, break it up into multiple lines. Use the `'\'` character at the end of a line to tell the Python interpreter that the statement continues the next line

CODING RULES : PEP 8

Naming Conventions

Use grammatically correct variable names, the class name should start with an ***uppercase*** and must follow camelCase convention if more than two words are to be used.

In the same way, a function name should be joined with an ***underscore***, and it must be ***lowercase***.

In method arguments, always use self as the first argument to declare an instance variable.

CODING RULES : PEP 8

Comment your code!

- File Comments: Every .py file should have a high-level comment at the top describing the file's contents and should include your name(s) and the date.
- Function Comments: Every function should have a comment describing:
  - what function does
  - what its parameter values are
  - what value(s) it returns
  - If a function has some tricky code, then use in-line comments to explain what it is doing.

Example :

```
def square_the_biggest(n1, n2):
    """
      Function: square_the_biggest
      ----------------------------
        Returns the square of the largest of its two input values
        n1: one real value
        n2: the other real value
        returns: the square of the larger of n1 and n2
    """
```

CODING RULES : PEP 8

In-line Comments :

Use inline comments sparingly.

- An inline comment is a comment on the same line as a statement. Inline comments should be separated by at least two spaces from the statement. They should start with a # and a single space.
- Inline comments are unnecessary and in fact distracting if they state the obvious. Don't do this:

```
x = x + 1                 # Increment x
```

But sometimes, this is useful:

```
x = x + 1                 # Compensate for border
```

Class Comments:

- Every Class should have a high-level comment describing what it does, and each of its method functions should have a comments similar to those of regular functions.

CODING RULES : PEP 8

Documentation Strings:

Conventions for writing good documentation strings (a.k.a. “docstrings”) are immortalized in PEP 257.

- Write docstrings for all public modules, functions, classes, and methods. Docstrings are not necessary for non-public methods, but you should have a comment that describes what the method does. This comment should appear after the def line.
- Note that most importantly, the """ that ends a multiline docstring should be on a line by itself:

```
"""Return a foobang

Optional plotz says to frobnicate the bizbaz first.
"""
```

- For one liner docstrings, please keep the closing """ on the same line:

```
"""Return an ex-parrot."""
```

Sources & ressources

- Style Guide for Python Code
  - https://www.python.org/dev/peps/pep-0008/
  - https://llego.dev/posts/writing-clean-pep-8-compliant-code-better-collaboration/

- UML Diagrams
  - https://drawio-app.com/
  - https://www.visual-paradigm.com/guide/uml-unified-modeling-language/what-is-uml/#:~:text=UML%2C%20short%20for%20Unified%20Modeling,business%20modeling%20and%20other%20non%2D