# The are 3 types of using copiots for generating codes

## Using suggestion realtime

Please refer to Copilot handons and challenge

***https://gitlab.com/phantichhoang/copilot-training***

## Using chat prompt

Some useful chat prompts

**1. Basic Class Template**

**Prompt:**

Create a class in [**language**] called [**ClassName**] that implements the following methods:

- `method1`: [brief description of method1 functionality]

- `method2`: [brief description of method2 functionality]

The class should have a constructor that initializes [***attributes***] and handle [***input validation, error handling, etc.]***.

**Example:**

Create a class in Python called `Employee` that implements the following methods:

- `get\_full\_name`: returns the full name by combining first and last names.

- `get\_annual\_salary`: calculates the employee's annual salary based on monthly salary.

The class should have a constructor that initializes first name, last name, and monthly salary and handle missing values by setting defaults.

**2. Function with Error Handling**

**Prompt:**

Write a function in [**language**] called `[**function\_name**]` that:

- Takes [parameters] as inputs.

- Performs [task/operation].

- Implements error handling for [error types].

- Returns [output].

**Example:**

Write a function in Python called `read\_file` that:

- Takes a file path as input.

- Opens the file and reads the content.

- Implements error handling for file not found, permission denied, and invalid file formats.

- Returns the content of the file or an error message.

**3. Data Processing Module**

**Prompt:**

Create a [**language**] module that processes data from [**data source or format, e.g., CSV, JSON, API**].

The module should:

- Load data from [source] using [library or method].

- Clean and preprocess the data by [describe operations: handling missing values, normalizing, etc.].

- Return [desired output].

**Example:**

Create a Python module that processes data from a CSV file.

The module should:

- Load data using pandas.

- Handle missing values by filling them with the column mean.

- Normalize the numeric columns.

- Return the cleaned DataFrame.

**4. API Module Creation**

**Prompt:**

Create an API module in [language] that performs the following actions:

- Connects to an external API (e.g., [API name]).

- Sends a [GET/POST] request to retrieve data from [endpoint].

- Handles possible errors like network issues or invalid responses.

- Returns the data in [format].

**Example:**

Create an API module in Python that:

- Connects to the GitHub API.

- Sends a GET request to retrieve public repositories for a given user.

- Handles network errors, invalid API keys, and rate limits.

- Returns the repository data in JSON format.

**5. Database Interaction Module**

**Prompt:**

Write a module in [language] that interacts with a [database type, e.g., MySQL, SQLite].

The module should:

- Connect to the database.

- Perform basic CRUD operations (Create, Read, Update, Delete).

- Handle database errors and return appropriate messages or results.

**Example:**

Write a Python module that interacts with an SQLite database.

The module should:

- Connect to the SQLite database.

- Perform basic CRUD operations on a table called `users` (Create user, Read user info, Update user data, Delete user).

- Handle errors such as connection failures, SQL injection, and invalid queries.

**Example:**

Write a Python module that interacts with an SQLite database.

The module should:

- Connect to the SQLite database.

- Perform basic CRUD operations on a table called `users` (Create user, Read user info, Update user data, Delete user).

- Handle errors such as connection failures, SQL injection, and invalid queries.

**6. Unit Testing Template**

Create unit tests in [testing framework] for the module [module name]. The test cases should:

- Verify the correct output for [functions/methods].

- Test edge cases for [functionality].

- Ensure error handling works correctly for invalid inputs.

**Example:**

Create unit tests in PyTest for the `math\_operations.py` module. The test cases should:

- Verify correct outputs for `add`, `subtract`, `multiply`, and `divide` functions.

- Test edge cases like dividing by zero.

- Ensure error handling works when non-numeric inputs are passed.

**7. Event-Driven Programming Template**

**Prompt:**

Create an event-driven system in [language] where:

- Events [event1, event2, etc.] trigger specific functions.

- Define event handlers for each event.

- The system logs each event and its outcome.

**Example:**

Create an event-driven system in JavaScript where:

- Events like 'click', 'hover', and 'submit' trigger specific functions.

- Define event handlers for each event.

- The system logs the event name and the outcome in the console.

**8. Logging Module Template**

**Prompt:**

Write a logging module in [language] that:

- Creates logs for different levels (INFO, WARNING, ERROR).

- Allows for custom log messages with timestamps.

- Stores the logs in a [file, database, or remote server].

**Example:**

Write a logging module in Python that:

- Creates logs for different levels (INFO, WARNING, ERROR).

- Adds timestamps to each log message.

- Stores the logs in a file called `app.log`.

These prompt templates can be further adjusted to fit the specific requirements of the code module you need. If you let me know the language and the specific task, I can help refine these prompts further!

## From UML to code

Need all UML define in XML format (continue researching)

A screenshot of a computer program

Description automatically generated