Pylint is a widely used static code analysis tool for Python. Its primary purpose is to help developers ensure their Python code adheres to a consistent coding standard and follows best practices. Pylint analyzes Python code to detect and report various types of issues such as syntax errors, coding style violations, potential bugs, and other code quality problems.

Here are some key features and functionalities of Pylint:

1. Static Code Analysis: Pylint performs static analysis of Python code without actually executing it. This means it examines the code structure, syntax, and patterns to identify issues.

2. Coding Standard Enforcement: Pylint checks code against a predefined set of coding standards. By default, it follows the PEP 8 style guide, which is the official style guide for Python code. However, it's highly customizable and can be configured to enforce different coding standards or customized rules.

3. Error Detection: Pylint detects various types of errors, including syntax errors, logical errors, uninitialized variables, unused variables, and other potential sources of bugs.

4. Code Style Checking: In addition to enforcing coding standards, Pylint checks for adherence to code styling conventions. It ensures consistent formatting, such as indentation, line lengths, and naming conventions.

5. Code Complexity Analysis: Pylint measures code complexity using metrics such as cyclomatic complexity, which helps identify overly complex code that may be difficult to maintain or understand.

6. Integration with Development Workflow: Pylint can be integrated into various development workflows, including text editors, integrated development environments (IDEs), continuous integration (CI) systems, and version control systems. This allows developers to receive real-time feedback on code quality and style as they write code.

7. Customization: Pylint is highly customizable, allowing developers to configure it according to their specific project requirements. This includes adjusting severity levels for different types of issues, defining custom coding standards, and excluding certain files or directories from analysis.

Overall, Pylint is a valuable tool for Python developers to maintain code quality, enforce coding standards, and identify potential issues early in the development process, ultimately leading to more robust and maintainable codebases.

This is a GitHub Actions workflow YAML file that sets up a continuous integration (CI) pipeline for a Python project using Pylint for code analysis. Let's break down each part of the workflow:

```yaml
name: Pylint
```
This sets the name of the workflow to "Pylint".

```yaml
on: [push]
```
This specifies that the workflow should be triggered on every push event to the repository.

```yaml
jobs:
  build:
    runs-on: ubuntu-latest
```
Defines a job named "build" that will run on an Ubuntu environment.

```yaml
strategy:
  matrix:
    python-version: ["3.8", "3.9", "3.10", "3.11"]
```
Configures a matrix strategy where the job will run for each Python version specified in the array: 3.8, 3.9, 3.10, and 3.11.

```yaml
steps:
  - uses: actions/checkout@v3
```
This step checks out the code from the repository into the runner's workspace.

```yaml
  - name: Set up Python ${{ matrix.python-version }}
    uses: actions/setup-python@v3
    with:
      python-version: ${{ matrix.python-version }}
```
Sets up the specified Python version in the matrix for the job using the setup-python action provided by GitHub Actions.

```yaml
```

```
  - name: Install dependencies
    run: |
      python -m pip install --upgrade pip
      pip install pylint
```
Installs dependencies required for the job, including upgrading pip and installing Pylint using pip.

```yaml
  - name: Analysing the code with pylint
    run: |
      pylint $(git ls-files '*.py')
```
This step executes the Pylint command to analyze Python files in the repository. `git ls-files '*.py'` lists all Python files in the repository, and then Pylint is executed on those files. Pylint will check for syntax errors, code style violations, and other issues in the Python codebase.

Overall, this workflow sets up a CI pipeline that runs Pylint code analysis on each push event for Python code using different Python versions specified in the matrix.

The specific rules of pylint can be found in it documentation here:
https://pylint.pycqa.org/en/latest/index.html