

# Code Regulations

Without Changes:

The screenshot shows the PyCharm IDE interface. On the left is the Project tool window displaying a file structure for 'G87.2026.T69.GE1' containing files like .gitignore, \_\_init\_\_.py, EnterpriseManagementException.py, EnterpriseManager.py, EnterpriseRequest.py, and pylintrc. The main editor window on the right contains Python code for 'EnterpriseManager.py'. The terminal window at the bottom shows the output of running pylint on the code, listing numerous errors related to missing docstrings and naming conventions.

```
Project: G87.2026.T69.GE1 (~/PycharmProjects/G87.2026.T69.GE1)
File: EnterpriseManager.py

1 import json
2 from EnterpriseManagementException import EnterpriseManagementException
3 from EnterpriseRequest import EnterpriseRequest
4
5 PRE = {"J":0, "A":1, "B":2, "C":3, "D":4, "E":5, "F":6, "G":7, "H":8, "I":9}
6
7 class EnterpriseManager: 1 usage & Linus Munn +1*
8     def __init__(self): & Charles2827Mad
9         pass
10
11     @staticmethod 2 usages & Linus Munn +1*
12     def validate_cif(cif):
13         # PLEASE INCLUDE HERE THE CODE FOR VALIDATING THE GUID
14         # RETURN TRUE IF THE GUID IS RIGHT, OR FALSE IN OTHER CASE
15         if len(cif) != 9 or cif[0] not in PRE or not cif[1:].isdigit():
16             return False
17         even_pos_sum = 0
18         odd_sum = 0
19         # Positions 1-7 are digits (skip prefix at position 0)
```

Terminal Local + ~

```
(venv) linusmunn@Linuss-MacBook-Pro G87.2026.T69.GE1 % pylint EnterpriseManager.py
*****
Module G87.2026.T69.GE1.EnterpriseManager
EnterpriseManager.py:1:0: C0114: Missing module docstring (missing-module-docstring)
EnterpriseManager.py:1:0: C0103: Module name "EnterpriseManager" doesn't conform to snake_case naming style (invalid-name)
EnterpriseManager.py:2:0: E0401: Unable to import 'EnterpriseManagementException' (import-error)
EnterpriseManager.py:3:0: E0401: Unable to import 'EnterpriseRequest' (import-error)
EnterpriseManager.py:7:0: C0115: Missing class docstring (missing-class-docstring)
EnterpriseManager.py:12:4: C0116: Missing function or method docstring (missing-function-docstring)
EnterpriseManager.py:40:4: C0116: Missing function or method docstring (missing-function-docstring)
EnterpriseManager.py:43:17: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
```

## Rule 1: Max Line Length

- Rule name: max-line-length
- Description: Maximum number of characters allowed per line of code.
- Original value: 100 characters
- New value: 120 characters
- Reasoning: Prevent lines from becoming overly long and hard to read.

- Positive Example:

```

10 @staticmethod 2 usages & Linus Munn +1*
11
12 def validate_cif(cif):
13     # PLEASE INCLUDE HERE THE CODE FOR VALIDATING THE GUID
14     # RETURN TRUE IF THE GUID IS RIGHT, OR FALSE IN OTHER CASE
15     if len(cif) != 9 or cif[0] not in PRE or not cif[1:].isdigit():
16         return False
17     even_pos_sum = 0
18     odd_sum = 0
19     # Positions 1-7 are digits (skip prefix at position 0)
20     for position in range(1, 8):
21         num = int(cif[position])
22
23         if position % 2 == 0: # even position
24             even_pos_sum += num
25         else: # odd position
26             scaled = num * 2

```

Terminal Local + ▾

```

(venv) linusmunn@Linus-MacBook-Pro: G87.2026.T69.GE1 % pylint EnterpriseManager.py
***** Module G87.2026.T69.GE1.EnterpriseManager
EnterpriseManager.py:1:0: C0114: Missing module docstring (missing-module-docstring)
EnterpriseManager.py:1:0: C0103: Module name "EnterpriseManager" doesn't conform to snake_case naming style (invalid-name)
EnterpriseManager.py:2:0: E0401: Unable to import 'EnterpriseManagementException' (import-error)
EnterpriseManager.py:3:0: E0401: Unable to import 'EnterpriseRequest' (import-error)
EnterpriseManager.py:7:0: C0115: Missing class docstring (missing-class-docstring)
EnterpriseManager.py:12:4: C0116: Missing function or method docstring (missing-function-docstring)
EnterpriseManager.py:40:4: C0116: Missing function or method docstring (missing-function-docstring)
EnterpriseManager.py:43:17: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)

```

- Negative Example:

```

11 @staticmethod & Linus Munn +1*
12 R> def validate_cif_with_a_function_name_that_is_absolutely_most_definitely_way_too_long_and_should_be_shortened_now(cif):
13     # PLEASE INCLUDE HERE THE CODE FOR VALIDATING THE GUID
14     # RETURN TRUE IF THE GUID IS RIGHT, OR FALSE IN OTHER CASE
15     if len(cif) != 9 or cif[0] not in PRE or not cif[1:].isdigit():
16         return False
17     even_pos_sum = 0
18     odd_sum = 0
19     # Positions 1-7 are digits (skip prefix at position 0)
20     for position in range(1, 8):
21         num = int(cif[position])
22
23         if position % 2 == 0: # even position
24             even_pos_sum += num
25         else: # odd position
26             scaled = num * 2

```

Terminal Local + ▾

```

(venv) linusmunn@Linus-MacBook-Pro: G87.2026.T69.GE1 % pylint EnterpriseManager.py
***** Module G87.2026.T69.GE1.EnterpriseManager
EnterpriseManager.py:12:0: C0301: Line too long (123/120) (line-too-long)

```

## Rule 2: Max Args per function

- Rule name: max-args
- Description: Maximum number of arguments a function can take.
- Original value: 5 arguments
- New value: 3 arguments
- Reasoning: Limit number of arguments to prevent an overly long list of parameters.

- Positive Example

```

11     @staticmethod 2 usages & Linus Munn +1 *
12     def validate_cif(cif):
13         # PLEASE INCLUDE HERE THE CODE FOR VALIDATING THE GUID
14         # RETURN TRUE IF THE GUID IS RIGHT, OR FALSE IN OTHER CASE
15         if len(cif) != 9 or cif[0] not in PRE or not cif[1:].isdigit():
16             return False
17         even_pos_sum = 0
18         odd_sum = 0
19         # Positions 1-7 are digits (skip prefix at position 0)
20         for position in range(1, 8):
21             num = int(cif[position])
22
23             if position % 2 == 0: # even position
24                 even_pos_sum += num
25             else: # odd position
26                 scaled = num * 2

```

terminal Local + ▾

```

venv) linusmunn@Linuss-MacBook-Pro 687.2026.T69.GE1 % pylint EnterpriseManager.py
*****
Module G87.2026.T69.GE1.EnterpriseManager
EnterpriseManager.py:1:0: C0114: Missing module docstring (missing-module-docstring)
EnterpriseManager.py:1:0: C0103: Module name "EnterpriseManager" doesn't conform to snake_case naming style (invalid-name)
EnterpriseManager.py:2:0: E0401: Unable to import 'EnterpriseManagementException' (import-error)
EnterpriseManager.py:3:0: E0401: Unable to import 'EnterpriseRequest' (import-error)
EnterpriseManager.py:7:0: C0115: Missing class docstring (missing-class-docstring)
EnterpriseManager.py:12:4: C0116: Missing function or method docstring (missing-function-docstring)
EnterpriseManager.py:40:4: C0116: Missing function or method docstring (missing-function-docstring)
EnterpriseManager.py:43:17: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)

```

- Negative Example:

```

7     class EnterpriseManager: 1 usage & Linus Munn +1 *
8
9
10    @staticmethod 2 usages & Linus Munn +1 *
11    def validate_cif(cif, a, b, c):
12        # PLEASE INCLUDE HERE THE CODE FOR VALIDATING THE GUID
13        # RETURN TRUE IF THE GUID IS RIGHT, OR FALSE IN OTHER CASE
14        a = a
15        b = b
16        c = c
17        print(a, b, c)
18        if len(cif) != 9 or cif[0] not in PRE or not cif[1:].isdigit():
19            return False
20        even_pos_sum = 0
21        odd_sum = 0
22        # Positions 1-7 are digits (skip prefix at position 0)
23        for position in range(1, 8):
24            num = int(cif[position])

```

Terminal Local + ▾

Your code has been rated at 6.67/10 (previous run: 4.38/10, +2.29)

```

(venv) linusmunn@Linuss-MacBook-Pro 687.2026.T69.GE1 % pylint EnterpriseManager.py
*****
Module G87.2026.T69.GE1.EnterpriseManager
EnterpriseManager.py:1:0: C0114: Missing module docstring (missing-module-docstring)
EnterpriseManager.py:1:0: C0103: Module name "EnterpriseManager" doesn't conform to snake_case naming style (invalid-name)
EnterpriseManager.py:2:0: E0401: Unable to import 'EnterpriseManagementException' (import-error)
EnterpriseManager.py:3:0: E0401: Unable to import 'EnterpriseRequest' (import-error)
EnterpriseManager.py:7:0: C0115: Missing class docstring (missing-class-docstring)
EnterpriseManager.py:12:4: C0116: Missing function or method docstring (missing-function-docstring)
EnterpriseManager.py:12:4: R0913: Too many arguments (4/3) (too-many-arguments)

```

### Rule 3: Max number of Statements

- Rule name: max-statements

- Description: Maximum number of statements allowed in a function.
- Original value: 50 statements
- New value: 20 statements
- Reasoning: Keep functions short, easy to read/understand, and easy to test.
- Positive Example:

```

.venv  terminal Local + 
venv) linusmunn@linus-MacBook-Pro:~/Desktop/PycharmProjects/EnterpriseManager % pylint EnterpriseManager.py
*****
Module 687.2026.T69.GE1 EnterpriseManager
EnterpriseManager.py:1:0: C0114: Missing module docstring (missing-module-docstring)
EnterpriseManager.py:1:0: C0103: Module name "EnterpriseManager" doesn't conform to snake_case naming style (invalid-name)
EnterpriseManager.py:2:0: E0401: Unable to import 'EnterpriseManagementException' (import-error)
EnterpriseManager.py:3:0: E0401: Unable to import 'EnterpriseRequest' (import-error)
EnterpriseManager.py:7:0: C0115: Missing class docstring (missing-class-docstring)
EnterpriseManager.py:12:4: C0116: Missing function or method docstring (missing-function-docstring)
EnterpriseManager.py:40:4: C0116: Missing function or method docstring (missing-function-docstring)
EnterpriseManager.py:43:17: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)

```

- Negative Example:

```

.venv  terminal Local + 
EnterpriseManager.py:1:0: C0114: Missing module docstring (missing-module-docstring)
EnterpriseManager.py:1:0: C0103: Module name "EnterpriseManager" doesn't conform to snake_case naming style (invalid-name)
EnterpriseManager.py:2:0: E0401: Unable to import 'EnterpriseManagementException' (import-error)
EnterpriseManager.py:3:0: E0401: Unable to import 'EnterpriseRequest' (import-error)
EnterpriseManager.py:7:0: C0115: Missing class docstring (missing-class-docstring)
EnterpriseManager.py:12:4: C0116: Missing function or method docstring (missing-function-docstring)
EnterpriseManager.py:12:4: R0915: Too many statements (21/20) (too-many-statements)

```

## Rule 4: Max module lines

- Rule name: max-module-lines
- Description: Maximum number of lines allowed in a module
- Original value: 1000 lines
- New value: 75 (later changed to 80) lines
- Reasoning: Encourage breaking up the code into multiple modules if it exceeds a certain length.
- Positive Example:

```
class EnterpriseManager: 1 usage & Linus Munn +1*
    def __init__(self):  & Charles2827Mad
        pass

    @staticmethod 2 usages & Linus Munn +1*
    def validate_cif(cif):
        # PLEASE INCLUDE HERE THE CODE FOR VALIDATING THE GUID
        # RETURN TRUE IF THE GUID IS RIGHT, OR FALSE IN OTHER CASE
        if len(cif) != 9 or cif[0] not in PRE or not cif[1:].isdigit():
            return False

Your code has been rated at 6.46/10 (previous run: 6.67/10, -0.21)

(venv) linusmann@Linuss-MacBook-Pro G87.2026.T69.GE1 % pylint EnterpriseManager.py
***** Module 687.2026.T69.GE1.EnterpriseManager
EnterpriseManager.py:1:0: C0114: Missing module docstring (missing-module-docstring)
EnterpriseManager.py:1:0: C0103: Module name "EnterpriseManager" doesn't conform to snake_case naming style (invalid-name)
EnterpriseManager.py:4:0: E0401: Unable to import 'EnterpriseManagementException' (import-error)
EnterpriseManager.py:5:0: E0401: Unable to import 'EnterpriseRequest' (import-error)
```

- Negative Example:

```
def validate_cif(cif):
    print(cif)
    print(cif)
    print(cif)

    # PLEASE INCLUDE HERE THE CODE FOR VALIDATING THE GUID
    # RETURN TRUE IF THE GUID IS RIGHT, OR FALSE IN OTHER CASE
    if len(cif) != 9 or cif[0] not in PRE or not cif[1:].isdigit():
        return False
    even_pos_sum = 0
    odd_sum = 0
    # Positions 1-7 are digits (skip prefix at position 0)
    for nposition in range(1, 8):
        if nposition % 2 == 0:
            even_pos_sum += int(cif[nposition])
        else:
            odd_sum += int(cif[nposition])

        if even_pos_sum > 9 or odd_sum > 9:
            return False

    if even_pos_sum == odd_sum:
        return True
    else:
        return False

Your code has been rated at 6.46/10 (previous run: 6.67/10, -0.21)

() linusmann@Linuss-MacBook-Pro G87.2026.T69.GE1 % pylint EnterpriseManager.py
***** Module 687.2026.T69.GE1.EnterpriseManager
EnterpriseManager.py:1:0: C0302: Too many lines in module (76/75) (too-many-lines)
```

#### Rule 5: Allow unused global variable

- Rule name: allow-global-unused-variables
  - Description: Allow global variables to go unused.
  - Original value: yes
  - New value: no
  - Reasoning: Prevent wasted or useless lines from clogging a file.
  - Positive Example:

The screenshot shows the PyCharm IDE interface. The left sidebar displays a project structure with files like `__init__.py`, `EnterpriseManagementException.py`, and `EnterpriseRequest.py`. The main code editor window contains Python code for an `EnterpriseManager` class. The terminal at the bottom shows the output of running `pylint enterprise_manager.py`, displaying various error and warning messages related to imports and docstrings.

```
G87.2026.T69.GE1 ~/PycharmProjects/G87.2026
1     """EnterpriseManager model contains the logic for CIF validation and JSON processing."""
2
3     import json
4     from EnterpriseManagementException import EnterpriseManagementException
5     from EnterpriseRequest import EnterpriseRequest
6
7     PRE = [{"J":0, "A":1, "B":2, "C":3, "D":4, "E":5, "F":6, "G":7, "H":8, "I":9}]
8
9     class EnterpriseManager: 1 usage  &Linus Munn +1
10        """EnterpriseManager class contains the logic for CIF validation."""
11
12        def __init__(self):  &Charles2827Mad
13            pass
14
15        @staticmethod 2 usages  &Linus Munn +1
16        def validate_cif(cif):
17
18            # PLEASE INCLUDE HERE THE CODE FOR VALIDATING THE GUID
19            # RETURN TRUE IF THE GUID IS RIGHT, OR FALSE IN OTHER CASE
```

Terminal Local + ▾

```
7bb09b7..823295b main -> main
(venv) linusmunn@Linus-MacBook-Pro G87.2026.T69.GE1 % pylint enterprise_manager.py
*****
Module G87.2026.T69.GE1.enterprise_manager
enterprise_manager.py:4:0: E0401: Unable to import 'EnterpriseManagementException' (import-error)
enterprise_manager.py:5:0: E0401: Unable to import 'EnterpriseRequest' (import-error)
enterprise_manager.py:16:4: C0116: Missing function or method docstring (missing-function-docstring)
enterprise_manager.py:45:4: C0116: Missing function or method docstring (missing-function-docstring)
enterprise_manager.py:48:17: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
```

- #### - Negative Example:

```
priseManagementException.py
priseRequest.py
trc
Libraries
es and Consoles

PRE = {"J":0, "A":1, "B":2, "C":3, "D":4, "E":5, "F":6, "G":7, "H":8, "I":9}
A = 2

class EnterpriseManager: 1 usage & Linus Munn +1 *
    """EnterpriseManager class contains the logic for CIF validation."""

    def __init__(self): & Charles2827Mad
        pass

    @staticmethod 2 usages & Linus Munn +1 *
    def validate_cif(cif):
        # PLEASE INCLUDE HERE THE CODE FOR VALIDATING THE GUID

Local x + v
```

husmunn@Linus-MacBook-Pro:~/Desktop\$ pylint enterprise\_manager.py

\*\*\* Module G87.2026.T69.GE1.enterprise\_manager

e\_manager.py:4:0: E0401: Unable to import 'EnterpriseManagementException' (import-error)

e\_manager.py:5:0: E0401: Unable to import 'EnterpriseRequest' (import-error)

e\_manager.py:17:4: C0116: Missing function or method docstring (missing-function-docstring)

e\_manager.py:46:4: C0116: Missing function or method docstring (missing-function-docstring)

e\_manager.py:49:17: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)

e\_manager.py:8:0: W0612: Unused variable 'A' (unused-variable)

## Rule 6:

- Rule name: disable
- Description: Disables the import-error message
- Original value: disable = ... null
- New value: disable = ... import-error
- Reasoning: Brings pylint up positively by 2.08 points
- Positive Example:

The screenshot shows the PyCharm IDE interface. The project structure on the left includes files like `__init__.py`, `enterprise_manager.py`, `EnterpriseManagementException.py`, `EnterpriseRequest.py`, and `pylintrc`. The code editor window displays `enterprise_manager.py` with the following content:

```
"""EnterpriseManager model contains the logic for CIF validation and JSON processing."""
import json
from EnterpriseManagementException import EnterpriseManagementException
from EnterpriseRequest import EnterpriseRequest

PRE = {"J":0, "A":1, "B":2, "C":3, "D":4, "E":5, "F":6, "G":7, "H":8, "I":9}

class EnterpriseManager: 1 usage & Linus Munn +1
    """EnterpriseManager class contains the logic for CIF validation."""

    def __init__(self): 2 usages & Charles2827Mad
        pass

    @staticmethod 2 usages & Linus Munn +1
    def validate_cif(cif):
```

The terminal at the bottom shows the command `pylint enterprise_manager.py` and its output:

```
charlessimons@Charless-MacBook-Pro-2:~/PycharmProjects/G87.2026.T69.GE1$ pylint enterprise_manager.py
***** Module /Users/charlessimons/PycharmProjects/G87.2026.T69.GE1/pylintrc
pylintrc:1:0: E0015: Unrecognized option found: check-fixture-in-docstring (unrecognized-option)
***** Module G87.2026.T69.GE1.enterprise_manager
enterprise_manager.py:4:4: C0116: Missing function or method docstring (missing-function-docstring)
enterprise_manager.py:45:4: C0116: Missing function or method docstring (missing-function-docstring)
enterprise_manager.py:48:17: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)

Your code has been rated at 9.38/10 (previous run: 7.29/10, +2.08)
```

- Negative Example:

The screenshot shows the PyCharm IDE interface with the same project structure. The code editor window displays `enterprise_manager.py` with the same content as the positive example. The terminal at the bottom shows the command `pylint enterprise_manager.py` and its output:

```
charlessimons@Charless-MacBook-Pro-2:~/PycharmProjects/G87.2026.T69.GE1$ pylint enterprise_manager.py
***** Module /Users/charlessimons/PycharmProjects/G87.2026.T69.GE1/pylintrc
pylintrc:1:0: E0015: Unrecognized option found: check-fixture-in-docstring (unrecognized-option)
pylintrc:1:0: W0512: Unknown option value for '-disable', expected a valid pylint message and got 'use-implicit-booleanness-not-comparison-to-zero'
***** Module G87.2026.T69.GE1.enterprise_manager
enterprise_manager.py:4:0: E0401: Unable to import 'EnterpriseManagementException' (import-error)
enterprise_manager.py:5:0: E0401: Unable to import 'EnterpriseRequest' (import-error)
enterprise_manager.py:16:4: C0116: Missing function or method docstring (missing-function-docstring)
enterprise_manager.py:45:4: C0116: Missing function or method docstring (missing-function-docstring)
enterprise_manager.py:48:17: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)

Your code has been rated at 7.29/10 (previous run: 7.29/10, +0.00)
```

## Rule 7:

- Rule name: docstring-min-length
- Description: Sets whether or not a docstring is required, given the function length
- Original value: docstring-min-length=-1
- New value: docstring-min-length=33
- Reasoning: 30 is the length of our longest function “def validate\_cif(cif):”
- Positive Example:

The screenshot shows the PyCharm IDE interface. The project tree on the left lists files like `enterprise_manager.py`, `pylintrc`, and `requirements.txt`. The main editor window displays the `enterprise_manager.py` code. The terminal at the bottom shows the command `pylint enterprise_manager.py` running and outputting results. The output includes:

```
(.venv) charlessimons@Charless-MacBook-Pro-2 G87.2026.T69.GE1 % pylint enterprise_manager.py
*****
Module /Users/charlessimons/PycharmProjects/G87.2026.T69.GE1/pylintrc
pylintc:1:0: E0015: Unrecognized option found: check-fimxe-in-docstring (unrecognized-option)
*****
Module G87.2026.T69.GE1.enterprise_manager
enterprise_manager.py:17: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)

-----
Your code has been rated at 9.79/10 (previous run: 9.38/10, +0.42)
```

The status bar at the bottom indicates Python 3.9 (G87.2026.T69.GE1).

- Negative Example:

The screenshot shows the PyCharm IDE interface. The project tree on the left lists files like `enterprise_manager.py`, `pylintrc`, and `requirements.txt`. The main editor window displays the `enterprise_manager.py` code. The terminal at the bottom shows the command `pylint enterprise_manager.py` running and outputting results. The output includes:

```
(.venv) charlessimons@Charless-MacBook-Pro-2 G87.2026.T69.GE1 % pylint enterprise_manager.py
*****
Module /Users/charlessimons/PycharmProjects/G87.2026.T69.GE1/pylintrc
pylintc:1:0: E0015: Unrecognized option found: check-fimxe-in-docstring (unrecognized-option)
*****
Module G87.2026.T69.GE1.enterprise_manager
enterprise_manager.py:4: C0110: Missing function or method docstring (missing-function-docstring)
enterprise_manager.py:4: C0110: Missing function or method docstring (missing-function-docstring)
enterprise_manager.py:17: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)

-----
Your code has been rated at 9.38/10 (previous run: 9.38/10, +0.00)
```

The status bar at the bottom indicates Python 3.9 (G87.2026.T69.GE1).

## Rule 8:

- Rule name: disable
- Description: Throws an error if imports are not in the top level
- Original value: disable = ... null
- New value: disable = ... import-outside-toplevel
- Reasoning: An import was made in the validate\_cif() function to use .inspect to determine function length to correctly write docstring-min-length. Brings pylint score up by .2
- Positive Example:

The screenshot shows a PyCharm interface with a pull request for 'enterprise\_manager.py'. The code editor displays the following changes:

```
9  class EnterpriseManager: 2 usages & Linus Munn +1*
10  """EnterpriseManager class contains the logic for CIF validation."""
11  def __init__(self): & Charles2827Mad
12      pass
13
14  @staticmethod 3 usages & Linus Munn +1*
15  def validate_cif(cif):
16      # MODIFIED CODE:
17      import inspect
18      print("validate_cif lines:", len(inspect.getsource(EnterpriseManager.validate_cif).splitlines()))
19
20      # PLEASE INCLUDE HERE THE CODE FOR VALIDATING THE GUID
21      # RETURN TRUE IF THE GUID IS RIGHT, OR FALSE IN OTHER CASE
22
23      if len(cif) != 9 or cif[0] not in PRE or not cif[1].isdigit():
24          return False
25      even_pos_sum = 0
26      odd_sum = 0
27      # Positions 1-7 are digits (skip prefix at position 0)
```

The terminal output shows:

```
charlessimons@Charles-MacBook-Pro-2: 687.2026.T69.GE1 % pylint enterprise_manager.py
***** Module /Users/charlessimons/PycharmProjects/687.2026.T69.GE1/pylintc
E0113:1:0: E0113: Unrecognized option found: check-fixme-in-docstring (unrecognized-option)
***** Module 687.2026.T69.GE1.enterprise_manager
enterprise_manager.py:51:17: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)

Your code has been rated at 9.66/10 (previous run: 9.66/10, +0.20)

charlessimons@Charles-MacBook-Pro-2: 687.2026.T69.GE1 %
```

- Negative Example:

The screenshot shows a PyCharm interface with a pull request for 'enterprise\_manager.py'. The code editor displays the following changes:

```
9  class EnterpriseManager: 2 usages & Linus Munn +1*
10  """EnterpriseManager class contains the logic for CIF validation."""
11  def __init__(self): & Charles2827Mad
12      pass
13
14  @staticmethod 3 usages & Linus Munn +1*
15  def validate_cif(cif):
16      # MODIFIED CODE:
17      import inspect
18      print("validate_cif lines:", len(inspect.getsource(EnterpriseManager.validate_cif).splitlines()))
19
20      # PLEASE INCLUDE HERE THE CODE FOR VALIDATING THE GUID
21      # RETURN TRUE IF THE GUID IS RIGHT, OR FALSE IN OTHER CASE
22
23      if len(cif) != 9 or cif[0] not in PRE or not cif[1].isdigit():
24          return False
25      even_pos_sum = 0
26      odd_sum = 0
27      # Positions 1-7 are digits (skip prefix at position 0)
```

The terminal output shows:

```
charlessimons@Charles-MacBook-Pro-2: 687.2026.T69.GE1 % pylint enterprise_manager.py
***** Module /Users/charlessimons/PycharmProjects/687.2026.T69.GE1/pylintc
E0113:1:0: E0113: Unrecognized option found: check-fixme-in-docstring (unrecognized-option)
***** Module 687.2026.T69.GE1.enterprise_manager
enterprise_manager.py:17:0: C0415: Import outside toplevel (inspect) (import-outside-toplevel)
enterprise_manager.py:51:17: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)

Your code has been rated at 9.46/10 (previous run: 9.66/10, -0.20)

charlessimons@Charles-MacBook-Pro-2: 687.2026.T69.GE1 %
```

## Rule 9:

- Rule name: disable
- Description: Our code uses “open” without specified encoding
- Original value: disable = ... null
- New value: disable = ... unspecified-encoding
- Reasoning: Disabling this in pylint brings our score up by .2
- Positive Example:

The screenshot shows the PyCharm IDE interface with a Python file named `enterprise_manager.py` open. The code contains a try block with an open statement. A yellow circle highlights the `open(fi)` call. The terminal output shows the pylint command being run and a warning about an unrecognized option.

```
charlessimons@CharLess-MacBook-Pro-2:~/PycharmProjects/887.2026.T69.GE1$ pylint enterprise_manager.py
***** Module 887.2026.T69.GE1.enterprise_manager
enterprise_manager.py:51:17: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
```

- Negative Example:

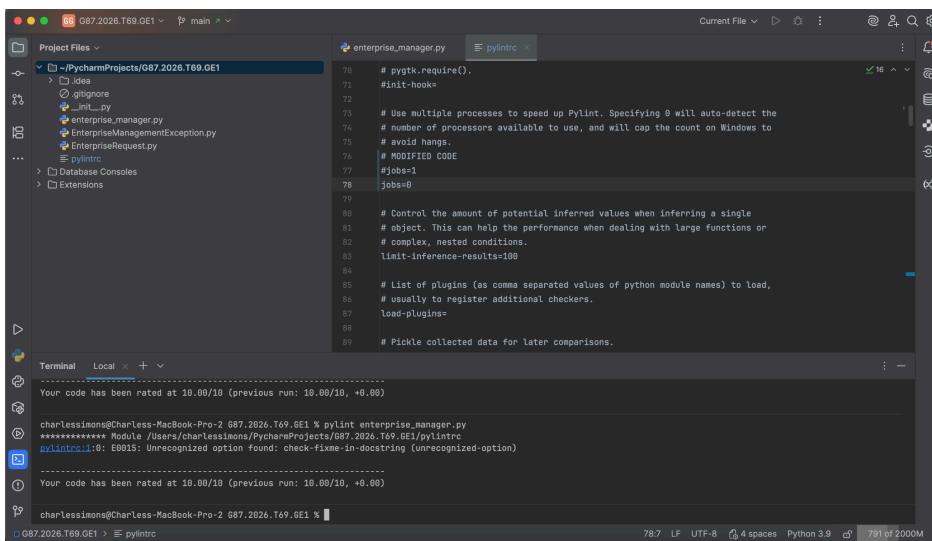
The screenshot shows the PyCharm IDE interface with the same `enterprise_manager.py` file. The code is identical to the positive example, but the `open` statement does not have an encoding parameter. A yellow circle highlights the `open(fi)` call. The terminal output shows the pylint command being run and a warning about an unrecognized option, along with a specific W1514 warning for the missing encoding.

```
charlessimons@CharLess-MacBook-Pro-2:~/PycharmProjects/887.2026.T69.GE1$ pylint enterprise_manager.py
***** Module 887.2026.T69.GE1.enterprise_manager
enterprise_manager.py:51:17: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)

charlessimons@CharLess-MacBook-Pro-2:~/PycharmProjects/887.2026.T69.GE1$ pylint enterprise_manager.py
***** Module 887.2026.T69.GE1.enterprise_manager
enterprise_manager.py:51:17: W1514: Using open without explicitly specifying an encoding (unspecified-encoding)
```

## Rule 10:

- Rule name: jobs
- Description: Determines whether or not multiple processes are used to speed up pylint
- Original value: jobs=1
- New value: jobs=0
- Reasoning: Setting to 0 auto-detects the number of processors available to use.
- Positive Example:



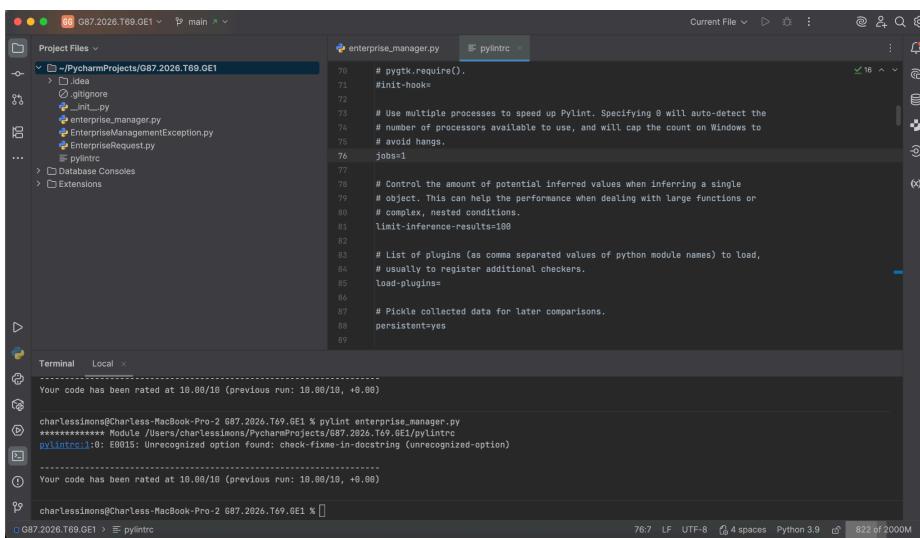
```
# pylint.require().
#init-hook=
#
# Use multiple processes to speed up Pylint. Specifying 0 will auto-detect the
# number of processors available to use, and will cap the count on Windows to
# avoid hangs.
# MODIFIED CODE
#jobs=1
#jobs=0

# Control the amount of potential inferred values when inferring a single
# object. This can help the performance when dealing with large functions or
# complex, nested conditions.
#limit-inference-results=100

# List of plugins (as comma separated values of python module names) to load,
# usually to register additional checkers.
load-plugins=

# Pickle collected data for later comparisons.
persistent=no
```

- Negative Example:



```
# pylint.require().
#init-hook=
#
# Use multiple processes to speed up Pylint. Specifying 0 will auto-detect the
# number of processors available to use, and will cap the count on Windows to
# avoid hangs.
#jobs=1

# Control the amount of potential inferred values when inferring a single
# object. This can help the performance when dealing with large functions or
# complex, nested conditions.
#limit-inference-results=100

# List of plugins (as comma separated values of python module names) to load,
# usually to register additional checkers.
load-plugins=

# Pickle collected data for later comparisons.
persistent=yes
```

## Rule 11:

- Rule name: attr-naming-style
- Description: Sets the naming style for attributes
- Original value: attr-naming-style = snake\_case
- New value: attr-naming-style = camelCase
- Reasoning: Easy rule change, camelCase is preferred
- Positive Example:

The screenshot shows the PyCharm IDE interface. The code editor displays the file `enterprise_manager.py` with the following content:

```
# Naming style matching correct attribute names.
# MODIFIED CODE
# attr-naming-style=snake_case
attr-naming-style=camelCase

# Regular expression matching correct attribute names. Overrides attr-naming-
# style. If left empty, attribute names will be checked with the set naming
# style.
#attr-rgx=

# Bad variable names which should always be refused, separated by a comma.
bad-names=foo,
bar,
baz,
toto,
tutu,
tata

# Bad variable names regexes, separated by a comma. If names match any regex,
# they will always be refused
bad-names-rgxes=
```

The terminal window below shows the command `pylint enterprise_manager.py` being run, resulting in the error `E0015: Unrecognized option found: check-fimxe-in-docstring (unrecognized-option)`.

- Negative Example:

The screenshot shows the PyCharm IDE interface. The code editor displays the file `enterprise_manager.py` with the following content:

```
# Regular expression matching correct argument names. Overrides argument-
# naming-style. If left empty, argument names will be checked with the set
# naming style.
#argument-rgx=

# Naming style matching correct attribute names.
attr-naming-style=snake_case

# Regular expression matching correct attribute names. Overrides attr-naming-
# style. If left empty, attribute names will be checked with the set naming
# style.
#attr-rgx=

# Bad variable names which should always be refused, separated by a comma.
bad-names=foo,
bar,
baz,
toto,
```

The terminal window below shows the command `pylint enterprise_manager.py` being run, resulting in the error `E0015: Unrecognized option found: check-fimxe-in-docstring (unrecognized-option)`.

## Rule 12:

- Rule name: class-naming-style
- Description: Controls the naming convention for classes.
- Original value: PascalCase
- New value: UPPER\_CASE
- Reasoning: We decided it was easier to read the class name using upper case.
- Positive Example:

The screenshot shows the PyCharm IDE interface. The project tree on the left shows files like `__init__.py`, `enterprise_manager.py`, and `EnterpriseManagementException.py`. The code editor window displays `enterprise_manager.py` with the following content:

```
"""EnterpriseManager model contains the logic for CIF validation and JSON processing."""
import json
from EnterpriseManagementException import EnterpriseManagementException
from EnterpriseRequest import EnterpriseRequest

PRE = {"J": "B", "A": "A", "B": "Z", "C": "S", "D": "I", "E": "S", "F": "I", "G": "T", "H": "B", "I": "G"}  
class ENTERPRISE_MANAGER: 2 usages & Linus Munn +2*  
    """EnterpriseManager class contains the logic for CIF validation."""  
    def __init__(self): & Charles287/Mad  
        pass  
  
    @staticmethod 3 usages & Linus Munn +2*  
    def validate_cif(cif):  
        # MODIFIED CODE:  
        import inspect  
        print(validate_cif Lines:, len(inspect.getsource(ENTERPRISE_MANAGER.validate_cif).splitlines()))  
  
        # PLEASE INCLUDE HERE THE CODE FOR VALIDATING THE QUID  
        # RETURN TRUE IF THE QUID IS RIGHT, OR FALSE IN OTHER CASE
```

The terminal window at the bottom shows the command `pylint enterprise_manager.py` being run, resulting in the error `E0015: Unrecognized option found: check-func-in-docstring (unrecognized-option)`. The status bar indicates Python 3.9 (G87.2026.T15.GE1) and 961 of 2200M.

- Negative Example:

The screenshot shows the PyCharm IDE interface. The project tree on the left shows files like `__init__.py`, `enterprise_manager.py`, and `EnterpriseManagementException.py`. The code editor window displays `enterprise_manager.py` with the following content:

```
# class-naming-style = class-const-rgx*  
# Naming style matching correct class names.  
# MODIFIED CODE  
# Class-naming-style=PascalCase  
class-naming-style = UPPER_CASE  
  
# Regular expression matching correct class names. Overrides class-naming-  
# style. If left empty, class names will be checked with the set naming style.  
#class-rgx*  
  
# Naming style matching correct constant names.  
#const-naming-style=UPPER_CASE  
  
# Regular expression matching correct constant names. Overrides const-naming-  
# style. If left empty, constant names will be checked with the set naming  
# style.  
#const-rgx*  
  
# Minimum line length for functions/classes that require docstrings, shorter  
# ones are exempt.  
# MODIFIED CODE
```

The terminal window at the bottom shows the command `pylint enterprise_manager.py` being run, resulting in the error `C0103: Class name "EnterpriseManager" doesn't conform to UPPER_CASE naming style (invalid-name)`. The status bar indicates Python 3.9 (G87.2026.T15.GE1) and 168.28 LF UTF-8 4 spaces Python 3.9 (G87.2026.T15.GE1) 775 of 2100M.

## Rule 13:

- Rule name: const-naming-style
- Description: Controls the naming convention for variables treated as constants.
- Original value: UPPER\_CASE
- New value: PascalCase
- Reasoning: Wanted to differentiate the naming for constants and regular variables so they stand out.
- Positive Example:

```
enterprise_manager.py  pylint
  enterprise_manager model contains the logic for CIF validation and code processing.

  2   import json
  3   from EnterpriseManagementException import EnterpriseManagementException
  4   from EnterpriseRequest import EnterpriseRequest
  5
  6   PRE = {"J":0, "A":1, "B":2, "C":3, "D":4, "E":5, "F":6, "G":7, "H":8, "I":9}
  7
  8   hello_world="HelloWorld"
  9   print(hello_world)

10
11 class ENTERPRISE_MANAGER:
12     """EnterpriseManager class contains the logic for CIF validation."""
13     def __init__(self):
14         pass
15
16     @staticmethod
17     def validate_cif(cif):
18         # MODIFIED CODE:
19         import inspect
20         print("validate_cif lines:", len(inspect.getsource(ENTERPRISE_MANAGER.validate_cif).splitlines()))
21
22 # PLEASE TRAILING HERE THE CODE END WAITING THE CHAT

Your code has been rated at 9.82/10 (previous run: 9.65/10, +0.15)

charlessimons@Charless-MacBook-Pro-2 G87.2026.T15.GE1 %
```

- Negative Example:

```
enterprise_manager.py  pylint
  enterprise_manager model contains the logic for CIF validation and code processing.

  2   import json
  3   from EnterpriseManagementException import EnterpriseManagementException
  4   from EnterpriseRequest import EnterpriseRequest
  5
  6   PRE = {"J":0, "A":1, "B":2, "C":3, "D":4, "E":5, "F":6, "G":7, "H":8, "I":9}
  7
  8   hello_World="HelloWorld"
  9   print(hello_World)

10
11 class ENTERPRISE_MANAGER:
12     """EnterpriseManager class contains the logic for CIF validation."""
13     def __init__(self):
14         pass
15
16     @staticmethod
17     def validate_cif(cif):
18         # MODIFIED CODE:
19         import inspect
20         print("validate_cif lines:", len(inspect.getsource(ENTERPRISE_MANAGER.validate_cif).splitlines()))
21
22 # PLEASE TRAILING HERE THE CODE END WAITING THE CHAT

pylint: E0015: Unrecognized option found: check-fixme-in-dostring (unrecognized-option)
pylint: E0015: Unrecognized option found: check-fixme-in-dostring (unrecognized-option)
pylint: E0015: Unrecognized option found: check-fixme-in-dostring (unrecognized-option)
enterprise_manager.py:9:0: C0302: Too many lines in module (92/80) (too-many-lines)
enterprise_manager.py:9:0: C0103: Constant name "hello_World" doesn't conform to PascalCase naming style (invalid-name)

Your code has been rated at 9.62/10 (previous run: 9.41/10, +0.20)

charlessimons@Charless-MacBook-Pro-2 G87.2026.T15.GE1 %
```

## Rule 14:

- Rule name: function-naming-style
  - Description: Controls the naming convention for functions.
  - Original value: snake\_case
  - New value: UPPER\_CASE
  - Reasoning: Upper case was much clearer to read for our specific function names.
  - Positive Example:

The screenshot shows the PyCharm IDE interface with the following details:

- Project:** G87.2026.T15.GE1 [G87.2026.T69.GE1]
- File:** enterprise\_manager.py
- Tool:** pylint
- Code Snippet (enterprise\_manager.py):**

```
from enterprise_manager import EnterpriseManagementException
from EnterpriseRequest import EnterpriseRequest

PRE = ["J":0, "A":1, "B":2, "C":3, "D":4, "E":5, "F":6, "G":7, "H":8, "I":9}

def FUNC_PRINT(text): 1 usage new +
    print(text)
    return text

FUNC_PRINT("Hello World!")

class ENTERPRISE_MANAGER: 2 usages & Linus Munn +2
    """EnterpriseManager class contains the logic for CIF validation."""
    def __init__(self): & Charles287Mad
        pass

    @staticmethod 3 usages & Linus Munn +2 *
    def validate_cif(cif):
        # MODIFIED CODE:
        import inspect
        print("validate_cif lines:", len(inspect.getsource(ENTERPRISE_MANAGER.validate_cif).splitlines()))
```

- Terminal Output:**

```
charlessimons@charless-MacBook-Pro-2: G87.2026.T15.GE1 % pylint enterprise_manager.py
*****
Module /Users/charlessimons/PycharmProjects/G87.2026.T15.GE1/pylintrc
pylintrc:1:D: E0015: Unrecognized option found: check-fixture-in-docstring (unrecognized-option)
*****
Module G87.2026.T15.GE1.enterprise_manager
enterprise_manager.py:1:0: C0302: Too many lines in module (85/80) (too-many-lines)
```

- Bottom Status Bar:** Your code has been rated at 9.81/10 (previous run: 9.63/10, +0.19)

- #### - Negative Example:

The screenshot shows the PyCharm IDE interface. The left sidebar displays a project structure for 'G87.2026.T15.GE1' containing files like 'enterprise\_manager.py', 'EnterpriseManagementException.py', and 'EnterpriseRequest.py'. The right pane shows the code editor for 'enterprise\_manager.py' with syntax highlighting and code completion. The bottom terminal window displays the output of running pylint on the code, showing various warnings and errors. The status bar at the bottom indicates the current file is 'main.py'.

```
Project: G87.2026.T15.GE1 [G87.2026.T15.GE1] - /PycharmProjects/G87.2026.T15.GE1
File: enterprise_manager.py
  1 from EnterpriseManagementException import EnterpriseManagementException
  2 from EnterpriseRequest import EnterpriseRequest
  3
  4 PRE = [{"J":0, "A":1, "B":2, "U":3, "O":4, "E":5, "F":6, "G":7, "H":8, "I":9}
  5
  6     def func_print(text): 1 usage new *
  7         print(text)
  8
  9     return text
 10
 11
 12 func_print("Hello World!")
 13
 14 class ENTERPRISE_MANAGER: 2 usages & Linus Munn +2*
 15     """EnterpriseManager class contains the logic for CIF validation."""
 16     def __init__(self): & Charles2827Mad
 17         pass
 18
 19
 20     @staticmethod 3 usages & Linus Munn +2*
 21     def validate_cif(cif):
 22         # MODIFIED CODE:
 23         import inspect
 24         print("validate_cif lines:", len(inspect.getsource(ENTERPRISE_MANAGER.validate_cif).splitlines()))
 25
```

```
charlessimons@Charles-MacBook-Pro-2: G87.2026.T15.GE1 % pylint enterprise_manager.py
*****
Module /Users/charlessimons/PycharmProjects/G87.2026.T15.GE1/pylintc
pylintc:1: E0015: Unrecognized option found: check-fixture-in-docstring (unrecognized-option)
*****
Module G87.2026.T15.GE1.enterprise_manager
enterprise_manager.py:8: C0302: Too many lines in module (85/88) (too-many-lines)
enterprise_manager.py:9: C0103: Function name 'func_print' doesn't conform to UPPER_CASE naming style (invalid-name)
```

Your code has been rated at 9.43/10 (previous run: 9.81/10, -0.19)

## Rule 15:

- Rule name: module-naming-style
- Description: Naming convention for modules
- Original value: snake\_case
- New value: PascalCase
- Reasoning: This was the original style provided in the source code, so it seemed logical to change the rules to accept it.
- Positive Example:

```
from EnterpriseManagementException import EnterpriseManagementException
from EnterpriseRequest import EnterpriseRequest
PRE = {"J":0, "A":1, "B":2, "C":3, "D":4, "E":5, "F":6, "G":7, "H":8, "I":9}

def FUNC_PRINT(text): 1usage new *
    print(text)
    return text

FUNC_PRINT("Hello World!")

class ENTERPRISE_MANAGER: 2 usages & Linus Munn +2*
    """EnterpriseManager class contains the logic for CIF validation."""
    def __init__(self): & Charles2827Mad
        pass

    @staticmethod 3 usages & Linus Munn +2*
    def validate_cif(cif):
        # MODIFIED CODE:
        import inspect
        print("\n".join(inspect.getsource(ENTERPRISE_MANAGER.validate_cif).splitlines()))

```

charlessimons@charless-MacBook-Pro-2: G87.2026.T15.GE1 % pylint
\*\*\*\*\*
Module /Users/charlessimons/PycharmProjects/G87.2026.T15.GE1/pylintc
pylintc:10: E0015: Unrecognized option found: check-fixme-in-docstring (unrecognized-option)
\*\*\*\*\*
Module G87.2026.T15.GE1.EnterpriseManager
EnterpriseManager.py:10: C0302: Too many lines in module (85/80) (too-many-lines)

Your code has been rated at 9.81/10

charlessimons@charless-MacBook-Pro-2: G87.2026.T15.GE1 %

- Negative Example:

```
module-naming-style=snake_case
# Regular expression matching correct method names. Overrides method-naming-
# style. If left empty, method names will be checked with the set naming style.
#method-rgx

# Naming style matching correct module names.
# MODIFIED CODE
# module-naming-style=snake_case
module-naming-style= PascalCase

# Regular expression matching correct module names. Overrides module-naming-
# style. If left empty, module names will be checked with the set naming style.
#module-rgx

# Colon-delimited sets of names that determine each other's naming style when
# the name regexes allow several styles.
name-groups

# Regular expression which should only match function or class names that do
```

\*\*\*\*\*
Module /Users/charlessimons/PycharmProjects/G87.2026.T15.GE1/pylintc
pylintc:10: E0015: Unrecognized option found: check-fixme-in-docstring (unrecognized-option)
\*\*\*\*\*
Module G87.2026.T15.GE1.enterprise\_manager
enterprise\_manager.py:10: C0302: Too many lines in module (85/80) (too-many-lines)
enterprise\_manager.py:10: C0103: Module name "enterprise\_manager" doesn't conform to PascalCase naming style (invalid-name)

Your code has been rated at 9.65/10 (previous run: 9.81/10, -0.19)

charlessimons@charless-MacBook-Pro-2: G87.2026.T15.GE1 %

## After Changes: (Screenshot of passing linting with score 10/10)

The screenshot shows the PyCharm IDE interface. The code editor displays `EnterpriseManager.py` with the following content:

```
from EnterpriseManagementException import EnterpriseManagementException
from EnterpriseRequest import EnterpriseRequest

PRE = {"J":0, "A":1, "B":2, "C":3, "D":4, "E":5, "F":6, "G":7, "H":8, "I":9}

class ENTERPRISE_MANAGER:
    """EnterpriseManager class contains the logic for CIF validation."""
    def __init__(self):
        pass

    @staticmethod
    def validate_cif(cif):
        # MODIFIED CODE:
        import inspect
        print("validate_cif lines:", len(inspect.getsource(ENTERPRISE_MANAGER.validate_cif).splitlines()))

        # PLEASE INCLUDE HERE THE CODE FOR VALIDATING THE GUID
        # RETURN TRUE IF THE GUID IS RIGHT, OR FALSE IN OTHER CASE

        if len(cif) != 9 or cif[0] not in PRE or not cif[1:].isdigit():
            return False
        return True
```

The terminal window shows the command `pylint EnterpriseManager.py` was run, resulting in a score of 10.00/10.

```
Your code has been rated at 9.81/10
=====
charlessimons@Charless-MacBook-Pro-2:~/PycharmProjects/G87.2026.T15.GE1% pylint EnterpriseManager.py
***** Module /Users/charlessimons/PycharmProjects/G87.2026.T15.GE1/pylintrc
pylintrc:1:0: E0015: Unrecognized option found: check-fixme-in-docstring (unrecognized-option)
-----
Your code has been rated at 10.00/10 (previous run: 9.81/10, +0.19)
```

The status bar at the bottom indicates Python 3.9 (G87.2026.T15.GE1), 4 spaces, and 938 of 2300M.