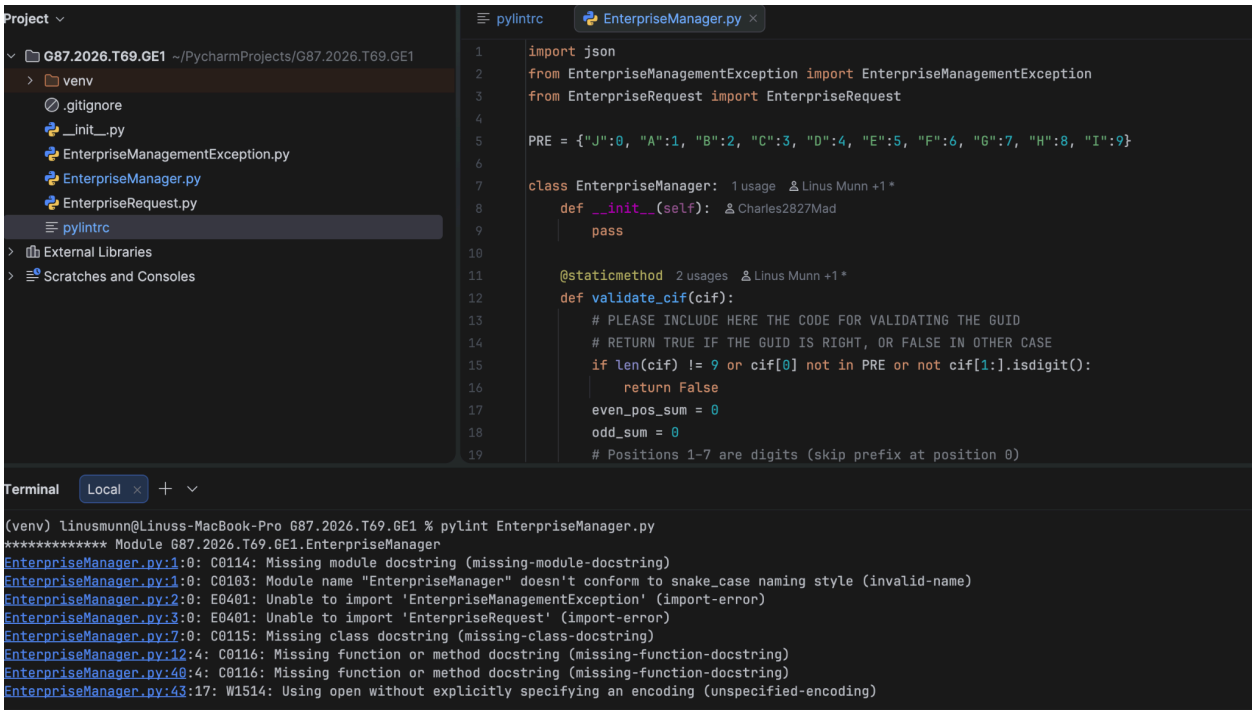
Code Regulations

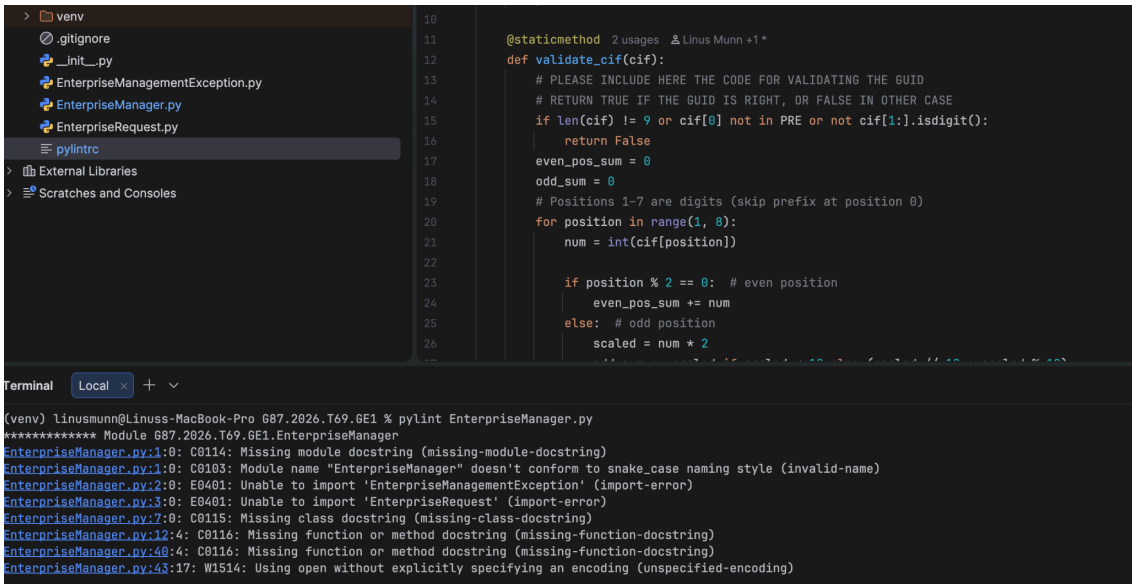
Without Changes:



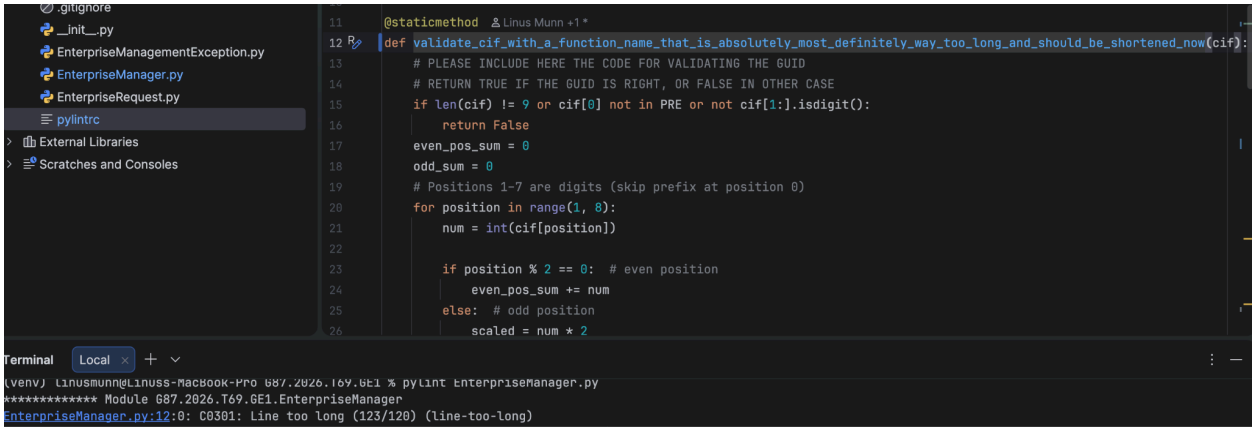
Rule 1: Max Line Length

- 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:



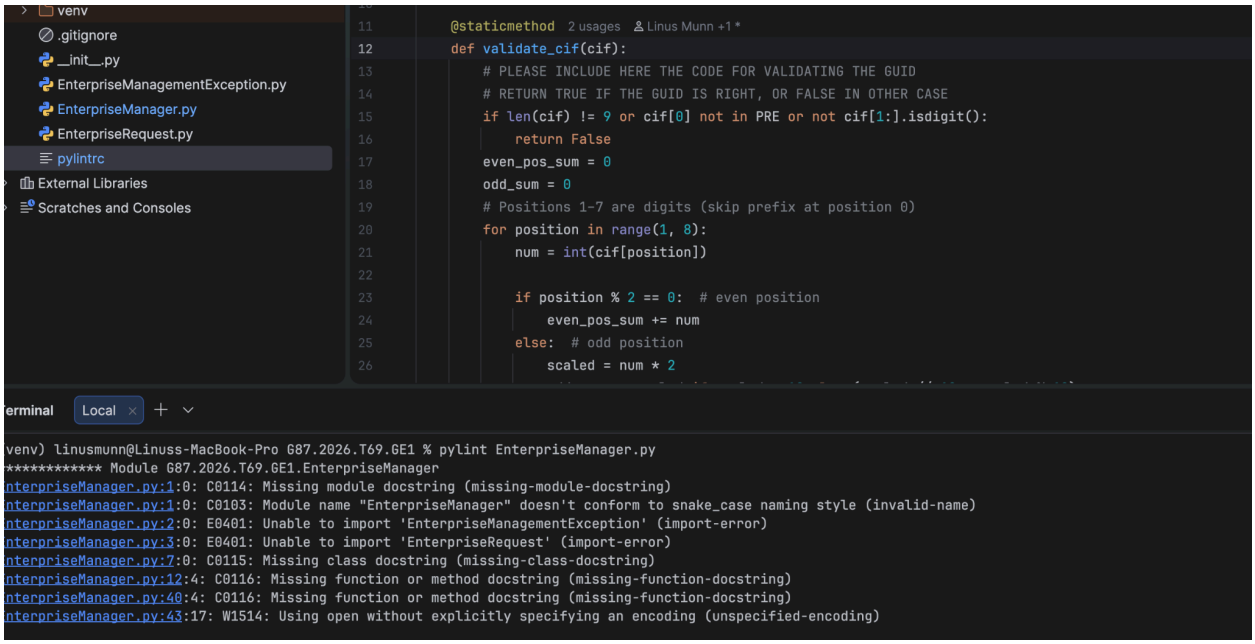
- Negative Example:



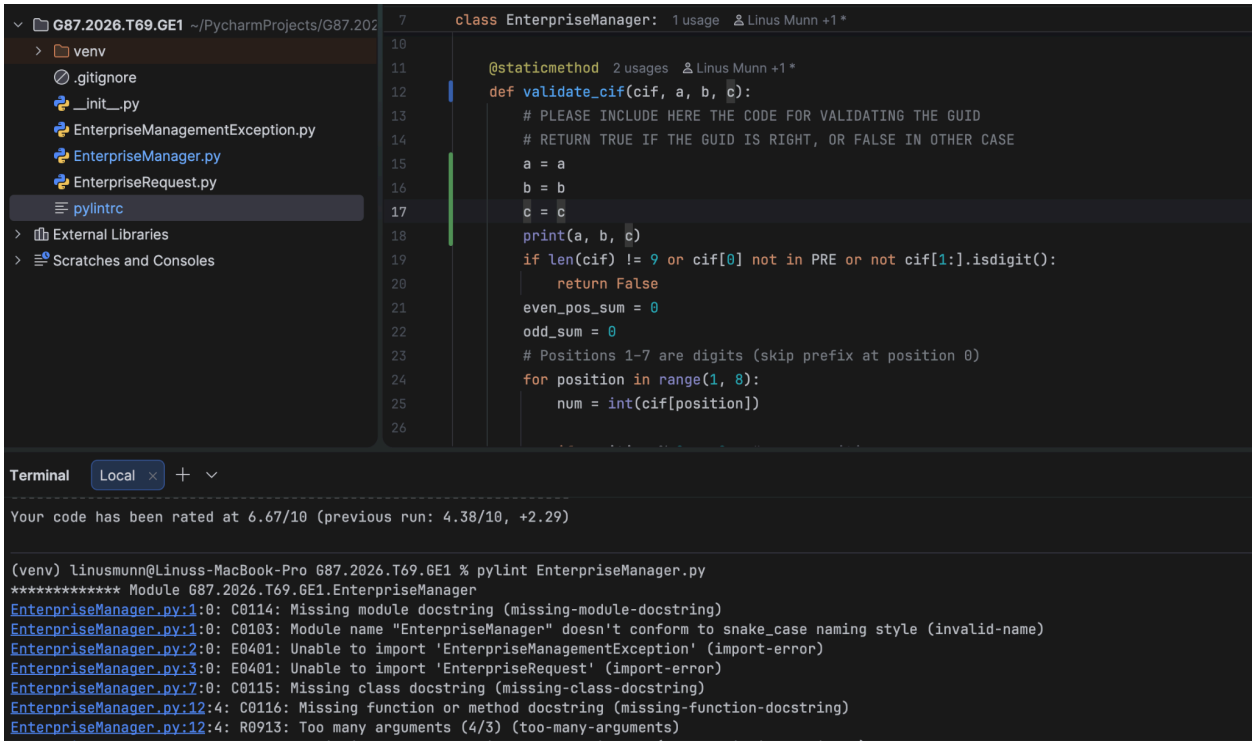
Rule 2: Max Args per function

- 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



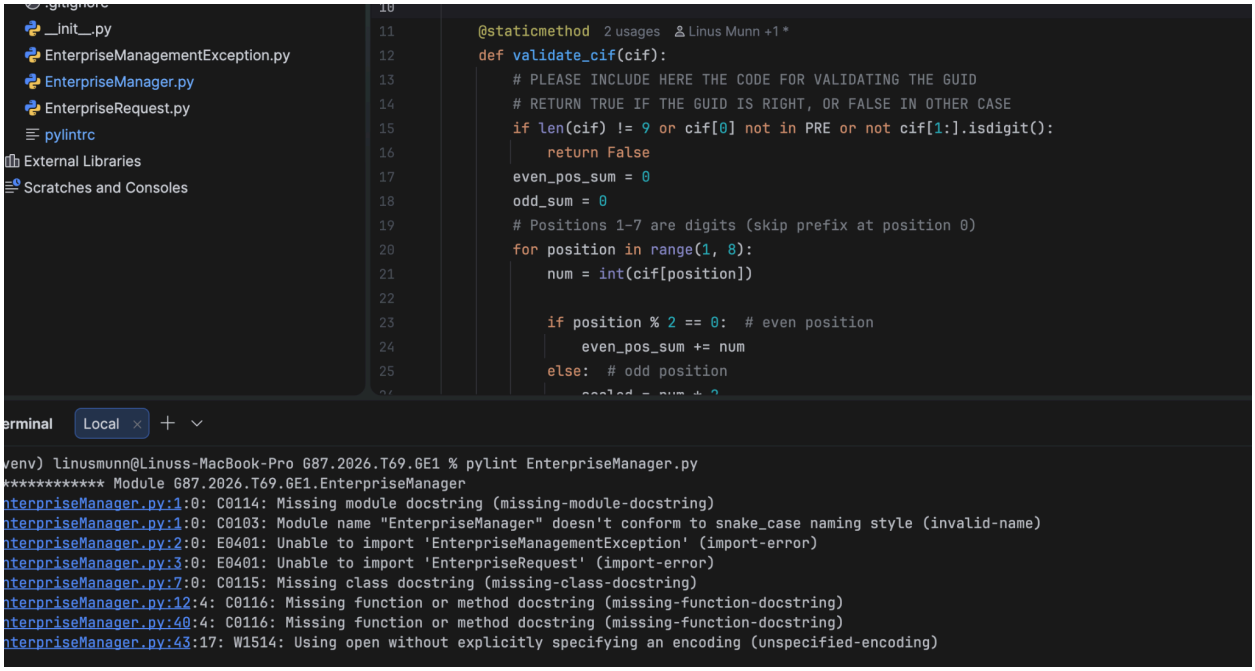
- Negative Example:



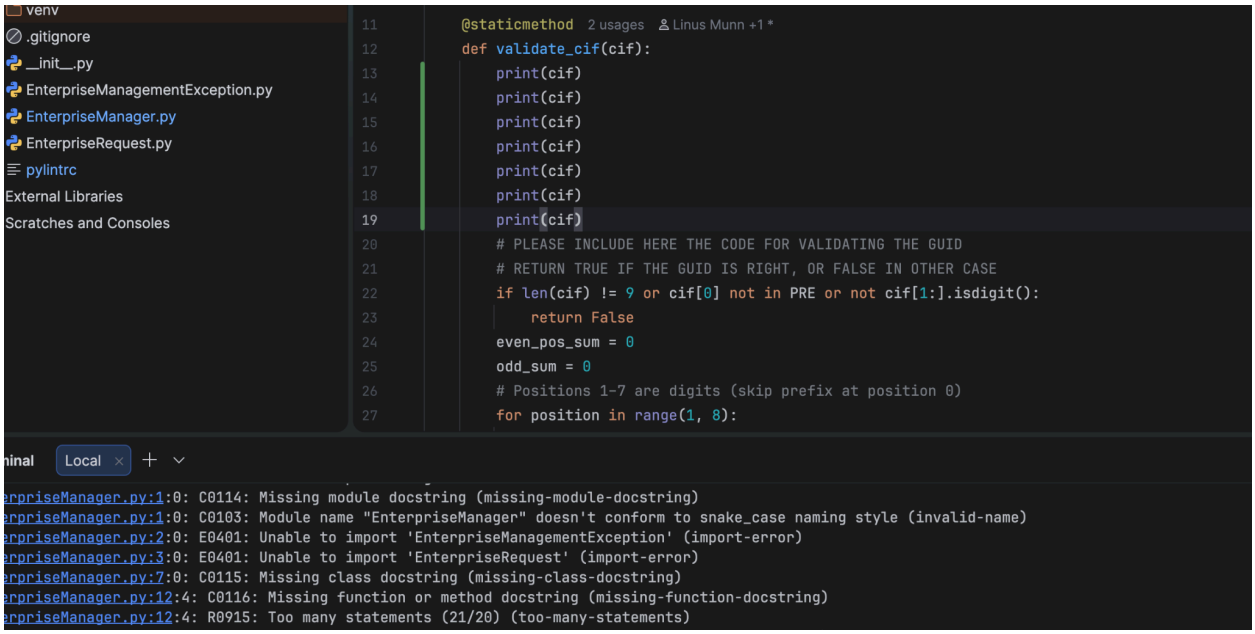
Rule 3: Max number of Statements

- 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:

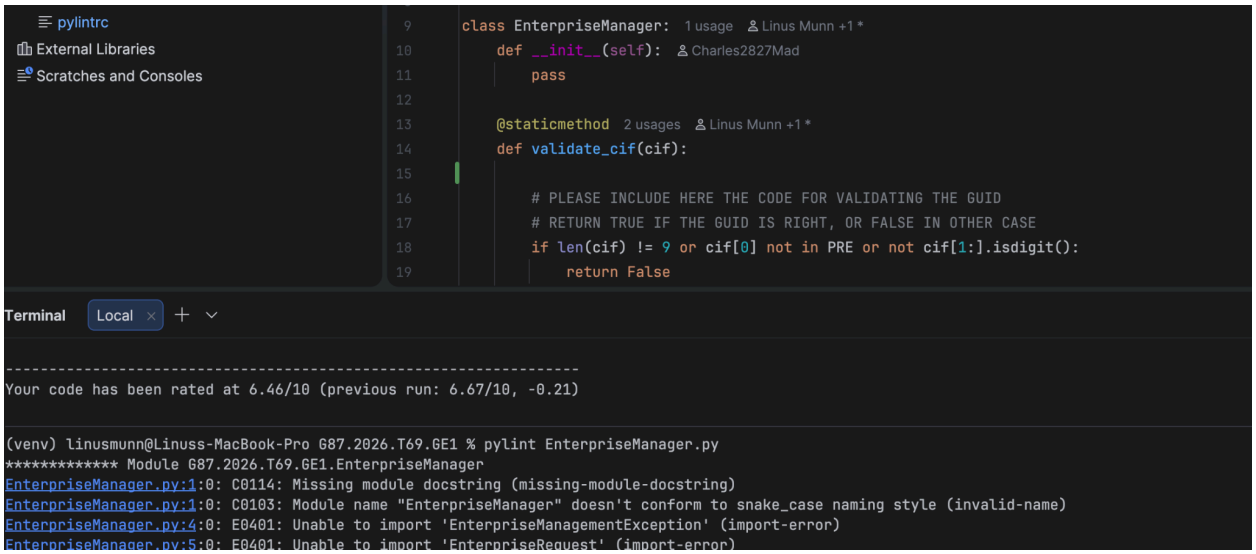


- Negative Example:

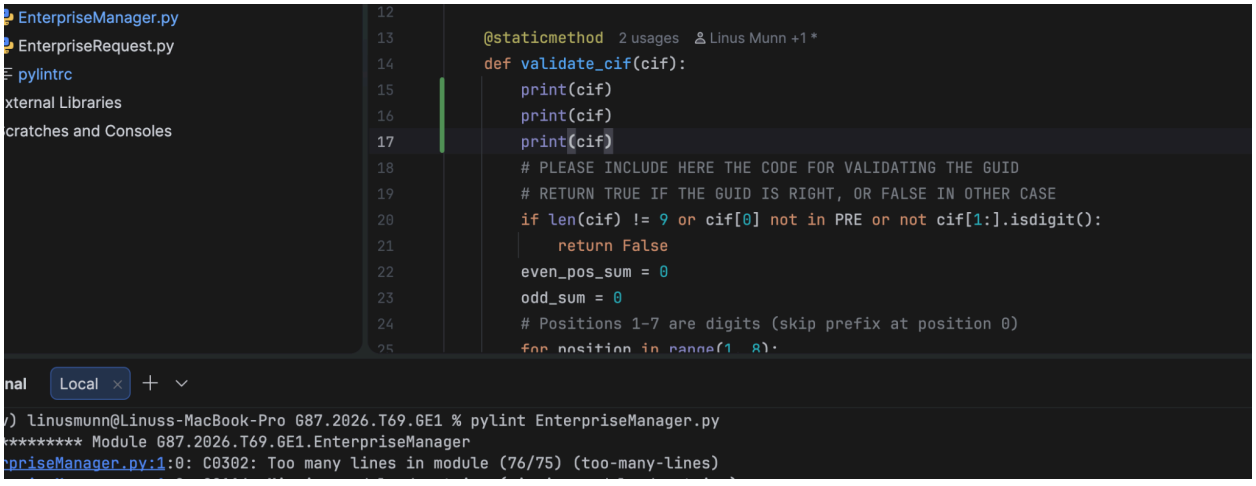


Rule 4: Max module lines

- 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:

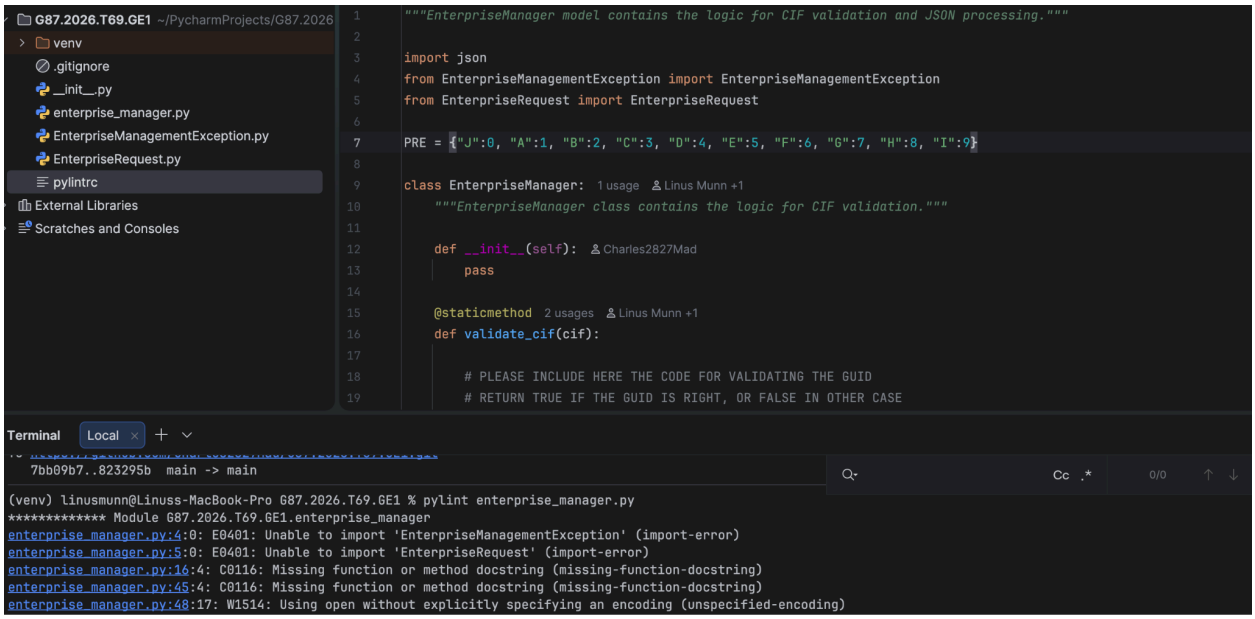


- Negative Example:

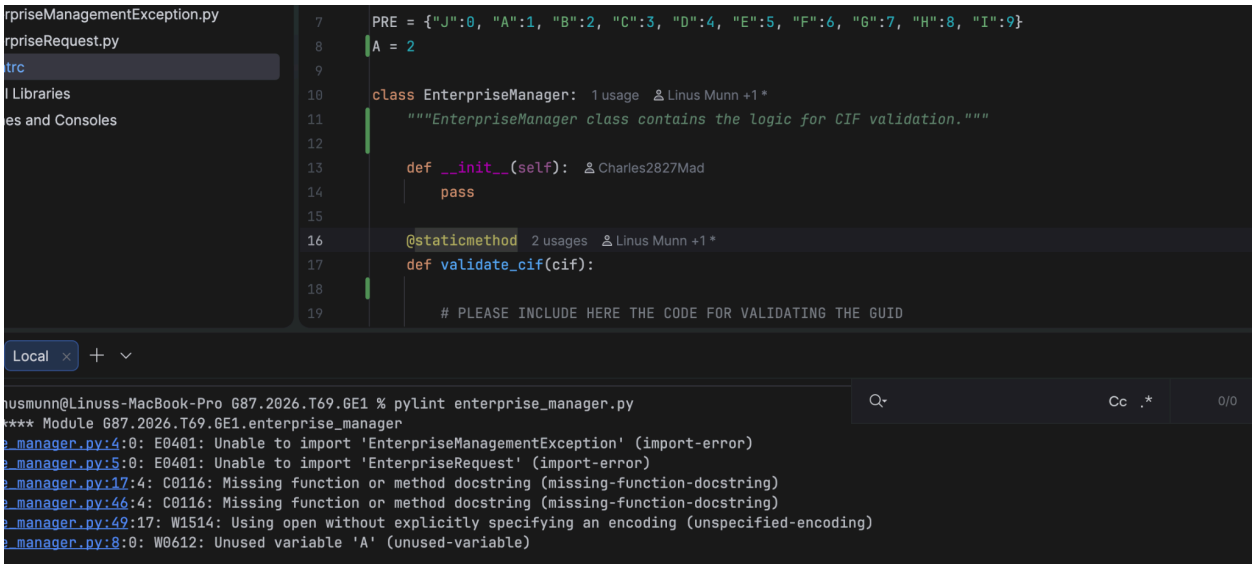


Rule 5: Allow unused global variable

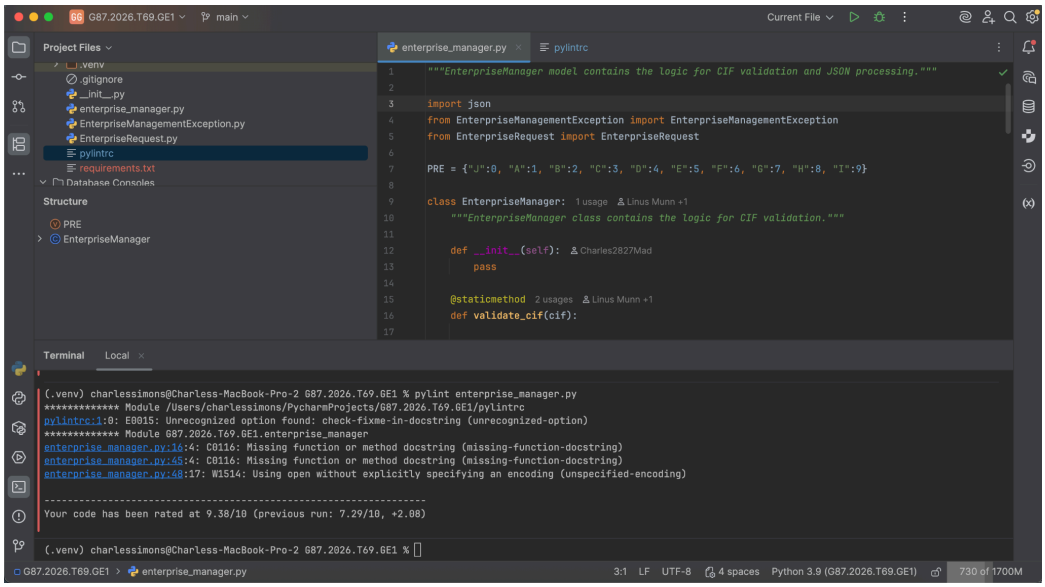
- Rule name: allow-global-unused-variables
- Description: Allow global variables to go unused.
- Original value: yes
- New value: no
- Reasoning: Prevent wasted or useless lines from clogging a file.
- Positive Example:

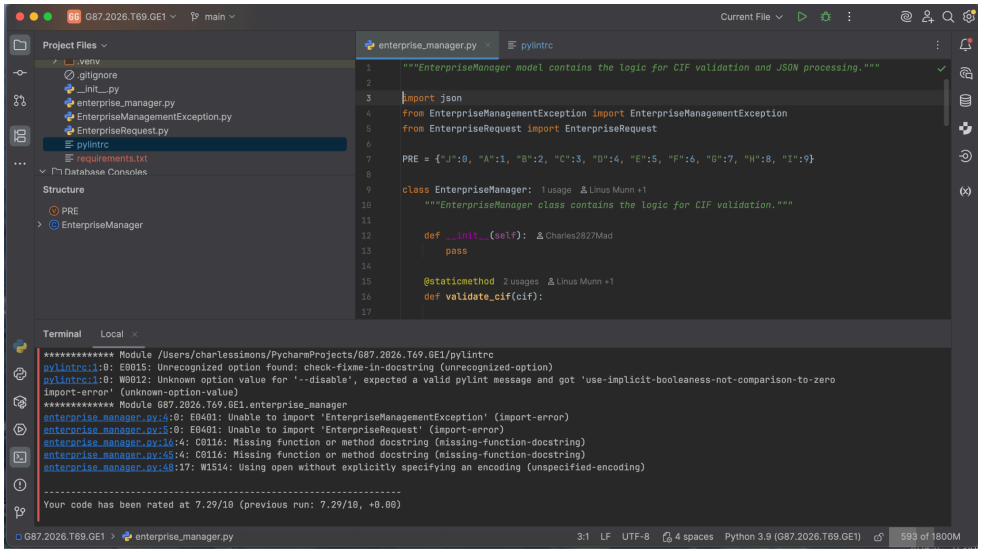


- Negative Example:



Rule 6:

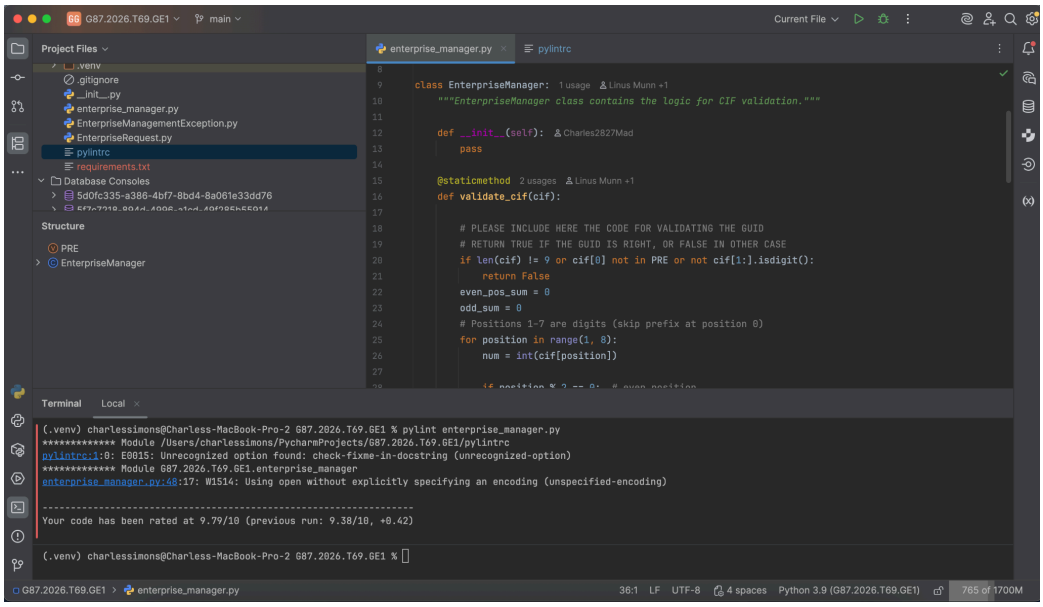
- Rule name: disable
- Description: Disables the import-error message
- Original value: disable = … null
- New value: disable = … import-error
- Reasoning: Brings pylint up positively by 2.08 points
- Positive Example:
- 

- Negative Example:
- 

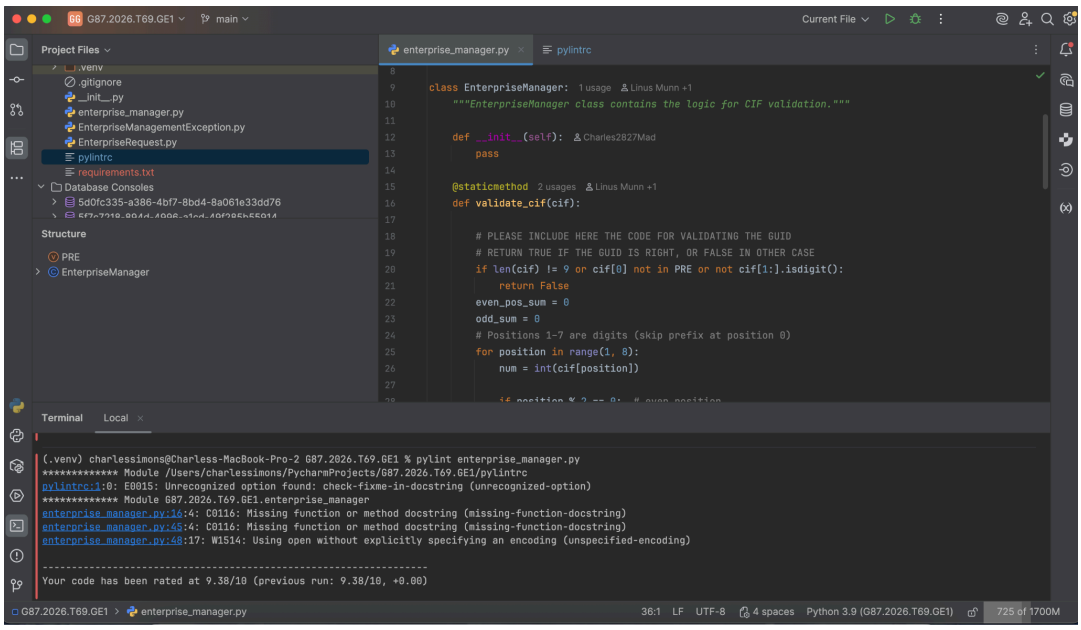
-

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:
-

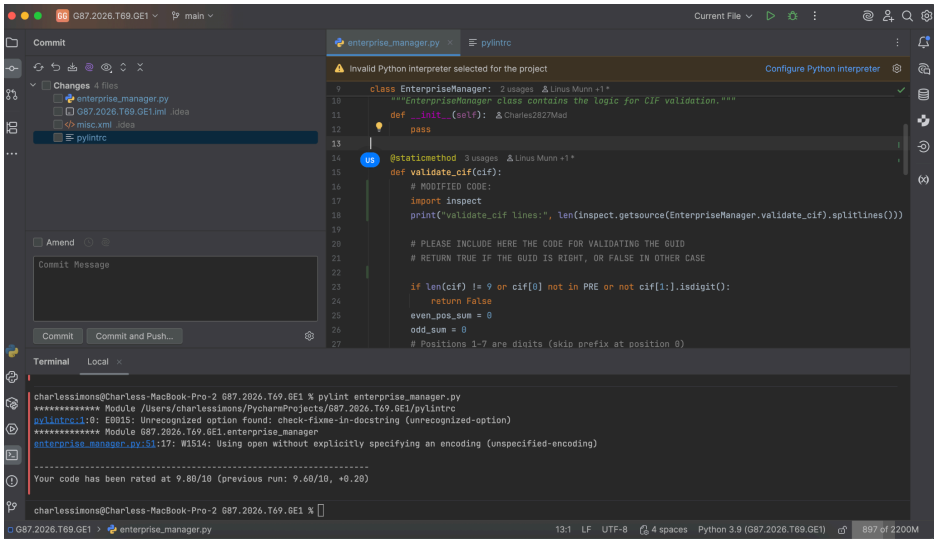


- Negative Example:

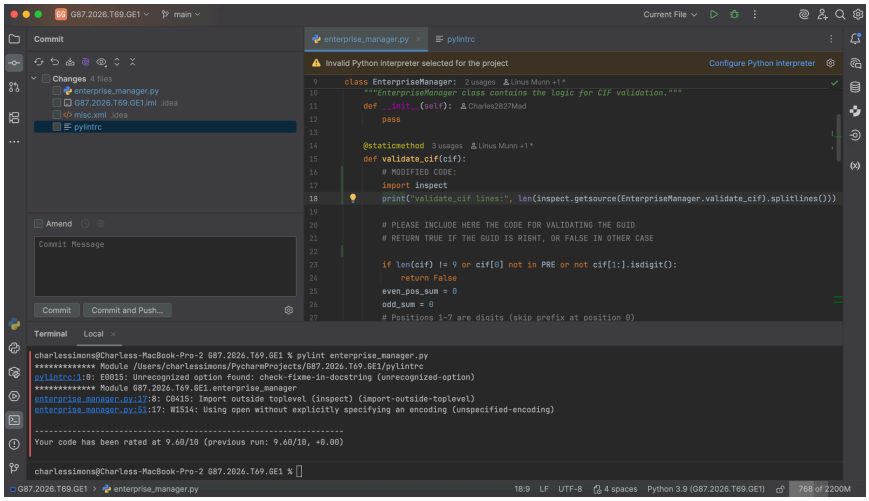


Rule 8:

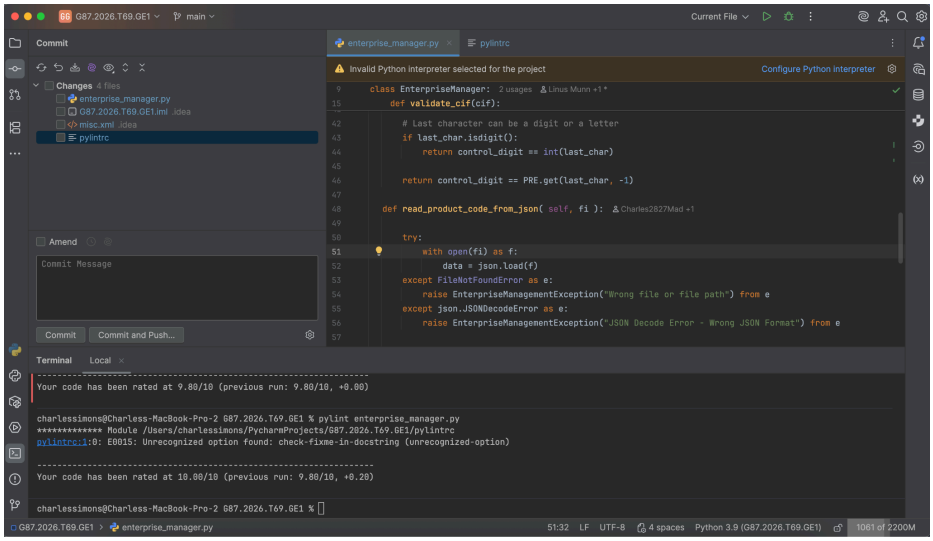
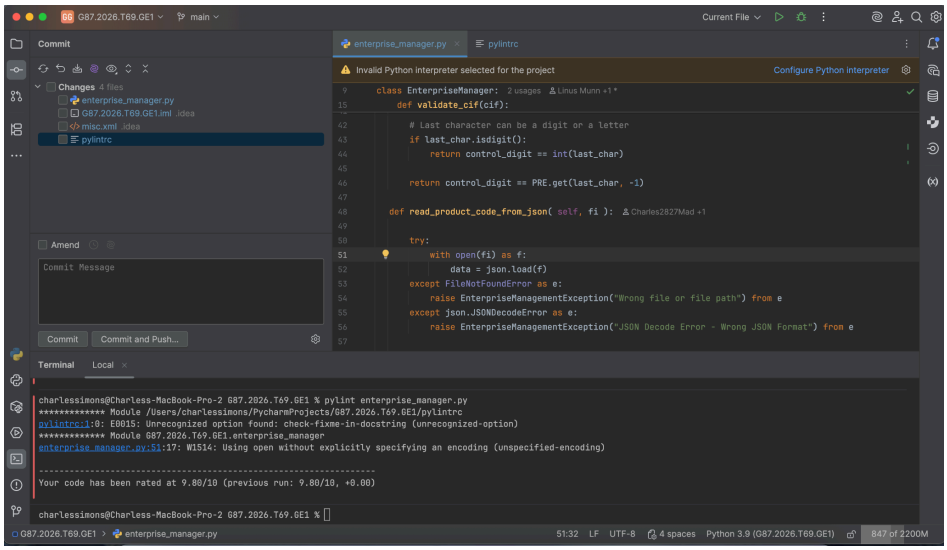
- Rule name: disable
- Description: Throws an error if imports are not in the top level
- Original value: disable = … null
- New value: disable = … import-outside-toplevel
- Reasoning: An import was made in the validate_cif() function to use .inspect to determine function length to correctly write docstring-min-length.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



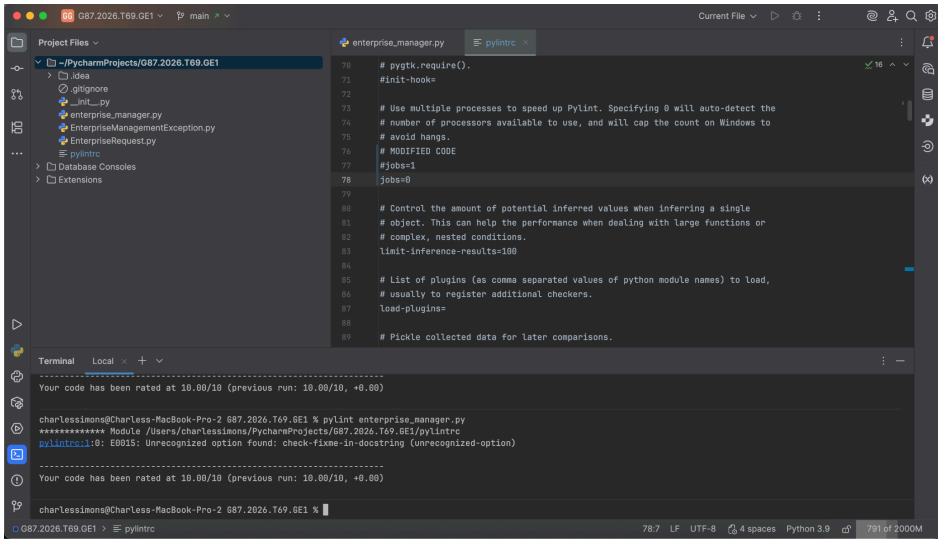
- Negative Example:
-

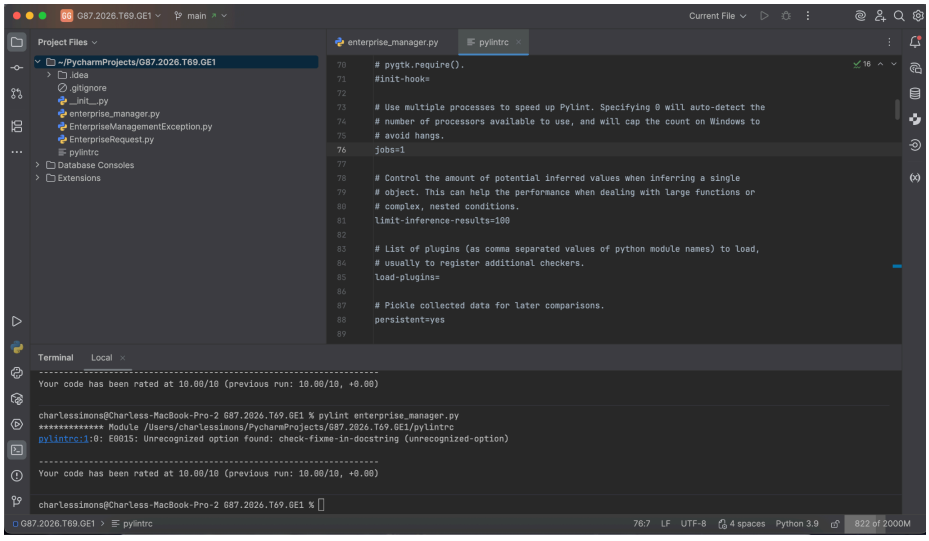


Rule 9:

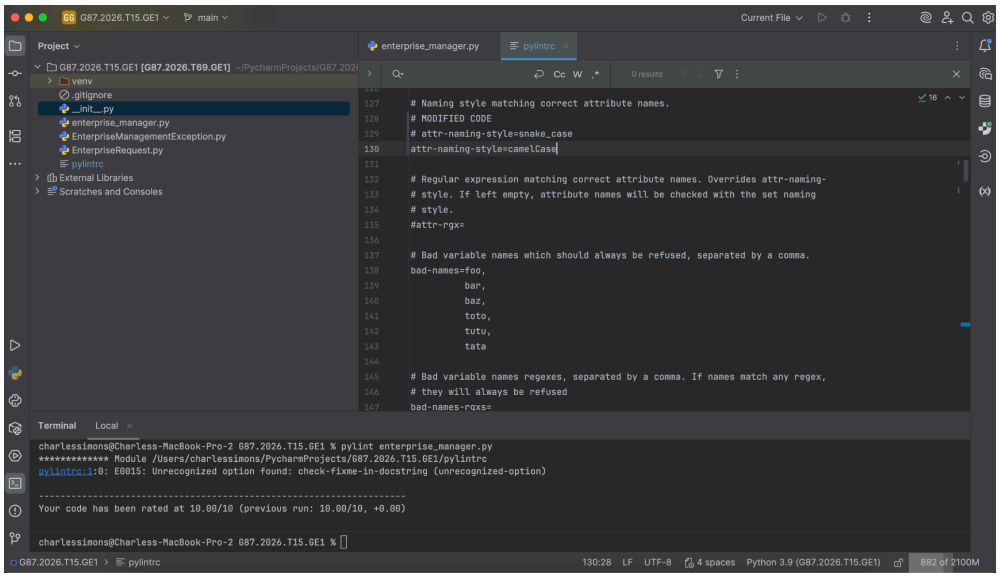
- Rule name: disable
- Description: Our code uses “open” without specified encoding
- Original value: disable = … null
- New value: disable = … unspecified-encoding
- Reasoning: Disabling this in pylint brings our score up by .2
- Positive Example:
  - 
- Negative Example:
  - 

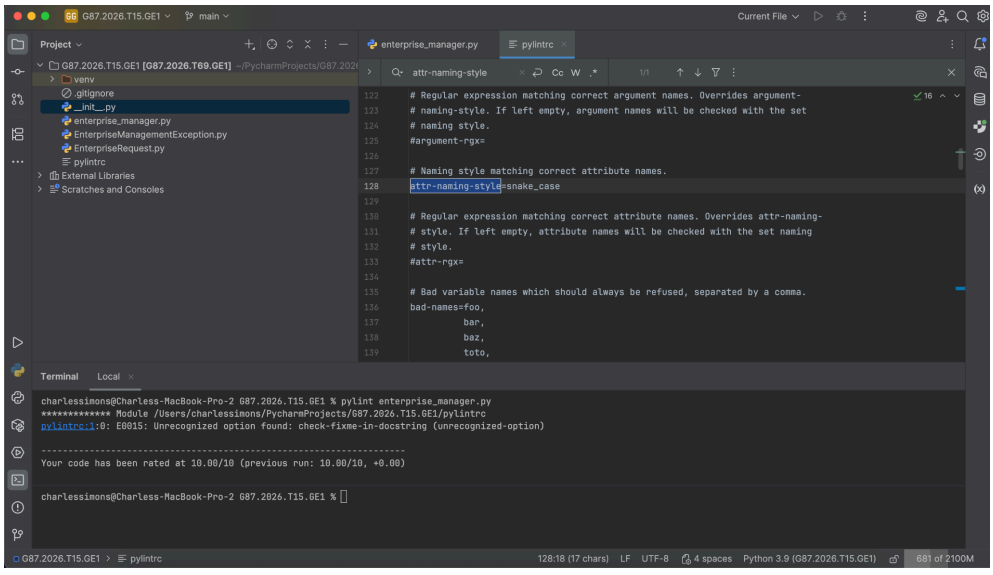
Rule 10:

- 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:
- 

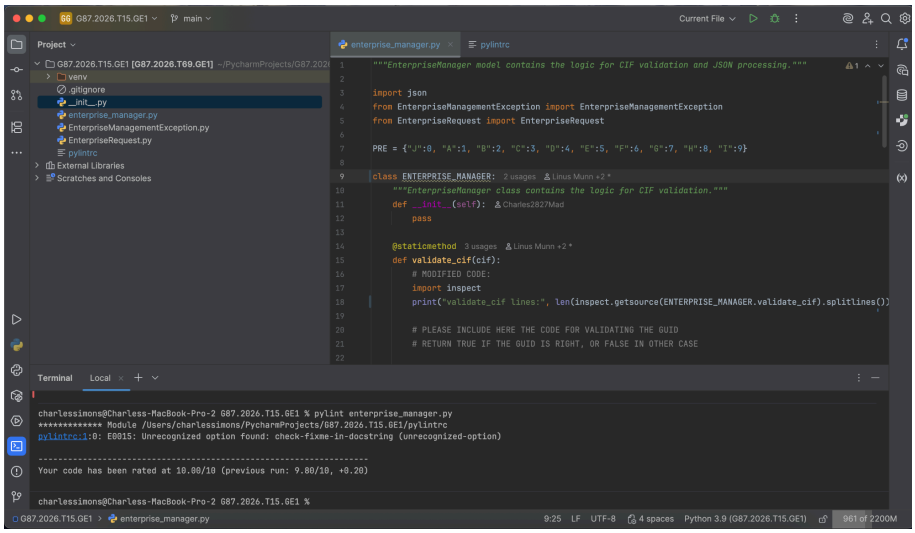
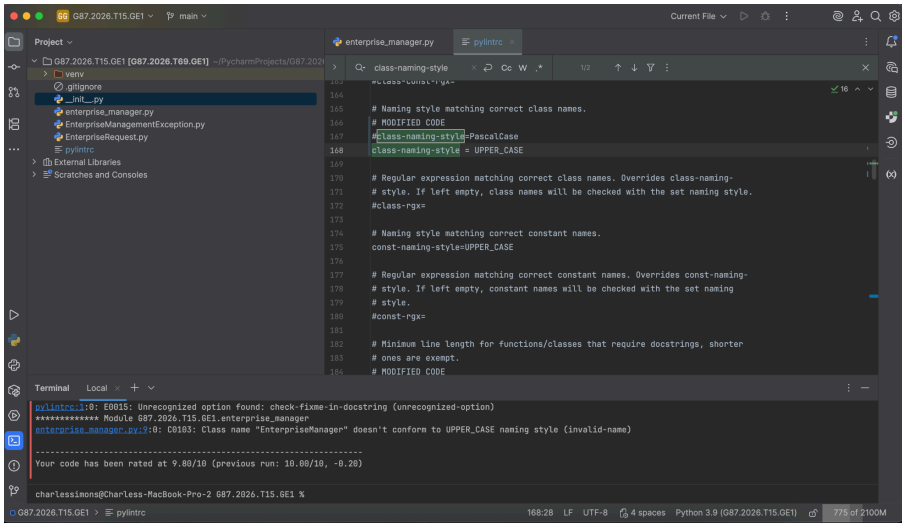
- Negative Example:
- 

Rule 11:

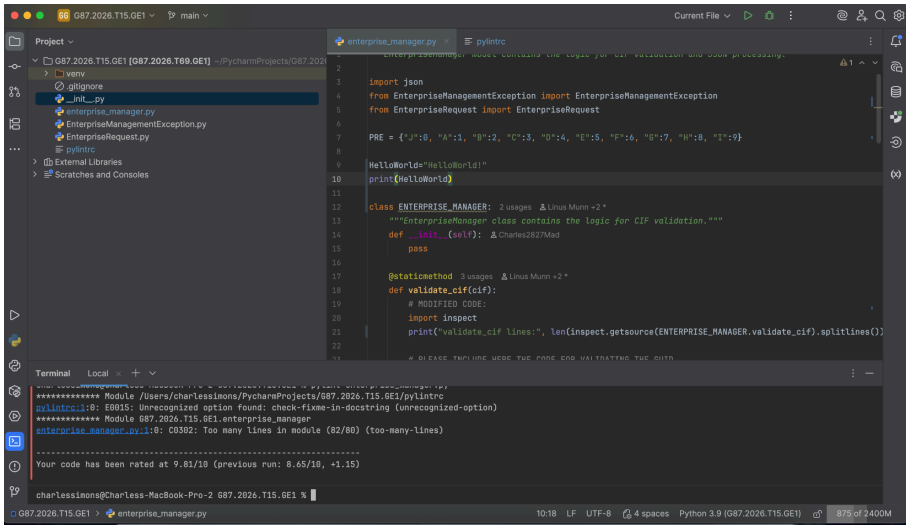
- Rule name: attr-naming-style
- Description: Sets the naming style for attributes
- Original value: attr-naming-style = snake_case
- New value:attr-naming-style = camelCase
- Reasoning: Easy rule change, camelCase is preffered
- Positive Example:
- 

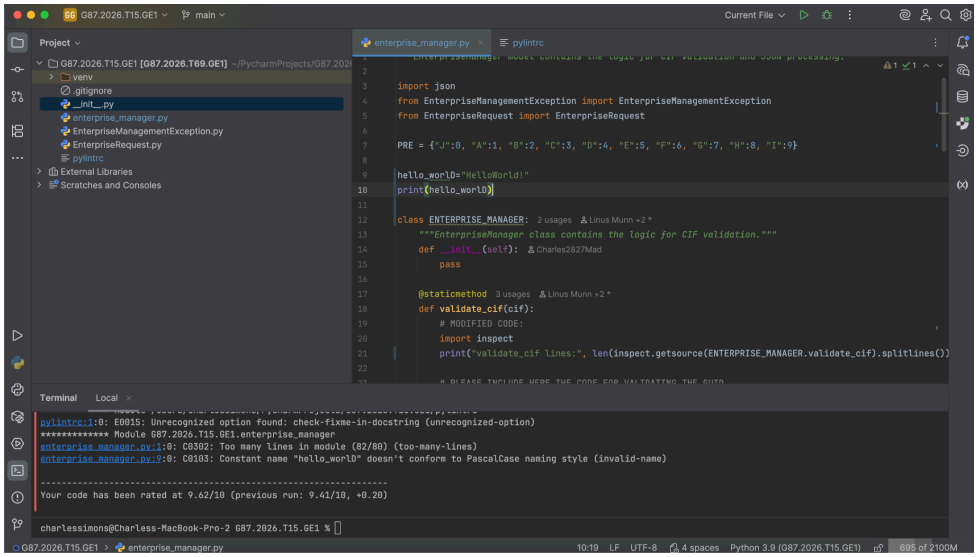
- Negative Example:
- 

Rule 12:

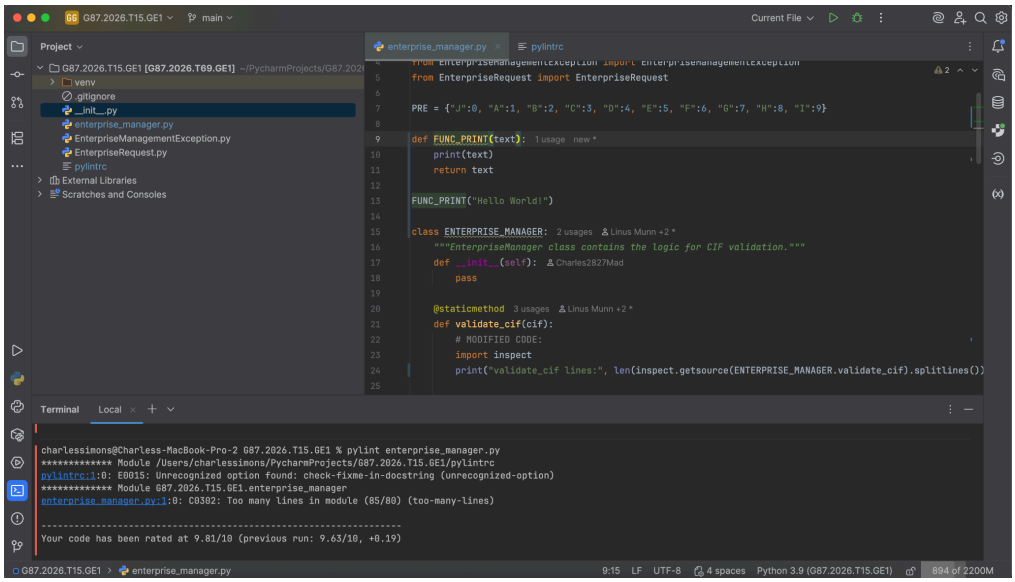
- Rule name: class-naming-style
- Description: Controls the naming convention for classes.
- Original value: PascalCase
- New value: UPPER_CASE
- Reasoning: We decided it was easier to read the class name using upper case.
- Positive Example:
  - 
- Negative Example:
  - 

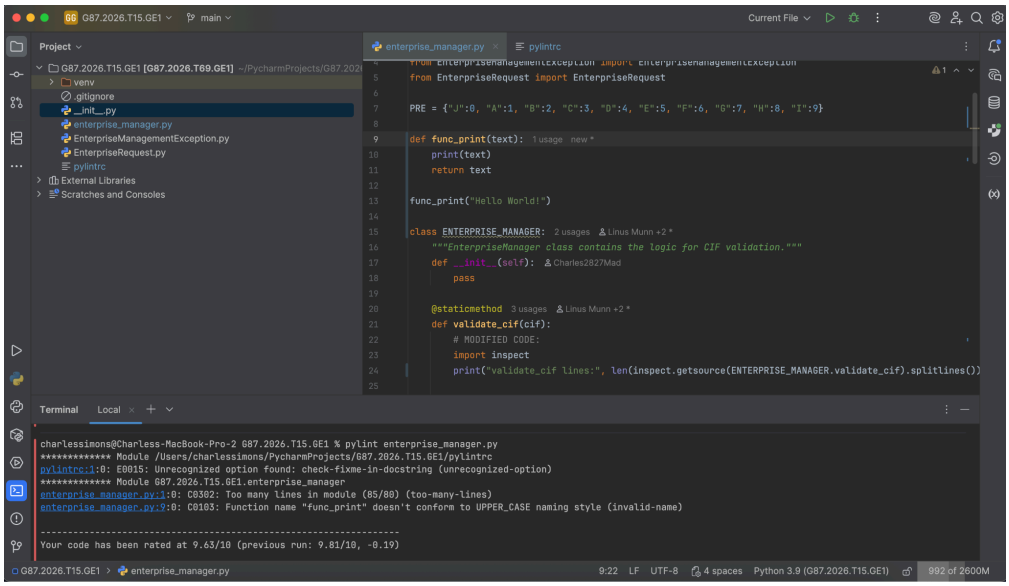
Rule 13:

- 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:
- 

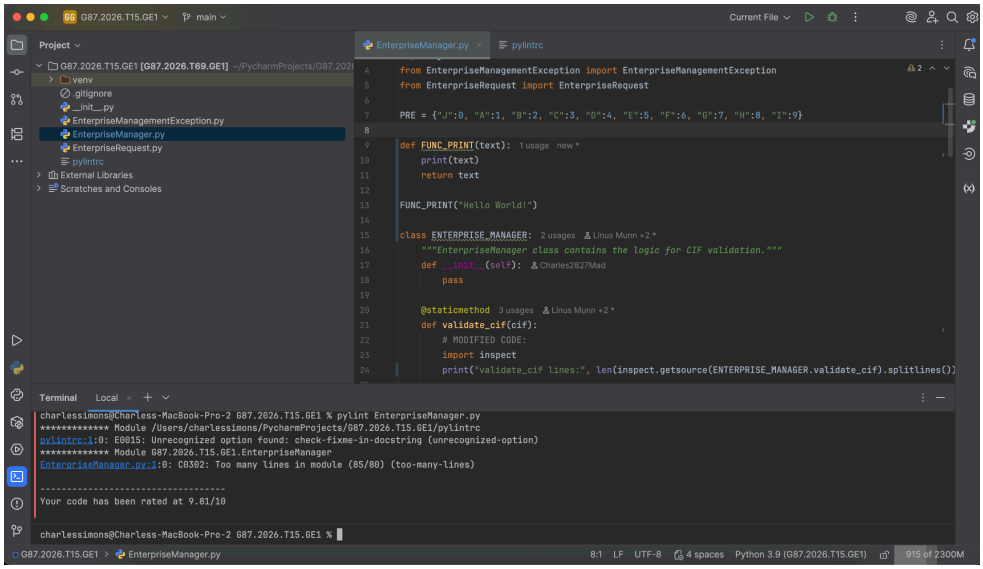
- Negative Example:
- 

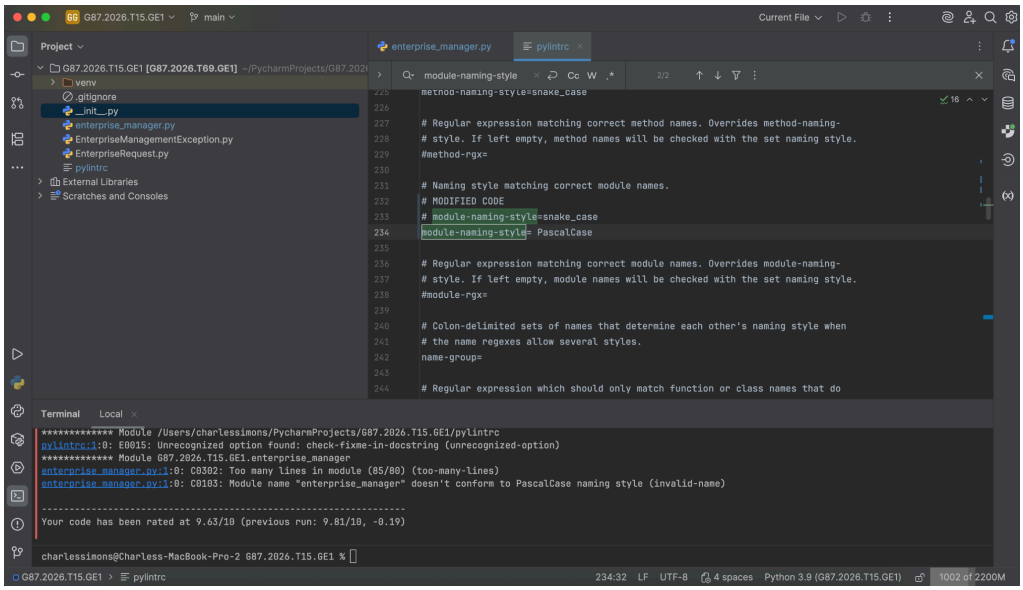
Rule 14:

- 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:
- 

- Negative Example:
- 

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:
- 

- Negative Example:
- 

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