

Programación Avanzada

IIC2233 2023-2

Hernán Valdivieso - Daniela Concha - Francisca Ibarra - Joaquín Tagle - Francisca Cattán



Experiencia 5: DCCorrector



Cansado de tener que revisar tus tareas manualmente para ver si cumpliste las reglas de PEP8 y no caer en descuentos, decides hacer un programa que los revise por ti y te avise de posibles errores... un *DCCorrector Automático*

Cómo funcionará

/codigo




- Interfaz.py
- Networking.py
- ...



linter.py

- ReglaLargoArchivo
- ReglaClasesMayuscula
- ...



Carpeta donde irá el código a revisar

Linter revisa reglas seleccionadas sobre un archivo de la carpeta

Linter hace print de un resumen de fallas encontradas

Archivos entregados

Linter.py

- Clase linter()
 - Init para cargar las reglas disponibles
 - Menus varios
 - Cargar un archivo a analizar
 - Aplicar regla sobre archivo

parametros.py

- Configuraciones del Linter

main.py

- Ejecuta el Linter

reglas_linter/

regla_base.py

- Clase abstracta Regla()
 - Tiene la estructura que debería seguir una regla, además de métodos y atributos generales.
 - Debemos actualizar las funciones que imprimen el resumen de faltas encontradas.

regla_nombreregla.py

- Cada regla será una nueva clase que hereda de Regla()
- Para cada una, debemos definir el comportamiento personalizado que seguirá la regla para detectar faltas en el código.

No modificar

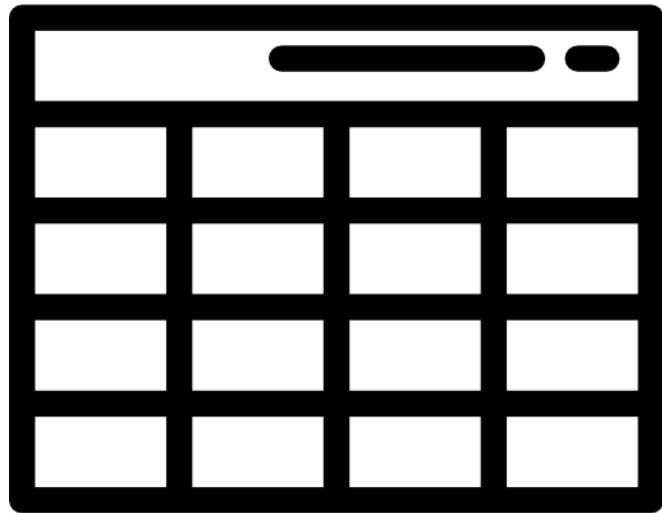
Esto modificaremos

¿Cómo procederemos?

1. El código ya viene con una regla de ejemplo implementada (ReglaLargoArchivo). Así que podemos revisar cómo se ven actualmente los *prints* de nuestro programa.
2. Modificaremos los *prints* de resumen de faltas, aprovechando *f-strings* para un mejor formateo.
3. Agregaremos reglas que solo necesitan saber de manejo de archivos y strings.
4. Usaremos RegEx para agregar algunas reglas más complejas

Mejoraremos nuestro *print* de detalles con *strings*

1. Vamos a `reglas_linter/regla_base.py`
2. Modifiquemos los métodos:
 - a. `imprimir_resumen`
 - b. `imprimit_resumen_nivel_archivo`
 - c. `imprimir_resumen_nivel_linea`
3. Revisemos como se ve nuestro print ahora al ejecutar la regla de máximo largo de archivo.



Añadir regla nueva

1. Vamos a `reglas_linter/regla_limite_caracteres_linea.py`
2. Sobreescribamos el método `revisar_regla()`
3. Ejecutemos nuestro linter para ver si programamos bien la regla.

Añadir regla nueva (usando Regex)

1. Vamos a `reglas_linter/regla_clases_mayuscula.py` y `reglas_linter/regla_espacio_coma.py`
2. Sobreescribamos el método `revisar_regla()`
 - Ojo, para estas necesitaremos usar Regex
3. Ejecutemos nuestro linter para ver si programamos bien las reglas.

BONUS: Regex más complejos

En la regla de espacio después de la coma, actualmente no estamos considerando el caso donde hay comas dentro de *strings*. Por ejemplo, la línea:

variable = “tengo una coma sin espacio,pero soy un string”

Sería detectada erroneamente como falta.

Tratemos de modificar la RegEx para ignorar estos casos.

Programación Avanzada

IIC2233 2023-2

Hernán Valdivieso - Daniela Concha - Francisca Ibarra - Joaquín Tagle - Francisca Cattán

