UNIVERSIDAD AUTÓNOMA DE YUCATÁN

FACULTAD DE MATEMATICAS

Semestre:

Agosto-diciembre

Materia:

Desarrollo Y Mantenimiento de Software

Proyecto Final:

Parte 1

Docente:

M. en C. Carlos Benito Mojica Ruiz.

Alumnos:

Santiago Efrain Itzincab Poot

Rogerio Emmanuel Canto Romero

Eduardo Alberto Gonzalez Ortega

Jafet Andree Mena Solis

Contenido

Unidad #1

Estándar de conteo

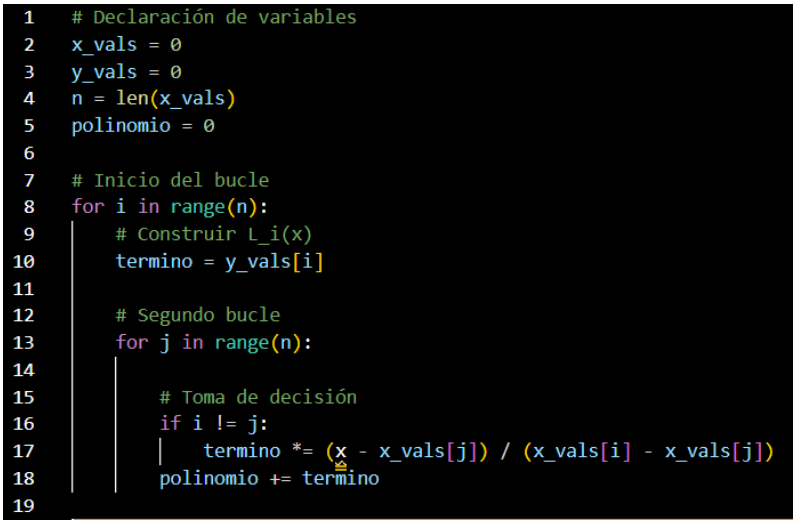
El siguiente estándar de conteo tiene como propósito servir como guía precisa para identificar y clasificar todas las líneas lógicas o físicas de un programa, basado en un paradigma de programación estructurada en el lenguaje de programación de Python.

Tipo de Conteo	**Tipo**	**Lenguaje**
Físico / Lógico	Físico	Python
Tipo de sentencia	**Incluir**	**Comentarios**
Ejecutables	Sí	
No ejecutables:		
Declaraciones	Si	
Directivas del compilador	Si	
Comentarios:		
-En una sola línea	No	
-Varias líneas	No	
-Líneas vacías	No	
-Comentarios en código	No	
Importaciones	Si	
Decoradores	Si	
Palabras reservadas	Si	

Tipo de conteo Físico / Lógico	**Tipo**	**Lenguaje**
	Lógico	Python
Tipo de sentencia	**Incluir**	**Comentarios**
Ejecutables		
if	Sí	
elif	No	
else	No	
for	Sí	
while	Sí	
def	Sí	
class	No	No es programación orientada a objetos, por tanto se espera que no haya declaración class.
try	Sí	
except	No	
with	Sí	
No ejecutables		
Comentarios		
- En una sola línea	No	
- Multilínea	No	
- Líneas en blanco o vacías	No	
Importaciones	No	
Decoradores	No	
Palabras reservadas	No	A excepción de las mencionadas anteriormente no se toman en consideración otro tipo de palabras reservadas

Ejemplo de Conteo de Líneas:

- Archivo con líneas vacías y comentarios:



```python
# Declaración de variables
x_vals = 0
y_vals = 0
n = len(x_vals)
polinomio = 0

# Inicio del bucle
for i in range(n):
    # Construir L_i(x)
    termino = y_vals[i]

    # Segundo bucle
    for j in range(n):

        # Toma de decisión
        if i != j:
            termino *= (x - x_vals[j]) / (x_vals[i] - x_vals[j])
        polinomio += termino

```

  - Líneas físicas: 10
  - Líneas lógicas: 3

Estándar de codificación

El estándar de codificación tiene como principal objetivo garantizar la calidad, mantenibilidad y uniformidad del código fuente en todos y cada uno de los proyectos realizados por el equipo de desarrollo, promoviendo de esta manera buenas prácticas y eficientes.

Longitud de líneas:

La longitud de todas las líneas de código deberá estar en el rango máximo de 80 caracteres; Se pueden hacer excepciones para líneas de código que necesiten unos caracteres más cuando sea completamente inevitable. Los espacios en blanco no son contabilizados como carácter así como los saltos en línea.

Comentarios:

Los comentarios deben ser frases completas y concisas explicando las decisiones que se tomaron durante el desarrollo del sistema.

Los comentarios de línea comienzan con un “#”, luego aplicar un espacio para empezar a escribir el comentario.

Siempre deben empezar con una letra en mayúscula, evitar empezar con el nombre de un identificador.

Los comentarios dentro de líneas con código están prohibidos, los comentarios de preferencia deben ir en una línea aparte.

Usar triple comillas para comentarios en bloques no está permitido; Usaremos # en cada línea del bloque de comentario, iniciando la primera palabra con mayúscula y un espacio después de #.

Espacios en blanco:

Debemos utilizar espacios en blanco entre elementos que están separados por comas; por ejemplo, las propiedades de una función.

```
# Función para construir el polinomio de Lagrange
def Lagrange_polynomial_symbolic(x_vals, y_vals):
```

Saltos de línea e indentación

Las funciones tendrán dos líneas vacías arriba de y debajo de las mismas.

Las estructuras de control tendrán una línea vacía antes de su declaración y después de que hayan sido terminadas.

Se usarán 4 espacios para hacer cada indentación; queda prohibido el uso de las tabulaciones.

Importaciones*:*

Las Importaciones deben ser declaradas en líneas distintas usando el siguiente orden de importación: librería estándar, librerías de terceros, librerías propias o locales.

Se recomienda siempre usar importaciones absolutas y evitar el uso de las importaciones relativas.

Identificadores:

Cuando nombramos una variable el nombre debe ser corto y explicito; la variable debe estar en minúsculas. Cuando declaramos una variable de dos o más palabras, la primera empieza en minúscula, y las demás empiezan en minúsculas o mayúsculas y van separadas por un “_” sin espacio (funcion_ejemplo).

Los identificadores de las funciones y las clases siempre empiezan con mayúsculas a diferencia de las variables, al igual que las variables si son conformadas por una

o más palabras se separan por un “_”, en caso de ser dos palabras la segunda después de _ puede ir en mayúscula o minúscula.

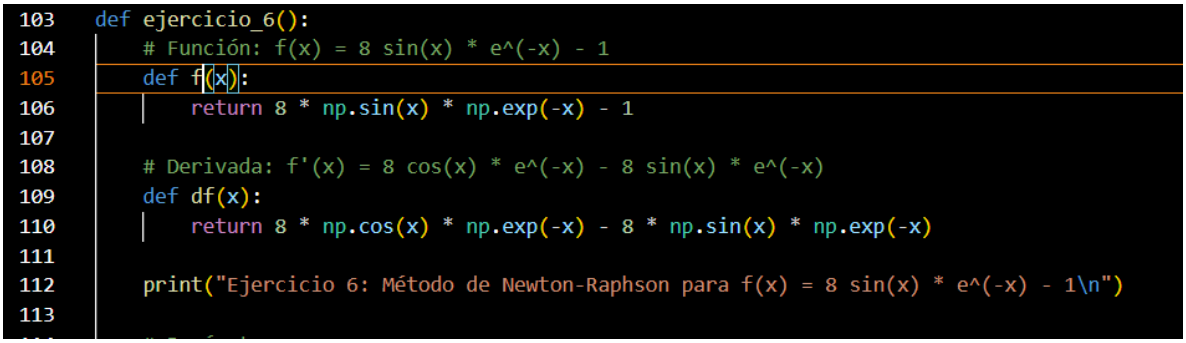
Estructura de Control:

solo puede haber una declaración de estructura de control por línea, es decir, no pueden haber anidadas más de una estructura de control por línea.

Uso de funciones:

Para las funciones se indica que no debe existir funciones dentro de funciones. Cada función será declarada aparte.

Mal uso:



Manual de usuario

Introducción:

Este script permite analizar archivos Python para verificar si cumplen con ciertos estándares de codificación y contar las líneas físicas y lógicas del código.

Requisitos Previos:

- Python instalado en tu equipo (versión 3.6 o superior recomendada).
- Biblioteca tkinter incluida (suele venir con Python por defecto).

Características Principales:

- Verificar estándares de codificación:
  Longitud de líneas (máximo 80 caracteres).
  Comentarios correctamente formateados.
  Indentación consistente (múltiplos de 4 espacios).
  Identificadores en minúsculas.
  Nombres funciones en mayúsculas al inicio.
- Contar líneas físicas (líneas que aparecen en el archivo).
- Contar líneas lógicas (aquellas que representan acciones de control o definiciones clave).

Instrucciones de Uso

1. Ejecutar el Script:

Guarda el código en un archivo .py (por ejemplo, proyecto1.py).

Opciones de ejecución:

- **Run Python File**: Se puede ejecutar con el botón de Run Python File si se está utilizando un IDE que soporte archivos Python.
- **Terminal:** Abrir una terminal en la ruta donde este guardado el archivo proyecto1.py e ingresar el comando *"python proyecto1.py".*

2. Ejecución:

- Seleccionar un Archivo:

Se abrirá un cuadro de diálogo para seleccionar un archivo Python. Navega hasta la ubicación del archivo que deseas analizar y selecciónalo.

3. Resultados:

Si el archivo cumple con los estándares, verás un mensaje indicando que no se encontraron errores.

Si hay errores, se mostrará una descripción detallada de cada problema, indicando la línea y el tipo de error.

Adicionalmente, el programa mostrará:

- Número de líneas físicas.
- Número de líneas lógicas.

4. Errores y Manejo

Si seleccionas un archivo no válido o inexistente, el programa notificará el error.

Otros errores inesperados serán capturados y mostrados en pantalla.

Descripción de Funciones

1. Verificar_estandares_archivo(archivo).

Analiza cada línea del archivo para identificar:

- Longitud mayor a 80 caracteres.
- Comentarios incorrectamente formateados (sin un espacio después de #, en minúsculas o vacíos).
- Indentación no múltiplo de 4 espacios.
- Uso incorrecto de comas (sin espacio después o con espacio antes).
- Identificadores que no están en minúsculas.
- Funciones sin nombres en minúsculas.

Parámetro:

- archivo: Ruta del archivo a analizar.

Salida:

- Imprime los errores encontrados (si existen) o confirma que no hay problemas.

2. Contar_lineas_fisicas(file_path).

Cuenta las líneas físicas en el archivo (incluyendo todas menos las vacías o comentarios).

Parámetro:

- file_path: Ruta del archivo a analizar.

Salida:

- Número de líneas físicas.

3. Contar_lineas_logicas(file_path).

Cuenta las líneas lógicas en el archivo, considerando palabras clave como if, for, while, entre otras.

Parámetro:

- file_path: Ruta del archivo a analizar.

Salida:

- Número de líneas lógicas.

4. Interfaz con tkinter:

Usa la biblioteca tkinter para mostrar un cuadro de diálogo de selección de archivos, permitiendo al usuario seleccionar fácilmente el archivo a analizar.

Posibles Errores

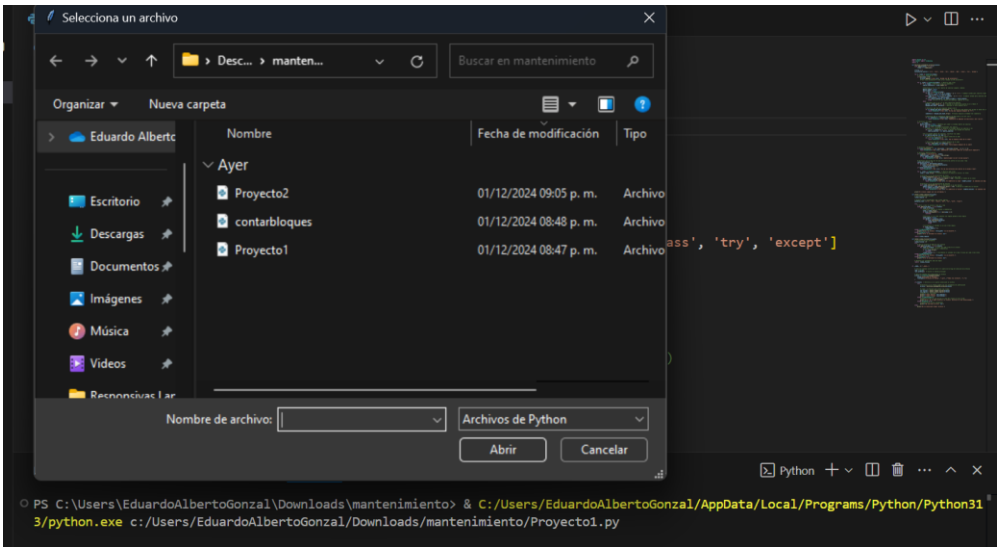
Archivo no encontrado: Ocurre si seleccionas un archivo inexistente.

Errores de formato: El programa reportará errores específicos relacionados con estándares de codificación.

Excepciones generales: Si ocurre un problema inesperado, se mostrará un mensaje con el detalle del error.

Ejemplo de uso:

Ejecutamos el programa y nos abrirá una ventana para elegir nuestro archivo a analizar



Una vez seleccionado el archivo el programa nos arrojará el resultado en la consola

```
3/python.exe c:/Users/EduardoAlbertoGonzal/Downloads/mantenimiento/Proyecto1.py
El archivo cumple con los estándares.
Líneas Lógicas: 44
Líneas Físicas: 152
PS C:\Users\EduardoAlbertoGonzal\Downloads\mantenimiento>
```

Manual Técnico

Este manual está dirigido a desarrolladores que deseen comprender la implementación técnica del script proporcionado, sus funciones, flujos de trabajo y cómo modificarlo o extenderlo.

Descripción General

Este script realiza un análisis estático de un archivo Python seleccionado por el usuario.

Se enfoca en:

- Verificar el cumplimiento de estándares de codificación.
- Contar líneas físicas y lógicas en el archivo.

Tecnologías Utilizadas:

- Python (versión 3.6 o superior).
- Biblioteca tkinter para la interfaz gráfica de selección de archivos.
- Módulo re para el manejo de expresiones regulares.

Arquitectura del Script

El script está compuesto por las siguientes secciones:

1. Importación de Bibliotecas
- tkinter y filedialog: Usados para implementar la selección de archivos mediante una GUI minimalista.
- re: Utilizado para realizar validaciones avanzadas en las líneas del archivo.
2. Funciones Principales
- Verificar_estandares_archivo: Validación de estándares de codificación.
- Contar_lineas_fisicas: Conteo de líneas físicas no vacías.
- Contar_lineas_logicas: Conteo de líneas que representan operaciones lógicas clave.
3. Flujo Principal
- Abre un cuadro de diálogo para seleccionar un archivo.

- Procesa el archivo seleccionado mediante las funciones anteriores.
- Muestra los resultados en la consola.

Descripción de las Funciones

1. Verificar_estandares_archivo(archivo)

Propósito: Valida que el archivo cumpla con los siguientes estándares:

- Líneas con una longitud máxima de 79 caracteres.
- Comentarios correctamente formateados:
- Un espacio después de #.
- Comentarios no vacíos y comenzando con mayúscula.
- Indentación en múltiplos de 4 espacios.
- Identificadores definidos en minúsculas.
- Nombres de funciones en mayúsculas.
- Estructuras de control (e.g., if, for) no combinadas en la misma línea.

Parámetros:

- archivo: Ruta del archivo a analizar.

Salida:

- Lanza excepciones detalladas si encuentra errores de formato.
- Imprime "El archivo cumple con los estándares" si no hay errores.

Algoritmo Interno:

a) Leer todas las líneas del archivo.
b) Analizar cada línea para identificar problemas relacionados con:

- Longitud.
- Comentarios.
- Indentación.
- Identificadores y nombres.
- Uso de estructuras de control.
- Levantar excepciones si alguna regla es violada.

2. Contar_lineas_fisicas(file_path)

Propósito: Cuenta las líneas físicas, excluyendo líneas vacías.

Parámetros:

- file_path: Ruta del archivo a analizar.

Salida:

- Número entero representando el conteo de líneas físicas.

Algoritmo Interno:

a) Abrir el archivo en modo de solo lectura.
b) Iterar línea por línea.
c) Incrementar el contador solo si la línea no está vacía.
d) Manejar excepciones como archivo inexistente.

3. Contar_lineas_logicas(file_path)

Propósito: Cuenta las líneas lógicas, considerando palabras clave como if, for, while, def, etc.

Parámetros:

- file_path: Ruta del archivo a analizar.

Salida:

- Número entero representando el conteo de líneas lógicas.

Algoritmo Interno:

a) Abrir el archivo en modo de solo lectura.
b) Iterar sobre cada línea, descartando comentarios y líneas vacías.
c) Identificar palabras clave al inicio de la línea.
d) Incrementar el contador por cada línea lógica encontrada.

4. tkinter y Selección de Archivos

Propósito: Permitir al usuario seleccionar un archivo para analizar mediante una interfaz gráfica.

Flujo:

- Crear una instancia de tk.Tk.
- Ocultar la ventana principal.
- Abrir un cuadro de diálogo usando filedialog.askopenfilename.
- Retornar la ruta del archivo seleccionado.

Flujo Principal del Script

1. Inicio del Programa
- Oculta la ventana principal de tkinter.
- Solicita al usuario que seleccione un archivo.
2. Verificación del Archivo
- Llama a Verificar_estandares_archivo.
- Reporta cualquier problema encontrado.
3. Conteo de Líneas
- Llama a Contar_lineas_fisicas y Contar_lineas_logicas.
- Imprime los resultados.
4. Manejo de Errores
- Captura excepciones relacionadas con la selección o lectura de archivos.
- Muestra mensajes detallados sobre cualquier error inesperado.

Consideraciones Técnicas

1. Rendimiento

- El análisis líneo por línea es eficiente para archivos pequeños y medianos.
- Para archivos grandes, podría optimizarse utilizando procesamiento paralelo o leyendo bloques de líneas.

2. Limitaciones

- Solo verifica estándares básicos de codificación.

- No analiza bloques de código más complejos, como la cohesión del diseño.

3. Extensibilidad

Para agregar nuevas reglas de verificación:

a) Modifica la función Verificar_estandares_archivo para incluir nuevas condiciones.
b) Utiliza expresiones regulares (re) o estructuras condicionales según sea necesario.

Unidad #2

Casos de prueba

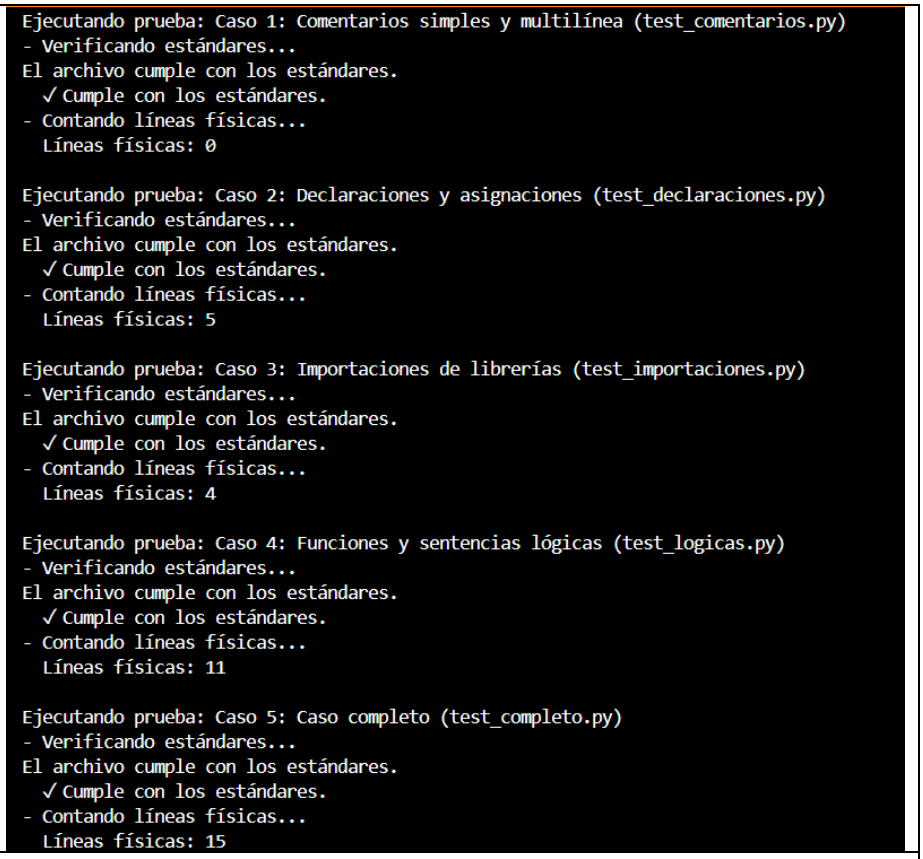
Los casos de prueba están diseñados para verificar que el código funcione sin errores, tal y como se debe esperar en diferentes situaciones. Así como que se verifique el funcionamiento del programa por parte del usuario.

El siguiente cuadro nos da una forma de registro de los casos de prueba:

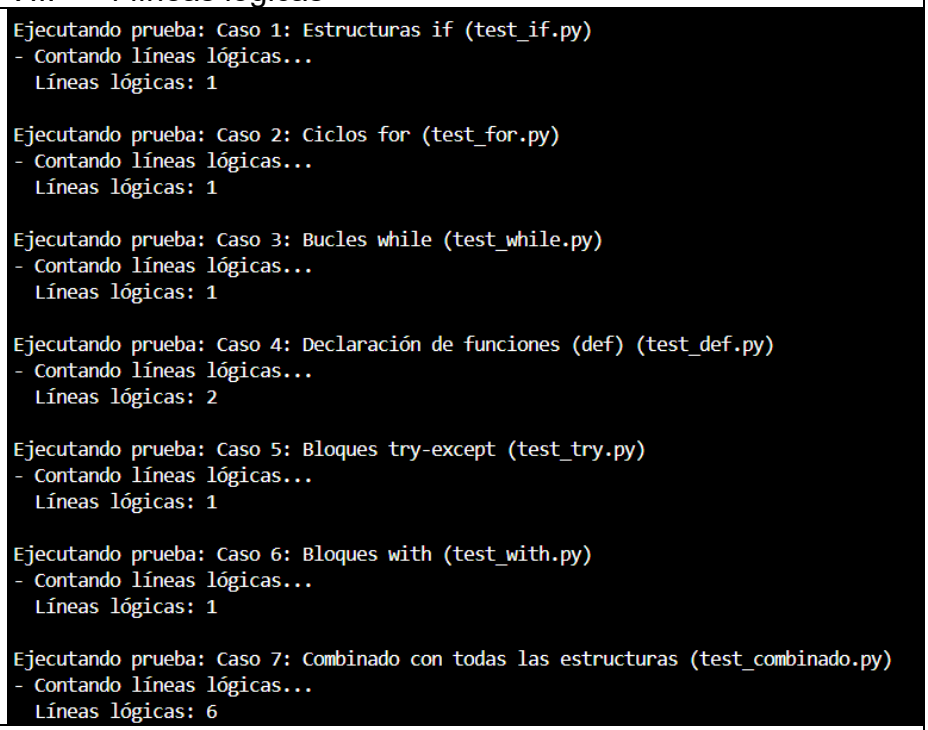
Tipo de prueba	
Nombre de la prueba/Numero objetivo de la prueba	Identificar único para la prueba
Descripción de la prueba	Describe las entradas y el procesamiento que va a tener el programa.
Condiciones de la prueba	Menciona las herramientas que se utilizaron para llevar a cabo en la prueba.
Resultados esperados	Lista de los resultados que se esperan obtener.
Resultados actuales	Lista de los resultados que se produjo durante la prueba.
Comentarios	Describe posibles comentarios que puedan ayudar a entender las pruebas.
Estado	Indicar el estado de la prueba: - Exitosa 1 - Fallida 0

Pruebas Unitarias:

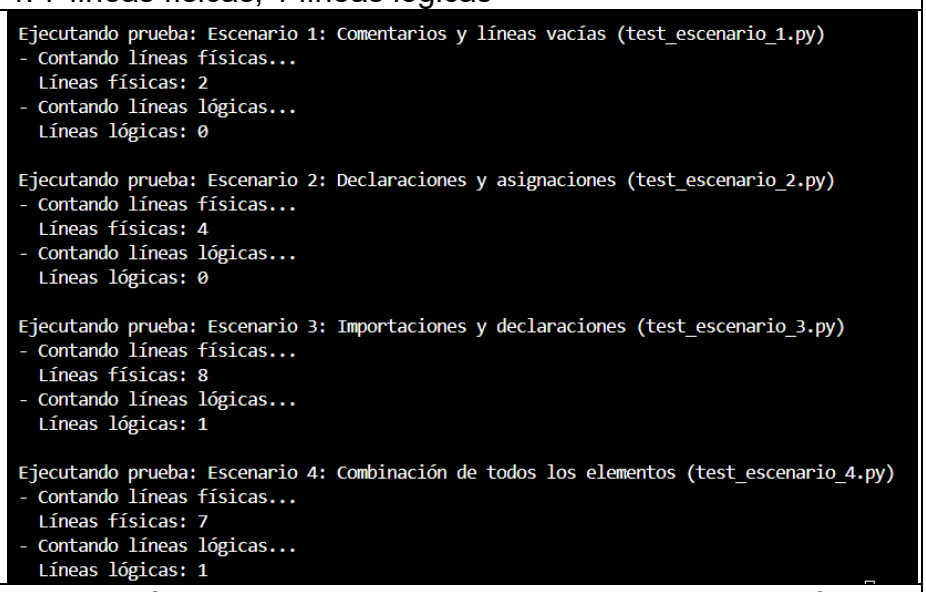
Tipo de prueba Prueba unitaria	
Nombre de la prueba/Numero objetivo de la prueba	P1 / Prueba_A Validar que el código sigue correctamente el estándar de conteo de líneas físicas.
Descripción de la prueba	El programa evalúa 5 escenarios predefinidos, diseñados para verificar los casos más relevantes relacionados con líneas físicas. **1. Comentarios:** Solo comentarios simples Se espera que no sean detectados como líneas físicas. **2. Declaraciones:** Declaraciones de variables, asignaciones e impresiones en consola, ignorando comentarios. **3. Importaciones:** Involucra importaciones de librerías o archivos externos, que deben contarse como líneas físicas. **4. Lógicas:** Contiene funciones, sentencias lógicas, impresiones en consola, docstrings y comentarios, sin afectar el conteo. **5. Completo:** Combina todos los escenarios previos en un solo caso, validando la consistencia en presencia de múltiples casos.
Condiciones de la prueba	Un script test1.py donde se tienen guardado las pruebas para realizar de manera automática sin necesidad de generar una por una.
Resultados esperados	1. 0 líneas físicas 2. 5 líneas físicas 3. 4 líneas físicas 4. 11 líneas físicas 5. 15 líneas físicas

Resultados actuales	
Comentarios	El código funciona correctamente en el conteo de líneas físicas.
Estado	1

Tipo de prueba Prueba unitaria	
Nombre de la prueba/Numero objetivo de la prueba	P2 / Prueba_B Validar que el código sigue correctamente el estándar de conteo de líneas lógicas.
Descripción de la prueba	Resultados esperados por escenario I. if: Contar las líneas lógicas asociadas a la palabra clave if, reconociendo las condiciones evaluadas como una sola línea lógica. II. for: Identificar ciclos for, donde el encabezado del bucle se cuenta como una línea lógica, sin incluir las operaciones dentro del bloque. III. while: Contar la línea lógica correspondiente al encabezado de un bucle while, ignorando el contenido del bloque repetitivo. IV. def: Reconocer la línea lógica asociada a la declaración de funciones mediante la palabra clave def. Cada función es una línea lógica.

	V. try: Evaluar las líneas lógicas correspondientes al bloque try que gestiona excepciones, considerando el encabezado try y no los bloques except. VI. with: Contar la línea lógica asociada a bloques with, que gestionan el contexto de operaciones con recursos. VII. Combinado: Evaluar un archivo que contiene todas las estructuras anteriores, verificando que cada encabezado if, for, while, def, class, try, y with sea identificado correctamente como una línea lógica.
Condiciones de la prueba	Un script test2.py donde se tienen guardado las pruebas para realizar de manera automática sin necesidad de generar una por una.
Resultados esperados	I. 1 líneas lógicas II. 1 líneas lógicas III. 1 líneas lógicas IV. 2 líneas lógicas V. 1 líneas lógicas VI. 1 líneas lógicas VII. 4 líneas lógicas
Resultados actuales	
Comentarios	El código funciona correctamente en el conteo de líneas físicas.
Estado	1

Pruebas de integración:

Tipo de prueba Prueba integración	
Nombre de la prueba/Numero objetivo de la prueba	P3 / Prueba_AB Validar que el código sigue correctamente el estándar de conteo de líneas físicas y lógicas.
Descripción de la prueba	El programa evalúa 4 escenarios predefinidos, diseñados para verificar los casos más relevantes relacionados con líneas físicas y lógicas. 1: Comentarios y líneas vacías 2: Declaraciones de variables, asignaciones e impresiones 3: Importaciones y declaraciones de clases y funciones 4: Combinación de todos los elementos anteriores
Condiciones de la prueba	Un script test3.py donde se tienen guardado las pruebas para realizar de manera automática sin necesidad de generar una por una.
Resultados esperados	1. 2 líneas físicas, 0 líneas lógicas 2. 4 líneas físicas, 0 líneas lógicas 3. 8 líneas físicas, 1 líneas lógicas 4. 7 líneas físicas, 1 líneas lógicas
Resultados actuales	
Comentarios	El código funciona correctamente en el conteo de líneas físicas y lógicas.
Estado	1

Unidad #3

Estimación de tamaño

Se utilizará la métrica de Puntos Funcionales para el tamaño y complejidad del sistema.

Se tomará en consideración el conteo de los siguientes elementos funcionales:

- Entradas lógicas
- Salidas
- Consulta (Querys)
- Archivos Lógicos Internos
- Archivos Lógicos Externos

Asignación del grado de complejidad con base a la siguiente tabla:

Elemento Funcional	Factor de Ponderación		
	Simple	Promedio	Complejo
Entradas Externas	3	4	6
Salidas Externas	4	5	7
Consultas Externas	3	4	6
Archivos Lógicos Externos	7	10	15
Archivos Lógicos Internos	5	7	10

Ecuación para el conteo de Puntos Funcionales sin Ajuste:

$$\boldsymbol{UFC = \sum Cantidad_{elemento} \cdot Peso_{elemento}}$$

La siguiente plantilla se utilizará para calcular el factor de complejidad técnica para los puntos funcionales sin ajustar:

Componentes del factor de complejidad técnica		
F_1	Fiabilidad de la copia de seguridad y recuperación	
F_2	Funciones distribuidas	
F_3	Configuración utilizada	
F_4	Facilidad operativa	
F_5	Complejidad de interfaz	
F_6	Reutilización	
F_7	Instalaciones múltiples	
F_8	Comunicaciones de datos	
F_9	Desempeño	
F_{10}	Entrada de datos en línea	
F_{11}	Actualización en línea	
F_{12}	Procesamiento complejo	
F_{13}	Facilidad de instalación	
F_{14}	Facilidad de cambio	
	Total	

Valores posibles para dar:

0-Irrelevante o sin influencia

1-Incidental

2-Moderado

3-Medio

4-Significativo

Ecuación para el factor de complejidad técnica:

$$\mathbf{TFC = 0.65 + 0.01 \sum_{i=1}^{14} F_i}$$

Ecuación para el conteo de Puntos Funcionales Ajustado

$$FP = UFC \cdot TFC$$

Por cada punto funcional encontrado se considerará:

$$1pf\ tarda\ aprox\ 10\ hrs$$

Descripción del sistema1:

Escribir un programa para contar las líneas lógicas y líneas físicas en un programa, omitiendo comentarios y líneas en blanco. Utilizar y proveer el estándar de conteo y codificación utilizado.

Programa	LOC Lógicas	LOC Físicas
ABC	20	123
XYZ	34	345

Parte 1: Elementos identificados:

1- Entrada del archivo Python a evaluar en el conteo.
2- Salida de total de líneas físicas, lógicas y si cumple con el estándar de codificación.

Parte 2: Conteo de los puntos de Función sin ajuste:

Elemento	Cantidad	Peso	Total
Entradas Externas	1	4	4
Salidas Externas	3	5	15
Consultas Externas	0	0	0
Archivos Lógicos Externos	0	0	0
Archivos Lógicos Internos	0	0	0
Total			19

$$UFC = (1 * 4) + (3 * 5) = 19$$

$$UFC = 19$$

Componentes del factor de complejidad técnica		
F_1	Fiabilidad de la copia de seguridad y recuperación	0
F_2	Funciones distribuidas	3
F_3	Configuración utilizada	0
F_4	Facilidad operativa	0
F_5	Complejidad de interfaz	0
F_6	Reutilización	3
F_7	Instalaciones múltiples	0
F_8	Comunicaciones de datos	0
F_9	Desempeño	0
F_{10}	Entrada de datos en línea	0
F_{11}	Actualización en línea	0
F_{12}	Procesamiento complejo	0
F_{13}	Facilidad de instalación	1
F_{14}	Facilidad de cambio	3
	Total	10

$$TFC = 0.65 + 0.01(10) = 0.75$$

$$FP = 19 \cdot\ 0.75 = \ 14.25 \approx 14$$

Tiempo esperado que puede tomar realizar el sistema:

$$Tiempo_{hrs} = 14 \cdot 10\ hrs = 140\ hrs\ \approx 140\ hrs$$

$$Tiempo_{dias} = \frac{140}{24} = 5.833\ días\ \approx 6\ días$$

Considerando que dos personas trabajaran en el desarrollo del programa podemos deducir el siguiente valor:

$$Tiempo_{dias} = \frac{6\ dias}{2\ desarrolladores}$$

3 días para cada desarrollador

Unidad #5

Aseguramiento de la calidad

Objetivo del Aseguramiento de la Calidad

El propósito principal del aseguramiento de la calidad es asegurar que el sistema:

- Cumpla con los requisitos tanto funcionales como no funcionales definidos previamente.
- Proporcione resultados coherentes y precisos en distintos entornos.
- Sea confiable, fácil de mantener y eficiente.

Este apartado tiene como fin detallar las actividades, roles, procesos y herramientas que se utilizarán para garantizar la calidad tanto técnica como administrativa del proyecto.

Metodología para el Aseguramiento de la Calidad

El aseguramiento de la calidad se implementará en dos niveles: **administrativo** y **técnico**, con el objetivo de asegurar que tanto el proceso de desarrollo como el producto final cumplan con los estándares establecidos.

Calidad Administrativa

A nivel administrativo, se llevarán a cabo las siguientes acciones:

- **Planificación del proyecto** de forma eficiente, con una adecuada asignación de recursos y un seguimiento continuo del progreso.
- **Claridad en los roles y responsabilidades**, asegurando que cada miembro del equipo tenga claras sus tareas y objetivos.
- **Revisiones periódicas** del avance del proyecto, controlando los plazos y los recursos, para garantizar que el proyecto se mantenga en el rumbo y cumpla con los estándares establecidos.

Calidad Técnica

En el ámbito técnico, se emplearán dos estrategias principales para garantizar la calidad del producto:

- **Revisión del código (inspecciones)**: Se realizarán inspecciones regulares del código para detectar posibles fallos en el diseño y la lógica, lo que permitirá corregir problemas en etapas tempranas del desarrollo.
- **Pruebas del sistema**: Se llevará a cabo un conjunto de pruebas, que incluyen:
    - **Pruebas unitarias** para asegurar el funcionamiento adecuado de cada componente individual del sistema.
    - **Pruebas de integración** para confirmar que las diferentes partes del sistema trabajen correctamente juntas.

Ambas estrategias, tanto las inspecciones como las pruebas, estarán interrelacionadas y se realizarán a lo largo de todo el ciclo de vida del desarrollo, asegurando la detección temprana de errores y la validación continua de que el producto cumple con los estándares de calidad.

Roles y responsabilidades

A continuación, se muestra los roles que tendrá cada persona del equipo y sus responsabilidades en el proceso del aseguramiento de la calidad (inspecciones).

Líder de calidad: responsable de planificar y supervisar las actividades de QA Santiago Efrain Itzincab Poot.

Moderador: Se encargará de coordinar las inspecciones de calidad.

Inspectores: Serán los que revisaran el código fuente y reportaran defectos.

Autor del código fuente: Persona que desarrolló el módulo que se está inspeccionando.

Secretario: Documenta los hallazgos durante las inspecciones.

Lector: Se encarga de leer el documento, puede ser igual un inspector quien ocupe este rol.

Inspecciones

Inspección 1

- Fecha de la inspección: 11/11/2024
- Modulo inspeccionado: Modulo de conteo de líneas lógicas y físicas
- Tipo de inspección: Revisión de código
- Objetivo de la inspección: Identificar defectos en la lógica del algoritmo de conteo de líneas físicas y lógicas.

Participantes y roles:

- Moderador: Eduardo Alberto Gonzalez Ortega
- Inspector: Jafet Andree Mena Solis.
- Autor: Santiago Efrain Itzincab Poot
- Secretario: Rogerio Emmanuel Canto Romero
- Lector: Jafet Andree Mena Solis

Resumen de la Inspección

Defectos encontrados: 2

Detalles de los Defectos

Defecto #1:

Descripción: El algoritmo considera a la declaración de clases como líneas lógicas cuando no deberían, ya que trabajamos en un entorno estructurado y no de objetos.
Impacto: Puede generar conteos incorrectos.

Acción propuesta: Actualizar la lógica para contar correctamente, con un estándar basado en estructurada.

Responsable: Santiago Efrain Itzincab Poot.

Defecto #2:

Descripción: El algoritmo considera los comentarios como líneas físicas.
Impacto: Puede generar conteos incorrectos.

Acción propuesta: Actualizar el algoritmo para que ya ignore los comentarios como líneas físicas.
Responsable: Santiago Efrain Itzincab Poot.

- Plazo estimado para las correcciones: (1 día).
- Revisión de seguimiento:

Seguimiento:

Los defectos han sido resueltos.

Notas adicionales: Se realizó una segunda revisión y los problemas fueron corregidos con éxito.

Inspección 2

- Fecha de la inspección: 13/11/2024
- Modulo inspeccionado: Modulo de conteo de líneas lógicas y físicas en pruebas
- Tipo de inspección: pruebas de código
- Objetivo de la inspección: Identificar defectos en la ejecución de conteo de líneas físicas y lógicas.

Participantes y roles:

- Moderador: Eduardo Alberto Gonzalez Ortega
- Inspector: Santiago Efrain Itzincab Poot, Rogerio Emmanuel Canto Romero
- Autor: Jafet Andree Mena Solis.
- Secretario: Rogerio Emmanuel Canto Romero

- Lector: Santiago Efrain Itzincab Poot

Resumen de la Inspección

Defectos encontrados: 1

Detalles de los Defectos

Defecto #1:

Descripción: Durante las pruebas unitarias se hacia el conteo tanto de líneas lógicas y físicas juntas, cuando deberían hacerse por separado.
Impacto: Puede afectar en la documentación.

Acción propuesta: dividir los casos de pruebas para solo evaluar físicas y lógicas una a la vez.
Responsable: Jafet Andree Mena Solis.

- Plazo estimado para las correcciones: (1 día).
- Revisión de seguimiento:

Seguimiento:

Las pruebas fueron realizadas por separado.

Notas adicionales: Se realizó una segunda revisión y los problemas fueron corregidos con éxito.

Inspección 3

- Fecha de la inspección: 15/11/2024
- Modulo inspeccionado: código fuente
- Tipo de inspección: revisión del código para verificar que cumple con los estándares de codificación.
- Objetivo de la inspección: identificar defectos del código fuente que tengan que ver con los estándares.

Participantes y roles:

- Moderador: Rogerio Emmanuel Canto Romero
- Inspector: Jafet Andree Mena Solis, Eduardo Alberto Gonzalez Ortega
- Autor: Santiago Efrain Itzincab Poot
- Secretario: Rogerio Emmanuel Canto Romero
- Lector: Eduardo Alberto Gonzalez Ortega

Resumen de la Inspección

Defectos encontrados: 2

Detalles de los Defectos

Defecto #1:

Descripción: existen comentarios dentro de líneas físicas del código.
Impacto: Puede afectar la legibilidad del programa.

Acción propuesta: poner en una línea aparte los comentarios que debe llevar el programa.
Responsable: Santiago Efrain Itzincab Poot

Defecto #1:

Descripción: Los nombres de las funciones empiezan con minúsculas.
Impacto: En los estándares se establece que deben iniciar con mayúsculas.

Acción propuesta: cambiar la inicial de los nombres de las funciones.
Responsable: Santiago Efrain Itzincab Poot

- Plazo estimado para las correcciones: (1 día).
- Revisión de seguimiento:

Seguimiento:

El código fuente ya cumple con los estándares.

Notas adicionales: Se realizó una segunda revisión y los problemas fueron corregidos con éxito.