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

Estándar de codificación

Manual de usuario

Manual Técnico

Unidad #2

Casos de prueba

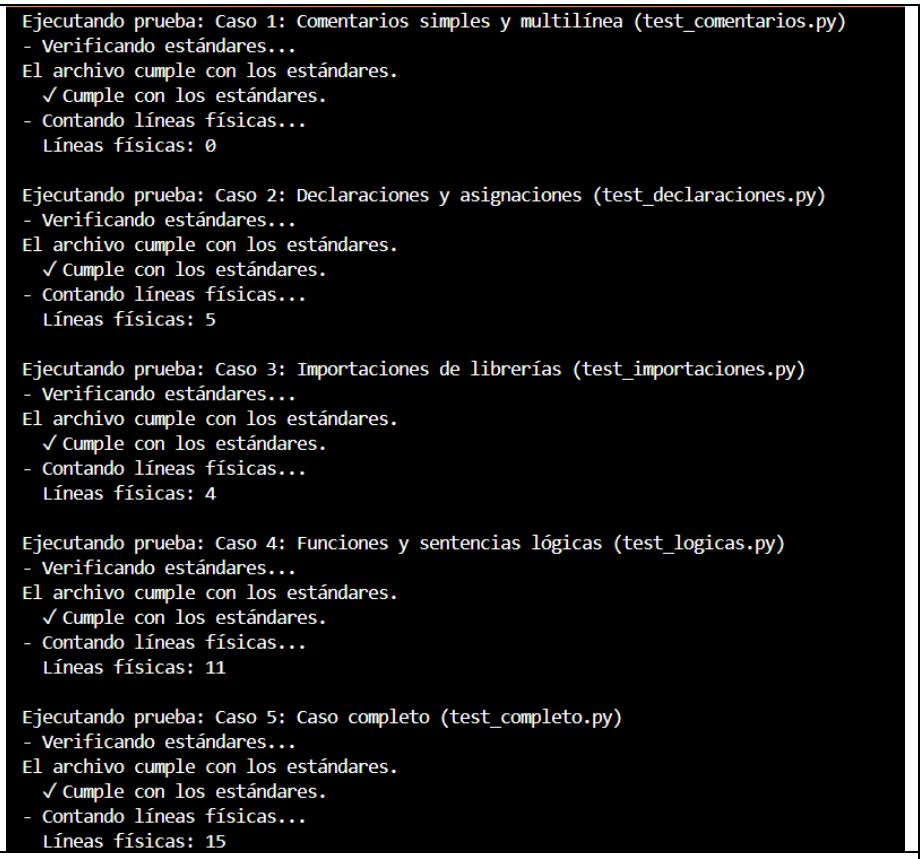
Los casos de prueba están diseñados para verificar que el código funcione sin errores, tal y como se debe esperar en diferentes situaciones. Así como que se verifique el funcionamiento del programa por parte del usuario.

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

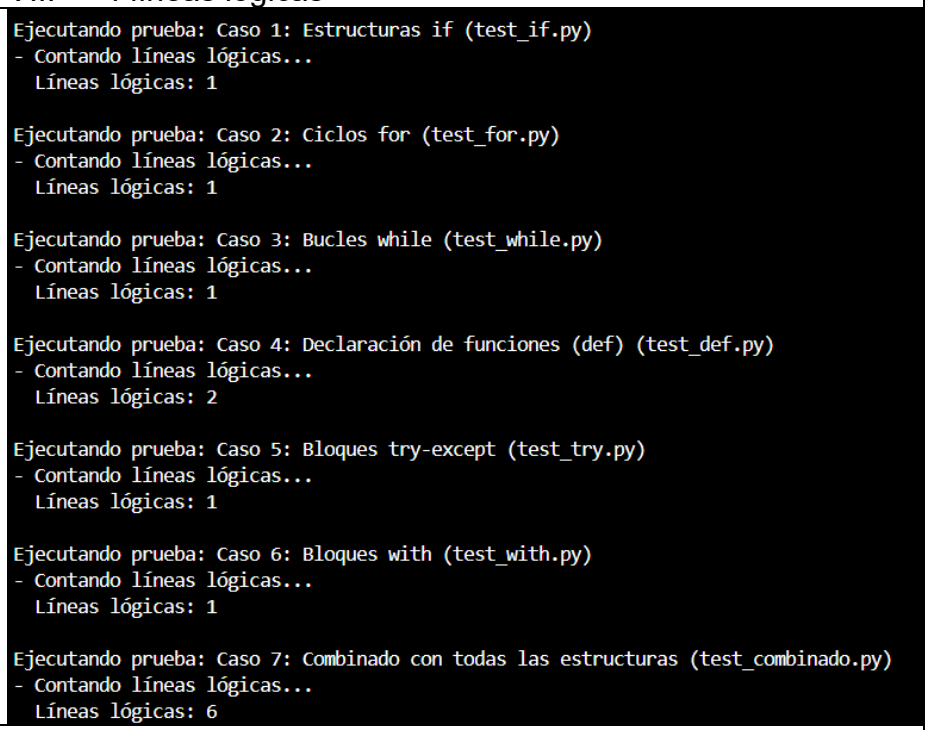
Tipo de prueba	
Nombre de la prueba/Numero objetivo de la prueba	Identificar único para la prueba
Descripción de la prueba	Describe las entradas y el procesamiento que va a tener el programa.
Condiciones de la prueba	Menciona las herramientas que se utilizaron para llevar a cabo en la prueba.
Resultados 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:

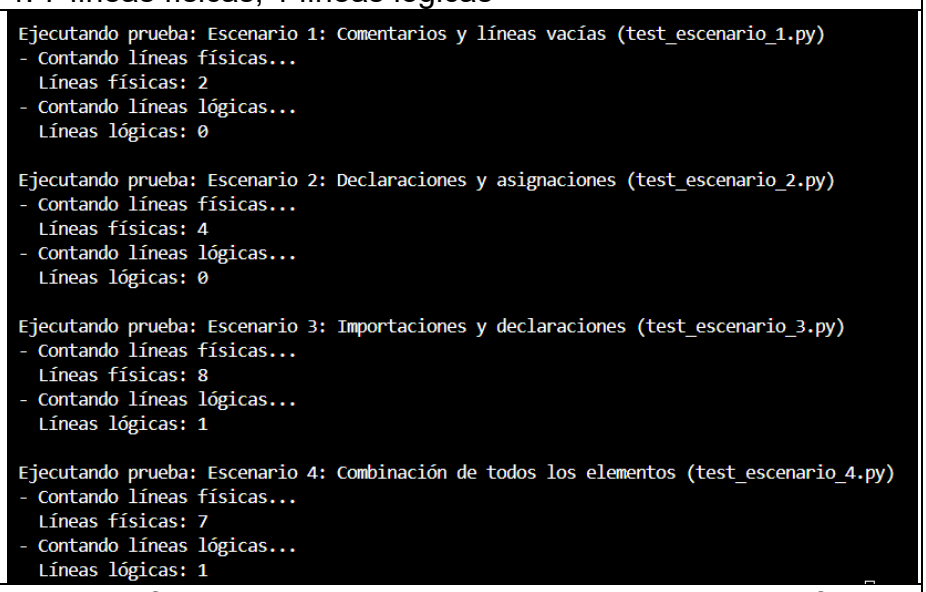
Tipo de prueba Prueba unitaria	
Nombre de la prueba/Numero objetivo de la prueba	P1 / Prueba_A Validar que el código sigue correctamente el estándar de conteo de líneas físicas.
Descripción de la prueba	El programa evalúa 5 escenarios predefinidos, diseñados para verificar los casos más relevantes relacionados 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. 11 líneas físicas 4. 4 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:

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

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	1	5	5
Consultas Externas	0	0	0
Archivos Lógicos Externos	0	0	0
Archivos Lógicos Internos	0	0	0
Total			9

$$UFC = (1 * 4) + (1 * 5) = 9$$

$$UFC = 9$$

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

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

$$FP = 9 \cdot \ 0.75 = 6.75 \ \approx 7$$

Tiempo esperado que puede tomar realizar el sistema:

$$Tiempo_{hrs} = 7 \cdot 10 \ hrs = 70 \ hrs \ \approx 70 \ hrs$$

$$Tiempo_{dias} = \frac{70}{24} = 2.916 \ días \ \approx 3 \ días$$

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

Inspecciones

Inspección 1

- Fecha de la inspección:
- 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:
- 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:
- 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.