Universidad Tecnológica de Durango

Tecnologías de la Información

Programación estructurada

Actividades

“Evidencias de Activiades y Tareas”

Alumnos:

- Barraza Torres Jesús Daniel

2°A BIS

Docente:

- Ing. Dagoberto Fiscal Gurrola, M.T.I.

Mayo 2025

Tabla de Ilustraciones

Actividad 1

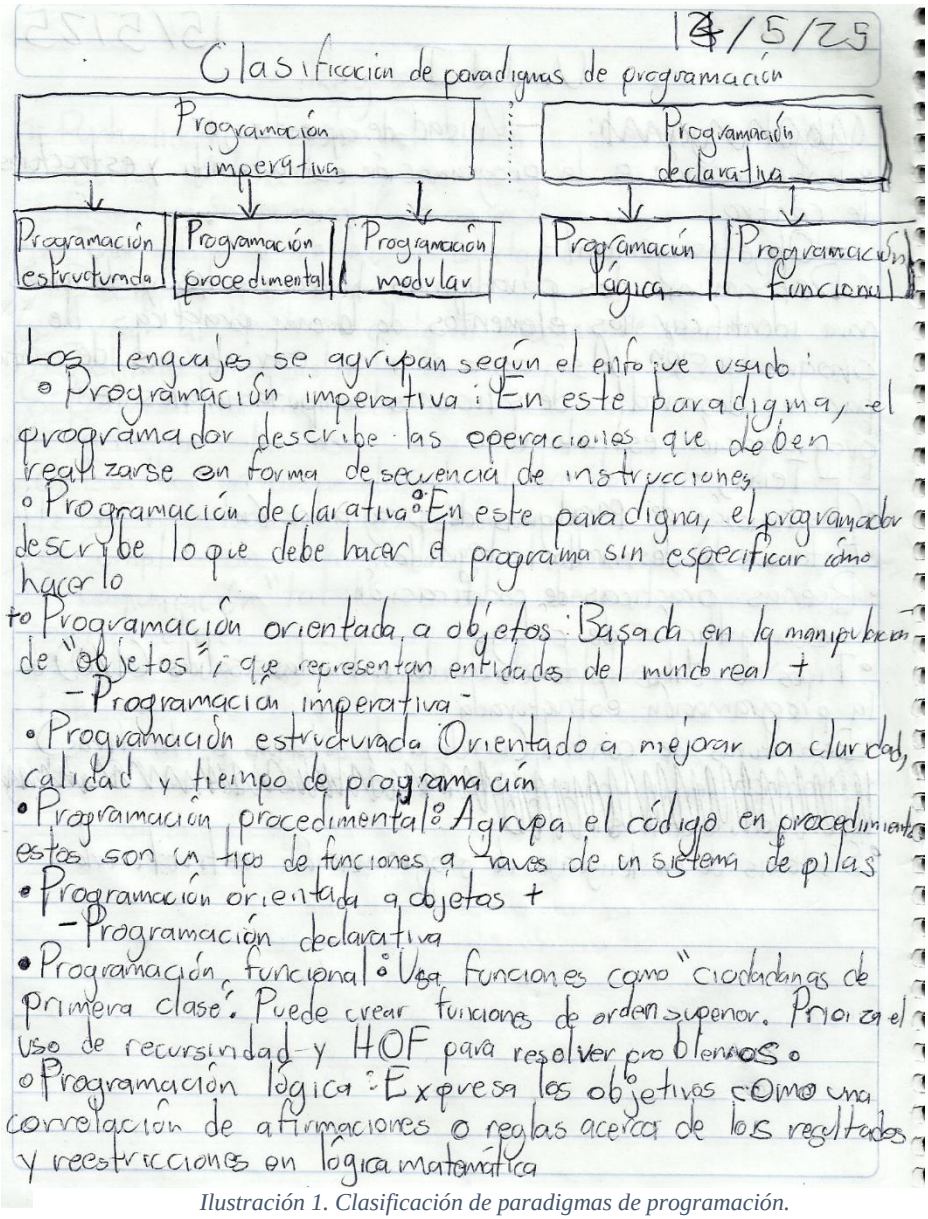
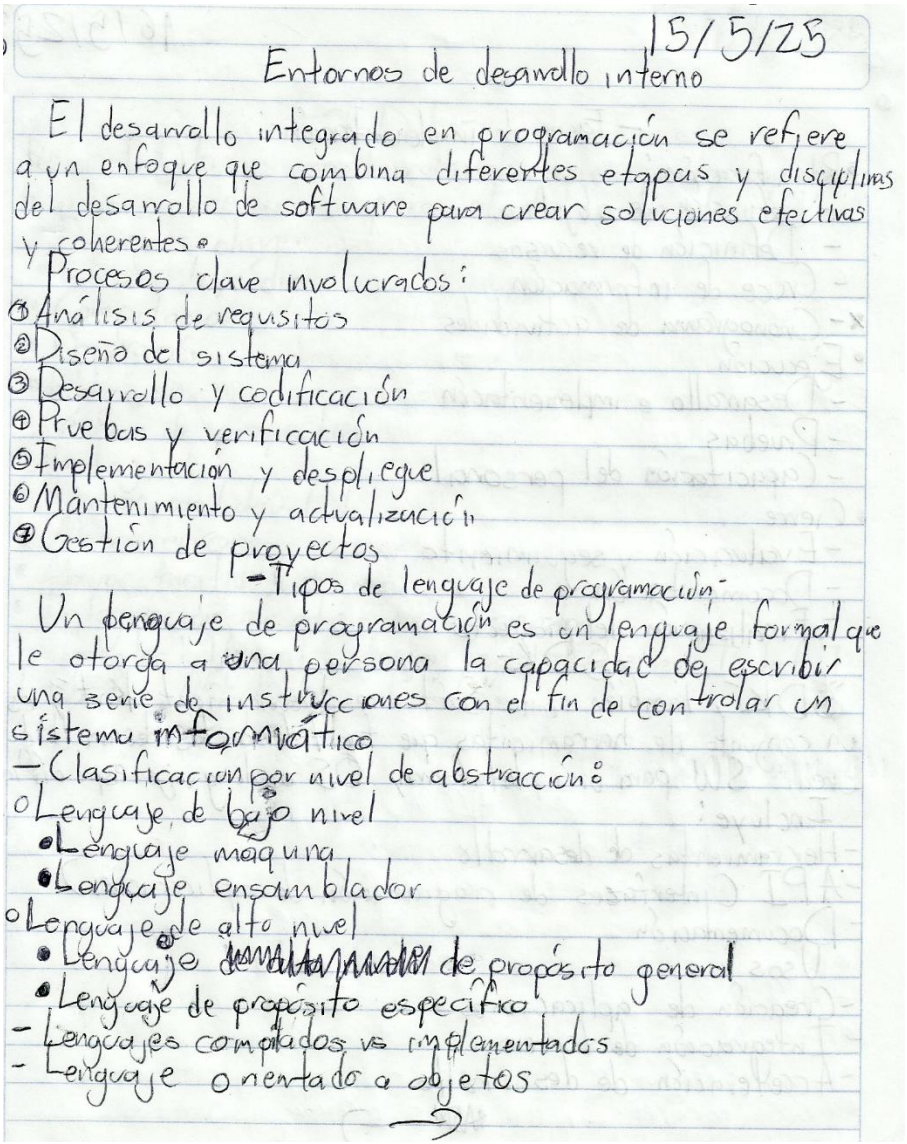


Ilustración 1. Clasificación de paradigmas de programación.

En esta ilustración podemos observar la clasificación de paradigmas de programación, los cuales se tratan de diversas maneras de programar, mostrando la manera de pensar de diversas personas debido a diversas funciones que el código debe de cumplir.

Actividad 2


15/5/25

Entornos de desarrollo interno

El desarrollo integrado en programación se refiere a un enfoque que combina diferentes etapas y disciplinas del desarrollo de software para crear soluciones efectivas y coherentes.

Procesos clave involucrados:

1. Análisis de requisitos
2. Diseño del sistema
3. Desarrollo y codificación
4. Pruebas y verificación
5. Implementación y despliegue
6. Mantenimiento y actualización
7. Gestión de proyectos

- Tipos de lenguaje de programación:

Un lenguaje de programación es un lenguaje formal que le otorga a una persona la capacidad de escribir una serie de instrucciones con el fin de controlar un sistema informático

- Clasificación por nivel de abstracción:

- Lenguaje de bajo nivel
  - Lenguaje máquina
  - Lenguaje ensamblador
- Lenguaje de alto nivel
  - Lenguaje de propósito general
  - Lenguaje de propósito específico

- Lenguajes compilados vs implementados
- Lenguaje orientado a objetos

Ilustración 2. Entornos de desarrollo interno pt.1

En esta ilustración podemos observar la parte de los entornos de desarrollo interno, los cuales son las diversas etapas y disciplinas del desarrollo de SW, a su vez, se encuentra el subtítulo de tipos de lenguajes de programación, que es muy explicativo por sí mismo.

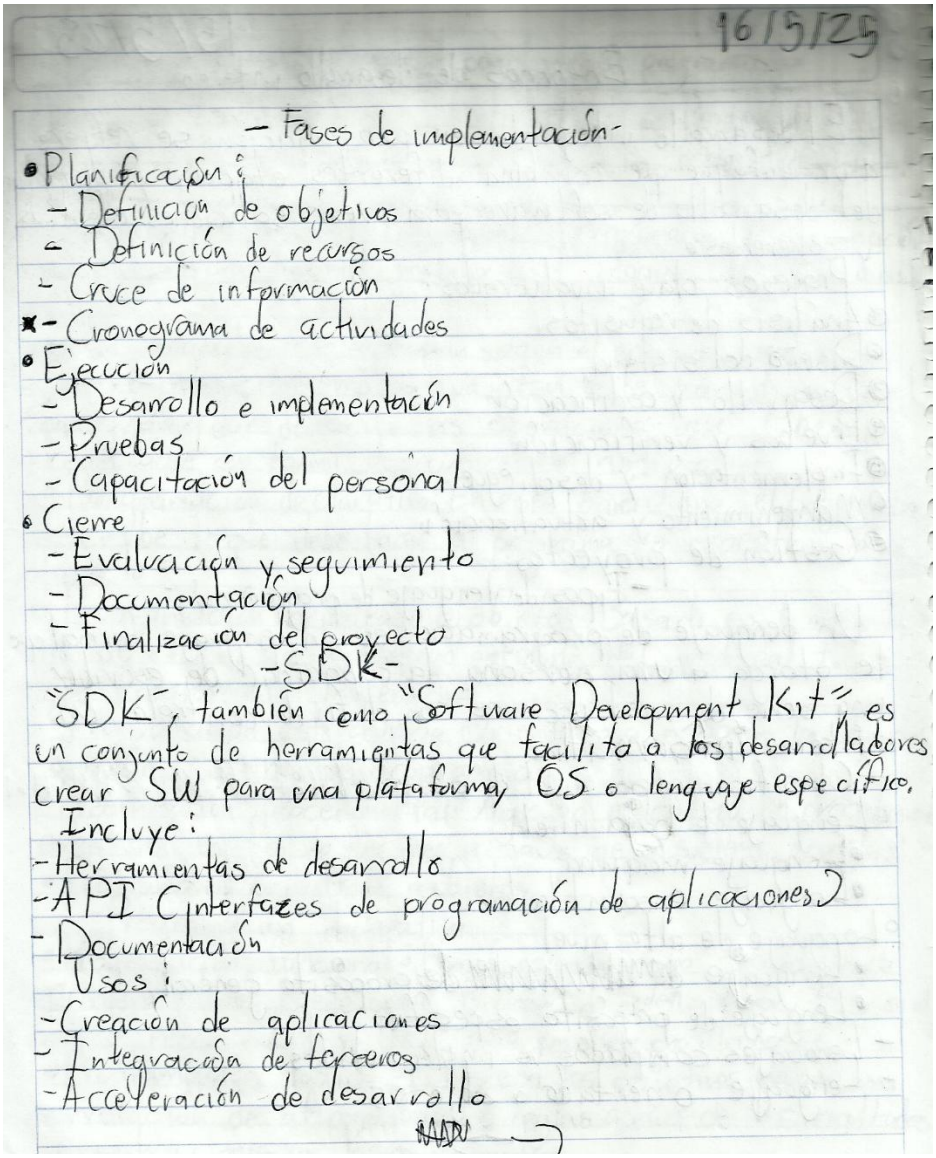


Ilustración 3. Entornos de desarrollo interno Pt.2

Continuando con el tema, en esta ilustración se encuentran las fases de implementación, las cuales son los pasos directos para el desarrollo interno de SW. Además se encuentra el SDK, el cual es básicamente una mejor manera de visualizar el código.

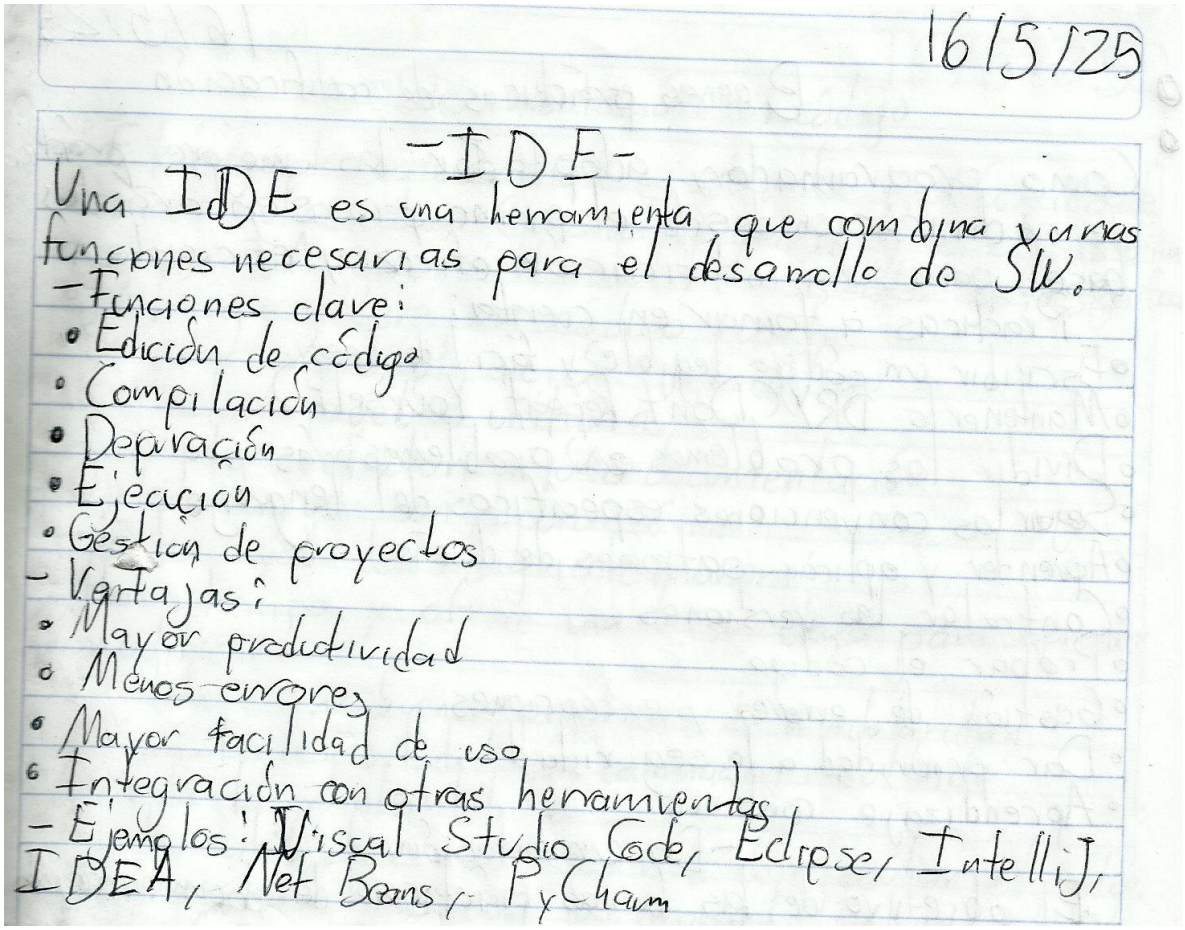
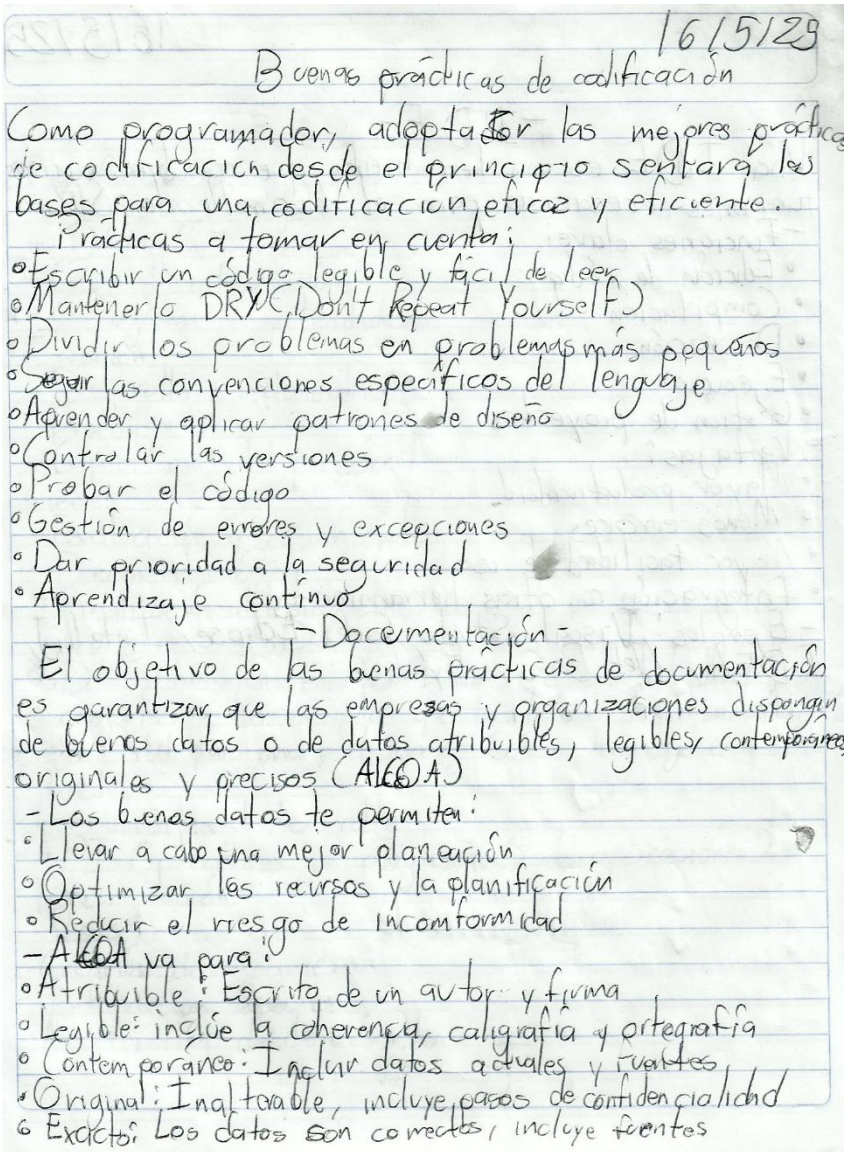


Ilustración 4. Entornos de desarrollo interno Pt.3

Finalizando con el tema, nos encontramos con una IDE, el cual es una herramienta que combina varias funciones para el desarrollo de SW; estos programas son esenciales y de utilidad para el programador debido a lo que ofrece y sus ventajas.

Actividad 3


16/5/23

Buenas prácticas de codificación

Como programador, adoptar las mejores prácticas de codificación desde el principio sentará las bases para una codificación eficaz y eficiente.

¡Practicas a tomar en cuenta!

- Escribir un código legible y fácil de leer
- Mantenerlo DRY (Don't Repeat Yourself)
- Dividir los problemas en problemas más pequeños
- Seguir las convenciones específicos del lenguaje
- Aprender y aplicar patrones de diseño
- Controlar las versiones
- Probar el código
- Gestión de errores y excepciones
- Dar prioridad a la seguridad
- Aprendizaje continuo

-Documentación-

El objetivo de las buenas prácticas de documentación es garantizar que las empresas y organizaciones dispongan de buenos datos o de datos atribuibles, legibles, contemporáneos originales y precisos (ALCOA)

-Los buenos datos te permiten:

- Llevar a cabo una mejor planeación
- Optimizar los recursos y la planificación
- Reducir el riesgo de incomformidad

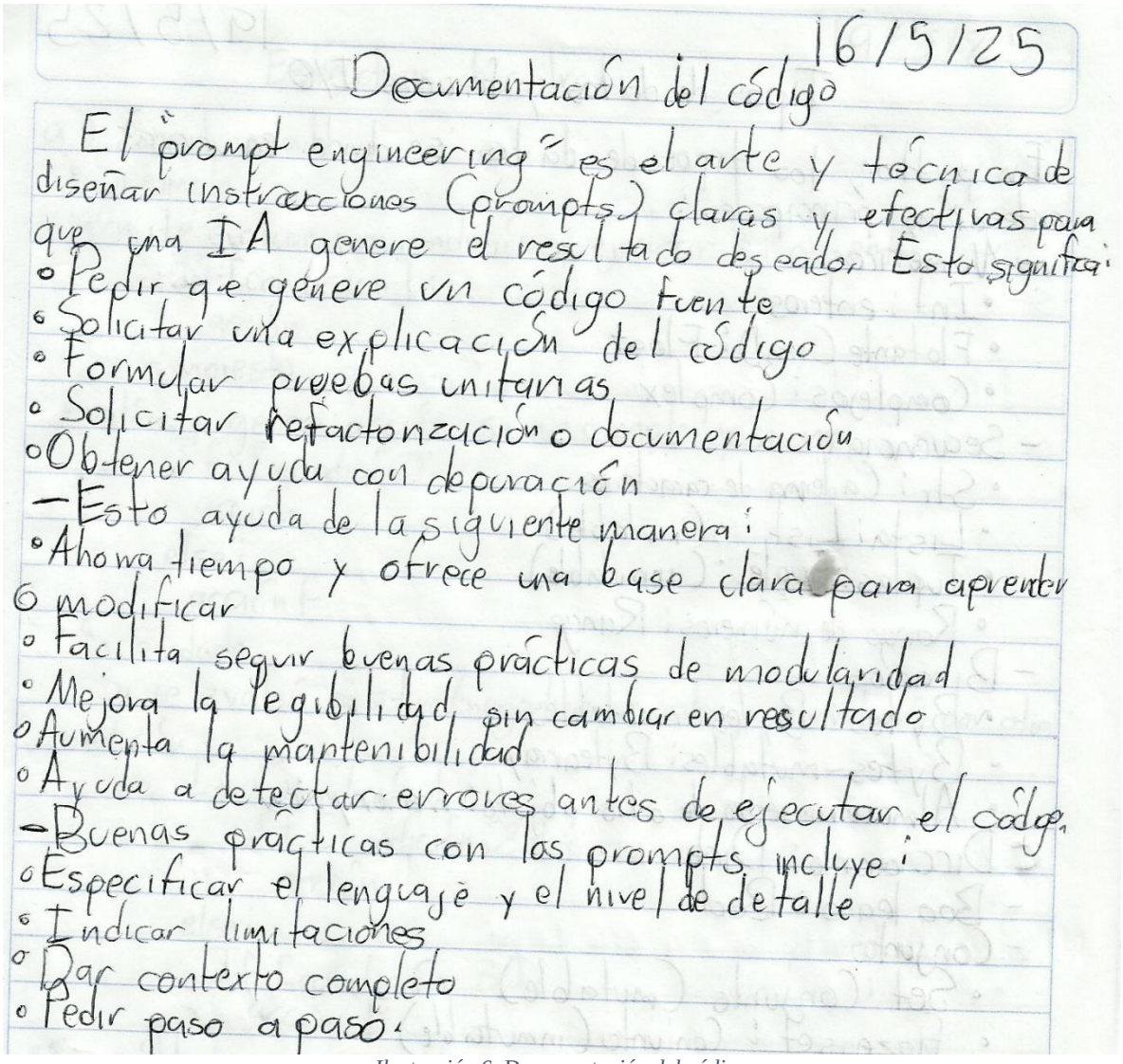
-ALCOA ya para:

- Atribuible: Escrito de un autor y firma
- Legible: incluye la coherencia, caligrafía y ortografía
- Contemporáneo: Incluir datos actuales y vigentes
- Original: Inalterable, incluye pasos de confidencialidad
- Exactos: Los datos son correctos, incluye fuentes

Ilustración 5. Buenas prácticas de codificación

En esta ilustración podemos ver los fundamentos de las buenas prácticas de programación, las cuales permiten tener un código estable y de buena calidad, junto a prácticas correctas para la documentación del mismo.

Actividad 4


16/5/25

Documentación del código

El "prompt engineering" es el arte y técnica de diseñar instrucciones (prompts) claras y efectivas para que una IA genere el resultado deseado. Esto significa:

- Pedir que genere un código fuente
- Solicitar una explicación del código
- Formular pruebas unitarias
- Solicitar refactorización o documentación
- Obtener ayuda con depuración

—Esto ayuda de la siguiente manera:

- Ahorra tiempo y ofrece una base clara para aprender o modificar
- Facilita seguir buenas prácticas de modularidad
- Mejora la legibilidad sin cambiar en resultado
- Aumenta la mantenibilidad
- Ayuda a detectar errores antes de ejecutar el código.

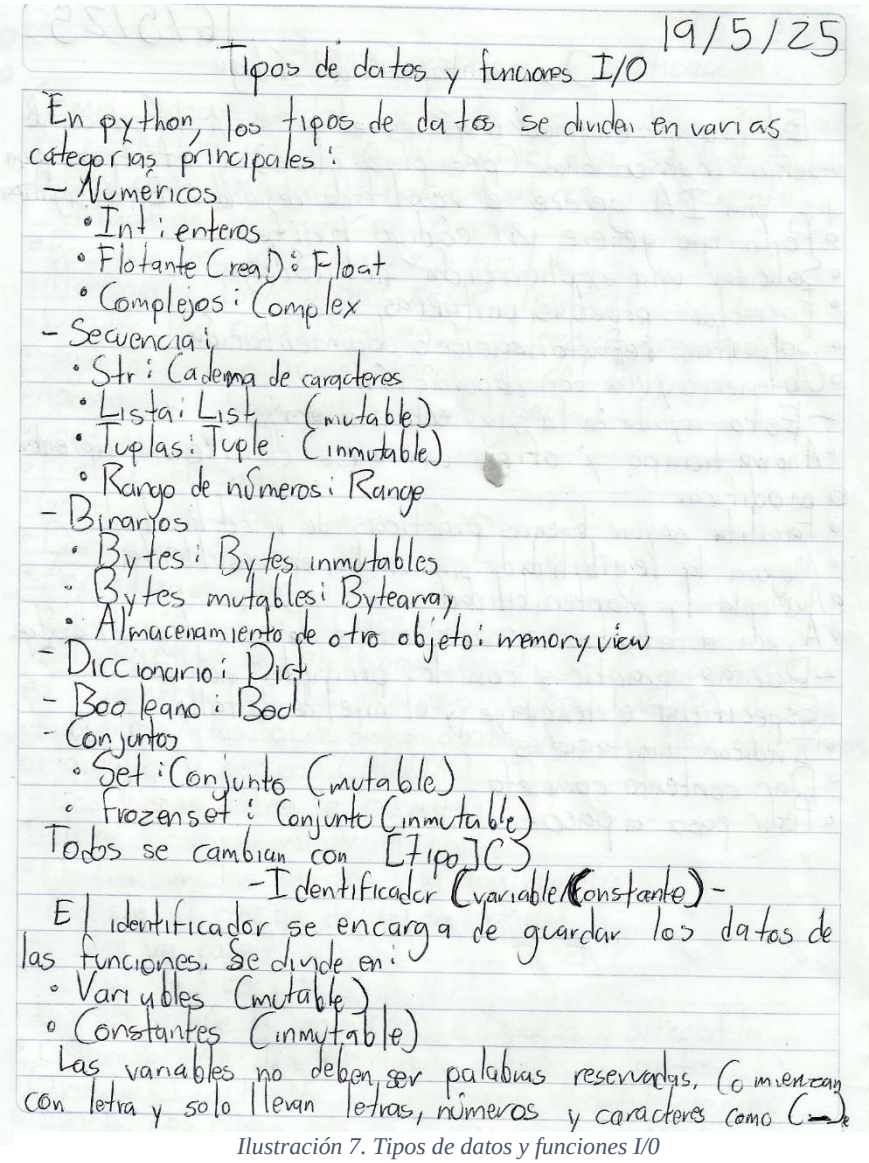
—Buenas prácticas con los prompts incluye:

- Especificar el lenguaje y el nivel de detalle
- Indicar limitaciones
- Dar contexto completo
- Pedir paso a paso.

Ilustración 6. Documentación del código

En esta ilustración podemos observar lo que viene siendo la documentación del código, atribuyendo al cómo funciona el código y a quien se le atribuye. Esta práctica permite la utilización y la modificación ágil del programa.

Actividad 5



19/5/25

Tipos de datos y funciones I/O

En python, los tipos de datos se dividen en varias categorías principales:

- Numéricos
  - Int: enteros
  - Flotante (real): Float
  - Complejos: Complex
- Secuencias
  - Str: Cadena de caracteres
  - Lista: List (mutable)
  - Tuplas: Tuple (inmutable)
  - Rango de números: Range
- Binarios
  - Bytes: Bytes inmutables
  - Bytes mutables: Bytearray
  - Almacenamiento de otro objeto: memory view
- Diccionario: Dict
- Booleano: Bool
- Conjuntos
  - Set: Conjunto (mutable)
  - Frozenset: Conjunto (inmutable)

Todos se cambian con [Tipo]()

-Identificador (variable/constante)-

El identificador se encarga de guardar los datos de las funciones. Se divide en:

- Variables (mutable)
- Constantes (inmutable)

Las variables no deben ser palabras reservadas, comienzan con letra y solo llevan letras, números y caracteres como (_)

Ilustración 7. Tipos de datos y funciones I/0

En esta ilustración podemos visualizar los tipos de datos que se pueden manejar en Python, por ejemplo, los datos numéricos, que albergan números y permiten realizar las operaciones matemáticas correspondientes.

Actividad 6

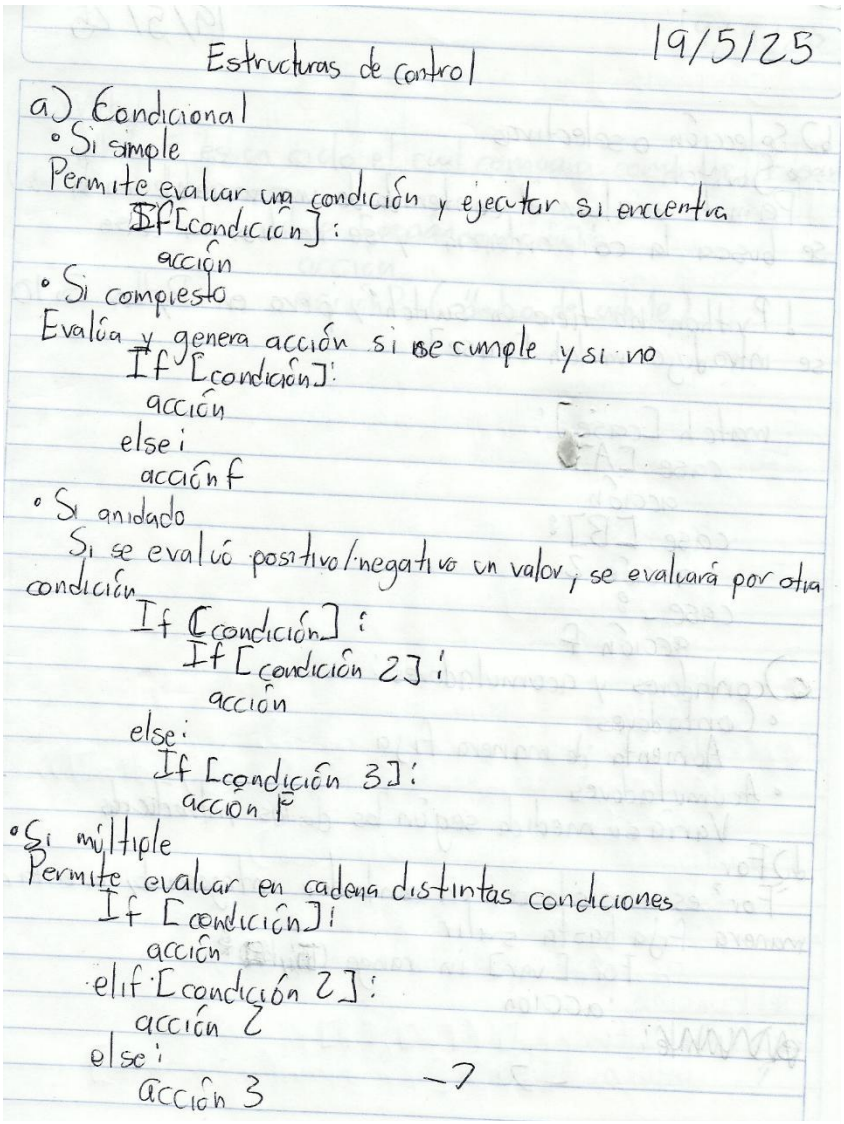


Ilustración 8. Estructuras de control Pt.1

En esta ilustración podemos observar las diversas estructuras de control del código, en este caso, se pueden observar las estructuras condicionales, siendo estas el Si simple, el Si compuesto, el Si anidado y el Si múltiple.

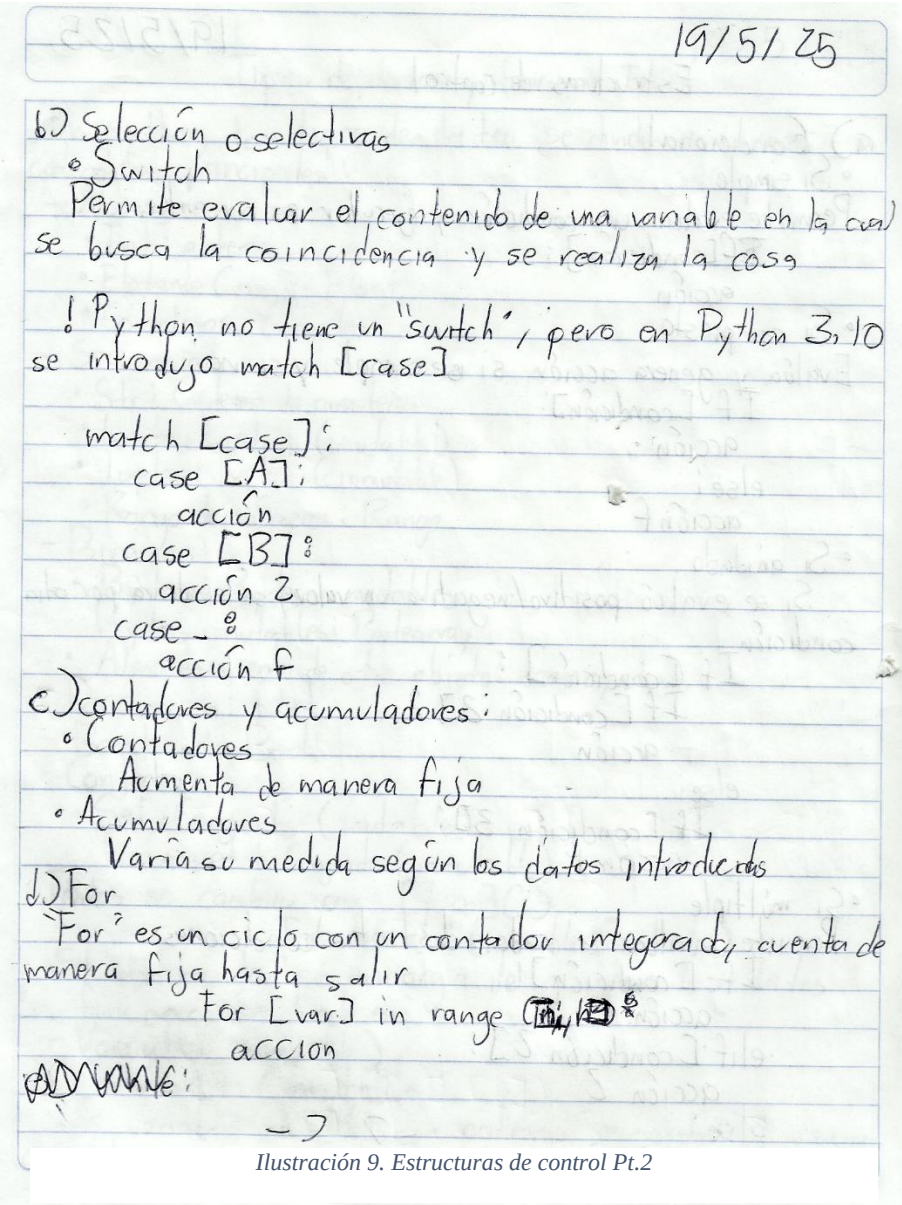


Ilustración 9. Estructuras de control Pt.2

Continuando con el tema, en esta ilustración se puede mostrar la estructura de selección (conocido como switch) el cual permite elegir de opciones al seleccionar una opción. También se describen los contadores y acumuladores.

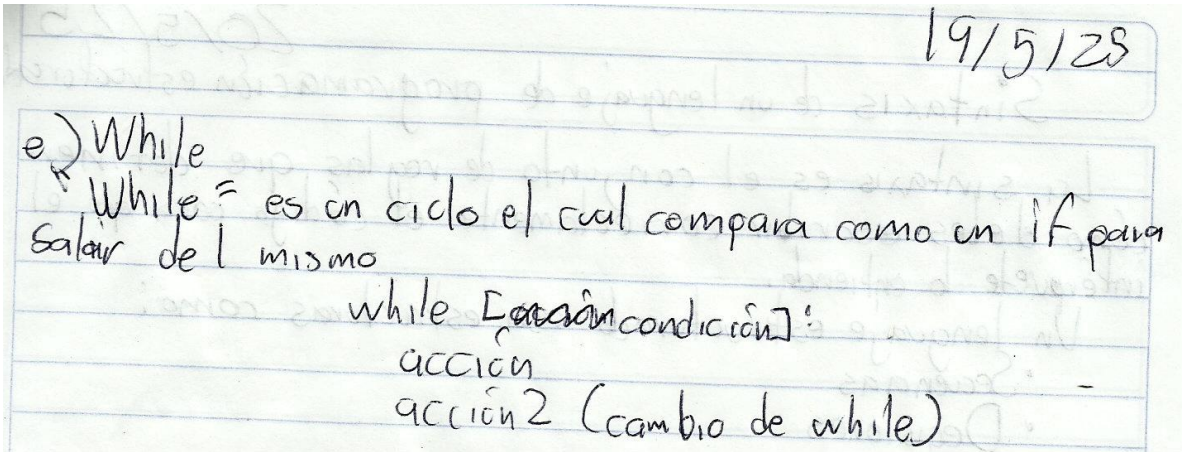
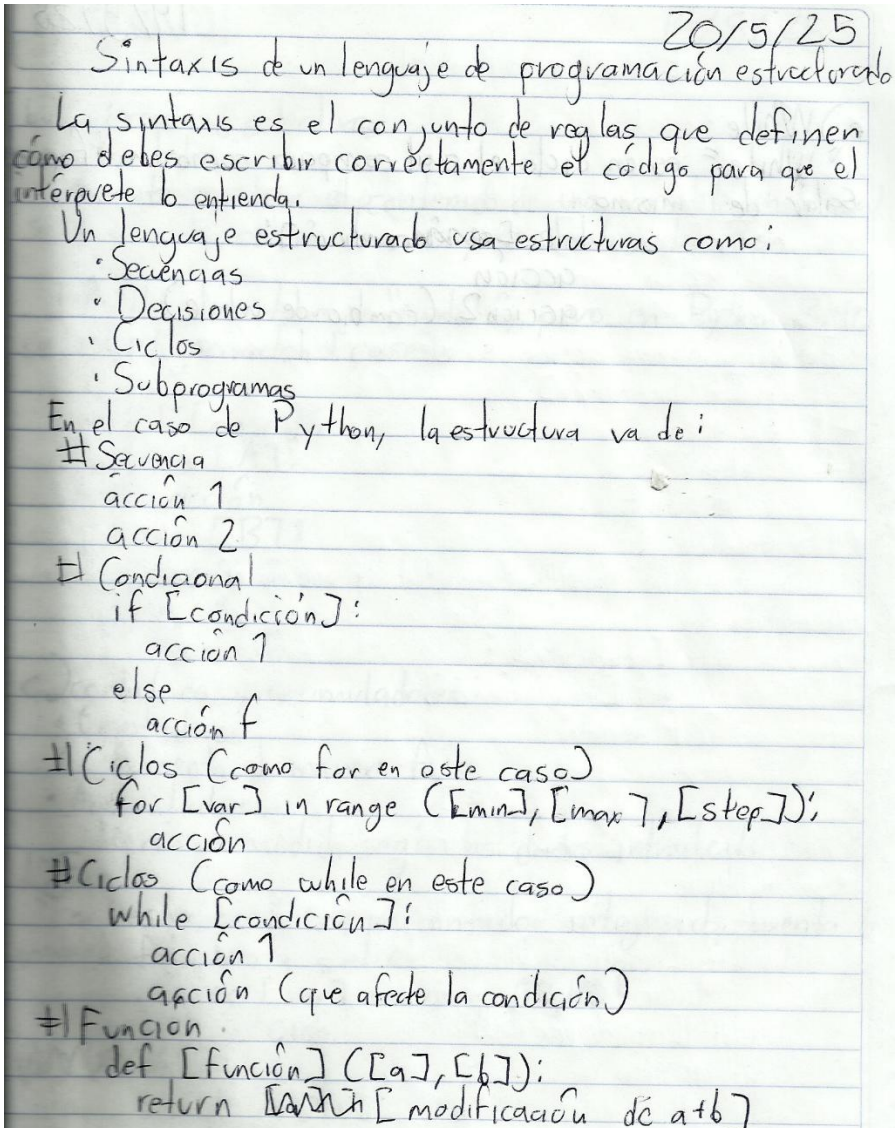


Ilustración 10. Estructuras de control Pt.3

Finalizando con el tema, en esta ilustración se encuentra el ciclo de comando “while” el cual es un tipo de ciclado el cual se basa en una condicional. Para evitar ciclados, se debe de modificar para que eventualmente la condición se cumpla y el ciclado termine.

Actividad 7


20/5/25
Sintaxis de un lenguaje de programación estructurado
La sintaxis es el conjunto de reglas que definen cómo debes escribir correctamente el código para que el intérprete lo entienda.
Un lenguaje estructurado usa estructuras como:
- Secuencias
- Decisiones
- Ciclos
- Subprogramas

En el caso de Python, la estructura va de:

```
#Secuencia
    acción 1
    acción 2
#Condicional
    if [condición]:
        acción 1
    else
        acción f
#Ciclos (como for en este caso)
    for [var] in range ([min],[max],[step]):
        acción
#Ciclos (como while en este caso)
    while [condición]:
        acción 1
        acción (que afecte la condición)
#Función
    def [función] ([a],[b]):
        return [modificación de a+b]
```

Ilustración 11. Sintaxis de un lenguaje de programación estructurado

En esta ilustración se puede encontrar las diversas maneras de escribir un lenguaje de programación en base a la programación estructurada. Junto a todas las herramientas vistas anteriormente, podemos crear estructuras complejas y funcionales.

Retroalimentación

Durante esta estadía de regreso de verdad no pensaba en la realización de los documentos de la manera que lo habíamos realizado hasta ahora, siendo que me siento completamente preparado para poder enfrentar lo que resta de tiempo.

Estos temas me sirvieron ya sea para repasar datos ya conocidos del dato anterior como para expandir datos que no conocía y que pueden llegar a ser de utilidad, gracias a las diversas investigaciones entiendo el uso de la programación estructurada.

Actualmente ando buscando la manera para poder realizar actividades más complejas (incluso más que la actividad final de la unidad pasada) con la eficacia que (creo) siempre me ha caracterizado para lograr salir adelante. Esta vez pienso llegar más lejos de lo que puedo ver y quizás alcanzar algo más.