UNIVERIDAD NACIONAL DE INGENIERIA

FACULTAD DE CIENCIAS

Escuela profesional de Ciencias de la Computación

Logotipo

Descripción generada automáticamente

PROYECTO FINAL

INTEGRANTES:

* Ramos Palomino Ronaldo Flabio 20230375K
* Cumpa Rodríguez Sofía Milagros 20240591H

PROFESOR:

* Yauri Lozano Eduardo

2024

ÍNDICE

1.Introduccion……………………………………………………………………………………3

2.Fundamento Teórico ……………………………………………………………………..4

2.1 Filosofía y Diseño………………………………………………………………….4

2.1.1 El Zen de Python…………………………………………………………4

2.1.2 Diseño………………………………………………………………………..4

2.2 Características de lenguaje…………………………………………………..5

2.3 Estructura de Datos Incorporados en Python………………………6

2.4 Funciones y Manejo de Errores……………………………………………7

2.4.1 Funciones…………………………………………………………………….7

2.4.2 Manejo de errores………………………………………………………8

2.5 Librerías y Herramientas………………………………………………………8

2.6 Biblioteca Estándar……………………………………………………………..8

2.7 Librerías de Terceros…………………………………………………………..9

3.Desarrollo…………………………………………………………………………………….10

3.1 Parte A…………………………………………………………………………….10

3.2 Parte B…………………………………………………………………………….11

4.Conclusiones ……………………………………………………………………………….14

**1.Introducción**

Este proyecto tiene como objetivo dar a los estudiantes una experiencia práctica en el uso de Python. Se busca que apliquen lo aprendido en C++ en un nuevo entorno, viendo las ventajas de Python en simplicidad, eficiencia y versatilidad en diferentes áreas tecnológicas. También permitirá trabajar con aplicaciones reales de Python, despertando el interés en inteligencia artificial, desarrollo web y ciencia de datos. Además, se espera que los participantes mejoren sus habilidades de programación y comprendan mejor el uso de distintos lenguajes en situaciones prácticas. El proyecto aspira a reforzar los conocimientos de los estudiantes y a fomentar su curiosidad y desarrollo profesional en áreas tecnológicas emergentes.

**2.Fundamento Teórico**

**2.1 Filosofía y Diseño**

2.1.1 El Zen de Python

El Zen de Python es una colección de aforismos que capturan la filosofía del diseño de Python. Se puede acceder a ellos escribiendo import this en el intérprete de Python. Aquí algunos de los aforismos más destacados:

* Plano es mejor que anidado
* Complejo es mejor que complciado
* Simple es mejor que complejo
* Bello es mejor que feo
* Explicito es mejor que implícito
* Espaciado es mejor que denso

2.1.2 Diseño

* Legibilidad del código Uno de los objetivos principales de Python es que el código sea fácil de leer y entender. Esto se logra mediante una sintaxis limpia y el uso de sangrías para definir bloques de código en lugar de llaves o palabras clave.
* Simplicidad y consistencia Python sigue la regla de ¨una forma obvia de hacer las cosas”. Esto promueve la consistencia y facilita el aprendizaje y uso del lenguaje.
* Tipado dinámico pero fuerte Python es dinámicamente tipado, lo que significa que no se necesita declarar explícitamente los tipos de las variables, pero mantiene un tipado fuerte, evitando operaciones entre tipos incompatibles.
* Interpretado Python es un lenguaje interpretado, lo que significa que el código se ejecuta línea por línea, permitiendo una rápida iteración y prueba de código.
* Multiplataforma Python es compatible con múltiples plataformas, lo que permite ejecutar el mismo código en diferentes sistemas operativos sin necesidad de modificaciones significativas.
* Biblioteca estándar rica Python viene con una amplia biblioteca estándar que cubre muchas áreas, desde manejo de archivos hasta protocolos de Internet, facilitando el desarrollo sin necesidad de dependencias externas.

En conclusión la filosofía y el diseño de Python están orientados a crear un lenguaje que sea fácil de aprender y usar, promoviendo la escritura de código claro, legible y mantenible. Estos principios han contribuido significativamente a su popularidad y adopción en una amplia variedad de campos, desde desarrollo web hasta ciencia de datos y machine learning.

2.2 Características del lenguaje

* Tipado dinámico y fuerte Python es un lenguaje de tipado dinámico, lo que significa que no es necesario declarar el tipo de una variable explícitamente; el tipo se determina automáticamente en tiempo de ejecución. Además, Python es fuertemente tipado, lo que implica que no permite la combinación implícita de tipos diferentes en una operación, evitando así errores ambiguos.
* Gestión automática de memoria Python maneja la memoria automáticamente a través de un recolector de basura. Utiliza el conteo de referencias y un recolector de basura generacional para gestionar la memoria de forma eficiente.
* Sintaxis clara y concisa Python utiliza una sintaxis clara y legible, con un fuerte énfasis en la legibilidad del código. Los bloques de código se definen por la indentación, no por llaves o palabras clave.
* Estructuras de datos incorporadas Python incluye estructuras de datos de alto nivel que son fáciles de usar y muy poderosas.

1. Listas: Colecciones ordenadas y mutables
2. Diccionarios: Colecciones desordenadas de pares clave-valor
3. Conjuntos: Colecciones desordenadas de elementos únicos
4. Tuplas: Colecciones ordenadas e inmutables
5. Programación Orientada a Objetos
6. Python soporta la programación orientada a objetos (POO), permitiendo la definición de clases y la creación de objetos.

* Manejo de excepciones Python tiene un sistema robusto para el manejo de excepciones, lo que permite gestionar errores de manera elegante y mantener el flujo del programa.
* Biblioteca estándar amplia Python viene con una amplia biblioteca estándar que incluye módulos para tareas comunes como manejo de archivos, manipulación de cadenas, operaciones matemáticas, y mucho más.
* Multiplataforma Python es compatible con múltiples plataformas (Windows, macOS, Linux), lo que permite escribir código que se puede ejecutar en diferentes sistemas operativos sin modificaciones significativas
* Extensibilidad Python se puede extender con módulos escritos en C o C++, lo que permite optimizar partes del código que necesitan mayor rendimiento
* Interpretado Python es un lenguaje interpretado, lo que significa que el código se ejecuta línea por línea, facilitando la depuración y la iteración rápida durante el desarrollo.

2.3 Estructuras de Datos Incorporados en Phyton

Python incluye estructuras de datos de alto nivel que ayudan a manejar y manipular datos de manera eficiente. A continuación se detallan las principales estructuras de datos incorporadas:

* Listas : Las listas son colecciones ordenadas y mutables que permiten almacenar una secuencia de elementos. Las listas pueden contener elementos de cualquier tipo y soportan operaciones como añadir, eliminar y modificar elementos. Características:

1. Ordenadas: Mantienen el orden de inserción de los elementos
2. Mutables: Se pueden modificar después de su creación
3. Indexadas: Se puede acceder a los elementos mediante ´índices

* Diccionarios : Los diccionarios son colecciones desordenadas de pares clave-valor, donde cada clave es única y se utiliza para acceder a su valor correspondiente. Los diccionarios son útiles para almacenar datos asociados, como un mapa o un conjunto de propiedades. Características:

1. Desordenados : No mantienen el orden de inserción (hasta Python 3.7, donde se garantizo el orden de inserción).
2. Mutables: Se pueden modificar después de su creación
3. Claves Unicas: Cada clave en el diccionario debe ser única

* Conjuntos : Los conjuntos son colecciones desordenadas de elementos únicos, lo que significa que no permiten elementos duplicados. Los conjuntos son útiles para operaciones matemáticas como la unión, intersección y diferencia.
* Tuplas : Las tuplas son colecciones ordenadas e inmutables, lo que significa que no se pueden modificar después de su creación. Las tuplas son útiles para almacenar datos heterogéneos y pueden ser utilizadas como claves en los diccionarios debido a su inmutabilidad. Características:

1. Ordenadas: Mantienen el orden de inserción de los elementos.
2. Inmutables: No se pueden modificar después de su creación.
3. Indexadas: Se puede acceder a los elementos mediante índices.

**2.4 Funciones y manejo de errores**

2.4.1 Funciones

* Definición de Funciones: Las funciones se definen utilizando la palabra clave def, seguida del nombre de la función y paréntesis. Dentro de los paréntesis se pueden incluir parámetros que la función recibiría como entrada.
* Llamada a Funciones: Una vez definida, una función puede ser llamada en cualquier parte del código pasando los argumentos necesarios
* Parámetros y Argumentos: Las funciones pueden tener parámetros que permiten recibir información. Los argumentos son los valores que se pasan a la función cuando se llama.
* Valores por Defecto: Los parámetros pueden tener valores por defecto, permitiendo que la función se llame sin pasar todos los argumentos. Esto proporciona flexibilidad en la forma en que se pueden llamar las funciones.
* Funciones Lambda: Las funciones lambda son funciones anónimas definidas con la palabra clave lambda. Son útiles para operaciones pequeñas y rápidas que no requieren una función completa.
* Funciones de Primera Clase: En Python, las funciones son ciudadanos de primera clase, lo que significa que pueden ser asignadas a variables, pasadas como argumentos a otras funciones y retornadas desde otras funciones.

**2.4.2 Manejo de Errores**

* Bloques try-except: Los bloques try y except se utilizan para capturar y manejar excepciones. El código que puede generar una excepción se coloca dentro del bloque try, y el código para manejar la excepción se coloca dentro del bloque except.
* Bloque finally: El bloque finally se ejecuta siempre, independientemente de si se lanzó una excepción o no. Es útil para liberar recursos o realizar tareas de limpieza necesarias.
* Levantar Excepciones: Se puede utilizar la palabra clave raise para lanzar una excepción manualmente, lo que permite gestionar situaciones específicas de error
* Excepciones Personalizadas: Python permite la creación de excepciones personalizadas definiendo nuevas clases que heredan de la clase base Exception. Esto es útil para manejar errores específicos de la aplicación de manera más precisa.

**2.5 Librerías y Herramientas**

Python se distingue no solo por su sintaxis simple y legible, sino también por su extensa colección de librerías y herramientas que facilitan el desarrollo en diversos dominios. A continuación, se describen algunas de las librerías y herramientas más relevantes en el ecosistema de Python.

**2.6 Biblioteca Estándar**

La biblioteca estándar de Python es una colección de módulos y paquetes que vienen incluidos con la instalación de Python. Estos módulos proporcionan soluciones para muchas tareas comunes, eliminando la necesidad de instalar librerías externas.

* os: Proporciona una forma de utilizar funcionalidades dependientes del sistema operativo, como manipulación de archivos y directorios.
* sys: Permite acceder a variables y funciones específicas del interprete de Python
* datetime: Ofrece clases para manipular fechas y horas de manera sencilla.
* json: Permite trabajar con datos en formato JSON (JavaScript Object Notation)
* math: Proporciona acceso a funciones matemáticas como trigonometría, logaritmos, y más
* urllib: Facilita el trabajo con URL, incluyendo la recuperación de datos desde la web.

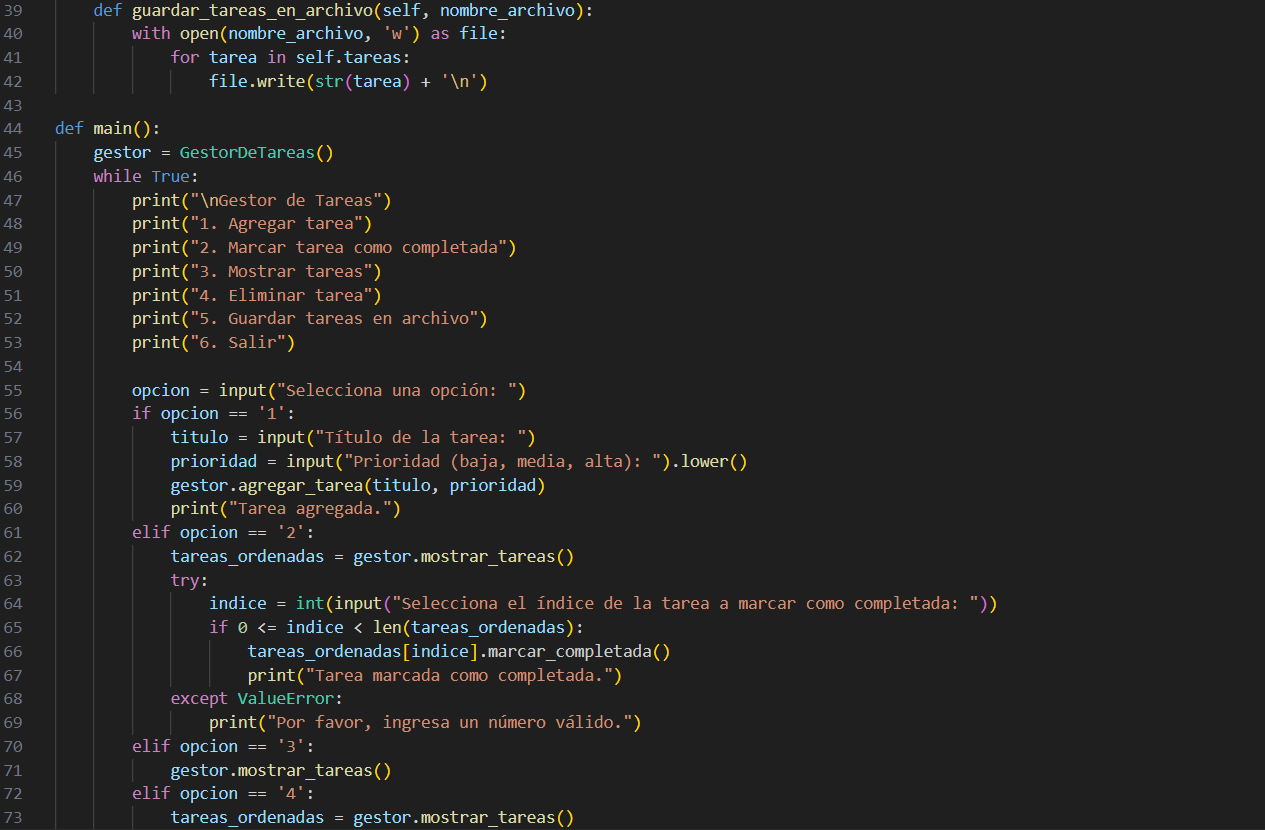
**2.7 Librería de Terceros**

Además de la biblioteca estándar, Python cuenta con una amplia gama de librerías de terceros que extienden su funcionalidad. Algunas de las más populares incluyen:

* NumPy: Proporciona soporte para grandes matrices y matrices multidimensionales, junto con una colección de funciones matemáticas de alto nivel para operar con estos arreglos.
* pandas: Ofrece estructuras de datos y herramientas de análisis de datos de alto rendimiento, especialmente útiles para la manipulación de datos tabulares.
* matplotlib: Permite crear visualizaciones estáticas, animadas e interactivas en Python. requests: Simplifica el envío de solicitudes HTTP, siendo mucho más intuitivo que los módulos de la biblioteca estándar.
* scikit-learn: Proporciona herramientas simples y eficientes para análisis de datos y minería de datos, incluyendo algoritmos de aprendizaje automático.
* Flask: Un micro framework para desarrollo web, que permite crear aplicaciones web de manera sencilla y rápida.
* Django: Un framework de alto nivel para el desarrollo web, que fomenta el desarrollo rápido y el diseño limpio y pragmático.

**3. Desarrollo:**

Gestor de Tareas con Priorización: Una aplicación para organizar tareas con prioridad (baja, media, alta). Permite agregar tareas, marcarlas como completadas, y mostrar una lista de tareas ordenadas por prioridad, adicionalmente implementar 3 funcionalidades más a criterio del estudiante. Resaltan Módulos para manejo de datos, lógica de ordenamiento y la interfaz de presentación al cliente.

Texto

Descripción generada automáticamente **3.1 Parte A:**

Texto

Descripción generada automáticamente

**3.2 Parte B:**

En esta parte debíamos implementar las librerías *pandas,* *matplotlib* y *numpy*, librerías descritas previamente.

Importar librerías:

Interfaz de usuario gráfica

Descripción generada automáticamente con confianza baja

Implementación de la librería “*pandas*”:

Captura de pantalla de un celular

Descripción generada automáticamente

Implementación de la librería “*matplotlib*”:

Texto

Descripción generada automáticamente

Subir el archivo a *github*:

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Texto

Descripción generada automáticamente

Primero, descargué e instalé Git desde su página oficial. Luego, creé una carpeta en mi computadora para almacenar los archivos que quería subir a GitHub.

Abrí la terminal que viene incluida con el programa de Git (Git Bash) y navegué hasta la carpeta usando el comando adecuado para acceder a su directorio. Dentro de esta carpeta, inicialicé Git, convirtiéndola en un repositorio local.

Antes de enlazar mi repositorio local con GitHub, configuré mi nombre de usuario y mi correo electrónico en Git para que pudiera identificar mis cambios. Esto es esencial para trabajar correctamente con el control de versiones.

Después, creé un repositorio vacío en GitHub y lo enlacé con mi repositorio local utilizando el URL del repositorio remoto. Este paso permitió que mi trabajo local estuviera sincronizado con el repositorio en GitHub.

Agregué los archivos que quería subir a la carpeta local y verifiqué su estado en la terminal. Luego, preparé estos archivos para su envío, realicé un commit con un mensaje explicativo y, finalmente, los subí al repositorio remoto en GitHub.

Con estos pasos, logré subir mi programa a GitHub. Aseguré configurar correctamente mi identidad (nombre de usuario y correo electrónico) y vincular el repositorio local con el remoto, para tener el control de los cambios y mantener todo sincronizado.

**4.CONCLUSIONES:**

Python se ha convertido en uno de los lenguajes de programación más versátiles y populares hoy en día. Su diseño, centrado en la simplicidad y la legibilidad, lo hace accesible para principiantes y programadores experimentados. Las características clave de Python, como el tipado dinámico, la gestión automática de memoria y las estructuras de datos incorporadas, mejoran su facilidad de uso y eficiencia.

La amplia biblioteca estándar de Python y su ecosistema de librerías de terceros permiten a los desarrolladores trabajar en diversas aplicaciones, desde desarrollo web y automatización hasta análisis de datos y aprendizaje automático. Además, la comunidad activa y el enfoque en la colaboración han impulsado el desarrollo continuo del lenguaje, haciendo que Python sea una herramienta esencial para desarrolladores modernos. Su combinación de simplicidad y versatilidad lo convierte en una excelente opción para muchos proyectos.

La implementación de las librerías **Pandas** y **Matplotlib** resultó ser una herramienta poderosa para enriquecer el programa. Mientras que **Pandas** permitió gestionar los datos de las tareas de forma estructurada y profesional, facilitando su almacenamiento y exportación en formatos como CSV, **Matplotlib** aportó un enfoque visual al programa, permitiendo representar gráficamente la distribución de las tareas según su prioridad. Estas herramientas no solo optimizan el rendimiento del código, sino que también hacen que los datos sean más accesibles e interpretables para los usuarios.

Por otro lado, el uso de **GitHub** como repositorio remoto ha demostrado ser una plataforma eficiente para gestionar el control de versiones y la colaboración en proyectos. Subir un programa a GitHub trae ventajas claras, como la seguridad en el almacenamiento, la posibilidad de revertir cambios y la facilidad de compartir el trabajo con otros. Sin embargo, también se presentan desafíos, como la configuración inicial de las credenciales, la comprensión de los comandos de Git y la vinculación adecuada entre el repositorio local y el remoto, que pueden ser barreras para los principiantes. Superar estos retos no solo fortalece las habilidades técnicas, sino que también fomenta mejores prácticas en el desarrollo de software.