

**Fundamentos de Python**

Unidad 2. Funciones y estructuras de datos

PROJECTOS

1. Calculadora de Edad
2. Sistema de Evaluación

ESTUDIANTE

Armando Gabriel Jacinto López

INSTITUCION

INFOTEC

FECHA

16/09/2025

**1.- Introducción**

En el desarrollo de software moderno, **Python** se ha consolidado como uno de los lenguajes de programación más versátiles y de rápido crecimiento. Su sintaxis clara y legible lo hace ideal tanto para principiantes como para desarrolladores experimentados. Para crear programas eficientes, modulares y fáciles de mantener, es fundamental dominar dos conceptos clave: las **funciones** y las **estructuras de datos**. Las funciones nos permiten organizar el código en bloques reutilizables, evitando la repetición y mejorando la legibilidad. Por su parte, las estructuras de datos son las herramientas que nos permiten almacenar y organizar información de manera lógica y eficiente.

El presente informe analiza la aplicación de estos conceptos a través de dos programas: el primero, un sistema para calcular la edad de un cliente a partir de su fecha de nacimiento, y el segundo, un sistema de gestión académica que almacena información de estudiantes y sus calificaciones. Ambos proyectos demuestran cómo el uso adecuado de funciones y estructuras de datos permite resolver problemas complejos de manera estructurada y optimizada.

**2. Desarrollo**

**2.1. Calculadora de Edad**

Este programa tiene como objetivo principal calcular la edad de una persona a partir de su fecha de nacimiento. Para lograrlo, el código hace un uso efectivo de **funciones** y una **estructura de datos** básica.

* **Funciones:** El programa está organizado alrededor de una única función principal, calculate\_age(). Esta función encapsula toda la lógica del problema: desde la captura de la entrada del usuario hasta la validación de la fecha y el cálculo de la edad. Al agrupar estas tareas dentro de una función, se logra un código más ordenado y fácil de entender. La función es llamada al final del script, lo que demuestra su papel central en la ejecución del programa.

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración 1 Función principal y validación de las entradas

* **Estructuras de Datos:** El programa utiliza una **lista (list)** de manera implícita, aunque no se declare directamente con corchetes. Cuando el usuario ingresa la fecha de nacimiento en formato de cadena de texto ("dd-mm-aaaa"), el método split("-") la divide en una lista de cadenas, por ejemplo, ['25', '10', '1995']. Esto permite acceder a cada componente de la fecha (día, mes, año) individualmente y convertirlos a números enteros para su posterior validación y cálculo. Esta es una forma simple pero poderosa de usar una estructura de datos para manipular información.

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración 2 Obtención de la edad del usuario

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración 3 Ejecución del programa

**2.2. Sistema Evaluación**

Este programa es más complejo, ya que busca gestionar la información de múltiples estudiantes y sus calificaciones. Para manejar esta complejidad, se apoya fuertemente en **funciones** y **diccionarios**.

* Texto

  El contenido generado por IA puede ser incorrecto.**Funciones:** El programa utiliza dos funciones. La función principal, grades\_system(), orquesta todo el proceso: pide el número de estudiantes y materias, y luego entra en bucles para solicitar los datos de cada alumno y sus calificaciones. Además, se crea una función auxiliar llamada verify\_inputs(), cuya única tarea es validar que las entradas no estén vacías. El uso de esta función es un excelente ejemplo de **reutilización de código**, ya que se llama múltiples veces para validar diferentes datos de entrada, como el nombre del estudiante, la matrícula, el nombre de la materia y la calificación. Esto evita la duplicación de código y hace que la lógica de la función principal sea más limpia.

Ilustración 4 Función principal, entrada, y validación de datos

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración 5 Función para verificar los datos de entrada

* Pantalla de computadora con letras

  El contenido generado por IA puede ser incorrecto.**Estructuras de Datos:** La columna vertebral de este programa es el uso de **diccionarios anidados**. El programa utiliza un diccionario principal llamado students, donde la **clave** es la matrícula del estudiante y el **valor** es otro diccionario. Este diccionario anidado contiene el nombre del estudiante y, a su vez, otro diccionario para almacenar sus materias. Cada materia es una clave, y su valor es un diccionario con la calificación (grade) y el estado (state). Esta estructura jerárquica es ideal para organizar datos complejos y relacionados de forma lógica. Permite un acceso rápido a la información de un estudiante usando su matrícula como clave, lo que es mucho más eficiente que buscar en una lista de estudiantes.

Ilustración 6 Ciclo for para añadir información sobre los estudiantes

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración 7 Ciclo for para añadir información sobre las materias

Texto

El contenido generado por IA puede ser incorrecto.

Ilustración 8 Ejecución del programa, y diccionario final

**3. Conclusiones**

Los dos programas demuestran de manera práctica cómo las **funciones** y las **estructuras de datos** son pilares fundamentales de la programación en Python. Las funciones, al modularizar la lógica, permiten que el código sea más legible, reutilizable y fácil de mantener, como se vio con la función de validación en el segundo programa.

Por otro lado, la elección de la estructura de datos adecuada es crucial para la eficiencia. El primer programa utiliza una **lista** de manera sencilla para procesar una cadena, mientras que el segundo programa aprovecha el poder de los **diccionarios anidados** para gestionar una base de datos de estudiantes de forma eficiente. En conjunto, ambos proyectos son una clara demostración de que un sólido entendimiento de estos conceptos es esencial para desarrollar programas robustos y bien diseñados.