Proyecto Integrador: Temas 1 al 6

Sistema de Gestión de Recursos Humanos (POO)

Curso: Python de 0 a Interfaces Gráficas

1. Objetivo General

Desarrollar una aplicación de consola completa que gestione una base de datos de empleados, aplicando la Programación Orientada a Objetos para modelar los datos y la Programación Estructurada para el control del flujo.

1.1. Competencias a Evaluar

- **Temas 1-3:** Uso de `input`, validaciones con `if/else` y menús con `while`.
- **Tema 4:** Uso de **Listas** para almacenar múltiples objetos.
- **Tema 5:** Organización del código en **Funciones** independientes.
- **Tema 6:** Creación de una **Clase** con atributos y métodos que modifiquen su estado.

---

2. Descripción del Proyecto

Eres el desarrollador principal de una empresa y necesitas crear un sistema para administrar a los empleados.

El sistema debe permitir:

1. Registrar nuevos empleados.
2. Ver la nómina completa (lista de empleados).
3. Buscar un empleado por su número de ID.
4. Aumentar el salario de un empleado específico (usando un método del objeto).

3. Instrucciones de Desarrollo

3.1. Paso 1: La Clase `Empleado` (El Molde)

Crea una clase que represente a una persona en la empresa.

- **Constructor (`__init__`):** Debe recibir y guardar:
  - `id` (entero, identificador único).

- `nombre` (texto).
- `puesto` (texto).
- `salario` (flotante).

- **Método `__str__`:** Debe devolver una cadena bonita, ej: `"[ID: 101] Ana Perez - Gerente - $50000"`.
- **Método `aumentar_salario(self, porcentaje)`:** Recibe un porcentaje (ej. 10 para 10 %) y actualiza el atributo `salario`.

3.2. Paso 2: Funciones de Gestión (La Lógica)

Crea funciones para manejar la lista de empleados. La lista debe crearse en el bloque principal y pasarse como parámetro.

registrar_empleado(lista_emp): Pide los datos al usuario, crea un **Objeto** `Empleado` y lo agrega a la lista con `.append()`.

mostrar_nomina(lista_emp): Recorre la lista con un bucle `for` e imprime cada objeto.

buscar_y_aumentar(lista_emp):

1. Pide un ID al usuario.
2. Recorre la lista buscando ese ID.
3. Si lo encuentra, muestra al empleado y pide el porcentaje de aumento.
4. Llama al método `.aumentar_salario()` de ESE objeto encontrado.

3.3. Paso 3: El Menú Principal

Crea un bucle `while True` que muestre las opciones y llame a las funciones correspondientes.

4. Código Base (Esqueleto)

```python
# --- 1. CLASE EMPLEADO ---
class Empleado:
    def __init__(self, id_emp, nombre, puesto, salario):
        self.id = id_emp
        self.nombre = nombre
        self.puesto = puesto
        self.salario = salario

    def aumentar_salario(self, porcentaje):
        # Formula: salario + (salario * (porcentaje / 100))
        aumento = self.salario * (porcentaje / 100)
        self.salario += aumento
        print(f"Salario actualizado a ${self.salario}")

    def __str__(self):
        return f"[ID: {self.id}] {self.nombre} | {self.puesto}"

# --- 2. FUNCIONES ---
def registrar_empleado(lista):
    print("--- Nuevo Empleado ---")
    # TODO: Pedir datos, crear objeto Empleado() y agregarlo a 'lista'
    pass

def mostrar_nomina(lista):
    print("--- Nomina Actual ---")
    # TODO: Recorrer lista e imprimir
    pass

# --- 3. MAIN ---
def main():
    base_datos = [] # Lista vacia para guardar objetos

    while True:
        print("\n--- RRHH SYSTEM ---")
        print("1. Registrar Empleado")
        print("2. Ver Nomina")
        print("3. Aumentar Salario")
        print("4. Salir")

        opc = input("Opcion: ")

        if opc == "1":
            registrar_empleado(base_datos)
        elif opc == "2":
            mostrar_nomina(base_datos)
        # ... completar opciones ...

if __name__ == "__main__":
    main()
```