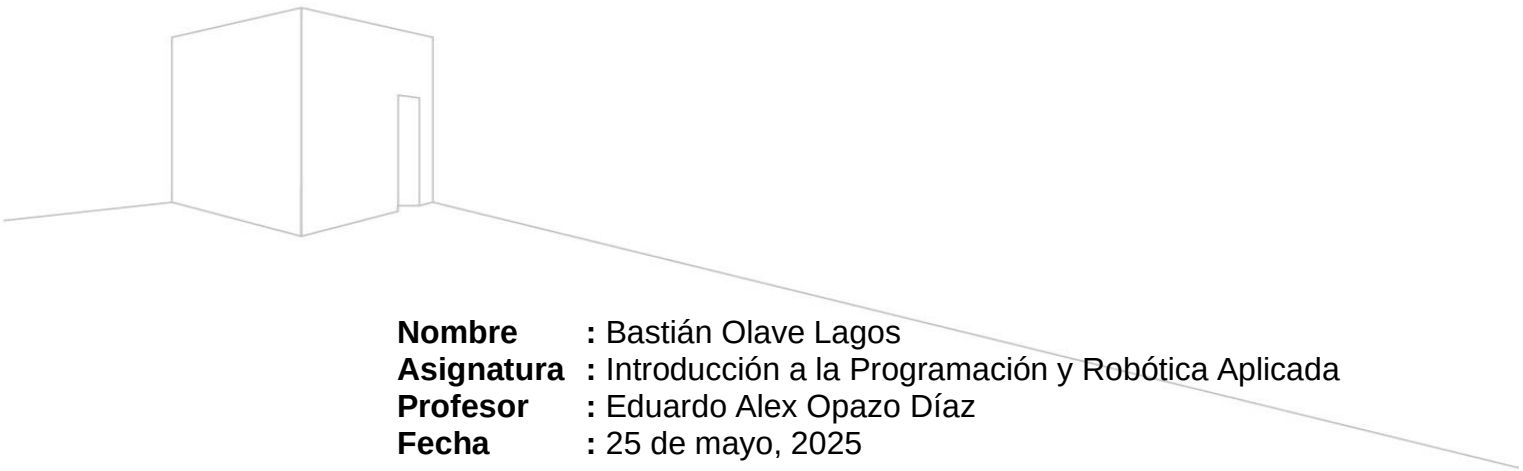
Sistema Automatizado de Gestión de Información para un Problema Real



Nombre : Bastián Olave Lagos
Asignatura : Introducción a la Programación y Robótica Aplicada
Profesor : Eduardo Alex Opazo Díaz
Fecha : 25 de mayo, 2025

Índice

I. Introducción

El avance de la tecnología ha facilitado la automatización de tareas repetitivas y la gestión eficiente de la información, permitiendo el desarrollo de soluciones simples pero efectivas para diversas problemáticas del entorno cotidiano. En este proyecto se presenta una aplicación desarrollada en lenguaje Python que aborda una necesidad concreta del ámbito académico: el control de asistencia en clases.
Muchas instituciones y docentes enfrentan el desafío de llevar un registro ordenado y confiable de la asistencia estudiantil. Frente a ello, se propone un sistema que permite crear cursos, registrar estudiantes, marcar asistencia diaria e identificar automáticamente a aquellos alumnos con baja participación, mediante el cálculo de porcentajes de asistencia y alertas visuales.

II. Desarrollo

Objetivo del Proyecto

Diseñar una aplicación en Python que funcione como un sistema automatizado para el control de asistencia en clases, permitiendo a docentes o responsables académicos registrar cursos, estudiantes y asistencia diaria, así como generar reportes con porcentajes de participación.

Características del Sistema

- Interfaz por consola con menú interactivo.
- Registro de cursos y estudiantes asociados.
- Marcado de asistencia con fecha personalizada.
- Validación de entradas mediante bloques try-except.
- Cálculo del porcentaje de asistencia por alumno.
- Generación de reportes con alertas si el porcentaje es bajo (menos del 70%).

Estructura del Programa

- Clase **Funciones.py** que encapsula la lógica principal del sistema.
- Diccionario global **Cursos** que organiza los datos por curso.
- Métodos específicos para:
    - Agregar Curso
    - Agregar estudiante a un curso
    - Registrar asistencia diaria
    - Ver porcentaje de asistencia por estudiante
- Uso de estructuras de control **(if, while, for)** y estructuras de datos **(listas, diccionarios)** para manejar la información de forma dinámica,

Validaciones y flujos

- Cada ingreso de datos se valida según su tipo (número, texto, opciones validas)
- El sistema permite repetir acciones hasta que el usuario decida salir
- El menú guía al usuario paso a paso en un flujo lógico e intuitivo
- Se respeta la lógica funcional definida en el planteamiento del problema, facilitando una experiencia simple y directa para el usuario final.

III. Código Fuente:

Para **main.py:**

```
# Codigo main para el control de asistencia de clases
# Bastian Olave Lagos
from os import system
from funciones import *

# func es una instancia de la clase funciones
# que contiene todos los metodos para el control de asistencia
func=funciones()

# Se crea un bucle infinito con el menu
while True:
    try:
        system('cls')
        print("\n\t\t\t\t\t\tControl de Asistencia de Clases")
        opcion=int(input("Ingresa una opción:\n\n"
                "1) Agregar Curso\n"
                "2) Agregar Alumno\n"
                "3) Marcar Asistencia\n"
                "4) Ver porcentaje de asistencia\n"
                "5) Salir\n"
                "-> "))
        if opcion==1:
            func.agregar_curso()
        elif opcion==2:
            func.agregar_alumno()
        elif opcion==3:
             func.agregar_asistencia()
        elif opcion==4:
             func.ver_porcentaje_asistencia()
        elif opcion==5:
            input("Saliendo del sistema...")
            break
        else:
             input("Por favor, ingresa una de las opciones")
    except ValueError:
         input("Por favor, ingrese un numero del 1 al 5.")
```

Para **funciones.py:**

```
# Codigo con las funciones para el control de asistencia de clases y usar con main.py
# Bastian Olave Lagos

#Diccionario para almacenar los cursos
cursos={}

# Se crea una clase para almacenar los metodos
class funciones:

    # A continuacion se definen los metodos para agregar cursos, alumnos y asistencia
    # y para ver el porcentaje de asistencia

    def agregar_curso(self):
        nombre = input("Ingresa el nombre del curso: ").strip().title()
        if nombre in cursos:
            print("Error!\nEl Curso ya existe!")
        else:
            cursos[nombre]=[]
            print("Curso agregado con exito!")
        input("Presione enter para continuar...")

    def agregar_alumno(self):
        if not cursos:
            print("No existen cursos creados!")
            return
        print("Curso disponible: ")
        for i, curso in enumerate(cursos, 1):
            print(f"{i}){curso}")
        try:
            opcion=int(input("Seleccion el numero del curso donde agregar a alumno: "))
            lista_cursos=list(cursos.keys())
            if opcion<1 or opcion>len(lista_cursos):
                input("Opcion invalida")
                return
            curso_elegido=lista_cursos[opcion-1]
        except ValueError:
            input("Error, debes ingresar un numero!")
            return
        nombre_alumno=input("Ingresa el nombre del alumno: ").strip().title()
        alumnos=cursos[curso_elegido]
        for alumno in alumnos:
            if alumno["nombre"]==nombre_alumno:
                input("El alumno ya existe en este curso")
                return
        alumnos.append({"nombre": nombre_alumno, "asistencias": []})
        print("Alumno agregado al curso!")
        input("Presione enter para continuar...")
```

```
def agregar_asistencia(self):
    if not cursos:
        input("No existen cursos creados!")
        return
    print("Cursos disponibles: ")
    for i, curso in enumerate(cursos, 1):
        print(f"{i}){curso}")
    try:
        opcion=int(input("Seleccion el numero del curso donde agregar asistencia: "))
        lista_cursos=list(cursos.keys())
        if opcion<1 or opcion>len(lista_cursos):
            input("Opcion invalida")
            return
        curso_elegido=lista_cursos[opcion-1]
    except ValueError:
        input("Error, debes ingresar un numero!")
        return
    alumnos=cursos[curso_elegido]
    if not alumnos:
        input("No hay alumnos en este curso")
        return
    fecha=input("Ingresa la fecha de la asistencia (dd/mm/aaaa): ").strip()
    print("Marcando asistencia oara el curso ", curso_elegido)
    for alumno in alumnos:
        while True:
            estado=input(f"El alumno {alumno['nombre']} asistió? (s/n): ").strip().lower()
            if estado in ["s", "n"]:
                if estado=="s":
                    alumno["asistencias"].append({"fecha": fecha, "presente": True})
                else:
                    alumno["asistencias"].append({"fecha": fecha, "presente": False})
                break
            else:
                print("Error, ingrese s o n")
    print("Asistencia marcada con exito!")
    input("Presione enter para continuar...")

def ver_porcentaje_asistencia(self):
    if not cursos:
        input("No hay cursos creados.")
        return
    print("Cursos disponibles:")
    for i, curso in enumerate(cursos, 1):
        print(f"{i}) {curso}")

    try:
        opcion = int(input("Selecciona el número del curso que quieres revisar: "))
        lista_cursos = list(cursos.keys())
        if opcion < 1 or opcion > len(lista_cursos):
```

```
                print("Opción inválida.")
                return
            curso_elegido = lista_cursos[opcion - 1]
        except ValueError:
            print("Entrada inválida.")
            return
        alumnos = cursos[curso_elegido]
        if not alumnos:
            print(f"No hay alumnos en el curso '{curso_elegido}'.")
            return
        print(f"\nPorcentaje de asistencia - Curso: {curso_elegido}\n")
        print("{:<20} {:<10} {:<10} {:<10}".format("Alumno", "Asistió", "Total", "%"))
        print("-" * 55)
        for alumno in alumnos:
            asistencias = alumno["asistencias"]
            total = len(asistencias)
            presentes = sum(1 for a in asistencias if a["presente"])
            porcentaje = (presentes / total * 100) if total > 0 else 0
            estado = "!" if porcentaje < 70 else ""
            print("{:<20} {:<10} {:<10} {:<5.1f}% {}".format(alumno['nombre'], presentes, total,
porcentaje, estado))
        print("\n! indica asistencia menor al 70%.")
        input("Presione enter para continuar...")
```

IV. Conclusión

En conclusión, el desarrollo del sistema de control de asistencia en código Python permite abordar efectivamente una necesidad muy común dentro de un entorno del contexto académico que muchos establecimientos sufren, automatizando el registro y participación de los estudiantes. Esta solución entregada cumple con los requerimientos establecidos: recibe y valida entradas dadas por los usuarios, utiliza estructuras para mayor control, posee funciones y manejo de los datos mediante listas y diccionarios. Además, se logro crear un Menú iterativo y claro, facilitando el manejo con instrucciones claras para los usuarios.