Proyecto 1 Python

Sistema de gestión de asistencia académica {

<Por="Ada_Leal_Parra"/>

}

Contenidos

Objetivo {

Se solicita el desarrollo de un programa para la consola o terminal y escrito en el lenguaje de programación Python, que funcione como una aplicación de consola, para gestionar la asistencia académica. El sistema debe cumplir con los siguientes requisitos:

Descripción del problema {

- El registro manual de la asistencia de los estudiantes ha generado ineficiencias en la obtención de información fiable, afectando tanto los procesos académicos como administrativos. La solución propuesta debe automatizar este proceso, permitiendo el acceso a la información relevante y generando informes de manera clara y precisa.

}

Contexto general {

ACME Education, una institución educativa dedicada a la formación técnica y profesional ha decidido modernizar su sistema de gestión de asistencia. Este sistema permitirá un control automatizado de la asistencia de sus estudiantes,

facilitando la generación de informes que apoyen la mejora continua tanto de los procesos académicos como administrativos.

Para este fin, ACME Education ha encargado el desarrollo de una solución informática llamada Sistema de Gestión de Asistencia Académica (SISGESA).

Este sistema se implementará como un programa de consola, permitiendo a los usuarios interactuar con las diversas funciones a través de un menú.

}

Investigacion

OS

- **Usos**
- **Automatización de tareas: Puedes escribir scripts para automatizar la creación, modificación y eliminación de archivos y directorios.**
- **Desarrollo de aplicaciones: Al desarrollar aplicaciones que interactúan con el sistema, os permite manejar archivos y directorios de manera eficiente.**
- **Recopilación de información del sistema: Puedes obtener información sobre el entorno de ejecución de tu script, como las variables de entorno o el directorio de trabajo actual.**
-

}

PROYECTO1PYTHON {

interfaz

menu

modelo

asistencia

consultas

docentes

estudiantes

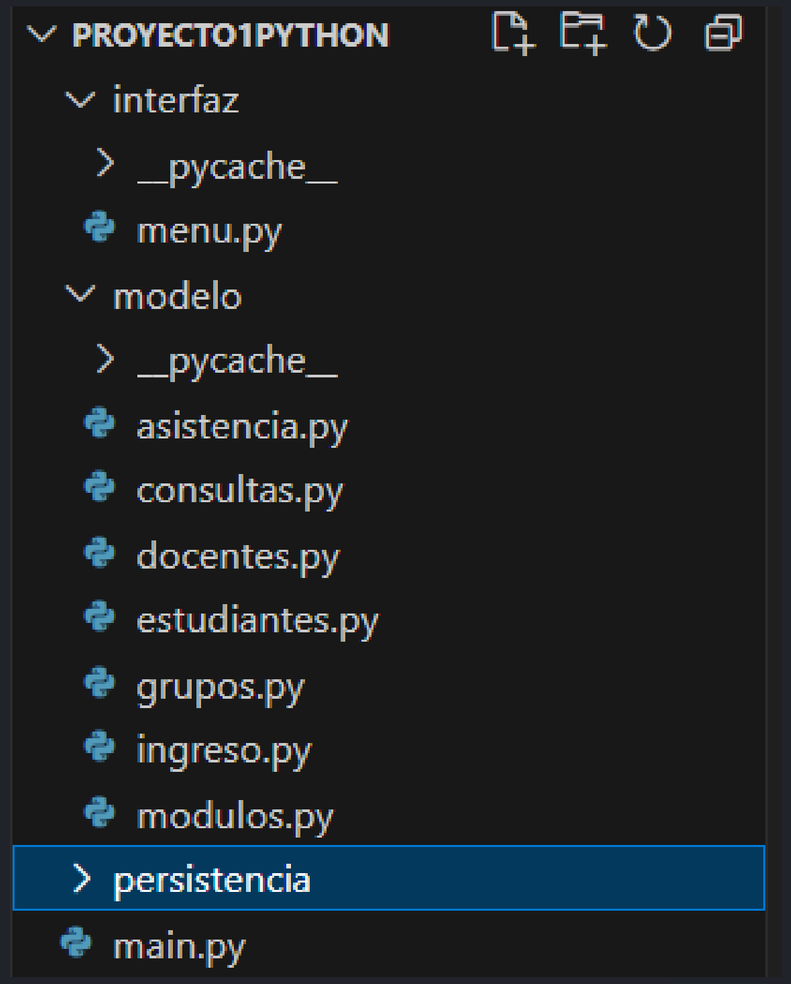
grupos

ingreso

modulos

persistencia

archivos.json



}

funcion para contraseña {

```
def iniciarSesion(): #Solicita usuario y contrasena
    usuario = input(">>> Ingrese su nombre de usuario <<<: ") #Ese
    contrasenaIntentada = input(">>> Ingrese la contraseña <<<: ")

    if not ingreso.verificarContraseña(contrasenaIntentada): # Llar
        print("Contraseña incorrecta. Saliendo.")
        return False # Retorna False indicando que la autenticació

    return True # Si la contraseña es correcta, retorna True para

def main(): # Función principal del programa.
    if iniciarSesion():  # Si la función iniciarSesion devuelve Tru
        menu.gestionarOpciones() # Llama a la función 'gestionarOp
    else:
        return  # Si la autenticación falla, la función 'main' sim

if __name__ == "__main__": # Comprueba si el script se está ejecut
    main()  # Llama a la función 'main' para iniciar el programa.
```

}

Estructuras de datos utilizadas.

Listas:
Se utilizan para almacenar colecciones de datos, como estudiantes, grupos y módulos. Por ejemplo, al cargar estudiantes desde el archivo JSON, se crea una lista que contiene diccionarios (cada diccionario representa un estudiante).

Diccionarios:
Se utilizan para representar objetos que tienen propiedades clave-valor. En este caso, cada estudiante, grupo y módulo se representan como un diccionario que contiene atributos como código, nombre, sexo, edad, etc.

Strings:
Se utilizan para almacenar texto, como nombres, códigos y mensajes de entrada/salida. Cada entrada del usuario se maneja como una cadena de texto.

}

Estructuras de datos utilizadas.

Números enteros:

Se utilizan para almacenar datos numéricos, como el número de estudiantes que se desea registrar o la edad del estudiante.

Booleanos:
Uso: Aunque no se ven explícitamente en las estructuras de datos, las validaciones a menudo producen resultados booleanos (verdadero o falso) que se utilizan para controlar el flujo del programa.
Ejemplo: La validación de si el código ingresado es numérico o si ya existe un estudiante con ese código se basa en comparaciones que devuelven valores booleanos.

}

importaciones

archivos

```python
import json
import os
from datetime import datetime
```

```python
archivoAsistencia = 'persistencia/asistencia.json'
archivoEstudiantes = 'persistencia/estudiantes.json'
archivoModulos = 'persistencia/modulos.json'
```

}

Estructura utilizada para correr el menu

if

```python
def gestionarOpciones(): # Función que gestiona las opciones del menú.
    while True: # Bucle que mantiene el menú activo hasta decida salir.
        mostrarMenu()
        try:
            opcion = int(input("Seleccione una opción del menu principal: "))
            print("_" * 55)
            if opcion == 1:
                grupos.registrarGrupo()
                print("Registro de grupos seleccionado.")
            elif opcion == 2:
                modulos.registrarModulo()
                print("Registro de módulos seleccionado.")
            elif opcion == 3:
                estudiantes.registrarEstudiantes()
                print("Registro de estudiantes seleccionado.")
            elif opcion == 4:
                docentes.registrarDocente()
                print("Registro de docentes seleccionado.")
            elif opcion == 5:
                asistencia.registrarAsistencia()
                print("Registro de asistencia seleccionado.")
            elif opcion == 6:
```

}

<!--Pyton-->

Gracias {

}