Trabajo final
Lógica de programación

Juan Pablo Restrepo Uribe

October 28, 2024

Descripción

El presente ejercicio tiene como objetivo que los estudiantes apliquen los conceptos de lógica de programación desarrollando un sistema en Python para la gestión de notas académicas. El sistema deberá permitir registrar estudiantes y las asignaturas que cursan, ingresar y parametrizar las evaluaciones de cada asignatura, y generar reportes detallados sobre el rendimiento académico de los estudiantes. A través de este proyecto, se espera que los estudiantes implementen diversas estructuras de control y datos, tales como listas, diccionarios, ciclos, condicionales y funciones, de manera que puedan automatizar el proceso de gestión de notas de manera eficiente y flexible.

Generales

A continuación te presento algunos elementos generales que deberán tener en cuenta a la hora presentar el trabajo, recuerden que este trabajo se presenta en dos etapas, una consiste en una exposición, donde el objetivo principal de la misma es vender el producto que están creando (el sistema de registro de notas) y una segunda etapa donde el docente revisara la el código presentado y evalúala su funcionalidad y correcto uso.

1. Nombre del sistema: Comenzar con un nombre atractivo y profesional para el sistema, que represente su función principal.
2. Descripción general: Ofrecer una breve descripción del sistema, explicando su propósito y cómo soluciona el problema de gestión de notas en un entorno educativo.
3. Público objetivo: Identificar quiénes son los usuarios potenciales del sistema, por ejemplo, docentes, instituciones educativas, o incluso estudiantes que deseen organizar sus propias calificaciones.
4. 4. Principales características: Explicar las principales funcionalidades del sistema, presentándolas como "ventajas competitivas" del producto. Destacar elementos como:
    - Registro de estudiantes y asignaturas.
    - Ingreso de evaluaciones con porcentajes parametrizables.
    - Cálculo automático de promedios.
    - Generación de reportes detallados.
    - Estadísticas del curso.
    - Funcionalidad de búsqueda y actualización de datos.
5. Beneficios: Enfatizar los beneficios de utilizar el sistema, tanto para los profesores como para los estudiantes:
    - Ahorro de tiempo: Automatización de cálculos y generación de reportes.
    - Precisión: Reducción de errores al parametrizar evaluaciones y calcular promedios.
    - Flexibilidad: Personalización de porcentajes y umbrales de aprobación.
    - Organización: Mejora la organización de notas y permite acceso rápido a la información.
6. Comparativa con métodos tradicionales: Comparar el uso del sistema con métodos tradicionales (como el uso de hojas de cálculo o registro manual) y demostrar por qué este sistema es más eficiente, preciso y fácil de usar.

7. Escalabilidad y futuras mejoras: Explicar cómo el sistema puede crecer y qué mejoras podrían implementarse en futuras versiones:
    - Integración con plataformas educativas más grandes (como Moodle).
    - Funcionalidades adicionales como seguimiento del rendimiento a lo largo del tiempo o integración de notificaciones automáticas para los estudiantes.

Funciones

1. Ingreso de estudiantes y asignaturas: El sistema debe permitir registrar nuevos estudiantes y las asignaturas que cursan.
2. Ingreso de notas por asignatura: El sistema deberá permitir el ingreso de las notas de los estudiantes en las diferentes asignaturas que cursa. Recuerde que cada materia tiene una parametrización diferente (exámenes, trabajos, participación, etc.).
3. Parametrización de porcentajes: El sistema debe permitir parametrizar los porcentajes de cada asignatura (por ejemplo, examen parcial 1 20 %, examen final 20 % trabajos 30 %, participación 30 %).
4. Cálculo de promedio por asignatura: Calcular el promedio final de cada estudiante (nota definitiva) para cada asignatura, utilizando los porcentajes previamente definidos.
5. Cálculo de nota parcial por asignatura: Calcular la nota de estudiante cuando este aún no ha finalizado el curso (faltan notas).
6. Generación de reportes individuales: Permitir generar un reporte individual para cada estudiante, indicando su promedio en cada asignatura y si ha aprobado o no (según un umbral de aprobación que también debe ser parametrizable).
7. Listado general de estudiantes: Mostrar un listado general con todos los estudiantes y su promedio en cada asignatura.
8. Estadísticas del curso: Generar estadísticas generales del curso, como el promedio general de los estudiantes, la mejor y peor calificación por asignatura, y el porcentaje de estudiantes aprobados por asignatura.
9. Actualización de notas y parámetros: Permitir actualizar las notas y los parámetros (porcentajes de evaluaciones y umbrales de aprobación) en cualquier momento.
10. Función de búsqueda: Implementar una función de búsqueda para encontrar rápidamente a un estudiante y sus notas por asignatura.
11. Menú de opciones: El sistema debe ofrecer un menú con las diferentes funcionalidades mencionadas (registro, consulta, reportes, estadísticas, etc.) y permitir navegar entre ellas.

Adicionales

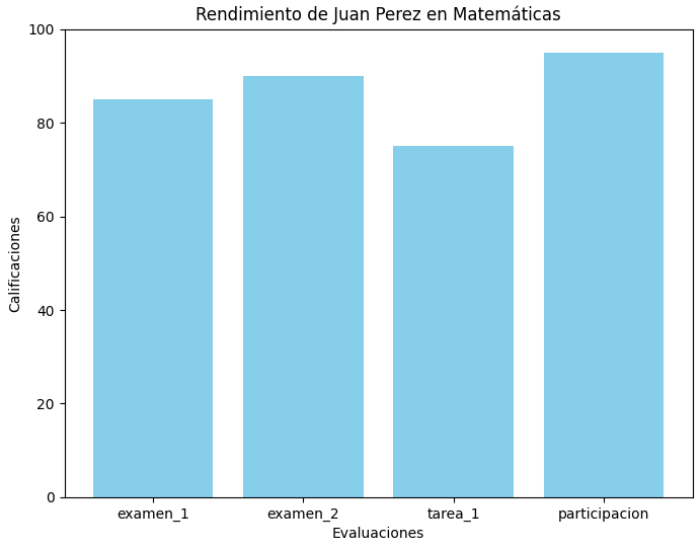
1. Implementar un sistema básico de autenticación, donde el usuario (profesor o administrador) tenga que iniciar sesión para acceder al sistema de gestión de notas. Esto introduce temas de seguridad y manejo de datos sensibles (incluir una nueva base de datos de usuarios).
2. Validar que las calificaciones ingresadas estén dentro de un rango válido (0 a 100 o 0 a 5) y que los porcentajes de evaluación sumen exactamente 100 %.
3. Incluir una funcionalidad para generar gráficos que representen el rendimiento académico (por ejemplo, una gráfica de barras o pastel que muestre el promedio de cada asignatura). Esto se puede hacer con librerías como matplotlib, dándoles una noción de visualización de datos.

```python
import matplotlib.pyplot as plt
# Base de datos de ejemplo aquí

def graficar_notas_estudiante(estudiante, asignatura):
    if estudiante in base_de_datos and asignatura in base_de_datos[estudiante]:
        notas = base_de_datos[estudiante][asignatura]["notas"]
```

```
        nombres_evaluaciones = list(notas.keys())
        calificaciones = [notas[evaluacion]["calificacion"] for evaluacion in nombres_evaluaciones]

        plt.figure(figsize=(8, 6))
        plt.bar(nombres_evaluaciones, calificaciones, color='skyblue')
        plt.xlabel('Evaluaciones')
        plt.ylabel('Calificaciones')
        plt.title(f'Rendimiento de {estudiante} en {asignatura}')
        plt.ylim(0, 100)
        plt.show()
    else:
        print(f"No se encontraron datos para {estudiante} en {asignatura}")
graficar_notas_estudiante("Juan Perez", "Matemáticas")
```



4. Manejo adecuado de los errores y excepciones en su código, como el ingreso de notas no numéricas o errores en el acceso a un estudiante o asignatura inexistente.

```
    def dividir_numeros():
    try:
        numerador = float(input("Ingrese el numerador: "))
        denominador = float(input("Ingrese el denominador: "))

        resultado = numerador / denominador

    except ZeroDivisionError:
        print("Error: No se puede dividir por cero.")

    except ValueError:
        print("Error: Por favor ingrese valores numéricos válidos.")

    else:
        print(f"El resultado de la división es: {resultado}")

    finally:
        print("Operación finalizada.")

dividir_numeros()

```

Notas

1. **Nota 1:** Puede usar bases de datos externas; sin embargo, eso no es de esta materia y aún no se ha visto, por lo cual le dejo un ejemplo de como podría organizar usted sus datos, esto no es una camisa de fuerza, hágalo como crea mejor.

```
base_de_datos = {
    "Juan Perez": {
        "Matem\'aticas": {
            "notas": {
                "examen_1": {"calificacion": 85, "porcentaje": 30},
                "examen_2": {"calificacion": 90, "porcentaje": 30},
                "tarea_1": {"calificacion": 75, "porcentaje": 20},
                "participacion": {"calificacion": 95, "porcentaje": 20}
            },
            "promedio_final": None
        },
        "Ciencias": {
            "notas": {
                "examen_1": {"calificacion": 80, "porcentaje": 50},
                "tarea_1": {"calificacion": 70, "porcentaje": 25},
                "participacion": {"calificacion": 85, "porcentaje": 25}
            },
            "promedio_final": None
        }
    },
    "Ana Gomez": {
        "Matematicas": {
            "notas": {
                "examen_1": {"calificacion": 90, "porcentaje": 30},
                "examen_2": {"calificacion": 88, "porcentaje": 30},
                "tarea_1": {"calificacion": 92, "porcentaje": 20},
                "participacion": {"calificacion": 100, "porcentaje": 20}
            },
            "promedio_final": None
        },
        "Ciencias": {
            "notas": {
                "examen_1": {"calificacion": 78, "porcentaje": 50},
                "tarea_1": {"calificacion": 85, "porcentaje": 25},
                "participacion": {"calificacion": 90, "porcentaje": 25}
            },
            "promedio_final": None
        }
    }
}

# Ejemplo de como acceder a las notas de un estudiante especifico:
print(base_de_datos["Juan Perez"]["Matematicas"]["notas"]["examen_1"]["calificacion"])
# Salida: 85

# Ejemplo de como actualizar una nota:
base_de_datos["Ana Gomez"]["Ciencias"]["notas"]["examen_1"]["calificacion"] = 82
```

2. **Nota 2:** Es muy probable que necesite más de una base de datos, por eso les dejo una segunda base de datos de ejemplo donde se agregan las notas.

```
base_de_datos = {
    "Juan Perez": {
        "Matematicas": {
            "examen_1": 85,
```

```
            "examen_2": 90,
            "tarea_1": 75,
            "participacion": 95
        },
        "Ciencias": {
            "examen_1": 80,
            "tarea_1": 70,
            "participacion": 85
        }
    },
    "Ana Gomez": {
        "Matem\'aticas": {
            "examen_1": 90,
            "examen_2": 88,
            "tarea_1": 92,
            "participacion": 100
        },
        "Ciencias": {
            "examen_1": 78,
            "tarea_1": 85,
            "participacion": 90
        }
    }
}
```

3. **Nota 3:** La idea es que todo el proyecto se desarrolle por medio de funciones.
4. **Nota 4:** Si requieren el uso de librerías, estas están permitidas.
5. **Nota 5:** Para los reportes individuales y de grupo, se pueden hacer por pantalla; sin embargo, les dejo un ejemplo de como se podría hacer en PDF.

```
from reportlab.lib.pagesizes import letter
from reportlab.lib import colors
from reportlab.platypus import SimpleDocTemplate, Table, TableStyle
def generar_pdf(base_de_datos, nombre_archivo):
    pdf = SimpleDocTemplate(nombre_archivo, pagesize=letter)
    elementos = []
    datos = [["Estudiante", "Asignatura", "Examen 1",
    "Examen 2", "Tarea 1", "Participaci\'on"]]

    for estudiante, materias in base_de_datos.items():
        for materia, notas in materias.items():
            fila = [
                estudiante,
                materia,
                notas.get("examen_1", "N/A"),
                notas.get("examen_2", "N/A"),
                notas.get("tarea_1", "N/A"),
                notas.get("participacion", "N/A")
            ]
            datos.append(fila)
    tabla = Table(datos)
    estilo = TableStyle([
        ('BACKGROUND', (0, 0), (-1, 0), colors.gray),
        ('TEXTCOLOR', (0, 0), (-1, 0), colors.whitesmoke),
        ('ALIGN', (0, 0), (-1, -1), 'CENTER'),
        ('GRID', (0, 0), (-1, -1), 1, colors.black),
        ('BACKGROUND', (0, 1), (-1, -1), colors.beige),
    ])
    tabla.setStyle(estilo)
    elementos.append(tabla)
```

```
    pdf.build(elementos)

generar_pdf(base_de_datos, "notas_estudiantes.pdf")
print("PDF generado con \'exito.")
```

6. **Nota 6:** Si desea eliminar lo que introdujo o se muestra por consola, puede hacerlo usando los comandos que se presentan a continuación (recuerde, es un ejemplo completo)

```
import time
from IPython.display import clear_output

for i in range(10):
    print(f"Iteración {i}")
    time.sleep(1)  # Esperar 1 segundo
    clear_output(wait=True)
```