PLANEACIÓN DE SEMESTRE INGENIERÍA INDUSTRIAL

Camila Lopez Cordero

1005321878

ALGORITMIA Y PROGRAMACIÓN

DOCENTE

Victor Hugo Mercado Ramos

UNIVERSIDAD DE ANTIOQUIA

FACULTAD DE INGENIERÍA

DEPARTAMENTO DE INGENIERÍA INDUSTRIAL

MEDELLÍN - ANTIOQUIA

2024_1

1. Documento de visión

Objetivo principal

El programa creado guardará cada nombre de una lista de 1000 estudiantes en una serie de archivos; cada uno de estos archivos corresponde a la asignatura que cursará cada estudiante de acuerdo al semestre para el cual se matriculó.

Objetivos específicos

- Generar una lista de 1000 nombres aleatorios para cada estudiante. No se puede repetir la misma combinación de nombres y apellidos dentro de la lista.
- Diseñar la planeación para cada grupo correspondiente a cada materia tomando en cuenta su semestre y su número de créditos como criterios principales.
- Facilitar la asignación de grupos para cada estudiante de acuerdo al semestre que está cursando.
- Simplificar el diseño del programa en cuestión a través del uso de diversas librerías de Python, principalmente orientadas al manejo de grupos de datos.

2. Especificación de requisitos

2.1 Lectura de los datos.

Tomando en cuenta que los datos que debemos leer se encuentran listados a modo de columna en un archivo de formato .csv, usaremos la función *open()* para guardar el archivo en una variable llamada *archivo* y la función *archivo.readlines()* para leer este archivo, tomar cada elemento de la "columna" y guardarlo en una lista de Python. Esto lo podremos aplicar para crear dos listas, una con los nombres y otra con los apellidos.

En la práctica, esto puede verse de la siguiente manera:

```
archivo = open('nombres.csv')
nombres = archivo.readlines()          Lista donde se encuentran los nombres
archivo = open('apellidos.csv')
apellidos = archivo.readlines()        Lista donde se encuentran los apellidos
```

Adicionalmente, también debemos eliminar el primer elemento de cada una de estas listas debido a que el primer elemento de cada archivo es el título del mismo (“Apellidos” y “Nombres”), y nuestras listas en este caso solamente necesitan los nombres como tal.

También es importante tener presente diversos factores que pueden influir al momento de leer y trabajar con los datos de los archivos:

- La función readlines() solamente sirve para extraer e ingresar a una lista elementos que se encuentren listados a modo de columna en una archivo tipo hoja de cálculo o separados por un salto de línea en un archivo plano de texto.

- En ambos archivos hay varios nombres repetidos, por tanto, también será necesario crear una función que tome las listas obtenidas y elimine los nombres que están repetidos, dando lugar a dos listas de nombres y apellidos aún más pequeñas.

  Para diseñar esta función emplearemos un ciclo que recorra la lista de nombres, identifique qué nombres se encuentran repetidos y luego los remueva con ayuda de la las funciones list.pop() o list.remove().

- Dado que los nombres masculinos y femeninos no deberían ir juntos al momento de crear un nombre (ejemplo, no debería haber alguien que se llame Isabella Juan), se dividirá la lista de nombres en otras dos listas:

  - nombresM ——— Nombres Masculinos
  - nombresF ——— Nombres Femeninos

  Para crear separar los nombres en estas dos listas crearemos una regla que evalúe cada elemento de la lista “nombres”, y que identifique si la cadena termina en “a”. Si el elemento en cuestión cumple este criterio, entonces será guardado en la lista “nombresF”, si no es el caso, guardará el elemento en la lista “nombresM”

  En este punto el programa tiene en cuenta todas las especificaciones que se necesitan para satisfacer la necesidad de generar una lista de nombres que cumple con las características que se nos piden.

  Adicionalmente, cada fragmento de esta primera parte del programa es una herramienta que nos permite manejar diversas excepciones que pueden dar más sentido al resultado que se quiere obtener: desde el hecho de que un nombre masculino no debe mezclarse con un nombre femenino, hasta el requisito de que un mismo nombre/apellido no se puede repetir.

2.2 Generación del archivo plano con la lista de nombres final.

Como debemos generar una lista de nombres completos aleatoria, emplearemos la librería "random" que nos permitirá obtener un número entero pseudoaleatorio dentro de un rango determinado a partir de la función "random.randint()". Estos números enteros se emplearán para generar las funciones que implementaremos para seleccionar dos elementos de cada lista tipo de lista (nombres y apellidos), haciendo posible que se seleccione una serie de 4 elementos de la siguiente manera.

Nombre Completo: Nombre + Nombre + Apellido + Apellido

Es importante tener en cuenta que un mismo Nombre Completo no podrá repetirse dentro de la lista de nombres completos, por tanto, al momento de generar cada nombre completo se hará uso de ciclos para comparar la combinación de nombres/apellidos que se está generando con la lista de las combinaciones que se están generando; si la combinación en cuestión ya existe, esta se descartará y se generará una diferente.

A su vez tampoco se podrá repetir un mismo apellido o un mismo nombre dentro de un nombre completo. Para lograr esto se puede guardar la lista de nombres y apellidos en una variable "temporal" que sea modificada la primera vez que se tome un nombre o un apellido para crear un Nombre Completo: una vez este nombre/apellido sea usado se eliminará de la lista temporal con la función list.remove() para garantizar que no se repita dentro de esta combinación, y a través de un ciclo se repetirá esta regla para cada combinación.

Esta serie de nombres completos será guardada en una lista de Python que llamaremos "nombresCompletos" que luego será recorrida a través de un ciclo para guardar cada nombre en un archivo plano de texto.

Ahora, para crear el archivo utilizaremos la función "open()" y guardaremos este archivo en una variable llamada "nombresFinales":

nombresFinales = open("Nombres Completos.txt", "x")

Es importante tener en cuenta que el primer parámetro es el nombre del archivo y su extensión, mientras que el segundo es necesario para que la función open() cree un nuevo archivo

Una vez se tenga el archivo "nombresFinales" y la lista de nombres "nombresCompletos", se utilizará un ciclo para recorrer la lista y empezar a añadir cada elemento de la misma al archivo usando la función archivo.write('elemento x de la lista').

En este segundo punto hacemos uso de las librerías que proporciona Python para escribir en archivos de forma simple y eficiente, y también hacemos uso de recursos ya aprendidos en el curso de Algoritmia y Programación como el manejo de ciclos y listas de Python para crear un programa que organiza los nombres de forma simple y eficiente.

2.3 Planeación para el siguiente semestre.

La librería que usaremos como herramienta principal para manipular y leer de forma apropiada y eficiente los datos en este punto, será la librería "Pandas", una librería especializada en la manipulación y el análisis de datos. Esta librería será muy útil pues proporciona operaciones diseñadas para manipular tablas de datos con facilidad, básicamente como Excel desde Python.

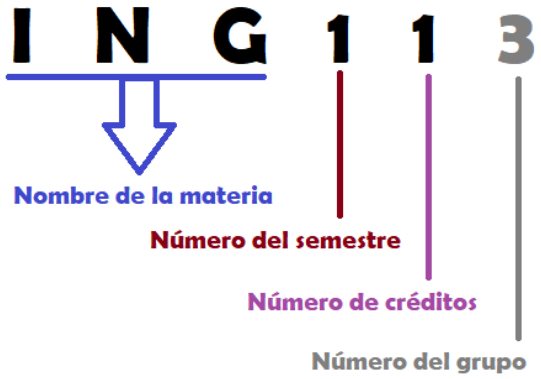
A continuación, un ejemplo de cómo se visualiza un "Dataframe" (una tabla de datos de dos dimensiones cuyos valores pueden ser leídos y también cambiar) impreso en GoogleColab:

	Nombre	Código de materia
0	Inglés1	ING1
1	Álgebra y Trigonometría	ALT
2	Gemotería Vectorial	GEV
3	Química	QUI

Para poder implementar la información particular de cada materia disponible dentro del pénsum, crearemos un archivo llamado "listaMaterias.csv" y listaremos cada grupo de materias con los datos pertenecientes a la misma de la siguiente manera:

Ingles1	ING11
AlgebraYTrigonometria	ALT13
Lectoescritura	LEC13
VivamosLaUniversidad	VIV11

En la columna izquierda estará el nombre completo de cada materia capitalizado mientras que en la columna izquierda se encontrarán los 5 primeros dígitos del código que corresponde a dicha materia cuya estructura será la siguiente:



Antes de proceder a crear la tabla con la configuración para el semestre siguiente, es muy importante tener presente la siguiente regla:

Para crear nuestro programa, asumimos que cada estudiante tomará todas las materias del Pénsum que es posible tomar para ese semestre. Ejemplo: todos los estudiantes del primer semestre deberán tomar las materias Álgebra y Trigonometría (3 créditos), Cálculo Diferencial (3 créditos), Geometría Vectorial y Analítica (3 créditos), Vivamos la Universidad (1 crédito), Inglés I (1 crédito), Lectoescritura (3 créditos) e Introducción a la Ingeniería Industrial (1 crédito), quedando cada estudiante con un total de 15 créditos por cursar.

La información referente a cada grupo de cada materia será guardada en una variable de tipo dataframe a través de la función pandas.DataFrame(), y nombraremos a esta variable "infoGrupos"

Ahora guardaremos todos los datos de el archivo listaMaterias.csv en una lista de Python en la cual cada elemento será una tupla, mientras que los elementos de esta tupla serán: (Nombre de la materia, Código), y se verá de la siguiente manera.

listaMaterías = [(nombre1, codigo1), (nombre2, codigo2), (nombre3, codigo3), …]

Una vez hecho esto, emplearemos un ciclo para recorrer listaMaterias, de esta manera tomaremos los valores que están en esta lista para evaluarlos y finalmente asignar un nuevo valor a cada categoría perteneciente a cada curso. Nuestro dataframe de infoGrupos debería quedar de la siguiente manera:

CA	**HTD**	**HTI**	**NTE**	**CC**	**TCA**	**FC**
ING111	16	16	140	3030303020	5	15/03/2024
LEC134	96	32	140	3030303020	5	15/03/2024

- *Código Asignatura (CA)*
- *Horas de trabajo docente (HTD)*
- *Horas de trabajo independiente (HTI)*
- *Número total de estudiantes (NTE)*
- *Código del curso (CC)*
- *Total de cursos asignados (TCA)*
- *Fecha de creación (FC)*

También es importante tener en cuenta que se crearán condiciones que cambien el valor de las variables "Porcentaje Proporcional" y "Cupos Aula" de acuerdo al semestre, como se puede ver a continuación. Es por esto que se creará una serie de condicionales que modifiquen estos valores de acuerdo al semestre que se esté evaluando:

Semestre	Porcentaje Proporcional	Cupos Aula
Semestre 1	14%	**30**
Semestre 2	13%	**30**
Semestre 3	12%	**30**
Semestre 4	11%	**25**
Semestre 5	10%	**25**
Semestre 6	10%	**25**
Semestre 7	9%	**20**
Semestre 8	8%	**20**
Semestre 9	7%	**20**
Semestre 10	6%	**10**

Ahora, para poder empezar a crear este dataframe se creará una serie de condiciones y operaciones que evalúen los códigos para cada materia, y de acuerdo a estas condiciones se asignará el valor adecuado para cada grupo:

Código Asignatura (CA):

El único dígito faltante para este código corresponde al número del grupo asignado. Para calcular el número de grupos para cada materia, tomaremos en cuenta lo siguiente.

- Evaluando el número del semestre, se crearán n grupos para cada materia:
  - Si (*Porcentaje Proporcional x 10) % Cupos Aula = 0*, entonces n = (*Porcentaje Proporcional x 10) / Cupos Aula*
  - Si (*Porcentaje Proporcional x 10) % Cupos Aula =! 0*, entonces *n* = (*Porcentaje Proporcional x 10 / Cupos Aula) + 1* , e ignoraremos la parte decimal de este resultado.

Horas de trabajo docente (HTD):

- Si el Número de créditos (quinto dígito del código) = 1 entonces HTD = 16
- Si el Número de créditos (quinto dígito del código) = 2 entonces HTD = 32
- Si el Número de créditos (quinto dígito del código) = 3 entonces HTD = 64
- Si el Número de créditos (quinto dígito del código) = 4 entonces HTD = 96

Horas de trabajo independiente (HTI):

- Si el Número de créditos (quinto dígito del código) = 1 entonces HTD = 32
- Si el Número de créditos (quinto dígito del código) = 2 entonces HTD = 64
- Si el Número de créditos (quinto dígito del código) = 3 entonces HTD = 80
- Si el Número de créditos (quinto dígito del código) = 4 entonces HTD = 120

Número total de estudiantes (NTE):

- NTE = Porcentaje proporcional x 10

Código del curso (CC):

Si tomamos en cuenta que hay n grupos, entonces se irá asignando un número en orden ascendente a cada grupo. Simplemente es importante tener en cuenta que siempre el último grupo será el de menos estudiantes. Ejemplo:

Ingles1 tiene 5 grupos, 4 grupos de 30 estudiantes y 1 grupo de 20:

- Grupo1, 30 estudiantes, CC = 1
- Grupo2, 30 estudiantes, CC = 2
- Grupo3, 30 estudiantes, CC = 3
- Grupo4, 30 estudiantes, CC = 4
- Grupo5, 20 estudiantes, CC = 5

Total de cursos asignados (TCA):

- Tomaremos el valor calculado en el primer ítem: *el sexto dígito del Código Asignatura (CA)*.

Fecha de Creación (FC):

- En este caso no se creará ninguna condición, simplemente utilizaremos la librería "datetime" (*from datetime import date*), y retornaremos la fecha actual utilizando la función "date.today()"

En este tercer punto, limitamos el alcance de nuestro programa para dejar en claro en qué tipo de situaciones puede ser usado.

Además, el uso de recursos adicionales como el archivo que contiene los nombres de las materias con los primeros 5 dígitos de su respectivo código, son herramientas que permiten manipular la información de forma eficiente, sin necesidad de escribir todos estos datos en el mismo código.

Todas los datos se encuentran conectados entre sí, hay conexión entre ellos, y las condiciones que creamos nos permiten clasificar la información cumpliendo con los requerimientos del programa que estamos diseñando.

2.4 Creación de listas para cada curso:

Debido a que en este punto no solamente debemos crear archivos, sino también directorios para organizar la información, es importante tener presente que haremos uso de la librería "os" (*import os*), que nos permitirá crear estos directorios a través de la función *os.mkdir('Nombre del directorio')*.

También, será necesario emplear la librería "openpyxl" para crear archivos de Excel en formato xlsx *(import openpyxl*) haciendo uso de la función "openpyxl.Workbook()".

Finalmente, importaremos la librería "csv" para poder crear y modificar archivos .csv.

Dicho esto, como primer paso, recorreremos la lista de datos que están en el archivo que creamos inicialmente (listaMaterias.csv), el cual contiene tanto el nombre completo de cada materia junto con los 5 primeros dígitos que se asignaron al código de cada grupo (CA). Empleando la información de cada elemento de esta lista de materias, crearemos la lista de directorios para cada materia de acuerdo al semestre de la siguiente manera:

- Ruta Trabajo Final
    - Número de Semestre
        - Nombre de Asignatura

Por ejemplo, para la materia Álgebra y Trigonometría se tomarán todos los datos que están en listaMaterias.csv y los directorios quedarán de la siguiente manera:

- Ruta Trabajo Final
    - Semestre 1
        - AlgebraYTrigonomteria

Ahora, como base para hacer este punto, recorreremos el data frame que creamos en el punto 3, "infoGrupos", y emplearemos la información que está en cada fila del mismo para crear dos archivos para cada grupo listado allí, uno en formato .csv y otro en formato .xlsx. Los archivos quedarán de la siguiente manera:

- Código del curso
- Nombre del curso sin espacio y capitalizado
- Cantidad de estudiantes
- Código del grupo

Por ejemplo, para el grupo del siguiente recuadro, el nombre de los archivos sería el siguiente:

CA	**HTD**	**HTI**	**NTE**	**CC**	**TCA**	**FC**
LEC134	96	32	140	4	5	15/03/2024

LEC134-Lectoescritura-30-4.csv

LEC134-Lectoescritura-30-4.xlsx

Debemos tener en cuenta lo siguiente para el 3er valor de estos nombres (cantidad estudiantes):

- Si CC < TCA entonces la cantidad de estudiantes será igual al número máximo de cupos por aula
- Si CC = TCA entonces la cantidad de estudiantes será igual al NTE – (número máximo de cupos por aula x (TCA - 1))

CC – Código Curso | TCA – Total de cursos asignados | NTE- Número total de estudiantes

Estos archivos se guardarán de la siguiente manera y empleando las siguientes funciones:

Por ejemplo, para el grupo 4 de Lectoescritura:

- CSV >
    1. open("TrabajoFinal\Semestre1\Lectoescritura*LEC134-Lectoescritura-30-4.csv*", "x") > *Creación del archivo*
    2. *Luego se modificará el archivo añadiendo los nombres que le corresponden*
- Excel >
    1. wb = openpyxl.Workbook() > *Creación del archivo*
    2. *Luego se modificará el archivo añadiendo los nombres que le corresponden*
    3. wb.save(TrabajoFinal\Semestre1\Lectoescritura*LEC134-Lectoescritura-30-4.xlsx*') > *Una vez modificado, se guardará el archivo en la ruta designada*

Una vez se tenga clara la estructura que se usará para crear los nombres de los archivos, y cómo y dónde se guardarán, podremos empezar a asignar cada estudiante generado aleatoriamente a cada grupo respectivamente. Para hacer esto, simplemente recorreremos la lista con los nombres completos ya creados que se mencionó en el primer punto (lista "nombresCompletos").

En este recorrido simplemente se creará una regla de acuerdo al número de estudiantes que se deba asignar para cada semestre, tomando en cuenta que tenemos una lista de 1000 estudiantes y tomando en cuenta nuevamente el porcentaje proporcional del siguiente recuadro:

Semestre	Porcentaje Proporcional	Cupos Aula
Semestre 1	14%	**30**
Semestre 2	13%	**30**
Semestre 3	12%	**30**
Semestre 4	11%	**25**
Semestre 5	10%	**25**
Semestre 6	10%	**25**
Semestre 7	9%	**20**

Semestre 8	8%	**20**
Semestre 9	7%	**20**
Semestre 10	6%	**10**

Entonces, los primeros 140 estudiantes de la lista “nombresCompletos” se asignarán todos a las materias del primer semestre; los siguientes 130 estudiantes se asignarán todos a las materias del segundo semestre, y así sucesivamente.

Durante el recorrido de la lista total de 1000 estudiantes, se irá asignando cada estudiante a su respectivo archivo mediante las siguientes funciones:

- Para los archivos .xlsx:

  wb = openpyxl.Workbook() > *Creación del archivo de Excel*

  hoja1 = wb.create_sheet("Hoja 1") > *Creación de la hoja en la que se escribirá*

  hoja["Nombre de la celda"] = Nombre del estudiante > *Escribir el nombre en la celda correspondiente*

- Para los archivos .csv:

  writer = csv.writer(archivoCsv) > *Se crea el archivo writer para llamar a las funciones de escritura en el archivo CSV.*

  writer.writerow(NombreEstudiante) > *Se llama a la función para escribir el nombre de X estudiante en la siguiente línea disponible*

Es importante tener presente que todos los nombres se listarán en forma de columna para cada uno de los archivos. Ejemplo:

JOHN ALEXANDER CARDONA CORREA
JUANA ANDREA RAMIREZ PEREZ
JUAN CAMILO LOPEZ OSORIO
ANDRES DAVID CUARTAS PENAGOS

Finalmente, en este último punto podemos ver cómo todos los recursos estructurados previamente nos permiten dar lugar a un producto final que cumple con las especificaciones de este proyecto. Cada fragmento del programa es una herramienta que facilita la planeación que se debe crear para el Semestre académico.

El uso de diversas librerías permite optimizar, no sólo recursos de hardware al emplear funciones preestablecidas, sino también tiempo por parte de quien elabora el programa.

Finalmente, la especificación de ciertas condiciones y excepciones, son factores que aportan para dar lugar a una solución apropiada que tiene en cuenta las necesidades de nuestro “usuario final”.