Proyecto de Nivel 4
Ejecución presupuestal de la Nación

Objetivo general

El objetivo de este proyecto es crear una aplicación para analizar información sobre la ejecución presupuestal que hizo el Estado colombiano en el año 2019. En el desarrollo de esta aplicación pondrá en práctica los conceptos del nivel 4.

Objetivos específicos

1. Implementar algoritmos para construir y recorrer matrices.
2. Utilizar las librerías pandas y matplotlib, así como consultar los sitios web oficiales donde se encuentra toda la documentación.
3. Descomponer un problema en subproblemas e implementar las funciones que los resuelven.

Instrucciones generales

La descripción de la aplicación le permitirá conocer el alcance, las funcionalidades esperadas y lo que debe realizar en este proyecto. **Tenga en cuenta que,** a lo largo de la descripción de la aplicación encontrará secciones con el título "**ATENCIÓN**" con indicaciones para conseguir que el resultado de su proyecto corresponda con lo esperado. Es importante que las siga cuidadosamente. Antes de empezar, le sugerimos leer con atención todo el proyecto. Mientras lo lee, trate de reconocer los conceptos del curso que tendrá que poner en práctica. Recuerda que este proyecto debe realizarse de forma **completamente individual**.

Descripción de la aplicación

La corrupción ha sido uno de los flagelos que más han afectado a Colombia durante toda su historia. Empresas, municipios, el Estado, e incluso personas del común en su día a día han replicado estas conductas que tanto daño hacen al país. No obstante, los esfuerzos para combatirla también han sido enormes. Líderes políticos y organismos gubernamentales han tratado, sin detenerse, de disminuir el efecto de la corrupción en la nación. Uno de estos muchos esfuerzos se ve reflejado en vigilar los recursos destinados en la celebración de contratos. La Agencia Nacional de Contratación Pública Colombia Compra Eficiente puso hace casi 10 años la plataforma SECOP II a disposición de todas las entidades del país para que registren todos sus contratos. Esto no solo ha ayudado a llevar un registro del dinero invertido, sino que les ha dado la oportunidad a entidades de menor tamaño (como alcaldías de pequeños municipios) de conocer qué empresas son confiables y cuáles son los precios que deberían pagar por productos y servicios. Como ciudadanos, todos nosotros podemos enterarnos de qué está comprando cada entidad, a quién, por cuánto dinero y utilizando qué tipo de método de contratación. Vale la pena aclarar que la información que se registra hoy en día en el SECOP II no representa ni la totalidad ni la mayoría de los contratos que se firman en el país, aunque se espera que en el futuro cercano se logre este objetivo.

En este proyecto usted trabajará con datos de una parte de los contratos registrados en el SECOP II, que está disponible a través del portal de datos abiertos del gobierno colombiano (https://www.datos.gov.co/Gastos-

Gubernamentales/SECOP-II-Contratos-Electr-nicos/jbjy-vk9h). En el archivo "2019.csv" usted encontrará una versión simplificada de los datos originales: suprimimos algunas entradas incompletas, contratos en borrador, y columnas que no vamos a utilizar en este proyecto. Además, eliminamos los contratos de años anteriores al 2019. Con este conjunto de datos usted puede participar en la difícil tarea de combatir la corrupción y conocer un poco más sobre cómo se invierten los recursos del país para construir un mejor futuro para todos.

¡Le deseamos la mejor de las suertes en esta revisión fiscal!

Su aplicación debe tener las siguientes partes:

Parte 1: Leer la información del archivo

Requerimiento 0: Cargar datos

Lo primero que debe hacer es permitir que se carguen los datos de un archivo csv a un DataFrame. Le debe preguntar al usuario el nombre del archivo, es decir que, **la función que implemente este requerimiento debe recibir como parámetro el nombre del archivo y debe retornar un DataFrame.**

Nota: Las tildes y eñes han sido suprimidas del archivo para evitar problemas de formatos al leer los datos. Los demás errores de ortografía, como el uso indebido de mayúsculas/minúsculas en algunas palabras, se deben a que los datos originales del SECOP II se encuentran así.

Las columnas del archivo y sus significados son las siguientes:

- NombreEntidad: nombre de la entidad que celebró el contrato.
- Departamento: nombre del departamento al que pertenece la entidad o "Distrito Capital de Bogota" si se trata de una entidad nacional. En caso de que no haya un departamento específico definido, este valor es "No Definido".
- Ciudad: nombre de la ciudad o municipio al que pertenece la entidad.
- Sector: sector económico al que pertenece la entidad. Ejemplos de estos sectores económicos son: "Servicio Publico", "Cultura", "Industria". Hay 24 sectores en total. En caso de que no haya un sector específico definido, este valor es "No aplica/No pertenece".
- Rama: rama estatal encargada de celebrar el contrato. Las posibles ramas son: "Ejecutivo", "Legislativo", "Judicial" y "Corporación Autonoma".
- DestinoGasto: destino del dinero del contrato al interior de la entidad. Este puede ser una "Inversion" o dinero para "Funcionamiento". En caso de que no haya un destino específico definido, este valor es "No Definido".
- EstadoContrato: estado actual del contrato. Este puede ser: "Activo", "modificado", "cedido", "Cerrado", "terminado", "Suspendido".
- DescripciondelProceso: descripción del motivo del contrato.
- TipodeContrato: tipo de contrato a nivel legal. Hay 21 tipos de contrato posibles. Algunos ejemplos son: "Prestacion de servicios", "Consultoria", "Comision", entre otros. En caso de que no haya un tipo de contrato específico definido, este valor es "No Especificado".
- ModalidaddeContratacion: modalidad en la que se adjudicó el contrato. Hay 14 modalidades de contratación diferentes. Algunos ejemplos son: "Licitacion publica", "Concurso de meritos abierto", "Contratacion directa", entre otras.
- FechaFirma: fecha de firma del contrato.
- FechadeIniciodelContrato: fecha de inicio del contrato.
- FechaFindelContrato: fecha de finalización del contrato.
- TipoDocProveedor: tipo de documento del proveedor. Por ejemplo, los proveedores que son personas naturales se identifican con su "Cedula de ciudadania" mientras que las empresas se identifican con un "NIT".
- ProveedorAdjudicado: nombre del proveedor a quien se adjudicó el contrato.

- EsGrupo: valor booleano que indica si el proveedor del contrato es un consorcio o un grupo temporal (True) o no lo es (False).
- EsPyme: valor booleano que indica si el proveedor del contrato es una PyME (pequeña y mediana empresa) (True) o no lo es (False).
- ValordelContrato: valor total (en millones de pesos) del contrato.
- ValorPendientedePago: valor (en millones de pesos) del saldo pendiente por pagar al proveedor del contrato.
- ValorPagado: valor (en millones de pesos) que el Estado ya ha pagado al proveedor del contrato.
- DiasAdicionados: días añadidos a la duración inicial del contrato.
- Anho: año de celebración del contrato.

ATENCIÓN: cuando esté estudiando el problema y el archivo, recuerde que las funciones describe() y unique() pueden serle de utilidad. La función describe() aplicada sobre un DataFrame retorna información estadística de todas las columnas numéricas. La función unique(), aplicada sobre una columna, retorna una lista con los valores únicos que aparezcan en esa columna.

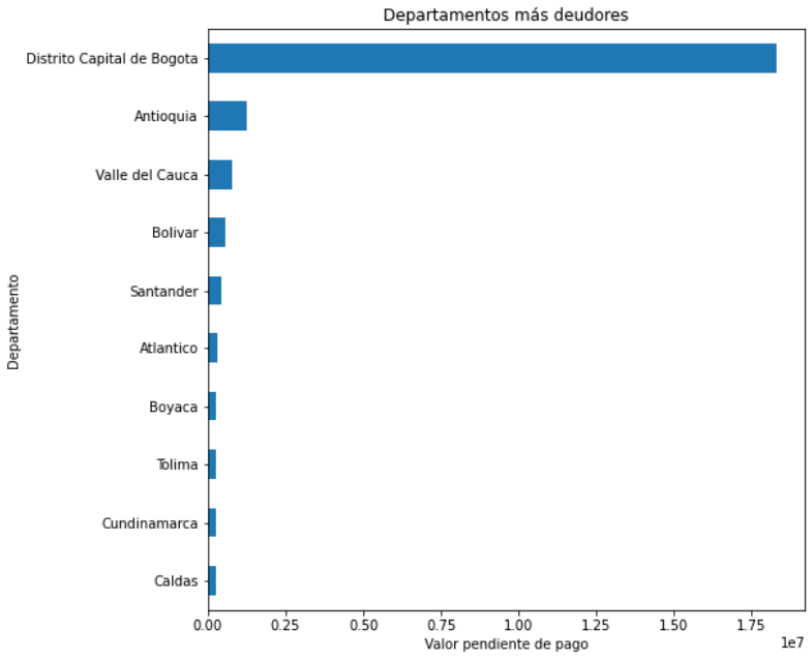
Parte 2: Consultar los contratos más costosos y la deuda por departamento

Requerimiento 1: Contratos más costosos

El primer requerimiento de la aplicación es consultar los contratos con mayor valor. Para esto, se debe crear un nuevo DataFrame a partir del original, el cual contenga la información de los 10 contratos con mayor valor en el DataFrame. De cada contrato, solo nos interesa la información de las siguientes 4 columnas: entidad contratante, departamento de la entidad, nombre del proveedor al cual se adjudicó el contrato y valor del contrato.

Requerimiento 2: Deuda por departamento

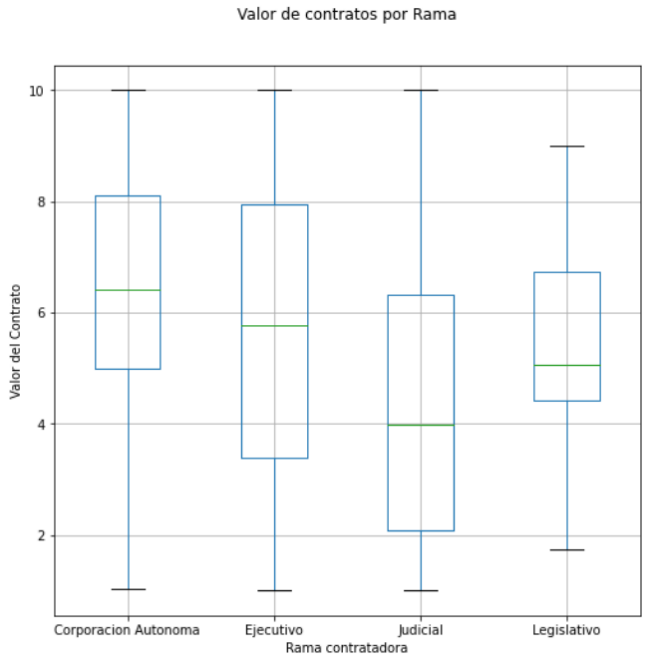
El segundo requerimiento es conocer cuáles son los departamentos que más dinero deben de los contratos celebrados. Para esto se debe crear una gráfica de barras horizontales, que muestre los 10 departamentos que tienen un mayor saldo pendiente por pagar a sus diferentes proveedores. Recuerde que esto se puede calcular sumando los valores encontrados en la columna *ValorPendientedePago*. La siguiente figura muestra la apariencia que debe tener esta gráfica.



Parte 3: Estudiar los valores de los contratos por rama del Estado

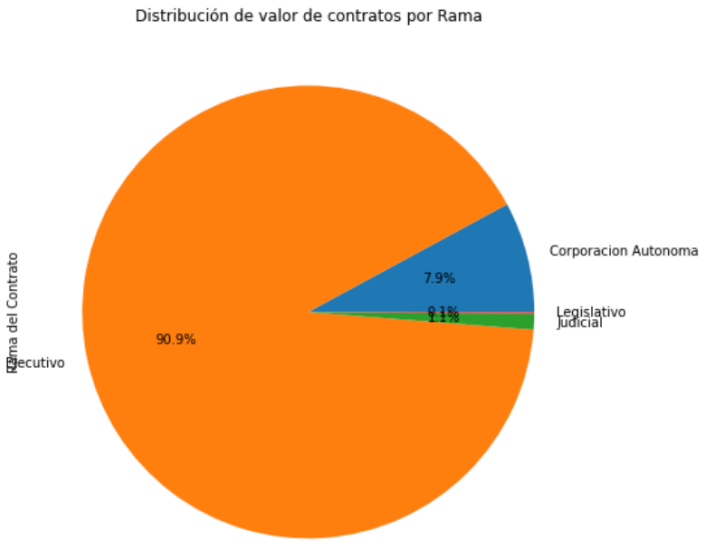
Requerimiento 3: Valor total de los contratos por cada rama

El tercer requerimiento de la aplicación debe permitir relacionar el valor de los contratos y la rama que los celebró. Para esto se debe crear una gráfica de tipo *boxplot*, usando la columna *ValordelContrato* y agrupando los datos de acuerdo con la columna *Rama*. Debido a que algunos de los contratos tienen un valor considerablemente mayor a los demás[1], el usuario debe poder escoger el rango de los valores de los contratos que desea visualizar, de modo que se deben poder fijar los límites superiores e inferiores para los datos que se muestran en la gráfica. La siguiente figura muestra la apariencia de esta gráfica cuando se utiliza como límite inferior 1 millón de pesos y como límite superior 10 millones de pesos.



Requerimiento 4: Repartición porcentual del valor total de los contratos entre las diferentes ramas del Estado

El cuarto requerimiento de la aplicación es permitir generar una gráfica de tipo *pie* que muestre la repartición porcentual del valor total de los contratos del Estado entre las diferentes ramas que los celebran. Recuerde que esta información se encuentra en la columna *Rama* del DataFrame. La siguiente figura muestra la apariencia que debe tener esta gráfica.

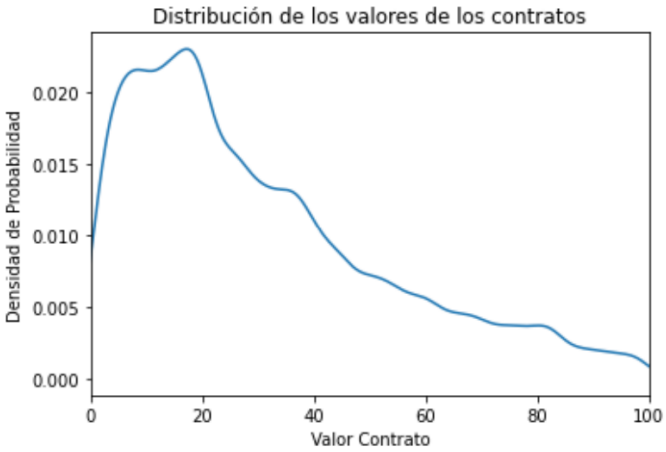


[1] Piense, por ejemplo, en los contratos relacionados con el metro de Bogotá.

ATENCIÓN: Puede configurar el tamaño de la figura con el parámetro figsize=(8,8) al momento de crear la figura.

Requerimiento 5: Distribución de los valores de los contratos

El quinto requerimiento consiste en mostrar la distribución de los valores de los contratos. Para esto se debe generar una gráfica de tipo KDE usando la columna *ValordelContrato*. Las gráficas de tipo KDE (kernel density estimation) son una forma de estimar la función de densidad de probabilidad de una variable aleatoria, que en este caso, se trata del valor en millones de pesos de un contrato. Puesto que existen contratos que son extremadamente más costosos que todos los demás, para poder tener una visualización adecuada de la gráfica, usted debe filtrar el DataFrame para quedarse sólo con los contratos que tengan un valor inferior a 100 millones de pesos. Defina límites explícitos para el eje X usando el parámetro xlim y una tupla con el menor valor (0) y el mayor valor (100) que deberían aparecer. La siguiente figura muestra la apariencia que debe tener esta gráfica.



Parte 4: Estudiar los valores de los contratos por sector y por departamento

En esta última parte de la aplicación no se trabajará directamente sobre el DataFrame original, sino que se creará una matriz a partir de este. La matriz debe relacionar el monto de los contratos de cada uno de los departamentos y los sectores en los cuales se desarrollan dichos contratos.

Requerimiento 6: Construcción de la matriz de departamentos vs sectores

Este requerimiento consiste en construir una matriz que cruce departamentos con sectores. La matriz debe tener la siguiente estructura:

	Ambiente y ...	No aplica/No...	Inclusion Soc...	Ley de Justicia	...	Relaciones Ext...
Putumayo	10527.10...	5880.95...	0	0	...	0
Distrito Capi...	304527.85...	1405949.32...	1129528.56...	771661.38...	...	457054.94...
Cundinamarca	4789.10...	57378.44...	1337.89...	5463.20...	...	0
Tolima	24409.57...	33404.75...	0	11524.56...	...	0
...	...	...	...	...	...	...
Vaupes	0	0	0	0	...	0

La primera fila (fila 0) tiene los nombres de cada uno de los diferentes sectores de inversión definidos en la columna *Sector* del DataFrame, mientras que la primera columna (columna 0) tiene los nombres de los departamentos tal como están definidos en la columna *Departamento* del DataFrame. Cada posición (f, c) en la matriz representa el valor total de los contratos firmados por el departamento *f* y desarrollados en el sector *c*. Por ejemplo, si "Antioquia" está en la fila 9, y "Cultura" se encuentra en la columna 8, la casilla de la posición (9, 8) representa la suma del valor de todos los contratos del departamento de Antioquia desarrollados en el sector de Cultura. Note también que la suma de una fila completa de la matriz nos da como resultado el valor total de todos los contratos del respectivo departamento, mientras que la suma de una columna completa de la matriz nos da como resultado el valor total de los contratos invertidos en el respectivo sector.

La función que usted implementará en este punto debe recibir por parámetro el DataFrame original y retornar una matriz con la estructura e información descritas anteriormente. **Es importante que antes de proceder a construir la matriz, realice un filtro sobre el DataFrame para excluir las filas que tienen como valor en la columna *Departamento* un "No Definido"**.

ATENCIÓN:

- Recuerde que tiene a su disposición la función unique() para obtener todos los valores únicos de una columna en un DataFrame.
- Para comprobar que los valores de la matriz creada son los correctos puede utilizar filtros, agrupaciones y sumas sobre el DataFrame original y verificar que estos arrojen el mismo resultado que el almacenado en su matriz.
- Puede ser de mucha utilidad pensar en descomponer el problema en varios problemas más pequeños e implementar la solución usando varias funciones.

Nota importante: Dado que el tamaño del DataFrame es bastante grande, es muy probable que la construcción de la matriz tarde más tiempo del que suelen tardar otras funciones. Si el tiempo supera los 2 minutos es muy probable que haya un problema en el código. Le recomendamos que detenga el programa y revise si existe algún error en sus funciones. También le recomendamos que pruebe sus funciones con un subconjunto de los datos (por ejemplo sólo con los contratos que tengan un valor inferior a 1 millón de pesos).

Utilizando como base la matriz anterior, usted deberá implementar los siguientes requerimientos.

Requerimiento 7: Sectores en los que el estado invierte más y menos, de acuerdo al valor de sus contratos

Este requerimiento debe calcular en cuáles sectores el estado ha gastado más y menos dinero teniendo en cuenta todos los contratos celebrados en cada sector. Para esto, el usuario debe indicar si desea conocer el sector de mayor gasto o el sector de menor gasto y suministrar la matriz de departamentos vs. sectores. La función que implemente esta opción debe retornar una tupla que tenga en la primera posición el nombre del sector y en la segunda posición el valor total de los contratos. Estos dos datos se le deben informar al usuario.

Requerimiento 8: Valor total de los contratos de un departamento

Este requerimiento consiste en calcular el valor total de los contratos de un departamento a partir de la matriz. Para esto, el usuario debe indicar el nombre del departamento a consultar y suministrar la matriz de departamentos vs. sectores. Se le debe informar al usuario el valor total de los contratos de dicho departamento.

Requerimiento 9: Departamentos con mayor gasto

Este requerimiento debe mostrar cuáles son los 10 departamentos que tienen un mayor gasto. Esto es, los 10 departamentos con el mayor valor total de sus contratos. El usuario debe suministrar la matriz de departamentos vs. sectores. Adicionalmente, es obligatorio hacer uso de la opción anterior, que permite calcular la ejecución presupuestal total de un departamento. Se retornará un **diccionario** cuyas llaves son los nombres de los 10 departamentos con mayor gasto, y los valores son sus respectivos totales.

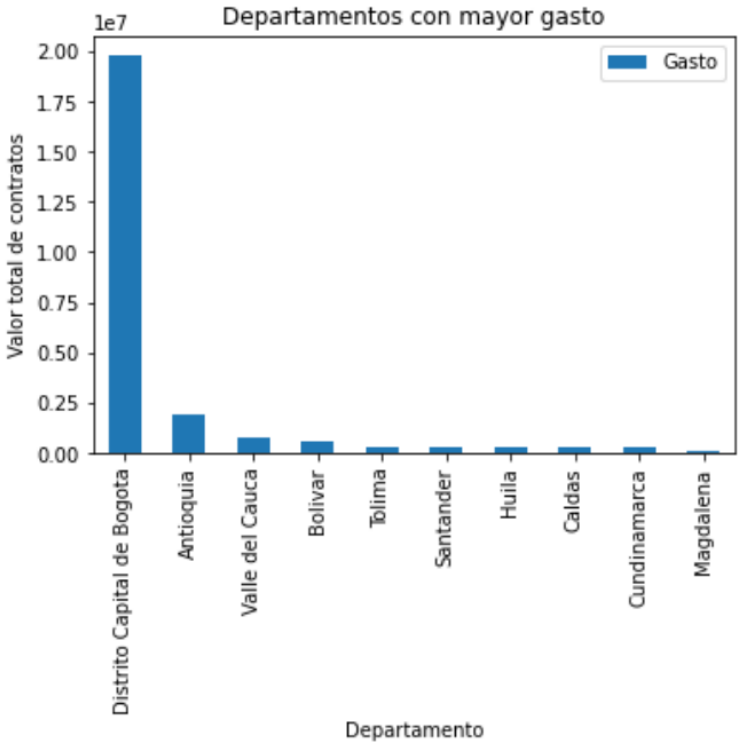
ATENCIÓN: Para calcular con mayor facilidad el resultado, le recomendamos guardar el valor total de los contratos de cada departamento en un diccionario auxiliar y usar este diccionario para armar el diccionario final.

Una vez obtenidos los 10 departamentos con mayor gasto, se debe mostrar la información **en una gráfica de barras**. Para la construcción de esta gráfica, primero construiremos un nuevo DataFrame a partir del diccionario que se acaba de generar. El siguiente fragmento de código permite construir un DataFrame con las características deseadas a partir de un diccionario:

```
df = pd.DataFrame.from_dict(diccionario,orient='index',columns=['Gasto'])
```

En la instrucción anterior “diccionario” es el nombre de la variable donde se guardó el diccionario con los 10 departamentos con mayor gasto y “Gasto” es el nombre que se le va a dar a la columna en el nuevo DataFrame que contendrá los valores del diccionario.

Luego de construir este nuevo DataFrame, el siguiente paso es generar una gráfica de barras. La gráfica resultante debe tener la siguiente apariencia:



Requerimiento 10: Departamentos más dedicados a un sector

Este último requerimiento de la aplicación consiste en calcular los 5 departamentos más dedicados a un sector. La dedicación de un departamento a un sector se calcula obteniendo el porcentaje del gasto total del departamento (esto es, el valor total de todos sus contratos) que está dedicado a dicho sector. Es decir, aquellos departamentos que, al aplicar la siguiente ecuación, tengan un mayor resultado:

$$\%\ gasto\ sector = \frac{valor\ total\ de\ los\ contratos\ del\ Departamento\ dedicados\ al\ sector}{valor\ total\ de\ todos\ los\ contratos\ del\ Departamento} * 100$$

El usuario debe suministrar la matriz de departamentos vs. sectores y el nombre de un sector de su interés. La función que usted desarrolle debe construir un diccionario que tenga como llaves los nombres de los 5 departamentos que lideran en dicho sector de acuerdo con esta estadística, y como valores sus respectivos porcentajes dedicados al sector de interés.

Una vez obtenidos los 5 departamentos con mayor gasto, se debe mostrar la información **sobre un mapa de Colombia.** Para esta visualización utilizaremos el archivo **mapa.png** (que se encuentra adjunto a este enunciado), el cual es una imagen del mapa de Colombia en blanco y negro con resolución de 800x800 píxeles. La idea es indicar dentro del mapa los departamentos que hayamos encontrado como los de mayor dedicación al sector. Para cada uno de estos Departamentos, se pintará un pequeño cuadrado de 13x13 píxeles centrado en las coordenadas (en píxeles) del Departamento dentro el mapa. Las coordenadas de los 33 departamentos se encuentran en el archivo **coordenadas.txt** (que se encuentra adjunto a este enunciado).

Para cargar el mapa como una matriz de píxeles y luego visualizarlo puede guiarse por el siguiente fragmento de código:

```
import matplotlib.image as mpimg
mapa = mpimg.imread("mapa.png").tolist()
plt.imshow(mapa)
plt.show()
```

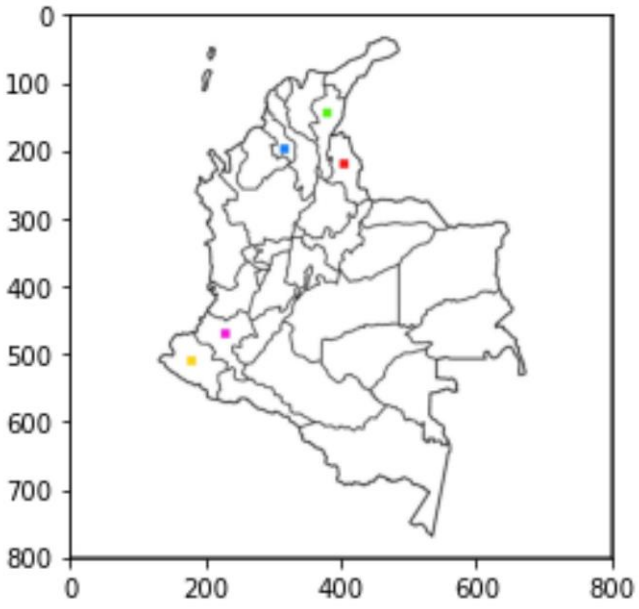
La siguiente función le permitirá cargar el archivo de coordenadas y retornar un diccionario, cuyas llaves son los nombres de los departamentos y los valores son tuplas con las coordenadas (x,y) de cada departamento. El nombre del departamento en este archivo es idéntico al que se encuentra en el DataFrame y en la matriz.

```
def cargar_coordenadas(nombre_archivo:str)->dict:
    deptos = {}
    archivo = open(nombre_archivo, encoding="utf8")
    titulos = archivo.readline()
    linea = archivo.readline()
    while len(linea) > 0:
        linea = linea.strip()
        datos = linea.split(";")
        deptos[datos[0]] = (int(datos[1]),int(datos[2]))
        linea = archivo.readline()
    return deptos
```

Una vez cargado el mapa y construido el diccionario de coordenadas, el siguiente paso es utilizar dichas coordenadas para pintar en el mapa los 5 departamentos más dedicados a este sector. Para esta visualización se desea que cada puesto en la clasificación (de los departamentos más dedicados a un sector) esté asociado a un color RGB en particular. La siguiente tabla indica los colores deseados.

Puesto	Valores RGB	Color
1	[0.94, 0.10, 0.10]	Rojo
2	[0.94, 0.10, 0.85]	Magenta
3	[0.10, 0.50, 0.94]	Azul
4	[0.34, 0.94, 0.10]	Verde
5	[0.99, 0.82, 0.09]	Amarillo

Cada uno de los cuadrados en el mapa debe utilizar uno estos colores, y se debe respetar el orden de la clasificación. La siguiente gráfica muestra el mapa para el sector de “Ley de Justicia”, en donde el top 5 de estados está compuesto en ese orden por: Norte de Santander, Cauca, Sucre, Cesar y Narinho:



ATENCIÓN: Tenga en cuenta que puede suceder que alguno de los sectores no alcance a tener contratos en al menos 5 departamentos. Es decir que el número de departamentos que han celebrado contratos para dicho sector es menor a 5. En este caso el diccionario debe solo incluir tantas parejas llave-valor como departamentos haya, y no incluir ningún departamento que tenga 0% de sus gastos dedicados a tal sector.

Actividad 1: Preparación del ambiente de trabajo

1. Cree una carpeta para trabajar, poniéndole su nombre o login.
2. Descargue de BrightSpace el archivo "contratos.zip" que contiene los archivos: "2019.csv" con los datos a procesar, "mapa.png" con el mapa en blanco y negro para la gráfica del mapa, y "coordenadas.txt" con las coordenadas de cada departamento en el mapa.
3. Abra Spyder y cambie la carpeta de trabajo para que sea la carpeta donde descargó el archivo con los datos.

Actividad 2: Construir el módulo de funciones

4. Usando Spyder, cree en su carpeta de trabajo un nuevo archivo con el nombre "contratos.py". En este archivo usted va a construir el módulo en el que va a implementar las funciones que responden a los requerimientos de la aplicación. **Defina, documente e implemente** las funciones en su nuevo archivo. Usted puede crear cuántas funciones considere necesarias dentro de su librería o módulo. Mínimo debe haber una función por cada uno de los requerimientos del programa.

Actividad 3: Construir la interfaz de usuario basada en consola

5. Implemente la interfaz de la aplicación en un nuevo archivo llamado "consola.py". Esta interfaz debe seguir la misma estructura de las consolas que hemos implementado en laboratorios y proyectos anteriores. Esto es: debe existir una función llamada iniciar_aplicacion() para que muestre el menú usando la función mostrar_menu() y permita al usuario seleccionar una opción. El menú que se despliegue debe permitir al usuario ejecutar todas las acciones de la descripción de la aplicación, así como salir (terminar) del programa. Por cada una de las funciones principales de su programa, debe existir una función en la consola que la ejecute, pidiendo previamente los datos necesarios al usuario (si aplica) e imprimiendo por pantalla el resultado de la función. Se sugiere nombrar estas funciones como ejecutar_XX, donde XX es la respectiva función de su módulo. Cuando haya implementado la función iniciar_aplicacion()corra su programa y verifique que se comporta adecuadamente, le permite al usuario seleccionar las opciones que quiere ejecutar, y termina el programa cuando se le indique.

Actividad 4: Probar el correcto funcionamiento de su programa

6. **Ejecute el programa** y **pruebe** cada una de las funciones para asegurarse que esté funcionando. Puede probar el correcto funcionamiento de su programa cargando la información que se encuentra en el archivo "2019.csv" o creando su propio archivo de prueba de menor tamaño (respetando el mismo formato) que le permita corroborar que los resultados arrojados por su programa son correctos.

Entrega

7. Comprima los dos archivos: contratos.py y consola.py en un solo archivo .zip. El archivo comprimido debe llamarse "**N4-PROY-login.zip**", donde login es su nombre de usuario de Uniandes.
8. Entregue el archivo comprimido a través de BrightSpace en la actividad designada como "**Proyecto N4**".