INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

LICENCIATURA EN CIENCIA DE DATOS

UNIDAD DE APRENDIZAJE

DESARROLLO DE APLICACIONES PARA ANÁLISIS DE DATOS

PRÁCTICA 3

NOMBRE DE LOS ALUMNOS:

DE LUNA OCAMPO YANINA

PROFESOR:

ITURIEL FLORES

GRUPO:

4CDM1

FECHA:

04/05/2022

En esta práctica pusimos a prueba el uso de pandas y DataFrames con algunas sus funciones aprendidas durante la clase.

Los pasos que se siguieron fueron los siguientes:

1. Primero importamos las librerías que utilizaremos durante toda la práctica. El de "*csv*" se utilizó para poder importar el, como dice el nombre, csv brindado en la práctica. El "*describe*" se usó para una función del DataFrame que se explicará adelante y por último "*pandas*", que nos ayuda al análisis de datos y proporcionando estructuras de datos flexibles.

```
# Importamos librerias
from pydoc import describe
import pandas as pd
import csv
```

2. Leemos el csv para poder pasarlo posteriormente al DataFrame.

```
datos = pd.read_csv('Book1.csv')
```

3. Vemos si los datos se cargaron correctamente, para esto utilizamos el comando "*head*" y poder visualizar las primeros 5 columnas de todo el DataFrame.

```
# Ver si se cargo completamente
datos.head()
```

	animal	age	visits	priority
0	cat	2.5	1	yes
1	cat	3	3	yes
2	snake	0.5	2	no
3	dog	-	3	yes
4	dog	5	2	no

4. En la siguiente instrucción se realizaron dos pasos, el primero fue convertir el csv importado, a DataFrame como se solicitó.
   La primera instrucción empezando la práctica es cambiar los index a las primeras letras del abecedario hasta que se terminen las filas del DataFrame, en este caso fue hasta j. Aquí es importante recalcar que al momento de cambiarle los index, cambié el nombre del conjunto.

```
# Cargar el archivo y hacerlo DataFrame
df = pd.DataFrame(datos)

# Inciso a
# A cada fila asigarle una letra como nombre o indice. Ejemplo: primera fila = 'a', segunda
rnm = df.rename(index={0:'a',1:'b',2:'c',3:'d',4:'e',5:'f',6:'g',7:'h',8:'i',9:'j',10:'k'})
rnm
```

	animal	age	visits	priority
a	cat	2.5	1	yes
b	cat	3	3	yes
c	snake	0.5	2	no
d	dog	-	3	yes
e	dog	5	2	no
f	cat	2	3	no
g	snake	4.5	1	no
h	cat	-	1	yes
i	dog	7	2	no
j	dog	3	1	no

5. Mostramos la información básica del DataFrame, junto con: incluido el número de filas, los nombres de las columnas, el número y el tipo de valores en cada columna. # Aquí es en donde se utiliza la librería de pydoc para obtener este, nos brinda como se ve en la imagen: desviación estándar, media, cuartiles, mínimo, máximo, etc.

```
# Inciso b
# Mostrar la informacion basica del df,
rnm.describe()
```

	visits
count	10.000000
mean	1.900000
std	0.875595
min	1.000000
25%	1.000000
50%	2.000000
75%	2.750000
max	3.000000

Con .info obtuvimos el nombre de las columnas y el tipo de datos que se almacena en estas.

```
rnm.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Index: 10 entries, a to j
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype 
---  ------    --------------  ----- 
 0   animal    10 non-null     object
 1   age       10 non-null     object
 2   visits    10 non-null     int64 
 3   priority  10 non-null     object
dtypes: int64(1), object(3)
memory usage: 400.0+ bytes
```

Obtenemos cuántas filas y columnas son, con un format las unimos e imprimimos los valores.

```
fil = len(rnm.index)
col = len(rnm.columns)
print(f'El número de filas que hay son: {fil} y el de columnas son: {col}')
```

```
El número de filas que hay son: 10 y el de columnas son: 4
```

Aquí, con dtypes vemos el tipo de datos que tiene cada columna, de la misma forma como pudimos ver en el .info.

```
rnm.dtypes
```

```
animal      object
age         object
visits       int64
priority    object
dtype: object
```

6. Para obtener el nombre de columnas y filas se utiliza .columns.values e index.values, nuevamente con un format unimos las respuestas y las mostramos.

```
nomCol = rnm.columns.values
nomFil = rnm.index.values
print(f'El nombre de las columnas son: {nomCol}.')
print(f'El nombre de las filas son: {nomFil}.')
```

```
El nombre de las columnas son: ['animal' 'age' 'visits' 'priority'].
El nombre de las filas son: ['a' 'b' 'c' 'd' 'e' 'f' 'g' 'h' 'i' 'j'].
```

7. Siguiendo con las instrucciones, solo mostramos las columnas “animal” y ”age” con todas sus filas. Para esto usamos .loc[] con los parámetros estipulados.

```
# Inciso c
# Mostrar solo las columnas animal y age con todas sus filas.
rnm.loc[:, ['animal', 'age']]
```

	animal	age
a	cat	2.5
b	cat	3
c	snake	0.5
d	dog	-
e	dog	5
f	cat	2
g	snake	4.5
h	cat	-
i	dog	7
j	dog	3

8. Mostramos las columnas con las filas pares, para esto se usa la función .iloc[] con sus parámetros correctamente estipulados.

```
# Inciso d
# Mostrar todas las columnas con únicamente las filas pares.
# matrix[row][column][skip]
rnm.iloc[::2]
```

	animal	age	visits	priority
a	cat	2.5	1	yes
c	snake	0.5	2	no
e	dog	5	2	no
g	snake	4.5	1	no
i	dog	7	2	no

9. Mostramos de igual forma como arriba, solo las columnas age y visits pero únicamente con las filas impares.

```
# Mostrar solo las columnas a
# Inciso e
rnm.iloc[[1,3,5,7,9], [1,2]]
```

	age	visits
b	3	3
d	-	3
f	2	3
h	-	1
j	3	1

10. Mostrar todas las columnas con las filas que no contengan un dato para edad. Ponemos los na_values = “-”, ya que no lo toma como entero, si no como string.

```
# Inciso g
# Mostrar todas las columnas con las filas que no c
data = pd.read_csv('Book1.csv', na_values = ['-'])
print(data["age"].isna())
print("\n\n-----------------------------\n\n")
print(data[data["age"].isna()==True])
```

```
0    False
1    False
2    False
3     True
4    False
5    False
6    False
7     True
8    False
9    False
Name: age, dtype: bool
```

Podemos imprimirlos así o solo las columnas que tiene los valores nulos.

```
  animal  age  visits priority
3    dog  NaN       3      yes
7    cat  NaN       1      yes
```

11. Mostrar todas las columnas con las filas cuya edad sea mayor a 3. Accedemos a la fila y comparamos, ya quitando los nulos para que no tenga problema.

```
# Inciso f
# Mostrar todas las columnas co
print(data[(data["age"] > 3)])
```

```
  animal  age  visits priority
4    dog  5.0       2       no
6  snake  4.5       1       no
8    dog  7.0       2       no
```

12. Sustituir las celdas que no contengan un dato para edad con el promedio del resto de edades. Para esto usamos la función. mean()

```
# Inciso h
# Sustituir las celdas que no contengan un dato para
data["age"] = data["age"].fillna(data["age"].mean())
print(data)
```

```
  animal     age  visits priority
0    cat  2.5000       1      yes
1    cat  3.0000       3      yes
2  snake  0.5000       2       no
3    dog  3.4375       3      yes
4    dog  5.0000       2       no
5    cat  2.0000       3       no
6  snake  4.5000       1       no
7    cat  3.4375       1      yes
8    dog  7.0000       2       no
9    dog  3.0000       1       no
```

13. Mostrar todas las columnas con las filas cuya edad este sea mayor a 2 y menor o igual 5. Usamos lo mismo para este pero con “&” para poder comparar las dos.

```
# Inciso i
# Mostrar todas las columnas con las filas cuya edad
print(data[(data["age"] > 2) & (data["age"] <= 5)])
```

```
  animal     age  visits priority
0    cat  2.5000       1      yes
1    cat  3.0000       3      yes
3    dog  3.4375       3      yes
4    dog  5.0000       2       no
6  snake  4.5000       1       no
7    cat  3.4375       1      yes
9    dog  3.0000       1       no
```

14. Sacamos la suma de todas las visitas con la función .sum de la columna del dataset de “visits”.

```
# Inciso j
# Obtener la suma de todas
suma = rnm['visits'].sum()
suma
```

```
19
```

15. Calcular el promedio de la edad de cada tipo diferente de animal. Tip: Investigar y usar el método "groupby".

```
# Inciso k
# Calcular el promedio de la edad de cada t
promAni = data.groupby(by='animal').mean()
promAni
```

animal	age	visits
cat	2.734375	2.0
dog	4.609375	2.0
snake	2.500000	1.5

16. Desplegar el data frame ordenado por por dos criterios, primero por edad en orden descendente y después por visitas en orden ascendente. Tip: Investigar y usar el método "sort_values".

descendente

```
data.sort_values(by='age', ascending=False)
```

	animal	age	visits	priority
8	dog	7.0000	2	no
4	dog	5.0000	2	no
6	snake	4.5000	1	no
3	dog	3.4375	3	yes
7	cat	3.4375	1	yes
1	cat	3.0000	3	yes
9	dog	3.0000	1	no
0	cat	2.5000	1	yes
5	cat	2.0000	3	no
2	snake	0.5000	2	no

ascendente

```
# ascendente
data.sort_values(by=['visits'])
```

	animal	age	visits	priority
0	cat	2.5000	1	yes
6	snake	4.5000	1	no
7	cat	3.4375	1	yes
9	dog	3.0000	1	no
2	snake	0.5000	2	no
4	dog	5.0000	2	no
8	dog	7.0000	2	no
1	cat	3.0000	3	yes
3	dog	3.4375	3	yes
5	cat	2.0000	3	no

17. Imprimimos los datos para la siguiente instrucción y ver el cambio realizado.

```
# Inciso m
print(data)
```

```
  animal     age  visits priority
0    cat  2.5000       1      yes
1    cat  3.0000       3      yes
2  snake  0.5000       2       no
3    dog  3.4375       3      yes
4    dog  5.0000       2       no
5    cat  2.0000       3       no
6  snake  4.5000       1       no
7    cat  3.4375       1      yes
8    dog  7.0000       2       no
9    dog  3.0000       1       no
```

Reemplazar sí y no en la columna de prioridad con su valor booleano correspondiente. Utilizamos el .replace múltiple para poder cambiar dichos valores.

```
data["priority"] = data["priority"].replace(["yes", "no"], ["1", "0"])
print(data)
```

```
  animal     age  visits priority
0    cat  2.5000       1        1
1    cat  3.0000       3        1
2  snake  0.5000       2        0
3    dog  3.4375       3        1
4    dog  5.0000       2        0
5    cat  2.0000       3        0
6  snake  4.5000       1        0
7    cat  3.4375       1        1
8    dog  7.0000       2        0
9    dog  3.0000       1        0
```

18. Reemplazar las celdas de la columna de animales por su respectivo nombre en español. Volvemos a utilizar el .replace múltiple para lograr realizar esta instrucción.

```
newName = data['animal'].replace(['cat', 'dog', 'snake'],['gato', 'perro', 'serpiente'])
newName
```

```
0         gato
1         gato
2    serpiente
3        perro
4        perro
5         gato
6    serpiente
7         gato
8        perro
9        perro
Name: animal, dtype: object
```

INSTITUTO POLITÉCNICO NACIONAL

ESCUELA SUPERIOR DE CÓMPUTO

LICENCIATURA EN CIENCIA DE DATOS

UNIDAD DE APRENDIZAJE

DESARROLLO DE APLICACIONES PARA ANÁLISIS DE DATOS

PRÁCTICA 3

NOMBRE DE LOS ALUMNOS:

DE LUNA OCAMPO YANINA

PROFESOR:

ITURIEL FLORES

GRUPO:

4CDM1

FECHA:

04/05/2022

En esta práctica aplicaremos lo aprendido acerca de ReGex, viéndolo en diferentes puntos e instrucciones.

Los pasos aplicados fueron los siguientes:

1. Utilizando el txt dado en la práctica, lo mandamos llamar y le ponemos la función .read() para que, como sería en su traducción, leerlo. Añadiendo el split, para convertir la cadena en lista.

```
r = open("C:/Users/yanin/Desktop/Práctica3/DAAD P3/expenses.txt")

arch = r.read()
arch = arch.split('\n')
```

2. Procedemos a declarar una lista para que se añadan todas las sentencias que se imprimirán.

```
lista = []
for i in arch:
    lista.append(i)
```

3. Empezamos buscando los patrones que coincidan con una expresión regular.

```
# a)
pat = r'D'
```

Visualizamos los resultados, confirmando que se cumple el patrón solicitado.

```
import re

for line in lista:
    if re.search(pat, line) != None:
        print(line)
```

```
Amount:Category:Date:Description
142.12:meal:20170226:host dinner with ABC clients, Al, Bob, Cy, Dave, Ellie
6.53:meal:20170302:Dunkin Donuts, drive to Big Inc. near DC
1247.49:supply:20170306:Dell 7000 laptop/workstation
6.99:supply:20170306:HDMI cable
212.06:util:20170308:Duquesne Light
```

4. Empezamos buscando los patrones que coincidan con una expresión regular.

```
# b)
pat = r'\''
```

Visualizamos los resultados, confirmando que se cumple el patrón solicitado.

```
import re

for line in lista:
    if re.search(pat, line) != None:
        print(line)
```

```
33.07:meal:20170303:dinner, Uncle Julio's
32.27:meal:20170317:lunch at Clyde's with Fred and Gina, Big Inc.
```

5. Empezamos buscando los patrones que coincidan con una expresión regular.

```
# c)
pat = r'\"'
```

Visualizamos los resultados, confirmando que se cumple el patrón solicitado.

```
import re

for line in lista:
    if re.search(pat, line) != None:
        print(line)
```

```
79.99:supply:20170227:spare 20" monitor
```

6. Empezamos buscando los patrones que coincidan con una expresión regular.

```
# d. Agregue un patrón que despliegues líneas que comiencen con el número 7 (siete).
pat = r'^7'
```

Visualizamos los resultados, confirmando que se cumple el patrón solicitado.

```
import re

for line in lista:
    if re.search(pat, line) != None:
        print(line)
```

```
79.81:meal:20170222:lunch with ABC Corp. clients Al, Bob, and Cy
7.59:supply:20170227:Python book (used)
79.99:supply:20170227:spare 20" monitor
```

7. Empezamos buscando los patrones que coincidan con una expresión regular.

```
# e. Agregue un patrón que despliegue líneas que terminen con una “r” o una “t”.
pat = r'[r|t]$'
```

Visualizamos los resultados, confirmando que se cumple el patrón solicitado.

```
import re

for line in lista:
    if re.search(pat, line) != None:
        print(line)
```

```
23.25:meal:20170223:dinner at Logan Airport
79.99:supply:20170227:spare 20" monitor
212.06:util:20170308:Duquesne Light
```

8. Empezamos buscando los patrones que coincidan con una expresión regular.

```
# f. Agregue un patrón que despliegue líneas que contengan un punto "."
pat = r'\.'
```

Visualizamos los resultados, confirmando que se cumple el patrón solicitado.

```
import re

for line in lista:
    if re.search(pat, line) != None:
        print(line)
```

```
5.25:supply:20170222:box of staples
79.81:meal:20170222:lunch with ABC Corp. clients Al, Bob, and Cy
43.00:travel:20170222:cab back to office
383.75:travel:20170223:flight to Boston, to visit ABC Corp.
55.00:travel:20170223:cab to ABC Corp. in Cambridge, MA
23.25:meal:20170223:dinner at Logan Airport
318.47:supply:20170224:paper, toner, pens, paperclips, tape
142.12:meal:20170226:host dinner with ABC clients, Al, Bob, Cy, Dave, Ellie
303.94:util:20170227:Peoples Gas
121.07:util:20170227:Verizon Wireless
7.59:supply:20170227:Python book (used)
79.99:supply:20170227:spare 20" monitor
49.86:supply:20170228:Stoch Cal for Finance II
6.53:meal:20170302:Dunkin Donuts, drive to Big Inc. near DC
127.23:meal:20170302:dinner, Tavern64
33.07:meal:20170303:dinner, Uncle Julio's
86.00:travel:20170304:mileage, drive to/from Big Inc., Reston, VA
22.00:travel:20170304:tolls
378.81:travel:20170304:Hyatt Hotel, Reston VA, for Big Inc. meeting
1247.49:supply:20170306:Dell 7000 laptop/workstation
6.99:supply:20170306:HDMI cable
212.06:util:20170308:Duquesne Light
23.86:supply:20170309:Practical Guide to Quant Finance Interviews
195.89:supply:20170309:black toner, HP 304A, 2-pack
86.00:travel:20170317:mileage, drive to/from Big Inc., Reston, VA
32.27:meal:20170317:lunch at Clyde's with Fred and Gina, Big Inc.
22.00:travel:20170317:tolls
119.56:util:20170319:Verizon Wireless
284.23:util:20170323:Peoples Gas
8.98:supply:20170325:Flair pens
```

9. Empezamos buscando los patrones que coincidan con una expresión regular.

```
# g. Agregue un patrón que despliegues líneas que contengan una "r" seguida por una "g". (La "r" y la "g" no necesariamente
#tienen que estar en posiciones consecutivas).
pat = r'r.*g'
```

Visualizamos los resultados, confirmando que se cumple el patrón solicitado.

```
import re

for line in lista:
    if re.search(pat, line) != None:
        print(line)
```

```
383.75:travel:20170223:flight to Boston, to visit ABC Corp.
55.00:travel:20170223:cab to ABC Corp. in Cambridge, MA
23.25:meal:20170223:dinner at Logan Airport
6.53:meal:20170302:Dunkin Donuts, drive to Big Inc. near DC
86.00:travel:20170304:mileage, drive to/from Big Inc., Reston, VA
378.81:travel:20170304:Hyatt Hotel, Reston VA, for Big Inc. meeting
86.00:travel:20170317:mileage, drive to/from Big Inc., Reston, VA
32.27:meal:20170317:lunch at Clyde's with Fred and Gina, Big Inc.
```

10. Empezamos buscando los patrones que coincidan con una expresión regular.

```
# h. Agregue un patrón que despliegue líneas que contengan dos caracteres consecutivos en mayúscula, por ejemplo: AA, DF, LM,
# YW, …, etcétera.
pat = r'.*[A-Z]{2}.*'
```

Visualizamos los resultados, confirmando que se cumple el patrón solicitado.

```
import re

for line in lista:
    if re.search(pat, line) != None:
        print(line)
```

```
79.81:meal:20170222:lunch with ABC Corp. clients Al, Bob, and Cy
383.75:travel:20170223:flight to Boston, to visit ABC Corp.
55.00:travel:20170223:cab to ABC Corp. in Cambridge, MA
142.12:meal:20170226:host dinner with ABC clients, Al, Bob, Cy, Dave, Ellie
49.86:supply:20170228:Stoch Cal for Finance II
6.53:meal:20170302:Dunkin Donuts, drive to Big Inc. near DC
86.00:travel:20170304:mileage, drive to/from Big Inc., Reston, VA
378.81:travel:20170304:Hyatt Hotel, Reston VA, for Big Inc. meeting
6.99:supply:20170306:HDMI cable
195.89:supply:20170309:black toner, HP 304A, 2-pack
86.00:travel:20170317:mileage, drive to/from Big Inc., Reston, VA
```

11. Empezamos buscando los patrones que coincidan con una expresión regular.

```
# i. Agregue un patrón que despliegue líneas que contengan una "," (coma).
pat = r'.*,.*'
```

Visualizamos los resultados, confirmando que se cumple el patrón solicitado.

```
import re

for line in lista:
    if re.search(pat, line) != None:
        print(line)
```

```
79.81:meal:20170222:lunch with ABC Corp. clients Al, Bob, and Cy
383.75:travel:20170223:flight to Boston, to visit ABC Corp.
55.00:travel:20170223:cab to ABC Corp. in Cambridge, MA
318.47:supply:20170224:paper, toner, pens, paperclips, tape
142.12:meal:20170226:host dinner with ABC clients, Al, Bob, Cy, Dave, Ellie
6.53:meal:20170302:Dunkin Donuts, drive to Big Inc. near DC
127.23:meal:20170302:dinner, Tavern64
33.07:meal:20170303:dinner, Uncle Julio's
86.00:travel:20170304:mileage, drive to/from Big Inc., Reston, VA
378.81:travel:20170304:Hyatt Hotel, Reston VA, for Big Inc. meeting
195.89:supply:20170309:black toner, HP 304A, 2-pack
86.00:travel:20170317:mileage, drive to/from Big Inc., Reston, VA
32.27:meal:20170317:lunch at Clyde's with Fred and Gina, Big Inc.
```

12. Empezamos buscando los patrones que coincidan con una expresión regular.

```
# j. Agregue un patrón que despliegues líneas que contengan tres o más comas. Las comas no necesariamente tienen que ser
# consecutivas.
pat = r'\,.*\,.*\,'
```

Visualizamos los resultados, confirmando que se cumple el patrón solicitado.

```
import re

for line in lista:
    if re.search(pat, line) != None:
        print(line)
```

```
318.47:supply:20170224:paper, toner, pens, paperclips, tape
142.12:meal:20170226:host dinner with ABC clients, Al, Bob, Cy, Dave, Ellie
86.00:travel:20170304:mileage, drive to/from Big Inc., Reston, VA
86.00:travel:20170317:mileage, drive to/from Big Inc., Reston, VA
```

13. Empezamos buscando los patrones que coincidan con una expresión regular.

```
# k. Agregue un patrón que despliegues líneas que NO contengan los caracteres v, w, x, y, o z.
pat = r'^((?![v-z]).)*$'
```

Visualizamos los resultados, confirmando que se cumple el patrón solicitado.

```
import re

for line in lista:
    if re.search(pat, line) != None:
        print(line)
```

```
23.25:meal:20170223:dinner at Logan Airport
303.94:util:20170227:Peoples Gas
33.07:meal:20170303:dinner, Uncle Julio's
212.06:util:20170308:Duquesne Light
284.23:util:20170323:Peoples Gas
```

14. Empezamos buscando los patrones que coincidan con una expresión regular.
    No me salió este pero hice dos intentos, de los cuales salieron un tanto similares a la instrucción pedida.

```
# PRENDIENTE
# l. Agregue un patrón que despliegue líneas que contenga cantidades de dinero entre 0.00 and 99.99. Trate de hacer la
# expresión regular lo más compacta posible
pat = r'[0-9]{0,2}\.[0-9]{2}'
```

Visualizamos los resultados, confirmando que se cumple el patrón solicitado.

```
5.25:supply:20170222:box of staples
79.81:meal:20170222:lunch with ABC Corp. clients Al, Bob, and Cy
43.00:travel:20170222:cab back to office
383.75:travel:20170223:flight to Boston, to visit ABC Corp.
55.00:travel:20170223:cab to ABC Corp. in Cambridge, MA
23.25:meal:20170223:dinner at Logan Airport
318.47:supply:20170224:paper, toner, pens, paperclips, tape
142.12:meal:20170226:host dinner with ABC clients, Al, Bob, Cy, Dave, Ellie
303.94:util:20170227:Peoples Gas
121.07:util:20170227:Verizon Wireless
7.59:supply:20170227:Python book (used)
79.99:supply:20170227:spare 20" monitor
49.86:supply:20170228:Stoch Cal for Finance II
6.53:meal:20170302:Dunkin Donuts, drive to Big Inc. near DC
127.23:meal:20170302:dinner, Tavern64
33.07:meal:20170303:dinner, Uncle Julio's
86.00:travel:20170304:mileage, drive to/from Big Inc., Reston, VA
22.00:travel:20170304:tolls
378.81:travel:20170304:Hyatt Hotel, Reston VA, for Big Inc. meeting
1247.49:supply:20170306:Dell 7000 laptop/workstation
6.99:supply:20170306:HDMI cable
212.06:util:20170308:Duquesne Light
23.86:supply:20170309:Practical Guide to Quant Finance Interviews
195.89:supply:20170309:black toner, HP 304A, 2-pack
86.00:travel:20170317:mileage, drive to/from Big Inc., Reston, VA
32.27:meal:20170317:lunch at Clyde's with Fred and Gina, Big Inc.

119.56:util:20170319:Verizon Wireless
284.23:util:20170323:Peoples Gas
8.98:supply:20170325:Flair pens
```

15. Empezamos buscando los patrones que coincidan con una expresión regular.

```
# m. Agregue un patrón que despliegue líneas que contengas exactamente tres comas.
pat = r'^[^,]+,[^,]+,[^,]+,[^,]+$'
```

Visualizamos los resultados, confirmando que se cumple el patrón solicitado.

```
import re

for line in lista:
    if re.search(pat, line) != None:
        print(line)
```

```
86.00:travel:20170304:mileage, drive to/from Big Inc., Reston, VA
86.00:travel:20170317:mileage, drive to/from Big Inc., Reston, VA
```

16. Empezamos buscando los patrones que coincidan con una expresión regular.

```
# n. Agregue un patrón que despliegue líneas que contengan el carácter "(". Es decir, un paréntesis de apertura.
pat = r'\('
```

Visualizamos los resultados, confirmando que se cumple el patrón solicitado.

```
import re

for line in lista:
    if re.search(pat, line) != None:
        print(line)
```

```
7.59:supply:20170227:Python book (used)
```

17. Empezamos buscando los patrones que coincidan con una expresión regular.

```
# o. Agregue un patrón que despliegue líneas que describen comidas que cuesten al menos 100.00.
pat = r'^([1-9][0-9][0-9].[0-9]{2})'
```

Visualizamos los resultados, confirmando que se cumple el patrón solicitado.

```
import re

for line in lista:
    if re.search(pat, line) != None:
        print(line)
```

```
383.75:travel:20170223:flight to Boston, to visit ABC Corp.
318.47:supply:20170224:paper, toner, pens, paperclips, tape
142.12:meal:20170226:host dinner with ABC clients, Al, Bob, Cy, Dave, Ellie
303.94:util:20170227:Peoples Gas
121.07:util:20170227:Verizon Wireless
127.23:meal:20170302:dinner, Tavern64
378.81:travel:20170304:Hyatt Hotel, Reston VA, for Big Inc. meeting
212.06:util:20170308:Duquesne Light
195.89:supply:20170309:black toner, HP 304A, 2-pack
119.56:util:20170319:Verizon Wireless
284.23:util:20170323:Peoples Gas
```

18. Empezamos buscando los patrones que coincidan con una expresión regular.

```
# p. Agregue un patrón que despliegue líneas que tengan una categoría de gastos compuestas por exactamente cuatro caracteres
# (El patrón debe funcionar inclusive si más líneas son agregadas al archivo, con nuevas categorías que no han sido definidas.
pat = r'.*supply.*[a-zA-Z]{4}'
```

Visualizamos los resultados, confirmando que se cumple el patrón solicitado.

```
5.25:supply:20170222:box of staples
318.47:supply:20170224:paper, toner, pens, paperclips, tape
7.59:supply:20170227:Python book (used)
79.99:supply:20170227:spare 20" monitor
49.86:supply:20170228:Stoch Cal for Finance II
1247.49:supply:20170306:Dell 7000 laptop/workstation
6.99:supply:20170306:HDMI cable
23.86:supply:20170309:Practical Guide to Quant Finance Interviews
195.89:supply:20170309:black toner, HP 304A, 2-pack
8.98:supply:20170325:Flair pens
```

19. Empezamos buscando los patrones que coincidan con una expresión regular.

```
# q. Agregue un patrón que despliegue líneas que para gastos ocurridos en marzo.
pat = r'.*201703'
```

Visualizamos los resultados, confirmando que se cumple el patrón solicitado.

```
import re

for line in lista:
    if re.search(pat, line) != None:
        print(line)
```

```
6.53:meal:20170302:Dunkin Donuts, drive to Big Inc. near DC
127.23:meal:20170302:dinner, Tavern64
33.07:meal:20170303:dinner, Uncle Julio's
86.00:travel:20170304:mileage, drive to/from Big Inc., Reston, VA
22.00:travel:20170304:tolls
378.81:travel:20170304:Hyatt Hotel, Reston VA, for Big Inc. meeting
1247.49:supply:20170306:Dell 7000 laptop/workstation
6.99:supply:20170306:HDMI cable
212.06:util:20170308:Duquesne Light
23.86:supply:20170309:Practical Guide to Quant Finance Interviews
195.89:supply:20170309:black toner, HP 304A, 2-pack
86.00:travel:20170317:mileage, drive to/from Big Inc., Reston, VA
32.27:meal:20170317:lunch at Clyde's with Fred and Gina, Big Inc.
22.00:travel:20170317:tolls
119.56:util:20170319:Verizon Wireless
284.23:util:20170323:Peoples Gas
8.98:supply:20170325:Flair pens
```

20. Empezamos buscando los patrones que coincidan con una expresión regular.

```
# r. Agregue un patrón que despliegue líneas que contengan una "a", seguida por una "b", seguida por una "c" (pueden haber
# otros caracteres la "a" la "b" y la "c").
pat = r'ABC'
```

Visualizamos los resultados, confirmando que se cumple el patrón solicitado.

```
import re

for line in lista:
    if re.search(pat, line) != None:
        print(line)
```

```
79.81:meal:20170222:lunch with ABC Corp. clients Al, Bob, and Cy
383.75:travel:20170223:flight to Boston, to visit ABC Corp.
55.00:travel:20170223:cab to ABC Corp. in Cambridge, MA
142.12:meal:20170226:host dinner with ABC clients, Al, Bob, Cy, Dave, Ellie
```

21. Empezamos buscando los patrones que coincidan con una expresión regular.

```
# s. Agregue un patrón que despliegue líneas que contengan secuencias de dos caracteres, seguidas por la MISMA secuencia de dos
# caracteres dos veces. Es decir, cada línea que coincide con la expresión regular debe contener al menos una secuencia de dos
# caracteres repetida al menos tres veces.
pat = r'(.*[a-z]{2}.*){3}'
```

Visualizamos los resultados, confirmando que se cumple el patrón solicitado.

```
Amount:Category:Date:Description
5.25:supply:20170222:box of staples
79.81:meal:20170222:lunch with ABC Corp. clients Al, Bob, and Cy
43.00:travel:20170222:cab back to office
383.75:travel:20170223:flight to Boston, to visit ABC Corp.
55.00:travel:20170223:cab to ABC Corp. in Cambridge, MA
23.25:meal:20170223:dinner at Logan Airport
318.47:supply:20170224:paper, toner, pens, paperclips, tape
142.12:meal:20170226:host dinner with ABC clients, Al, Bob, Cy, Dave, Ellie
303.94:util:20170227:Peoples Gas
121.07:util:20170227:Verizon Wireless
7.59:supply:20170227:Python book (used)
79.99:supply:20170227:spare 20" monitor
49.86:supply:20170228:Stoch Cal for Finance II
6.53:meal:20170302:Dunkin Donuts, drive to Big Inc. near DC
127.23:meal:20170302:dinner, Tavern64
33.07:meal:20170303:dinner, Uncle Julio's
86.00:travel:20170304:mileage, drive to/from Big Inc., Reston, VA
22.00:travel:20170304:tolls
378.81:travel:20170304:Hyatt Hotel, Reston VA, for Big Inc. meeting
1247.49:supply:20170306:Dell 7000 laptop/workstation
6.99:supply:20170306:HDMI cable
212.06:util:20170308:Duquesne Light
23.86:supply:20170309:Practical Guide to Quant Finance Interviews
195.89:supply:20170309:black toner, HP 304A, 2-pack
86.00:travel:20170317:mileage, drive to/from Big Inc., Reston, VA
32.27:meal:20170317:lunch at Clyde's with Fred and Gina, Big Inc.
22.00:travel:20170317:tolls
119.56:util:20170319:Verizon Wireless
284.23:util:20170323:Peoples Gas
8.98:supply:20170325:Flair pens
```

22. Empezamos buscando los patrones que coincidan con una expresión regular.

```
# t. Agregue un patrón que despliegue líneas que contengan en la descripción de gastos una "a" minúscula y un digito entre 0 y 9
# en cualquier orden. Es decir, el carácter "a" puede aparecer antes o después del digito.
pat = r'.*supply.*((a.*[0-9])|([0-9].*a))'
```

Visualizamos los resultados, confirmando que se cumple el patrón solicitado.

```
import re

for line in lista:
    if re.search(pat, line) != None:
        print(line)
```

```
5.25:supply:20170222:box of staples
318.47:supply:20170224:paper, toner, pens, paperclips, tape
79.99:supply:20170227:spare 20" monitor
49.86:supply:20170228:Stoch Cal for Finance II
1247.49:supply:20170306:Dell 7000 laptop/workstation
6.99:supply:20170306:HDMI cable
23.86:supply:20170309:Practical Guide to Quant Finance Interviews
195.89:supply:20170309:black toner, HP 304A, 2-pack
8.98:supply:20170325:Flair pens
```

23. Empezamos buscando los patrones que coincidan con una expresión regular.

```
# u. Agregue un patrón que despliegues líneas que NO contengan letras mayúsculas.
pat = r'(?!.*[A-Z])^.*$'
```

Visualizamos los resultados, confirmando que se cumple el patrón solicitado.

```
import re

for line in lista:
    if re.search(pat, line) != None:
        print(line)
```

```
5.25:supply:20170222:box of staples
43.00:travel:20170222:cab back to office
318.47:supply:20170224:paper, toner, pens, paperclips, tape
79.99:supply:20170227:spare 20" monitor
22.00:travel:20170304:tolls
22.00:travel:20170317:tolls
```

24. Empezamos buscando los patrones que coincidan con una expresión regular.

```
# v. Agregue un patrón que despliegue líneas que contengan el carácter "d", posiblemente seguido de otros caracteres, seguido de
# una "i". Coincidencias incluirían palabras tales como: diver, doily, drip, diplomat, etc.
pat = r'd.*i'
```

Visualizamos los resultados, confirmando que se cumple el patrón solicitado.

```
import re

for line in lista:
    if re.search(pat, line) != None:
        print(line)
```

```
23.25:meal:20170223:dinner at Logan Airport
142.12:meal:20170226:host dinner with ABC clients, Al, Bob, Cy, Dave, Ellie
6.53:meal:20170302:Dunkin Donuts, drive to Big Inc. near DC
127.23:meal:20170302:dinner, Tavern64
33.07:meal:20170303:dinner, Uncle Julio's
86.00:travel:20170304:mileage, drive to/from Big Inc., Reston, VA
23.86:supply:20170309:Practical Guide to Quant Finance Interviews
86.00:travel:20170317:mileage, drive to/from Big Inc., Reston, VA
32.27:meal:20170317:lunch at Clyde's with Fred and Gina, Big Inc.
```