Ejercicios de la Librería Pandas

Ejercicio 1

Escribir un programa que pregunte al usuario por las ventas de un rango de años y muestre por pantalla una serie con los datos de las ventas indexada por los años, antes y después de aplicarles un descuento del 10%.

```
import pandas as pd
import numpy as np


desde = int(input("Ingrese el año de inicio de su registro: "))
hasta = int(input("Ingrese el año de fin de su registro: "))
listado = []

for x in range(desde,hasta+1):
  reg = int(input(f"Ingrese las ventas del año {x}: "))
  listado.append(reg)
serie = pd.Series(np.array(listado)*0.9, index = range(desde,hasta+1))
print("")
print(f"Listado de Ventas desde {desde} hasta {hasta} con un descuento de 10%")
serie
```

```
Ingrese el año de inicio de su registro: 43
Ingrese el año de fin de su registro: 4

Listado de Ventas desde 43 hasta 4 con un descuento de 10%
Series([], dtype: float64)
```

Ejercicio 2

Escribir una función que reciba una diccionario con las notas de los alumnos en curso en un examen y devuelva una serie con la nota mínima, la máxima, media y la desviación típica.

```
notas = {"Juan": 4,"Pepe": 5,"Jacinto": 2,"Maria": 3.4,"Alejandro": 6.7,"Tomas": 3.8}
def estadisticas(notas):
  listado = []
  for x in notas:
    listado.append(notas[x])
```

```
  listado = np.array(listado)

  minima = listado.min()
  maxima = listado.max()
  media = listado.mean()
  desviacion = listado.std()

  serie = pd.Series([minima,maxima,media,desviacion],index=['Mínima',"Máxima","Media","Desvia
  #suma = serie[0].sum()
  #cuenta = serie.value_counts()
  return serie
estadisticas(notas)
```

```
Mínima              2.000000
Máxima              6.700000
Media               4.150000
Desviación Típica   1.446548
dtype: float64
```

Ejercicio 3

Escribir una función que reciba una diccionario con las notas de los alumnos en curso en un examen y devuelva una serie con las notas de los alumnos aprobados ordenadas de mayor a menor.

```
import pandas as pd

def alumnAprobados(notas):
    notas = pd.Series(notas)
    return notas[notas >= 5.0].sort_values(ascending=False)

notas = {'Juan':3.2, 'Maria':6.5, 'Pedro':4.4, 'Ana': 7.0, 'Luis': 5.6}
print(alumnAprobados(notas))
```

```
Ana       7.0
Maria    6.5
Luis      5.6
dtype: float64
```

Ejercicio 4

Escribir programa que genere y muestre por pantalla un DataFrame con los datos de la tabla siguiente:

Mes	Ventas	Gastos

Mes	Ventas	Gastos
Enero	30500	22000
Febrero	35600	23400
Marzo	28300	18100
Abril	33900	20700

```
import pandas as pd

datos = {'Mes':['Enero', 'Febrero', 'Marzo', 'Abril'], 'Ventas':[30500, 35600, 28300, 33900],
contabilidad = pd.DataFrame(datos)
print(contabilidad)
```

```
        Mes  Ventas  Gastos
0    Enero   30500   22000
1  Febrero   35600   23400
2    Marzo   28300   18100
3    Abril   33900   20700
```

Ejercicio 5

Escribir una función que reciba un DataFrame con el formato del ejercicio anterior, una lista de meses, y devuelva el balance (ventas - gastos) total en los meses indicados.

```
import pandas as pd
import random
from IPython.display import display

personas = {
    'Nombre':['Felipe', 'Laura', 'Erick', 'Maria', 'Jorge', 'Rosa'],
    'Apellido':['Bravo', 'Bustamante', 'Garcia', 'Dominguez', 'Ortiz', 'Bravo'],
    'Meses': ['Diciembre', 'Marzo', 'Junio', 'Enero', 'Septiembre', 'Octubre'],
    'Edad': [26,15,36,15,57,27],
    'Gastos': [800000,550000,744000,370000,629000,745596],
    'Ventas': [1456002, 802300, 1232300, 743300, 802300, 1202300],
    }

def calcularBalance(dataFrame):
  indicados = random.choices(personas['Meses'], k = 1)
  dataFrame['Total balance'] = dataFrame.Ventas - dataFrame.Gastos
  print(display(dataFrame[dataFrame.Meses.isin(indicados)]))


infoPersonas = pd.DataFrame(personas)
calcularBalance(infoPersonas)
```

	Nombre	Apellido	Meses	Edad	Gastos	Ventas	Total balance
2	Erick	Garcia	Junio	36	744000	1232300	488300

None

Ejercicio 6

El archivo cotizacion.csv contiene las cotizaciones de las empresas del IBEX35 con las siguientes columnas: Nombre (nombre de la empresa), Final (precio de la acción al cierre de bolsa), Máximo (precio máximo de la acción durante la jornada), Mínimo (precio mínimo de la acción durante la jornada), Volumen (Volumen al cierre de bolsa), Efectivo (capitalización al cierre en miles de euros). Construir una función que construya un DataFrame a partir del un archivo con el formato anterior y devuelva otro DataFrame con el mínimo, el máximo y la media de dada columna.

```
from google.colab import files

uploaded = files.upload()
for fn in uploaded.keys():
  print('User uploaded file "{name}" with length {length} bytes'.format(name=fn, length=len(u
```

Elegir archivos cotizacion.csv

- **cotizacion.csv**(text/csv) - 1552 bytes, last modified: 7/4/2022 - 100% done

```
Saving cotizacion.csv to cotizacion.csv
User uploaded file "cotizacion.csv" with length 1552 bytes
```

```
import pandas as pd

def resumen_cotizaciones(fichero):
    df = pd.read_csv(fichero, sep=';', decimal=',', thousands='.', index_col=0)
    return pd.DataFrame([df.min(), df.max(), df.mean()], index=['Mínimo', 'Máximo', 'Media'])

resumen_cotizaciones('/content/cotizacion.csv')
```

	Final	Máximo	Mínimo	Volumen	Efectivo
Mínimo	1.016500	4.067500	1.016500	1.221000e+03	2343.090
Máximo	19705.000000	19875.000000	19675.000000	3.612969e+07	145765.440
Media	2796.768757	3170.113357	3136.510471	4.252279e+06	31767.778

Ejercicio 7

El fichero titanic.csv contiene información sobre los pasajeros del Titanic. Escribir un programa con los siguientes requisitos:

1. Generar un DataFrame con los datos del archivo.
2. Mostrar por pantalla las dimensiones del DataFrame, el número de datos que contiene, los nombres de sus columnas y filas, los tipos de datos de las columnas, las 10 primeras filas y las 10 últimas filas
3. Mostrar por pantalla los datos del pasajero con identificador 148.
4. Mostrar por pantalla las filas pares del DataFrame.
5. Mostrar por pantalla los nombres de las personas que iban en primera clase ordenadas alfabéticamente.
6. Mostrar por pantalla el porcentaje de personas que sobrevivieron y murieron.
7. Mostrar por pantalla el porcentaje de personas que sobrevivieron en cada clase.
8. Eliminar del DataFrame los pasajeros con edad desconocida.
9. Mostrar por pantalla la edad media de las mujeres que viajaban en cada clase.
10. Añadir una nueva columna booleana para ver si el pasajero era menor de edad o no.
11. Mostrar por pantalla el porcentaje de menores y mayores de edad que sobrevivieron en cada clase.

```
from google.colab import files

uploaded = files.upload()
for fn in uploaded.keys():
    print('User uploaded file "{name}" with length {length} bytes'.format(name=fn, length=ler
```

Elegir archivos Ninguno archivo selec. Upload widget is only available when the cell has been executed in the current browser session. Please rerun this cell to enable.

```
Saving titanic.csv to titanic.csv
User uploaded file "titanic.csv" with length 61194 bytes
```

```
import pandas as pd
from IPython.display import display

titanic = pd.read_csv('/content/titanic.csv', index_col=0)

print(display(titanic))
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
			Futrelle, Mrs						

```
print('Dimension del DataFrame:', titanic.shape, '\n')
print('Número de elementos:', titanic.size, '\n')
print('Nombres de columnas del data frame:', titanic.columns, '\n')
print('Nombres de filas:', titanic.index, '\n' )
print('Tipos de datos:\n', titanic.dtypes, '\n')
print('Primeras 10 filas:\n', titanic.head(10), '\n' )
print('Últimas 10 filas:\n', titanic.tail(10), '\n')
```

```
8                            Palsson, Master. Gosta Leonard    male   2.0
9          Johnson, Mrs. Oscar W (Elisabeth Vilhelmina Berg)  female  27.0
10                        Nasser, Mrs. Nicholas (Adele Achem)  female  14.0

             SibSp  Parch            Ticket     Fare Cabin Embarked
PassengerId
1                1      0         A/5 21171   7.2500   NaN        S
2                1      0          PC 17599  71.2833   C85        C
3                0      0  STON/O2. 3101282   7.9250   NaN        S
4                1      0            113803  53.1000  C123        S
5                0      0            373450   8.0500   NaN        S
6                0      0            330877   8.4583   NaN        Q
7                0      0             17463  51.8625   E46        S
8                3      1            349909  21.0750   NaN        S
9                0      2            347742  11.1333   NaN        S
10               1      0            237736  30.0708   NaN        C

Últimas 10 filas:
              Survived  Pclass                                          Name  \
PassengerId
882                  0       3                              Markun, Mr. Johann
883                  0       3                     Dahlberg, Miss. Gerda Ulrika
884                  0       2                   Banfield, Mr. Frederick James
885                  0       3                          Sutehall, Mr. Henry Jr
886                  0       3            Rice, Mrs. William (Margaret Norton)
887                  0       2                           Montvila, Rev. Juozas
888                  1       1                    Graham, Miss. Margaret Edith
```

```
...                  1       1                Graham, Miss. Margaret Edith
889                  0       3  Johnston, Miss. Catherine Helen "Carrie"
890                  1       1                     Behr, Mr. Karl Howell
891                  0       3                       Dooley, Mr. Patrick

                    Sex   Age  SibSp  Parch            Ticket     Fare Cabin  \
PassengerId
882                male  33.0      0      0            349257   7.8958   NaN
883              female  22.0      0      0              7552  10.5167   NaN
884                male  28.0      0      0  C.A./SOTON 34068  10.5000   NaN
885                male  25.0      0      0   SOTON/OQ 392076   7.0500   NaN
886              female  39.0      0      5            382652  29.1250   NaN
887                male  27.0      0      0            211536  13.0000   NaN
888              female  19.0      0      0            112053  30.0000   B42
889              female   NaN      1      2        W./C. 6607  23.4500   NaN
890                male  26.0      0      0            111369  30.0000  C148
891                male  32.0      0      0            370376   7.7500   NaN

             Embarked
PassengerId
882                 S
883                 S
884                 S
885                 S
886                 Q
887                 S
888                 S
889                 S
890                 C
891                 Q
```

```
titanic.loc[148]
```

```
Survived                                    0
Pclass                                      3
Name        Ford, Miss. Robina Maggie "Ruby"
Sex                                   female
Age                                      9.0
SibSp                                      2
Parch                                      2
Ticket                            W./C. 6608
Fare                                  34.375
Cabin                                    NaN
Embarked                                   S
Name: 148, dtype: object
```

```
titanic.iloc[lambda x: x.index % 2 == 0]
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
6	0	3	Moran, Mr. James	male	NaN	0	0	330877	8.4583

```
titanic.loc[titanic['Pclass']==1].sort_values('Name',ascending=True)
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
731	1	1	Allen, Miss. Elisabeth Walton	female	29.00	0	0	24160	211.3375
306	1	1	Allison, Master. Hudson Trevor	male	0.92	1	2	113781	151.5500
298	0	1	Allison, Miss. Helen Loraine	female	2.00	1	2	113781	151.5500
499	0	1	Allison, Mrs. Hudson J C (Bessie Waldo Daniels)	female	25.00	1	2	113781	151.5500
			Anderson						

```
total = titanic['Survived'].count()
lista_sobrevivientes = titanic[titanic['Survived']==1]
lista_muertos = titanic[titanic['Survived']==0]
si = lista_sobrevivientes['Survived'].count()
```

```
no = lista_muertos['Survived'].count()
print(f"Cantidad de personas que sobrevivieron: {si} de {total}, representando un {round(si/t
print(f"Cantidad de personas que murieron: {no} de {total}, representando un {round(no/total*
```

```
Cantidad de personas que sobrevivieron: 342 de 891, representando un 38.38%
Cantidad de personas que murieron: 549 de 891, representando un 61.62%
```

```
print(titanic.groupby('Pclass')['Survived'].value_counts(normalize=True))
```

```
Pclass  Survived
1       1           0.629630
        0           0.370370
2       0           0.527174
        1           0.472826
3       0           0.757637
        1           0.242363
Name: Survived, dtype: float64
```

```
titanic.dropna(subset=['Age'])
```

PassengerId	Survived	Pclass	Name	Sex	Age	SibSp	Parch	Ticket	Fare
1	0	3	Braund, Mr. Owen Harris	male	22.0	1	0	A/5 21171	7.2500
2	1	1	Cumings, Mrs. John Bradley (Florence Briggs Th...	female	38.0	1	0	PC 17599	71.2833
3	1	3	Heikkinen, Miss. Laina	female	26.0	0	0	STON/O2. 3101282	7.9250
4	1	1	Futrelle, Mrs. Jacques Heath (Lily May Peel)	female	35.0	1	0	113803	53.1000
			Allen, Mr						

```
print(titanic.groupby(['Pclass','Sex'])['Age'].mean().unstack()['female'])
```

```
Pclass
1    34.611765
2    28.722973
```

```
3    21.750000
Name: female, dtype: float64
```

```
titanic['Young'] = titanic['Age'] < 18
```

```
print(titanic.groupby(['Pclass', 'Young'])['Survived'].value_counts(normalize = True) * 100)
```

```
Pclass  Young  Survived
1       False  1           61.274510
               0           38.725490
        True   1           91.666667
               0            8.333333
2       False  0           59.006211
               1           40.993789
        True   1           91.304348
               0            8.695652
3       False  0           78.208232
               1           21.791768
        True   0           62.820513
               1           37.179487
Name: Survived, dtype: float64
```

Ejercicio 8

Los archivos emisiones-2016.csv, emisiones-2017.csv, emisiones-2018.csv y emisiones-2019.csv, contienen datos sobre las emisiones contaminates en la Ciudad Autónoma de Buenos Aires en los años 2016, 2017, 2018 y 2019 respectivamente. Escribir un programa con los siguientes requisitos:

1. Generar un DataFrame con los datos de los cuatro archivos.
2. Filtrar las columnas del DataFrame para quedarse con las columnas ESTACION, MAGNITUD, AÑO, MES y las correspondientes a los días D01, D02, etc.
3. Reestructurar el DataFrame para que los valores de los contaminantes de las columnas de los días aparezcan en una única columna.
4. Añadir una columna con la fecha a partir de la concatenación del año, el mes y el día (usar el módulo datetime).
5. Eliminar las filas con fechas no válidas (utilizar la función isnat del módulo numpy) y ordenar el DataFrame por estaciones, contaminantes y fecha.
6. Mostrar por pantalla las estaciones y los contaminantes disponibles en el DataFrame.
7. Crear una función que reciba una estación, un contaminante y un rango de fechas y devuelva una serie con las emisiones del contaminante dado en la estación y rango de fechas dado.
8. Mostrar un resumen descriptivo (mímino, máximo, media, etc) para cada contaminante.
9. Mostrar un resumen descriptivo para cada contaminante por distritos.
10. Crear una función que reciba una estación y un contaminante y devuelva un resumen descriptivo de las emisiones del contaminante indicado en la estadión indicada.

11. Crear una función que devuelva las emisiones medias mensuales de un contaminante y un año dados para todos las estaciones.
12. Crear un función que reciba una estación de medición y devuelva un DataFrame con las medias mensuales de los distintos tipos de contaminantes.

```
#Siguiendo las reglas del mentor, este ejercicio quedara pendiente para trabajos o tareas pos
```

✓ 0 s completado a las 8:18