Laboratorio de Introducción a Pandas

Completa el siguiente conjunto de ejercicios para consolidar tu conocimiento de los fundamentos de Pandas.

1. Importa Numpy y Pandas y asígnalos los alias `np` y `pd` respectivamente.

```
import pandas as pd
import numpy as np
```

2. Crea una Serie de Pandas que contenga los elementos de la lista a continuación.

```
lst = [5.7, 75.2, 74.4, 84.0, 66.5, 66.3, 55.8, 75.7, 29.1, 43.7]

# Tu código aquí
serie=pd.Series(lst)
serie
```

```
0     5.7
1    75.2
2    74.4
3    84.0
4    66.5
5    66.3
6    55.8
7    75.7
8    29.1
9    43.7
dtype: float64
```

3. Usa la indexación para devolver el tercer valor en la Serie anterior.

Sugerencia: Recuerda que la indexación comienza en 0.

```
# Tu código aquí
serie[2]
```

```
74.4
```

4. Crea un DataFrame de Pandas a partir de la lista de listas a continuación. Cada sublista debe representarse como una fila.

```
b = [[53.1, 95.0, 67.5, 35.0, 78.4],
     [61.3, 40.8, 30.8, 37.8, 87.6],
```

```
     [20.6, 73.2, 44.2, 14.6, 91.8],
     [57.4, 0.1, 96.1, 4.2, 69.5],
     [83.6, 20.5, 85.4, 22.8, 35.9],
     [49.0, 69.0, 0.1, 31.8, 89.1],
     [23.3, 40.7, 95.0, 83.8, 26.9],
     [27.6, 26.4, 53.8, 88.8, 68.5],
     [96.6, 96.4, 53.4, 72.4, 50.1],
     [73.7, 39.0, 43.2, 81.6, 34.7]]

# Tu código aquí
df_b=pd.DataFrame(b)
df_b
```

```
      0     1     2     3     4
0  53.1  95.0  67.5  35.0  78.4
1  61.3  40.8  30.8  37.8  87.6
2  20.6  73.2  44.2  14.6  91.8
3  57.4   0.1  96.1   4.2  69.5
4  83.6  20.5  85.4  22.8  35.9
5  49.0  69.0   0.1  31.8  89.1
6  23.3  40.7  95.0  83.8  26.9
7  27.6  26.4  53.8  88.8  68.5
8  96.6  96.4  53.4  72.4  50.1
9  73.7  39.0  43.2  81.6  34.7
```

5. Modifica el array sin usar funciones como transpose o reshape para generar el DataFrame que se muestra más adelante

```
b = [[53.1, 95.0, 67.5, 35.0, 78.4],
     [61.3, 40.8, 30.8, 37.8, 87.6],
     [20.6, 73.2, 44.2, 14.6, 91.8],
     [57.4, 0.1, 96.1, 4.2, 69.5],
     [83.6, 20.5, 85.4, 22.8, 35.9],
     [49.0, 69.0, 0.1, 31.8, 89.1],
     [23.3, 40.7, 95.0, 83.8, 26.9],
     [27.6, 26.4, 53.8, 88.8, 68.5],
     [96.6, 96.4, 53.4, 72.4, 50.1],
     [73.7, 39.0, 43.2, 81.6, 34.7]]

# Tu código aquí
datos={ "score_" + str(i): b[i] for i in range(len(b))}
df=pd.DataFrame(datos)
df
```

```
   score_0  score_1  score_2  score_3  score_4  score_5  score_6
score_7  \
0     53.1     61.3     20.6     57.4     83.6     49.0     23.3
27.6
1     95.0     40.8     73.2      0.1     20.5     69.0     40.7
26.4
```

```
2     67.5     30.8     44.2     96.1     85.4      0.1     95.0
53.8
3     35.0     37.8     14.6      4.2     22.8     31.8     83.8
88.8
4     78.4     87.6     91.8     69.5     35.9     89.1     26.9
68.5
```

```
   score_8  score_9
0     96.6     73.7
1     96.4     39.0
2     53.4     43.2
3     72.4     81.6
4     50.1     34.7
```

6. Crea un subconjunto de este DataFrame que contenga solo las columnas de Score 1, 3 y 5.

```
# Tu código aquí
df_135=df.iloc[:,[1,3,5]]
df_135
```

```
   score_1  score_3  score_5
0     61.3     57.4     49.0
1     40.8      0.1     69.0
2     30.8     96.1      0.1
3     37.8      4.2     31.8
4     87.6     69.5     89.1
```

7. Del DataFrame original, calcula el valor promedio de Score_3.

```
# Tu código aquí
mean_score_3=df.iloc[:,3].mean()
mean_score_3
```

```
45.459999999999994
```

8. Del DataFrame original, calcula el máximo valor del Score_4.

```
# Tu código aquí
max_score_4=df.iloc[:,4].max()
max_score_4
```

```
85.4
```

9. Del DataFrame original, calcula la media valor del Score_2.

```
# Tu código aquí
median_score_2=df.iloc[:,2].median()
median_score_2
```

```
44.2
```

10. Crea un DataFrame de Pandas a partir del diccionario de pedidos de productos a continuación.

```
orders = {'Description': ['LUNCH BAG APPLE DESIGN',
  'SET OF 60 VINTAGE LEAF CAKE CASES ',
  'RIBBON REEL STRIPES DESIGN ',
  'WORLD WAR 2 GLIDERS ASSTD DESIGNS',
  'PLAYING CARDS JUBILEE UNION JACK',
  'POPCORN HOLDER',
  'BOX OF VINTAGE ALPHABET BLOCKS',
  'PARTY BUNTING',
  'JAZZ HEARTS ADDRESS BOOK',
  'SET OF 4 SANTA PLACE SETTINGS'],
 'Quantity': [1, 24, 1, 2880, 2, 7, 1, 4, 10, 48],
 'UnitPrice': [1.65, 0.55, 1.65, 0.18, 1.25, 0.85, 11.95, 4.95, 0.19,
1.25],
 'Revenue': [1.65, 13.2, 1.65, 518.4, 2.5, 5.95, 11.95, 19.8, 1.9,
60.0]}

# Tu código aquí
df_orders=pd.DataFrame(orders)
df_orders
```

	Description	Quantity	UnitPrice	Revenue
0	LUNCH BAG APPLE DESIGN	1	1.65	1.65
1	SET OF 60 VINTAGE LEAF CAKE CASES	24	0.55	13.20
2	RIBBON REEL STRIPES DESIGN	1	1.65	1.65
3	WORLD WAR 2 GLIDERS ASSTD DESIGNS	2880	0.18	518.40
4	PLAYING CARDS JUBILEE UNION JACK	2	1.25	2.50
5	POPCORN HOLDER	7	0.85	5.95
6	BOX OF VINTAGE ALPHABET BLOCKS	1	11.95	11.95
7	PARTY BUNTING	4	4.95	19.80
8	JAZZ HEARTS ADDRESS BOOK	10	0.19	1.90
9	SET OF 4 SANTA PLACE SETTINGS	48	1.25	60.00

11. Calcula la cantidad total pedida y los ingresos generados a partir de estos pedidos.

```
# Tu código aquí
cantidad_total_pedida = df_orders['Quantity'].sum()
ingresos_generados = df_orders['Revenue'].sum()

print("Cantidad total pedida:", cantidad_total_pedida)
print("Ingresos generados:", ingresos_generados)
```

```
Cantidad total pedida: 2978
Ingresos generados: 637.0
```

12. Obten los precios del artículo más caro y del más barato pedidos e imprime la diferencia.

```
# Tu código aquí
precio_articulo_mas_caro = df_orders['UnitPrice'].max()
precio_articulo_mas_barato = df_orders['UnitPrice'].min()

diferencia_precios = precio_articulo_mas_caro -
precio_articulo_mas_barato

print("Precio del artículo más caro:", precio_articulo_mas_caro)
print("Precio del artículo más barato:", precio_articulo_mas_barato)
print("Diferencia de precios:", diferencia_precios)
```

```
Precio del artículo más caro: 11.95
Precio del artículo más barato: 0.18
Diferencia de precios: 11.77
```

Carguemos otro conjunto de datos para más ejercicios

```
# Tu código aquí
admissions = pd.read_csv('../Admission_Predict.csv')
```

Evaluemos el conjunto de datos mirando la función `head`.

```
# Tu código aquí
admissions.head()
```

```
   Serial No.  GRE Score  TOEFL Score  University Rating  SOP  LOR
CGPA  \
0           1        337          118                  4  4.5   4.5
9.65
1           2        316          104                  3  3.0   3.5
8.00
2           3        322          110                  3  3.5   2.5
8.67
3           4        314          103                  2  2.0   3.0
8.21
4           5        330          115                  5  4.5   3.0
9.34

   Research  Chance of Admit
0         1              0.92
1         1              0.72
2         1              0.80
3         0              0.65
4         1              0.90
```

1 - Antes de comenzar a trabajar con este conjunto de datos y evaluar los datos de admisiones de posgrado, verificaremos que no haya datos faltantes en el conjunto de datos. Haz esto en la celda de abajo.

```
# Tu código aquí
admissions.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 385 entries, 0 to 384
Data columns (total 9 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Serial No.         385 non-null    int64  
 1   GRE Score          385 non-null    int64  
 2   TOEFL Score        385 non-null    int64  
 3   University Rating  385 non-null    int64  
 4   SOP                385 non-null    float64
 5   LOR                385 non-null    float64
 6   CGPA               385 non-null    float64
 7   Research           385 non-null    int64  
 8   Chance of Admit    385 non-null    float64
dtypes: float64(4), int64(5)
memory usage: 27.2 KB
```

2 - Curiosamente, hay una columna que identifica de manera única a los solicitantes. Esta columna es la columna de número de serie. En lugar de tener nuestro propio índice, deberíamos hacer de esta columna nuestro índice. Haz esto en la celda de abajo. Mantén la columna en el marco de datos además de hacerla un índice.

```
# Tu código aquí

admissions = admissions.set_index('Serial No.', drop=False)
admissions
```

```
             Serial No.  GRE Score  TOEFL Score  University Rating  SOP  \
Serial No.                                                               
1                     1        337          118                  4  4.5   
2                     2        316          104                  3  3.0   
3                     3        322          110                  3  3.5   
4                     4        314          103                  2  2.0   
5                     5        330          115                  5  4.5   

             LOR   CGPA  Research  Chance of Admit  
Serial No.                                          
1             4.5  9.65         1             0.92  
2             3.5  8.87         1             0.76  
3             2.5  8.56         1             0.72  
4             3.0  8.43         0             0.80  
5             4.5  9.45         1             0.90  
```

```
3.0
...                 ...        ...          ...                  ...  ...
...
381                 381        324          110                    3  3.5
3.5
382                 382        325          107                    3  3.0
3.5
383                 383        330          116                    4  5.0
4.5
384                 384        312          103                    3  3.5
4.0
385                 385        333          117                    4  5.0
4.0

             CGPA  Research  Chance of Admit
Serial No.
1            9.65         1              0.92
2            8.00         1              0.72
3            8.67         1              0.80
4            8.21         0              0.65
5            9.34         1              0.90
...           ...       ...               ...
381          9.04         1              0.82
382          9.11         1              0.84
383          9.45         1              0.91
384          8.78         0              0.67
385          9.66         1              0.95

[385 rows x 9 columns]
```

Resulta que `GRE Score` y `CGPA` también identifican de manera única los datos. Muestra esto en la celda de abajo.

```
# Tu código aquí
duplicates_gre = admissions.duplicated(subset=['GRE Score'])
duplicates_cgpa = admissions.duplicated(subset=['CGPA'])

if not duplicates_gre.any() or not duplicates_cgpa.any():
    print("GRE Score o CGPA identifican de manera única los datos.")
else:
    print("Hay duplicados en al menos una de las columnas.")
```

```
Hay duplicados en al menos una de las columnas.
```

```
duplicates_gre_cgpa = admissions.duplicated(['GRE Score', 'CGPA'])

if not duplicates_gre_cgpa.any():
    print("GRE Score y CGPA identifican de manera única los datos.")
else:
    print("Hay duplicados basados en GRE Score y CGPA.")
```

```
GRE Score y CGPA identifican de manera única los datos.
```

3 - En esta parte del laboratorio, nos gustaría probar condiciones complejas en todo el conjunto de datos de una vez. Comencemos encontrando el número de filas donde el CGPA es mayor a 9 y el estudiante ha realizado una investigación.

```
# Tu código aquí
admissions_1=admissions[(admissions["CGPA"]>9) &
(admissions["Research"]==1)]
admissions_1.head()
```

```
            Serial No.  GRE Score  TOEFL Score  University Rating  SOP
LOR   \
Serial No.

1                    1        337          118                  4  4.5
4.5
5                    5        330          115                  5  4.5
3.0
11                  11        328          112                  4  4.0
4.5
20                  20        328          116                  5  5.0
5.0
21                  21        334          119                  5  5.0
4.5

            CGPA  Research  Chance of Admit
Serial No.
1           9.65         1             0.92
5           9.34         1             0.90
11          9.10         1             0.78
20          9.50         1             0.94
21          9.70         1             0.95
```

4 - Ahora devuelve todas las filas donde el CGPA es mayor a 9 y la puntuación SOP es menor a 3.5. Encuentra la probabilidad media de admisión para estos solicitantes.

```
# Tu código aquí
admissions_2=admissions[(admissions["CGPA"]>9) &
(admissions["SOP"]<3.5)]
admissions_2
probabilidad_media_admision = admissions_2.iloc[:,8].mean()
print("Probabilidad media de admisión para estos solicitantes:",
probabilidad_media_admision)
```