

Cuaderno demo UT2 - Ejercicios de ampliación



Adrián Yared Armas de la Nuez

Contenido

1. Enunciado.....	3
2. Ejercicios.....	3
2.1 Ejer 1.....	3
2.1.1 Código.....	3
2.1.2 Ejecución.....	3
2.2 Ejer 2.....	3
2.2.1 Código.....	3
2.2.2 Ejecución.....	3
2.3 Ejer 3.....	3
2.3.1 Código.....	3
2.3.2 Ejecución.....	4
2.4 Ejer 4.....	4
2.4.1 Código.....	4
2.4.2 Ejecución.....	4
2.5 Ejer 5.....	5
2.5.1 Código.....	5
2.5.2 Ejecución.....	5
2.6 Ejer 6.....	5
2.6.1 Código.....	5
2.6.2 Ejecución.....	5
2.7 Ejer 7.....	5
2.7.1 Código.....	5
2.7.2 Ejecución.....	6
2.8 Ejer 8.....	6
2.8.1 Código.....	6
2.8.2 Ejecución.....	6
2.9 Ejer 9.....	6
2.9.1 Mostrar los primeros 5 registros.....	6
2.9.2 Ejecución.....	7
2.9.3 Mostrar los últimos 5 registros.....	7
2.9.4 Ejecución.....	7
2.9.5 Mostrar el nombre de las filas (índices).....	7
2.9.6 Ejecución.....	7
2.9.7 Mostrar el nombre de las columnas.....	8
2.9.8 Ejecución.....	8
2.9.9 Obtener estadísticas básicas descriptivas.....	8

2.9.10 Ejecución.....	8
2.9.11 Obtener la transpuesta del DataFrame.....	8
2.9.12 Ejecución.....	8
2.10 Ejer 10.....	9
2.10.1 Código.....	9
2.10.2 Ejecución.....	9
2.11 Ejer 11.....	9
2.11.1 Código.....	9
2.11.2 Ejecución.....	10
2.12 Ejer 12.....	10
2.12.1 Código.....	10
2.12.2 Ejecución.....	11
3. Github y Colab.....	11



1. Enunciado

Utilizando como fuente de partida el cuaderno suministrado por el profesor y titulado Cuaderno demo UT2 - Ejercicios de ampliación

Url:

https://colab.research.google.com/drive/1fJiRWF1sEbMa1ykSpnocz8Dw4_LB-kw9?usp=sharing

Completar los ejercicios propuestos

2. Ejercicios

2.1 Ejer 1

2.1.1 Código

Ejer 1: Mostrar la columna 4
`x[:,3]`

2.1.2 Ejecución

```
✓ [170] # Ejer 1: Mostrar la columna 4
x[:,3]
→ array([ 4.,  8., 12.])
```

2.2 Ejer 2

2.2.1 Código

#Ejer 2: mostrar la fila 1
`x[0,:]`

2.2.2 Ejecución

```
✓ [172] #Ejer 2: mostrar la fila 1
x[0,:]
→ array([1., 2., 3., 4.])
```

2.3 Ejer 3

2.3.1 Código

#Ejer 3: Extraer la submatriz de las filas 1 y 3
`submatriz = x[[0, 2], :]` # Filas 1 y 3 (índices 0 y 2), todas las columnas

```
print(submatriz)
```

2.3.2 Ejecución

```
#Ejer 3: Extraer la submatriz de las filas 1 y 3
submatriz = x[[0, 2], :] # Filas 1 y 3 (índices 0 y 2), todas las columnas
print(submatriz)

[[ 1.  2.  3.  4.]
 [ 9. 10. 11. 12.]]
```

2.4 Ejer 4

2.4.1 Código

#Ejer 4: crear un dataframe con una lista (ficticia) de municipios de Gran canaria y el número de habitantes separados por géneros (Masculino, Femenino, etc...)
import pandas as pd

Crear los datos ficticios

```
datos = {
    'Municipio': ['Las Palmas de Gran Canaria', 'Telde', 'San Bartolomé de Tirajana',
                  'Santa Lucía de Tirajana', 'Arucas', 'Gáldar', 'Ingenio'],
    'Total_Habitantes': [383343, 102647, 53225, 68455, 37927, 24939, 30889],
    'Masculino': [189000, 50500, 26500, 33700, 18300, 12500, 15600],
    'Femenino': [194343, 52147, 26725, 34755, 19627, 12439, 15289],
    'Otros': [343, 147, 75, 55, 27, 39, 89]
}
```

Crear el DataFrame

```
df = pd.DataFrame(datos)
```

Mostrar el DataFrame

```
print(df)
```

2.4.2 Ejecución

	Municipio	Total_Habitantes	Masculino	Femenino	Otros
0	Las Palmas de Gran Canaria	383343	189000	194343	343
1	Telde	102647	50500	52147	147
2	San Bartolomé de Tirajana	53225	26500	26725	75
3	Santa Lucía de Tirajana	68455	33700	34755	55
4	Arucas	37927	18300	19627	27
5	Gáldar	24939	12500	12439	39
6	Ingenio	30889	15600	15289	89

2.5 Ejer 5

2.5.1 Código

#Ejer 5: Mostrar los tipos de datos asociados a cada columna
`print(df.dtypes)`

2.5.2 Ejecución

```
#Ejer 5: Mostrar los tipos de datos asociados a cada columna
print(df.dtypes)
```

Municipio	object
Total_Habitantes	int64
Masculino	int64
Femenino	int64
Otros	int64
dtype:	object

2.6 Ejer 6

2.6.1 Código

#Ejer 6: modificar uno o más valores de una de las columnas
Modificar valores de la columna 'Masculino' para dos municipios específicos
`df.loc[df['Municipio'] == 'Telde', 'Masculino'] = 51000`
`df.loc[df['Municipio'] == 'Arucas', 'Masculino'] = 18500`

Mostrar el DataFrame modificado
`print(df)`

2.6.2 Ejecución

	Municipio	Total_Habitantes	Masculino	Femenino	Otros
0	Las Palmas de Gran Canaria	383343	189000	194343	343
1	Telde	102647	51000	52147	147
2	San Bartolomé de Tirajana	53225	26500	26725	75
3	Santa Lucía de Tirajana	68455	33700	34755	55
4	Arucas	37927	18500	19627	27
5	Gáldar	24939	12500	12439	39
6	Ingenio	30889	15600	15289	89

2.7 Ejer 7

2.7.1 Código

#Ejer 7: Cambiar alguno de los valores de forma directa
Modificar directamente un valor en la columna 'Femenino'
`df.at[6, 'Femenino'] = 16000` # Cambia el valor en la fila 6, columna 'Femenino'

Mostrar el DataFrame modificado

```
print(df)
```

2.7.2 Ejecución

	Municipio	Total_Habitantes	Masculino	Femenino	Otros
0	Las Palmas de Gran Canaria	383343	189000	194343	343
1	Telde	102647	51000	52147	147
2	San Bartolomé de Tirajana	53225	26500	26725	75
3	Santa Lucía de Tirajana	68455	33700	34755	55
4	Arucas	37927	18500	19627	27
5	Gáldar	24939	12500	12439	39
6	Ingenio	30889	15600	16000	89

2.8 Ejer 8

2.8.1 Código

```
#Ejer 8: Obtener estadística basica descriptiva
# Obtener estadística descriptiva del DataFrame
estadisticas = df.describe()
```

```
# Mostrar las estadísticas
print(estadisticas)
```

2.8.2 Ejecución

	Total_Habitantes	Masculino	Femenino	Otros
count	7.000000	7.000000	7.000000	7.000000
mean	100203.571429	49542.857143	50862.285714	110.714286
std	127629.999863	62875.985271	64675.845570	109.753837
min	24939.000000	12500.000000	12439.000000	27.000000
25%	34408.000000	17050.000000	17813.500000	47.000000
50%	53225.000000	26500.000000	26725.000000	75.000000
75%	85551.000000	42350.000000	43451.000000	118.000000
max	383343.000000	189000.000000	194343.000000	343.000000

2.9 Ejer 9

2.9.1 Mostrar los primeros 5 registros

```
# Mostrar los primeros 5 registros
print("Primeros registros:")
print(df_fechas.head())
```

2.9.2 Ejecución

```
Primeros registros:
      col1      col2
2022-09-10 -0.266260 -0.266991
2022-09-11 -2.101022 -0.605660
2022-09-12 -1.083943 -1.098504
2022-09-13  0.595563  1.437030
2022-09-14 -0.121009 -0.996641
```

2.9.3 Mostrar los últimos 5 registros

```
# Mostrar los últimos 5 registros
print("\nÚltimos registros:")
print(df_fechas.tail())
```

2.9.4 Ejecución

```
Últimos registros:
      col1      col2
2022-11-14 -0.966897 -0.265822
2022-11-15  0.069343  0.224174
2022-11-16  1.696729 -1.475117
2022-11-17 -1.092085 -0.199636
2022-11-18 -0.984318 -0.717810
```

2.9.5 Mostrar el nombre de las filas (índices)

```
# Mostrar el nombre de las filas (índices)
print("\nNombres de las filas (índice):")
print(df_fechas.index)
```

2.9.6 Ejecución

```
Nombres de las filas (índice):
DatetimeIndex(['2022-09-10', '2022-09-11', '2022-09-12', '2022-09-13',
               '2022-09-14', '2022-09-15', '2022-09-16', '2022-09-17',
               '2022-09-18', '2022-09-19', '2022-09-20', '2022-09-21',
               '2022-09-22', '2022-09-23', '2022-09-24', '2022-09-25',
               '2022-09-26', '2022-09-27', '2022-09-28', '2022-09-29',
               '2022-09-30', '2022-10-01', '2022-10-02', '2022-10-03',
               '2022-10-04', '2022-10-05', '2022-10-06', '2022-10-07',
               '2022-10-08', '2022-10-09', '2022-10-10', '2022-10-11',
               '2022-10-12', '2022-10-13', '2022-10-14', '2022-10-15',
               '2022-10-16', '2022-10-17', '2022-10-18', '2022-10-19',
               '2022-10-20', '2022-10-21', '2022-10-22', '2022-10-23',
               '2022-10-24', '2022-10-25', '2022-10-26', '2022-10-27',
               '2022-10-28', '2022-10-29', '2022-10-30', '2022-10-31',
               '2022-11-01', '2022-11-02', '2022-11-03', '2022-11-04',
               '2022-11-05', '2022-11-06', '2022-11-07', '2022-11-08',
               '2022-11-09', '2022-11-10', '2022-11-11', '2022-11-12',
               '2022-11-13', '2022-11-14', '2022-11-15', '2022-11-16',
               '2022-11-17', '2022-11-18'],
              dtype='datetime64[ns]', freq='D')
```


2.9.7 Mostrar el nombre de las columnas

```
# Mostrar el nombre de las columnas
print("\nNombres de las columnas:")
print(df_fechas.columns)
```

2.9.8 Ejecución

```
Nombres de las columnas:
Index(['col1', 'col2'], dtype='object')
```

2.9.9 Obtener estadísticas básicas descriptivas

```
# Obtener estadísticas básicas descriptivas
print("\nEstadística básica descriptiva:")
print(df_fechas.describe())
```

2.9.10 Ejecución

```
Estadística básica descriptiva:
      col1      col2
count  70.000000  70.000000
mean    0.211590 -0.151428
std     1.029600  0.960072
min    -2.789635 -2.601681
25%    -0.230504 -0.699308
50%     0.267626 -0.217776
75%     0.756650  0.487699
max     2.844842  1.774463
```

2.9.11 Obtener la transpuesta del DataFrame

```
# Obtener la transpuesta del DataFrame
print("\nTranspuesta del DataFrame:")
print(df_fechas.T)
```

2.9.12 Ejecución

```
Transpuesta del DataFrame:
      2022-09-10  2022-09-11  2022-09-12  2022-09-13  2022-09-14  2022-09-15  \
col1   -0.266260  -2.101022  -1.083943   0.595563  -0.121009   1.024152
col2   -0.266991  -0.605660  -1.098504   1.437030  -0.996641  -0.425613

      2022-09-16  2022-09-17  2022-09-18  2022-09-19  ...  2022-11-09  \
col1    0.458394   0.331155   1.030374  -1.429984  ...   -2.789635
col2   -2.601681  -0.911283  -0.301777   0.470198  ...   -0.391851

      2022-11-10  2022-11-11  2022-11-12  2022-11-13  2022-11-14  2022-11-15  \
col1   -0.361362   1.146963   0.537562  -0.091270  -0.966897   0.069343
col2   -0.354948   1.231045  -0.235917   0.074059  -0.265822   0.224174

      2022-11-16  2022-11-17  2022-11-18
col1    1.696729  -1.092085  -0.984318
col2   -1.475117  -0.199636  -0.717810
```

2.10 Ejer 10

2.10.1 Código

#Ejer 10: Crear una función anónima denominada `mi_funcion_rango`, que pasándole por parámetro la columna "C" de `df` obtenga el rango
de los valores que forman parte de dicha columna

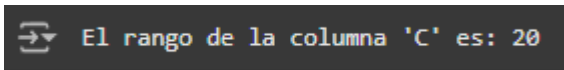
```
# Crear un DataFrame de ejemplo con una columna "C"  
df = pd.DataFrame({  
    "C": [10, 20, 15, 25, 30]  
})
```

```
# Crear la función anónima para calcular el rango  
mi_funcion_rango = lambda columna: columna.max() - columna.min()
```

```
# Aplicar la función a la columna "C"  
rango_columna_c = mi_funcion_rango(df["C"])
```

```
# Mostrar el rango calculado  
print("El rango de la columna 'C' es:", rango_columna_c)
```

2.10.2 Ejecución



2.11 Ejer 11

2.11.1 Código

```
#Ejer 11: Volver a cambiar a CANARIAS  
Datos.iloc[4,0]= 'CANARIAS'  
Datos
```

2.11.2 Ejecución

	Comunidad	Renta	DiferenciaRenta
0	ANDALUCIA	17747	-9.1
1	ARAGON	26512	-7.8
2	ASTURIAS	21149	-9.0
3	BALEARS	22048	-22.7
4	CANARIAS	17448	-18.4
5	CANTABRIA	22096	-9.3
6	CASTILLA Y LEON	23167	-7.0
7	CASTILLA Y LA MANCHA	19369	-7.1
8	CATALUNYA	27812	-10.9
9	VALENCIA	20792	-9.9
10	EXTREMADURA	18301	-5.2
11	GALICIA	21903	-8.1
12	MADRID	32048	-11.1
13	MURCIA	19838	-8.1
14	NAVARRA	29314	-8.5
15	PAIS VASCO	30401	-10.4
16	RIOJA	25714	-8.6
17	CEUTA	19559	-6.7
18	MELILLA	17900	-6.9

2.12 Ejer 12

2.12.1 Código

#Ejer 12: Filtrando los datos en el dataframe Datos, identificar las comunidades autónomas cuya renta

Apartado 1: Filtramos las comunidades con renta entre 26000€ y 28000€

```
comunidades_26000_28000 = Datos[(Datos["Renta"] >= 26000) & (Datos["Renta"] <= 28000)]
```

```
print("Comunidades autónomas con renta entre 26000€ y 28000€:")
```

```
print(comunidades_26000_28000[["Comunidad", "Renta"]])
```

Apartado 2: Filtramos las comunidades con renta superior a 30000€

```
comunidades_superior_30000 = Datos[Datos["Renta"] > 30000]
```

```
print("Comunidades autónomas con renta superior a 30000€:")
```

```
print(comunidades_superior_30000[["Comunidad", "Renta"]])
```

2.12.2 Ejecución

```
Comunidades autónomas con renta entre 26000€ y 28000€:
  Comunidad Renta
1   ARAGON  26512
8  CATALUNYA 27812
Comunidades autónomas con renta superior a 30000€:
  Comunidad Renta
12  MADRID  32048
15  PAIS VASCO 30401
```

3. Github y Colab

