

# Librerías Python

```
import numpy as np
import timeit
import pandas as pd
import matplotlib.pyplot as plt

#Numpy
array = np.random.randint(0, 100, size=10) # Generar un array de 10
números aleatorios entre 0 y 100
print("Lista generada:", array) # Mostrar el array generado
print("Media:", np.mean(array)) # Calcular y mostrar la media
print("Desviación estándar:", np.std(array)) # Calcular y mostrar la
desviación estándar
print("Máximo:", np.max(array)) # Calcular y mostrar el valor máximo
print("Mínimo:", np.min(array)) # Calcular y mostrar el valor mínimo
```

121



122 GUILLERMO FELIPE GARCÍA HERNÁNDEZ

You, 5 minu

PROBLEMS 7

OUTPUT

TERMINAL

...



+

✓



**ValueError: 'skyblue' is not a valid color value.**

PS C:\Users\U\_38010700\Desktop\gui\IA\07-10> py ejercicio.py

Lista generada: [ 7 45 88 80 6 58 63 55 59 82]

Media: 54.3

Desviación estándar: 27.077850727116434

Máximo: 88

Mínimo: 6



```
matriz = np.random.randint(0, 100, size=(3, 3)) # Generar una matriz
3x3 de números aleatorios entre 0 y 100
matriz_invertida = matriz[::-1, ::-1] # Invertir la matriz
print("Matriz original:\n", matriz) # Mostrar la matriz original
print("Matriz invertida:\n", matriz_invertida) # Mostrar la matriz
invertida
```

PROBLEMS 7

OUTPUT

TERMINAL

...

 py 

Mínimo: 0

Matriz original:

[[47 91 48]

[34 17 6]

[20 65 14]]

Matriz invertida:

[[14 65 20]

[ 6 17 34]

[48 91 47]]

```
array1 = np.random.randint(0, 100, size=5) # Generar el primer array de
5 números aleatorios entre 0 y 100
array2 = np.random.randint(0, 100, size=5) # Generar el segundo array
de 5 números aleatorios entre 0 y 100
suma_arrays = array1 + array2 # Sumar los dos arrays elemento a
elemento
resta_arrays = array1 - array2 # Restar los dos arrays elemento a
elemento
producto_arrays = array1 * array2 # Multiplicar los dos arrays elemento
a elemento
producto_punto = np.dot(array1, array2) # Calcular el producto punto de
los dos arrays
print("Array 1:", array1) # Mostrar el primer array
print("Array 2:", array2) # Mostrar el segundo array
print("Suma:", suma_arrays) # Mostrar la suma de los arrays
print("Resta:", resta_arrays) # Mostrar la resta de los arrays
print("Producto:", producto_arrays) # Mostrar el producto de los arrays
print("Producto punto:", producto_punto) # Mostrar el producto punto de
los arrays
```

121

122

GUILLERMO FELIPE GARCÍA HERNÁNDEZ

You,

PROBLEMS 7

OUTPUT

TERMINAL

...

py



[48 91 47]]

Array 1: [45 9 3 14 13]

Array 2: [55 38 85 22 83]

Suma: [100 47 88 36 96]

Resta: [-10 -29 -82 -8 -70]

Producto: [2475 342 255 308 1079]

Producto punto: 4459

```
codigo_listas = """
```

```
lista1= list(range(1_000_000))# Crear una lista de 1,000,000 elementos
```

```
lista2= list(range(1_000_000))# Crear otra lista de 1,000,000 elementos
```

```
suma = [lista1[i] + lista2[i] for i in range(len(lista1))] # Sumar las  
dos listas elemento a elemento usando comprensión de listas
```

```
"""
```

```
tiempo_listas = timeit.timeit(stmt=codigo_listas, number=10) # Medir el  
tiempo de ejecución del código con listas
```

```
print("Tiempo con listas:", tiempo_listas, "segundos") # Mostrar el  
tiempo de ejecución con listas
```

```
codigo_numpy = """
```

```
array_1 = np.arange(1_000_000)# Crear un array de 1,000,000 elementos
```

```
array_2 = np.arange(1_000_000)# Crear otro array de 1,000,000 elementos
```

```
suma = array_1 + array_2 # Sumar los dos arrays elemento a elemento  
usando NumPy
```

```
"""
```

```
tiempo_numpy = timeit.timeit(stmt=codigo_numpy, setup="import numpy as  
np", number=10) # Medir el tiempo de ejecución del código con NumPy
```

```
print("Tiempo con NumPy:", tiempo_numpy, "segundos") # Mostrar el  
tiempo de ejecución con NumPy
```

121

122

GUILLERMO FELIPE GARCÍA HERNÁNDEZ

PROBLEMS 7

OUTPUT

TERMINAL

...

py



Tiempo con listas: 1.1780873000006977 segundos

Tiempo con NumPy: 0.08366960000057588 segundos

```
#Pandas
df = pd.read_csv("Iris.csv") # Cargar el archivo CSV en un DataFrame

print("Primeras filas del DataFrame:\n", df.head()) # Mostrar las
primeras filas del DataFrame
```

122 | GUILLERMO FELIPE GARCÍA HERNÁNDEZ

PROBLEMS 7

OUTPUT

TERMINAL

...



py



Primeras filas del DataFrame:

	Id	SepalLengthCm	...	PetalWidthCm	Species
0	1	5.1	...	0.2	Iris-setosa
1	2	4.9	...	0.2	Iris-setosa
2	3	4.7	...	0.2	Iris-setosa
3	4	4.6	...	0.2	Iris-setosa
4	5	5.0	...	0.2	Iris-setosa

[5 rows x 6 columns]

<class 'pandas.core.frame.DataFrame'>

RangeIndex: 150 entries, 0 to 149

Data columns (total 6 columns):

#	Column	Non-Null Count	Dtype
0	Id	150 non-null	int64
1	SepalLengthCm	150 non-null	float64
2	SepalWidthCm	150 non-null	float64
3	PetalLengthCm	150 non-null	float64
4	PetalWidthCm	150 non-null	float64
5	Species	150 non-null	object

dtypes: float64(4), int64(1), object(1)

memory usage: 7.2+ KB

```
print("Información del DataFrame:\n", df.info()) # Mostrar información
del DataFrame
print("Estadísticas descriptivas:\n", df.describe()) # Mostrar
estadísticas descriptivas del DataFrame
```

dtypes: float64(4), int64(1), object(1)

memory usage: 7.2+ KB

Información del DataFrame:

None

Estadísticas descriptivas:

	Id	SepalLengthCm	...	PetalLengthCm	PetalWidthCm
count	150.000000	150.000000	...	150.000000	150.000000
mean	75.500000	5.843333	...	3.758667	1.198667
std	43.445368	0.828066	...	1.764420	0.763161
min	1.000000	4.300000	...	1.000000	0.100000
25%	38.250000	5.100000	...	1.600000	0.300000
50%	75.500000	5.800000	...	4.350000	1.300000
75%	112.750000	6.400000	...	5.100000	1.800000
max	150.000000	7.900000	...	6.900000	2.500000

[8 rows x 5 columns]

```
Index(['Id', 'SepalLengthCm', 'SepalWidthCm', 'PetalLengthCm', 'PetalWidthCm',
      'Species'],
      dtype='object')
['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']
```

```
print(df.columns) # Mostrar los nombres de las columnas del DataFrame
```

```
print(df["Species"].unique()) # Mostrar los valores únicos en la
columna "Species"
```

```
cond_longitud = df["SepalLengthCm"] > 5 # Condición para longitud del
sépalos mayor a 5 cm
```

```
cond_especie = df["Species"] == "Iris-setosa" # Condición para especie
"Iris-setosa"
```

```
filtro = df[cond_longitud & cond_especie] # Filtrar filas que cumplen
ambas condiciones
```

```
print("Filas que cumplen ambas condiciones:\n", filtro) # Mostrar las
filas que cumplen ambas condiciones
```

PROBLEMS 7

OUTPUT

TERMINAL

...



py



['Iris-setosa' 'Iris-versicolor' 'Iris-virginica']

Filas que cumplen ambas condiciones:

	Id	SepalLengthCm	...	PetalWidthCm	Species
0	1	5.1	...	0.2	Iris-setosa
5	6	5.4	...	0.4	Iris-setosa
10	11	5.4	...	0.2	Iris-setosa
14	15	5.8	...	0.2	Iris-setosa
15	16	5.7	...	0.4	Iris-setosa
16	17	5.4	...	0.4	Iris-setosa
17	18	5.1	...	0.3	Iris-setosa
18	19	5.7	...	0.3	Iris-setosa
19	20	5.1	...	0.3	Iris-setosa
20	21	5.4	...	0.2	Iris-setosa
21	22	5.1	...	0.4	Iris-setosa
23	24	5.1	...	0.5	Iris-setosa
27	28	5.2	...	0.2	Iris-setosa
28	29	5.2	...	0.2	Iris-setosa
31	32	5.4	...	0.4	Iris-setosa
32	33	5.2	...	0.1	Iris-setosa
33	34	5.5	...	0.2	Iris-setosa
36	37	5.5	...	0.2	Iris-setosa
39	40	5.1	...	0.2	Iris-setosa
44	45	5.1	...	0.4	Iris-setosa
46	47	5.1	...	0.2	Iris-setosa
48	49	5.3	...	0.2	Iris-setosa

```
df["petal_area"] = df["PetalLengthCm"] * df["PetalWidthCm"] # Crear una
nueva columna "petal_area" como el producto de "PetalLengthCm" y
"PetalWidthCm"
print("DataFrame con la nueva columna 'petal_area':\n", df.head()) #
Mostrar las primeras filas del DataFrame con la nueva columna
```

122

GUILLERMO FELIPE GARCÍA HERNÁNDEZ

PROBLEMS 7

OUTPUT

TERMINAL



py



[22 rows x 6 columns]

DataFrame con la nueva columna 'petal\_area':

	Id	SepalLengthCm	...	Species	petal_area
0	1	5.1	...	Iris-setosa	0.28
1	2	4.9	...	Iris-setosa	0.28
2	3	4.7	...	Iris-setosa	0.26
3	4	4.6	...	Iris-setosa	0.30
4	5	5.0	...	Iris-setosa	0.28

```
medias_por_especie = df.groupby("Species").mean(numeric_only=True) #  
Calcular la media de las columnas numéricas agrupadas por "Species"  
print("Medias por especie:\n", medias_por_especie.round(2)) # Mostrar  
las medias por especie
```

122

GUILLERMO FELIPE GARCÍA HERNÁNDEZ

PROBLEMS 7

OUTPUT

TERMINAL



py



Medias por especie:



	Id	SepalLengthCm	...	PetalWidthCm	petal_area
Species			...		
Iris-setosa	25.5	5.01	...	0.24	0.36
Iris-versicolor	75.5	5.94	...	1.33	5.72
Iris-setosa	25.5	5.01	...	0.24	0.36
Iris-setosa	25.5	5.01	...	0.24	0.36
Iris-versicolor	75.5	5.94	...	1.33	5.72
Iris-virginica	125.5	6.59	...	2.03	11.30

```
valor_maximo = df["PetalWidthCm"].max() # Encontrar el valor máximo en  
la columna "PetalWidthCm"  
print("Valor máximo de PetalWidthCm:", valor_maximo) # Mostrar el valor  
máximo encontrado  
fila_maxima = df[df["PetalWidthCm"] == valor_maximo] # Filtrar la fila  
que contiene el valor máximo  
especie_maxima = fila_maxima["Species"].values[0] # Obtener la especie  
correspondiente al valor máximo
```



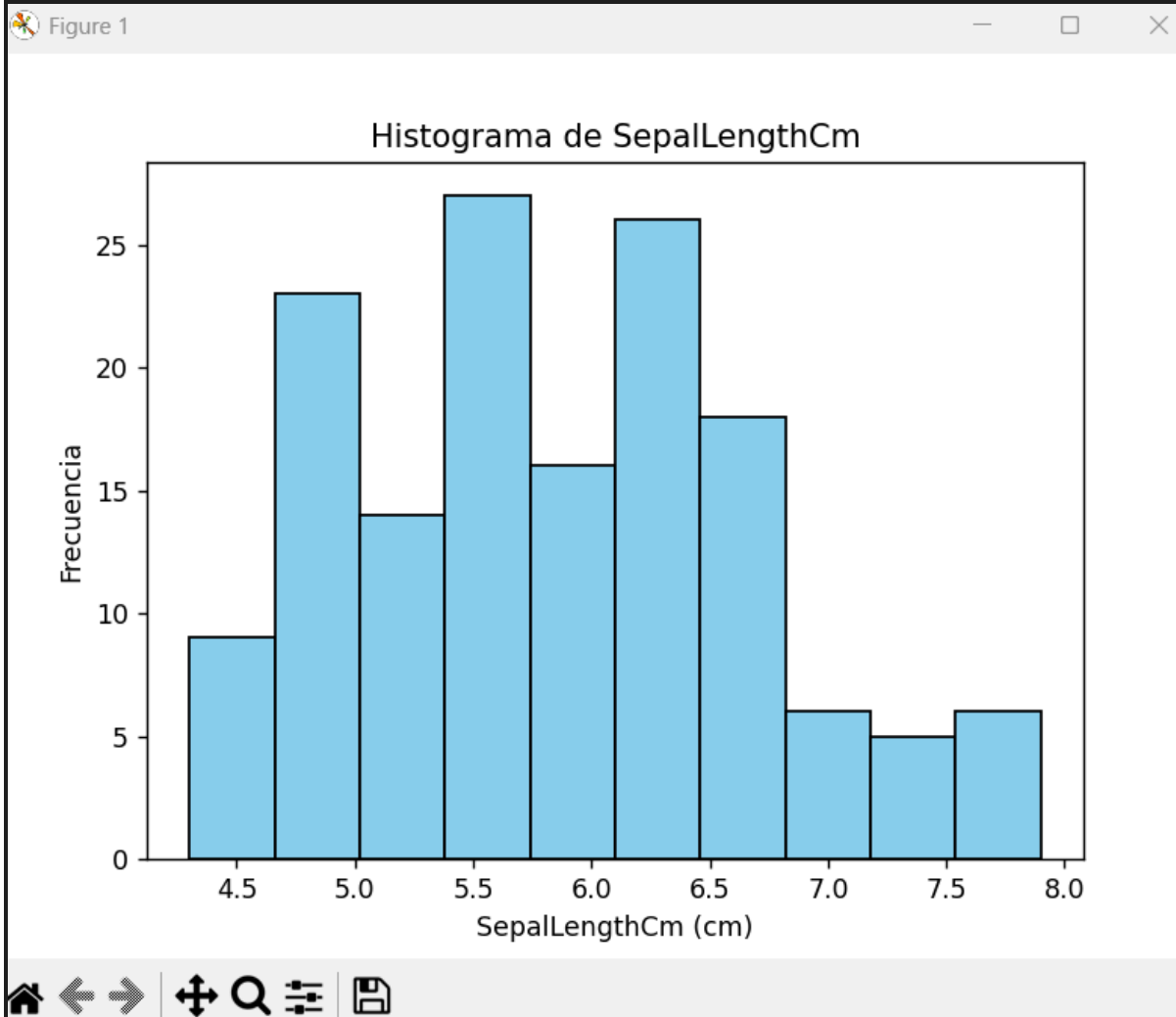
```
print("Especie con el valor máximo de PetalWidthCm:", especie_maxima) #  
Mostrar la especie correspondiente al valor máximo
```

122 | GUILLERMO FELIPE GARCÍA HERNÁNDEZ You, 20 minu

PROBLEMS 7 OUTPUT TERMINAL ...   + v   ...

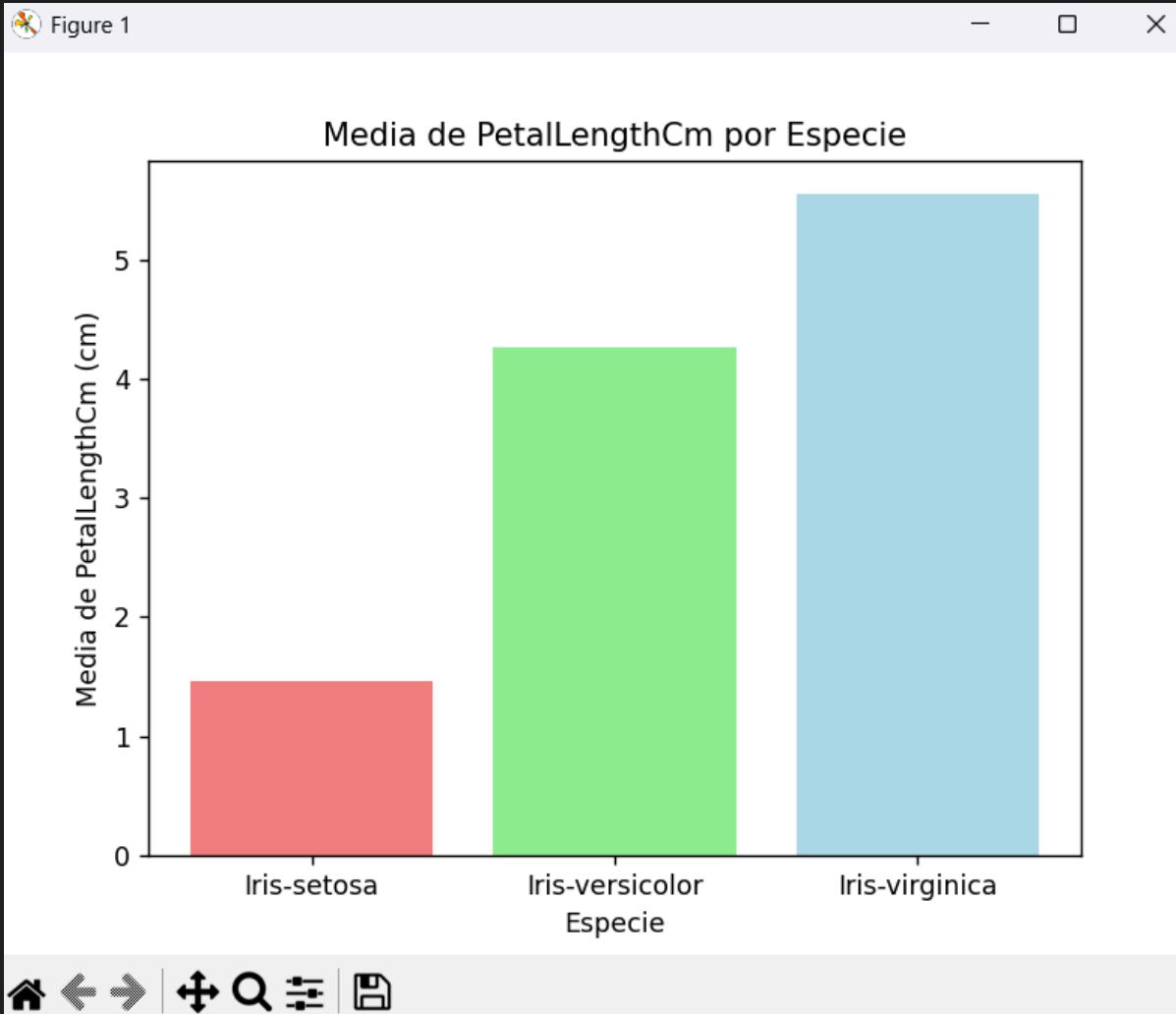
```
Valor máximo de PetalWidthCm: 2.5  
Especie con el valor máximo de PetalWidthCm: Iris-virginica
```

```
#Matplotlib  
plt.hist(df["SepalLengthCm"], bins=10, color='skyblue',  
edgecolor="black") # Crear un histograma de la columna "SepalLengthCm"  
plt.title("Histograma de SepalLengthCm") # Título del histograma  
plt.xlabel("SepalLengthCm (cm)") # Etiqueta del eje x  
plt.ylabel("Frecuencia") # Etiqueta del eje y  
plt.show() # Mostrar el histograma
```



```
medias = df.groupby("Species")["PetalLengthCm"].mean() # Calcular la
media de "PetalLengthCm" agrupada por "Species"
print(medias) # Mostrar las medias calculadas

plt.bar(medias.index, medias.values, color=['lightcoral', 'lightgreen',
'lightblue']) # Crear un gráfico de barras con las medias
plt.title("Media de PetalLengthCm por Especie") # Título del gráfico
plt.xlabel("Especie") # Etiqueta del eje x
plt.ylabel("Media de PetalLengthCm (cm)") # Etiqueta del eje y
plt.show() # Mostrar el gráfico de barras
```



```
especies = df["Species"].unique() # Obtener las especies únicas
colores = ['lightcoral', 'lightgreen', 'lightblue'] # Definir colores
para cada especie
for especie, color in zip(especies, colores): # Iterar sobre cada
especie y su color correspondiente
    subset = df[df["Species"] == especie] # Filtrar el DataFrame por la
especie actual
```

```
plt.scatter(subset["SepalLengthCm"], subset["PetalLengthCm"],  
label=especie, color=color) # Crear un gráfico de dispersión para la  
especie actual  
plt.title("SepalLengthCm vs PetalLengthCm") # Título del gráfico de  
dispersión  
plt.xlabel("SepalLengthCm (cm)") # Etiqueta del eje x  
plt.ylabel("PetalLengthCm (cm)") # Etiqueta del eje y  
plt.legend(title="Especies") # Mostrar la leyenda del gráfico  
plt.show() # Mostrar el gráfico de dispersión
```

