01 PAO25-25 - PYTHON, Data Types

Luis Pilaguano

Examen

In [ ]:
```
# Your imports HERE !!!!
```

01 - 1 pt

- Dado un array de elementos, escribir una función, que reciba dicho array como parámetro de entrada y devuelva el elemento repetido. Usando plain Python, sin ninguna librería.

In [4]:
```
array_src = [4, 8, 0, 3, 6, 2, 1, 5, 3, 9, 7]
expected_rep_element = 3
rep_element = repeated_element(array_src)

assert(expected_rep_element == rep_element)
# muestro el valor repetido de mi array
print("El valor repetido es ", rep_element)
```

```
El valor repetido es: 3
```

In [1]:
```
def repeated_element(array):
    guardar = set()
    # recorro mi array con un for
    for valor in array:
        if valor in guardar:
```

```
            return valor
        guardar.add(valor)
```

02 - 1pt

- Genera una matriz con valores aleatorios de 0 a 100 de dimensiones 16x16
- Calcula los siguentes estadísticos en la matriz anterior, por fila (el resultado de cada uno debe ser un array de dimensiones 1x16)
    - Suma de los elementos de cada fila
    - Media de los elementos de cada fila
    - Valores mínimos de cada fila
    - Valores máximos de cada fila

In [17]:
```
# inicio importando random
import random
# matriz 16 x 16
matriz = [[random.randint(0, 100) for _ in range(16)] for _ in range(16)]
suma = []
media = []
minimo = []
maximo = []
# calculo las operaciones para cada lista
for fila in matriz:
    suma.append(sum(fila))
    media.append(sum(fila) / len(fila))
    minimo.append(min(fila))
    maximo.append(max(fila))
# imprimo los resultados
print("Matriz 16 x 16")
print("----------------------------------------------------------------")
print("")
for fila in matriz:
    print(fila)
print("")
print("----------------------------------------------------------------")
print("")
print("SUMA: ", suma)
print("MEDIA: ", media)
print("MINIMO: ", minimo)
print("MAXIMO: ", maximo)
```

```
Matriz 16 x 16
----------------------------------------------------------------

[95, 76, 69, 20, 17, 80, 96, 70, 28, 44, 69, 42, 36, 72, 79, 59]
[0, 72, 14, 23, 58, 94, 44, 95, 92, 17, 30, 50, 44, 36, 59, 75]
[98, 65, 56, 66, 60, 95, 8, 4, 81, 94, 82, 9, 100, 14, 92, 60]
[60, 18, 73, 81, 6, 10, 16, 8, 34, 95, 20, 95, 80, 48, 66, 93]
[96, 58, 84, 78, 87, 49, 22, 96, 32, 95, 51, 80, 29, 16, 87, 71]
[33, 0, 90, 10, 27, 68, 60, 89, 35, 6, 58, 38, 41, 71, 55, 61]
[53, 28, 85, 42, 47, 15, 96, 44, 56, 42, 20, 27, 5, 96, 58, 80]
[49, 14, 59, 51, 44, 51, 79, 41, 36, 77, 26, 26, 41, 28, 57, 30]
[45, 40, 84, 6, 85, 74, 85, 94, 92, 97, 87, 3, 19, 86, 21, 31]
[16, 83, 95, 9, 14, 66, 76, 62, 44, 58, 20, 61, 10, 85, 12, 89]
[53, 0, 82, 51, 94, 32, 82, 22, 0, 46, 29, 70, 88, 63, 1, 96]
[74, 62, 40, 98, 12, 41, 85, 3, 17, 9, 78, 36, 68, 27, 95, 0]
[32, 16, 83, 81, 17, 75, 42, 8, 88, 51, 54, 59, 97, 73, 39, 0]
[63, 40, 28, 51, 1, 71, 72, 44, 71, 13, 73, 95, 62, 50, 99, 47]
[33, 24, 25, 100, 62, 91, 25, 71, 6, 24, 13, 21, 99, 51, 43, 14]
[69, 33, 71, 46, 48, 30, 31, 32, 3, 19, 79, 45, 46, 26, 88, 67]

----------------------------------------------------------------

SUMA:  [952, 803, 984, 803, 1031, 742, 794, 709, 949, 800, 809, 745, 815, 880, 70
2, 733]
MEDIA:  [59.5, 50.1875, 61.5, 50.1875, 64.4375, 46.375, 49.625, 44.3125, 59.3125,
50.0, 50.5625, 46.5625, 50.9375, 55.0, 43.875, 45.8125]
MINIMO:  [17, 0, 4, 6, 16, 0, 5, 14, 3, 9, 0, 0, 0, 1, 6, 3]
MAXIMO:  [96, 95, 100, 95, 96, 90, 96, 79, 97, 95, 96, 98, 97, 99, 100, 88]
```

03 - 1pt

- Generar un dataframe de 3x4 dimensiones con valores númericos aleatorios [0,10]
- Darle un índice semántico alfanumérico
- Aplicarle una función lambda que obtenga la diferencia entre el máximo y mínimo de cada fila

In [26]:
```python
import pandas as pd
import numpy as np

# Generar DataFrame 3x4 con valores aleatorios entre 0 y 10
data_frame = pd.DataFrame(np.random.randint(0, 11, size=(3, 4)))

# Asignar índice alfanumérico
data_frame.index = ['a1', 'a2', 'a3']

# Calcular diferencia entre máximo y mínimo por fila
data_frame['Rango'] = data_frame.apply(lambda fila: fila.max() - fila.min(), axi

# Mostrar el resultado
print("--------------------------------------------------")
print(data_frame)
```

```
--------------------------------------------------
    0  1  2  3  Rango
a1  4  3  7  7      4
a2  2  5  4  1      4
a3  7  5  1  4      6
```

04 - 1pt

- Crear un array de NumPy de dimensiones 8x5, con números enteros aleatorios en el rango [-100, 100]
- Ejecutar en este orden
    1. Reemplazar los valores de todos aquellos números múltiplos de 5 por 100
    2. Reemplazar los valores de todos aquellos números múltiplos de 3 por Nan
    3. Averiguar cuantos elementos tienen Nan por fila
    4. Reemplazar los valores de Nan por 0

In [32]:
```
import numpy as np

# Crear un array de NumPy de dimensiones 8x5, con números enteros aleatorios en
array = np.random.randint(-100, 101, size=(8, 5)).astype(float)

print("Array inicial: ", array)
# Reemplazar los valores de todos aquellos números múltiplos de 5 por 100
array[array % 5 == 0] = 100
print("números múltiplos de 5 por 100:", array)

# Reemplazar los valores de todos aquellos números múltiplos de 3 por Nan
array[(array % 3 == 0) & (array != 100)] = np.nan
print("múltiplos de 3 por Nan", array)

# Averiguar cuantos elementos tienen Nan por fila
nan_por_fila = np.sum(np.isnan(array), axis=1)
print("Cantidad de NaN por fila:", nan_por_fila)

# Reemplazar los valores de Nan por 0
array[np.isnan(array)] = 0
print("Array final con NaN reemplazado por 0:", array)
```

```
Array inicial:  [[-48.  71.  59.  97.  59.]
 [-33.  82.  83.  22.  44.]
 [-63. -77. -32.  15.  -3.]
 [ 97.  38.  43.  -4. 100.]
 [ 23.  86. -31.  -8. -98.]
 [ 47.  86.  63.  46. -11.]
 [ 94.  46.  47.  -5.  98.]
 [-49.  60.  67.  27. -62.]]
números múltiplos de 5 por 100: [[-48.  71.  59.  97.  59.]
 [-33.  82.  83.  22.  44.]
 [-63. -77. -32. 100.  -3.]
 [ 97.  38.  43.  -4. 100.]
 [ 23.  86. -31.  -8. -98.]
 [ 47.  86.  63.  46. -11.]
 [ 94.  46.  47. 100.  98.]
 [-49. 100.  67.  27. -62.]]
múltiplos de 3 por Nan [[ nan  71.  59.  97.  59.]
 [ nan  82.  83.  22.  44.]
 [ nan -77. -32. 100.  nan]
 [ 97.  38.  43.  -4. 100.]
 [ 23.  86. -31.  -8. -98.]
 [ 47.  86.  nan  46. -11.]
 [ 94.  46.  47. 100.  98.]
 [-49. 100.  67.  nan -62.]]
Cantidad de NaN por fila: [1 1 2 0 0 1 0 1]
Array final con NaN reemplazado por 0: [[  0.  71.  59.  97.  59.]
 [  0.  82.  83.  22.  44.]
 [  0. -77. -32. 100.   0.]
 [ 97.  38.  43.  -4. 100.]
 [ 23.  86. -31.  -8. -98.]
 [ 47.  86.   0.  46. -11.]
 [ 94.  46.  47. 100.  98.]
 [-49. 100.  67.   0. -62.]]
```

05 - 1pt

- Crear una función que reciba una lista de strings y devuelva una serie de pandas, y convierta el primer y último carácter de la palabra de cada elemento de la serie a mayúsculas

In [39]:
```python
languages = ['python', 'php', 'java', 'javascript', 'c++', 'sql']
expected_output = pd.Series(['PythoN', 'PhP', 'JavA', 'JavascripT', 'C++', 'SqL'
output = first_last_uppercase(languages)  # This calls your function first_last_
assert(expected_output.equals(output))  # This will fail if the two lists are di
```

In [38]:
```python
import pandas as pd

def first_last_uppercase(lista):
    # serie de pandas
    serie = pd.Series(lista)
    # .apply()
    return serie.apply(lambda palabra:
#(::::::::::::::::::::::::::::::::::::::::::::::::::)
```

07 - 1pt

- La serie de Fibonacci es una sucesión de números, en la cual cada número es la suma de los dos anteriores. Los dos primeros son siempre 0 y 1.
  - $F_0 = 0$
  - $F_1 = 1$
  - $F_n = F_{n-1} + F_{n-2}$
- Crear un generador infinito de números de Fibonacci.

In [ ]:
```
fibo = [0,1,1,2,3,5,8,13,21,34,55,89,144,233,377,610,987,1597]
g = fibonacci_generator()  # This calls tour function fibonacci_generator
expected_output = [next(g) for n in range(len(fibo))]
assert(fibo == expected_output)  # This will fail if the output is not as expect
```

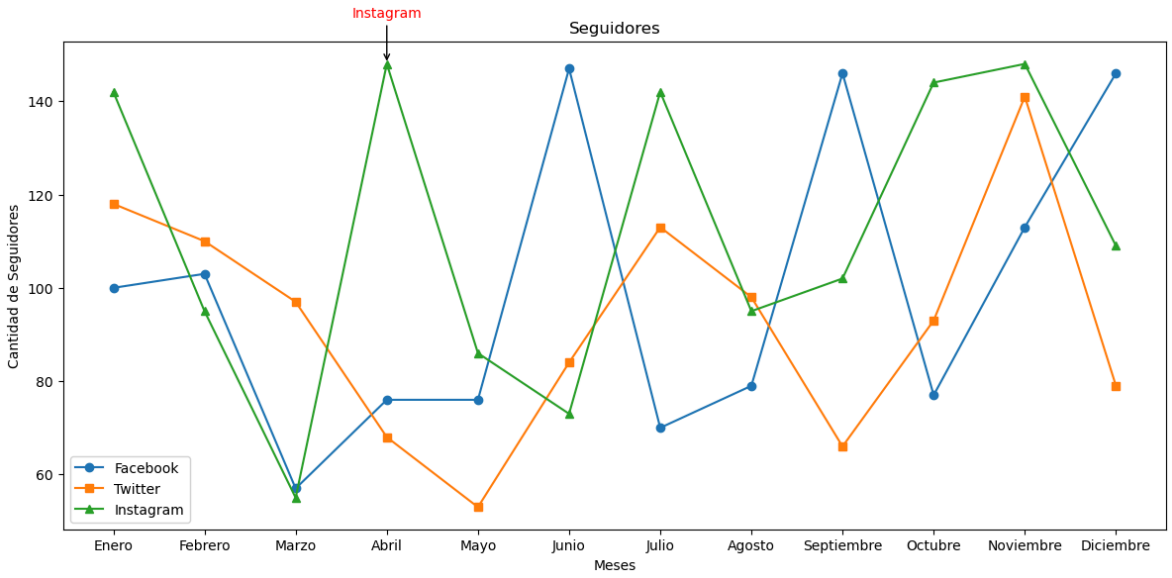
In [ ]:
```
# Your solution HERE !!!!
```

07 - 4pt

- Crear una lista de meses de Enero a Diciembre (eje X)
- Generar datos aleatorios para el eje Y, hacerlo 3 veces distintas (Y0, Y1 e Y2)
- Representar las 3 secuencias de datos aleatorios en una sola figura usando matplotlib
- Añadir una leyenda para poder identificar cada secuencia
- Nombrar las 3 secuencias de la siguiente manera ("Facebook", "Twitter", "Instagram")
- Añadir un título y nombres a los dos ejes
- Añadir una anotación (texto y flecha) a uno de los gráficos

In [35]:
```
import matplotlib.pyplot as plt
import numpy as np

# Crear una lista de meses de Enero a Diciembre (eje X)
meses = ['Enero', 'Febrero', 'Marzo', 'Abril', 'Mayo', 'Junio',
         'Julio', 'Agosto', 'Septiembre', 'Octubre', 'Noviembre', 'Diciembre']

# Generar datos aleatorios para el eje Y, hacerlo 3 veces distintas (Y0, Y1 e Y2

y0 = np.random.randint(50, 150, size=12)
y1 = np.random.randint(50, 150, size=12)
y2 = np.random.randint(50, 150, size=12)

# Representar las 3 secuencias de datos aleatorios en una sola figura usando mat
plt.figure(figsize=(12, 6))
plt.plot(meses, y0, marker='o', label="Facebook")
plt.plot(meses, y1, marker='s', label="Twitter")
plt.plot(meses, y2, marker='^', label="Instagram")

# Añadir una leyenda para poder identificar cada secuencia
plt.legend()

# }+
plt.title("Seguidores")
plt.xlabel("Meses")
plt.ylabel("Cantidad de Seguidores")

# Añadir una anotación (texto y flecha) a uno de los gráficos
```

```
mes_max_instagram = meses[np.argmax(y2)]
valor_max_instagram = np.max(y2)
plt.annotate(
    "Instagram",
    xy=(mes_max_instagram, valor_max_instagram),
    xytext=(mes_max_instagram, valor_max_instagram + 10),
    arrowprops=dict(facecolor='red', arrowstyle='->'),
    ha='center',
    color='red'
)

# graficos
plt.tight_layout()
plt.show()
```



Examen

In [ ]: