Tarea 1

June 3, 2025

1 Tarea 1

Tópicos Selectos De Matemáticas Aplicadas II: Análisis de Datos con Python Fecha de entrega: Viernes 6 de junio

Alan Badillo Salas

1. Crea un bucle para obtener el promedio de todos los elementos que se encuentran en las siguientes listas:

[49]:
```
L1=[7.2,7.8,6.8,8.0,None,'8.2',5.6,8.2,7.7,7.5,None,5.8]
L2=['6.8',None,6.8,6.1,7.9,9.4,None]
L3=[8.5,'9.0',None,7.7]
```

Solución:

Primero utilizaremos un método tradicional, nos percatamos que algunos valores son `None` y otros son números pero en texto, por lo que hay que ignorar a los valores que sean `None` y convertir aquellos cuyo tipo sea `str`

[50]:
```
suma = 0
contador = 0

for l in L1:
    if l is None:
        break
    if type(l) == str:
        l = float(l)
    # en este punto los valores `l` deberían ser números
    suma += l
    contador += 1

promedio1 = suma / contador

print(f"El promedio de L1 es: {promedio1:.2f}")
```

```
El promedio de L1 es: 7.45
```

En un segundo método podemos filtrar los elementos más rápidamente y hacer las conversiones mediante las listas condensadas:

Sintaxis:

`[<elemento> for <elemento> in <secuencia> if <condición sobre el elemento>]`

Esto genera una lista que dispone solo los elementos que cumplen la condición.

La ventaja de este segundo método es poder operar la lista completa con los operadores `sum(•)` y `len(•)`

[51]:
```
# Filtramos los elementos que no son `None` y los reconvertimos a `float`
L2_corregida = [float(l) for l in L2 if not l is None]

promedio2 = sum(L2_corregida) / len(L2_corregida)

print(f"El promedio de L2 es: {promedio2:.2f}")
```

```
El promedio de L2 es: 7.40
```

Una forma aún más compacta es usar la librería de Pandas para limpiar los datos y operarlos directamente.

Esto nos proporciona métodos estadísticos fundamentales para describir el eje de datos y con un código más compacto:

- `<serie>.dropna()` - Eliminamos los valores nulos
- `<serie>.astype()` - Reconvertimos los datos a `float`
- `<serie>.mean()` - Obtiene la media del eje de datos
- `<serie>.describe()` - Resume los estadísticos principales

[52]:
```
import pandas

S3 = pandas.Series(L3).dropna().astype(float)

S3.describe()
```

[52]:
```
count    3.000000
mean     8.400000
std      0.655744
min      7.700000
25%      8.100000
50%      8.500000
75%      8.750000
max      9.000000
dtype: float64
```

2. Crea un bucle que divida en 3 listas del mismo tamaño a los elementos de las listas anteriores ordenados de menor a mayor.

Para este proceso usaremos el método de pivote que consiste en calcular los índices donde se hará el corte de las listas, para luego rebanar las listas mediante la subselección de índices:

- `<lista>[i:j+1]` - Devuelve los elementos del índice `i` al `j`
- `<lista>[:a]` - Devuelve los elementos del índice `0` al `a - 1`
- `<lista>[b:]` - Devuelve los elementos del índice `b` al último

- `<lista>[a:b]` - Devuelve los elementos entre los índices `[a, b)` excluyéndolos

Por lo que, nuestro objetivo es encontrar los índices `a` y `b` para cada lista y aplicar los cortes `[0, a)`, `[a, b)` y `[b, N]` donde `N` es el tamaño de la lista o dejarlo en blanco para que se calcule solo

[53]:
```
from math import floor, ceil

# Recorremos cada lista
for L in [L1, L2, L3]:
    N = len(L) # Tamaño de la lista
    a = floor(N / 3) # Pivote inferior
    b = ceil((2 * N) / 3) # Pivote superior
    L_inferior = L[0:a]
    L_medio = L[a:b]
    L_superior = L[b:]

    print("Lista original:", L)
    print("-" * 90)
    print("Lista partida: ", L_inferior, L_medio, L_superior)

    print(end="\n\n")
```

```
Lista original: [7.2, 7.8, 6.8, 8.0, None, '8.2', 5.6, 8.2, 7.7, 7.5, None, 5.8]
------------------------------------------------------------------------------
----------
Lista partida:  [7.2, 7.8, 6.8, 8.0] [None, '8.2', 5.6, 8.2] [7.7, 7.5, None,
5.8]


Lista original: ['6.8', None, 6.8, 6.1, 7.9, 9.4, None]
------------------------------------------------------------------------------
----------
Lista partida:  ['6.8', None] [6.8, 6.1, 7.9] [9.4, None]


Lista original: [8.5, '9.0', None, 7.7]
------------------------------------------------------------------------------
----------
Lista partida:  [8.5] ['9.0', None] [7.7]
```

Nota: Observamos que si la puede cargarse con más elementos al centro

3. Crea una código que reciba del usuario nueve elementos de una matriz cuadrada de 3x3 y que retorne el determinante de la matriz.

Para este problema podemos capturar los 9 elementos uno a uno o de 3 en 3 separados por espacios

[54]:
```
matriz = []

for fila in range(3):
    vectorFila = []
    for columna in range(3):
        elemento = float(input(f"Ingresa la posición ({fila}, {columna})"))
        vectorFila.append(elemento)
    matriz.append(vectorFila)

matriz
```

[54]:
```
[[1.0, 2.0, 3.0], [4.0, 5.0, 6.0], [7.0, 8.0, 9.0]]
```

Mejora usando la librería `Pandas` y la lectura de `3` en `3`

[55]:
```
import pandas

A = pandas.DataFrame([
    (input(f"Línea {i} (3 valores separados por espacio simple):").split(" ")) \
        for i in range(3)
]).astype(float)

A
```

[55]:

	0	1	2
0	1.0	2.0	3.0
1	4.0	5.0	6.0
2	7.0	8.0	9.0

4. Crea un código que reciba una cadena de caracteres y arroje la respuesta de si es un estado ó una capital del país.

Para poder decidir si un texto es un estado o una capital, debemos establecer cuales son los estados y cuales las capitales, esta información la podemos disponer en un CSV para tomarlas como listas y luego recorrerlas buscando el texto objetivo.

Debemos considerar que la busqueda ingresada por el usuario podría no coincidir con las de las listas en caracteres especiales o mayúsculas y minúsculas.

[56]:
```
import pandas

entidades_capitales = pandas.read_csv("estados_capitales.csv")

entidades_capitales
```

[56]:

	Entidad Federativa	Capital	Municipio Capital
0	Aguascalientes	Aguascalientes	Aguascalientes
1	Baja California	Mexicali	Mexicali
2	Baja California Sur	La Paz	La Paz

3	Campeche	San Francisco Campeche	Campeche
4	Chiapas	Tuxtla Gutiérrez	Tuxtla Gutiérrez
5	Chihuahua	Chihuahua	Chihuahua
6	Ciudad de México	Ciudad de México	Ciudad de México
7	Coahuila	Saltillo	Saltillo
8	Colima	Colima	Colima
9	Durango	Durango	Durango
10	Guanajuato	Guanajuato	Guanajuato
11	Guerrero	Chilpancingo	Chilpancingo de los Bravo
12	Hidalgo	Pachuca	Pachuca de Soto
13	Jalisco	Guadalajara	Guadalajara
14	México	Toluca	Toluca
15	Michoacán	Morelia	Morelia
16	Morelos	Cuernavaca	Cuernavaca
17	Nayarit	Tepic	Tepic
18	Nuevo León	Monterrey	Monterrey
19	Oaxaca	Oaxaca de Juárez	Oaxaca de Juárez
20	Puebla	Puebla	Puebla
21	Querétaro	Querétaro	Querétaro
22	Quintana Roo	Chetumal	Othón P. Blanco
23	San Luis Potosí	San Luis Potosí	San Luis Potosí
24	Sinaloa	Culiacán	Culiacán
25	Sonora	Hermosillo	Hermosillo
26	Tabasco	Villahermosa	Centro
27	Tamaulipas	Ciudad Victoria	Victoria
28	Tlaxcala	Tlaxcala	Tlaxcala
29	Veracruz	Xalapa	Xalapa
30	Yucatán	Mérida	Mérida
31	Zacatecas	Zacatecas	Zacatecas

[57]:
```
estados = list(entidades_capitales["Entidad Federativa"].values)

estados
```

[57]:
```
['Aguascalientes',
 'Baja California',
 'Baja California Sur',
 'Campeche',
 'Chiapas',
 'Chihuahua',
 'Ciudad de México',
 'Coahuila',
 'Colima',
 'Durango',
 'Guanajuato',
 'Guerrero',
 'Hidalgo',
```

```
 'Jalisco',
 'México',
 'Michoacán',
 'Morelos',
 'Nayarit',
 'Nuevo León',
 'Oaxaca',
 'Puebla',
 'Querétaro',
 'Quintana Roo',
 'San Luis Potosí',
 'Sinaloa',
 'Sonora',
 'Tabasco',
 'Tamaulipas',
 'Tlaxcala',
 'Veracruz',
 'Yucatán',
 'Zacatecas']
```

[58]:
```
capitales = list(entidades_capitales["Capital"].values)

capitales
```

[58]:
```
['Aguascalientes',
 'Mexicali',
 'La Paz',
 'San Francisco Campeche',
 'Tuxtla Gutiérrez',
 'Chihuahua',
 'Ciudad de México',
 'Saltillo',
 'Colima',
 'Durango',
 'Guanajuato',
 'Chilpancingo',
 'Pachuca',
 'Guadalajara',
 'Toluca',
 'Morelia',
 'Cuernavaca',
 'Tepic',
 'Monterrey',
 'Oaxaca de Juárez',
 'Puebla',
 'Querétaro',
 'Chetumal',
```

```
 'San Luis Potosí',
 'Culiacán',
 'Hermosillo',
 'Villahermosa',
 'Ciudad Victoria',
 'Tlaxcala',
 'Xalapa',
 'Mérida',
 'Zacatecas']
```

Ahora que tenemos las listas de estados y capitales podemos hacer la búqueda

[59]:
```
busqueda = input("Ingresa el estado o la capital")

busquedaOriginal = busqueda

# Para mejorar la búsqueda podemos quitar espacios y acentos:
busqueda = busqueda.replace(" ", "")
busqueda = busqueda.replace("á", "a")
busqueda = busqueda.replace("é", "e")
busqueda = busqueda.replace("í", "i")
busqueda = busqueda.replace("ó", "o")
busqueda = busqueda.replace("ú", "u")
# Tambien convertimos a minúsculas
busqueda = busqueda.lower()

# Ahora recorremos los estados y capitales
for estado, capital in zip(estados, capitales):
    # Aplicamos la misma limpieza

    estadoOriginal = estado

    estado = estado.replace(" ", "")
    estado = estado.replace("á", "a")
    estado = estado.replace("é", "e")
    estado = estado.replace("í", "i")
    estado = estado.replace("ó", "o")
    estado = estado.replace("ú", "u")
    estado = estado.lower()

    capitalOriginal = capital

    capital = capital.replace(" ", "")
    capital = capital.replace("á", "a")
    capital = capital.replace("é", "e")
    capital = capital.replace("í", "i")
    capital = capital.replace("ó", "o")
```

```
    capital = capital.replace("ú", "u")
    capital = capital.lower()

    if estado.startswith(busqueda) or busqueda.startswith(estado):
        print(f"Se encontró al estado {estadoOriginal} mediante la búsqueda␣
↪<{busqueda}>")

    if capital.startswith(busqueda) or busqueda.startswith(capital):
        print(f"Se encontró al capital {estadoOriginal} mediante la búsqueda␣
↪<{busqueda}>")
```

```
Se encontró al estado Aguascalientes mediante la búsqueda <aguas>
Se encontró al capital Aguascalientes mediante la búsqueda <aguas>
```

5. Pide al usuario dos listas de números reales (misma longitud) y calcula su producto escalar usando un bucle for.

[60]:
```
N = int(input("Ingresa el tamaño de las listas:"))

lista1 = [float(input(f"Ingresa el elemento {i + 1} / {N} de la lista 1:")) for␣
↪i in range(N)]
lista2 = [float(input(f"Ingresa el elemento {i + 1} / {N} de la lista 2:")) for␣
↪i in range(N)]

suma = 0

for x1, x2 in zip(lista1, lista2):
    suma += x1 * x2

print(lista1)
print(lista2)
print(f"El producto punto es: {suma:.2f}")
```

```
[1.0, 2.0, 3.0]
[4.0, 5.0, 6.0]
El producto punto es: 32.00
```

6. Utiliza un bucle while True para pedir números al usuario y agregarlos a una lista. Cuando la suma total de los elementos sea mayor a 100, el programa debe detenerse y mostrar la lista y la suma final.

[61]:
```
lista = []

while True:
    x = float(input("Ingresa un número"))
    lista.append(x)
    if sum(lista) > 100:
        break
```

```
print(lista)
print(sum(lista))
```

```
[17.0, 23.0, 43.0, 23.0]
106.0
```

7. Crea una lista vacía y presenta al usuario un menú como este dentro de un while True:

[62]:
```
lista = []

while True:
    print("\nMenú:")
    print("1. Agregar número")
    print("2. Mostrar lista")
    print("3. Salir")

    opcion = input("Seleccione una opción: ")

    print(f"La opción seleccionada es: {opcion}")

    if opcion == "1":
        x = float(input("Ingresa el número:"))
        lista.append(x)
        print(f"Se añadió el elemento: {x}")
    elif opcion == "2":
        print(lista)
    elif opcion == "3":
        print()
        print("Adiós")
        break
    else:
        print("La opción no es válida")

    print()


print("Fin del programa")
```

```
Menú:
1. Agregar número
2. Mostrar lista
3. Salir
La opción seleccionada es: 12
La opción no es válida


Menú:
1. Agregar número
```

```
2. Mostrar lista
3. Salir
La opción seleccionada es: 1
Se añadió el elemento: 1.0


Menú:
1. Agregar número
2. Mostrar lista
3. Salir
La opción seleccionada es: 1
Se añadió el elemento: 34.0


Menú:
1. Agregar número
2. Mostrar lista
3. Salir
La opción seleccionada es: 1
Se añadió el elemento: 23.0


Menú:
1. Agregar número
2. Mostrar lista
3. Salir
La opción seleccionada es: 5
La opción no es válida


Menú:
1. Agregar número
2. Mostrar lista
3. Salir
La opción seleccionada es: 2
[1.0, 34.0, 23.0]


Menú:
1. Agregar número
2. Mostrar lista
3. Salir
La opción seleccionada es: 1
Se añadió el elemento: 34.0


Menú:
1. Agregar número
```

```
2. Mostrar lista
3. Salir
La opción seleccionada es: 2
[1.0, 34.0, 23.0, 34.0]


Menú:
1. Agregar número
2. Mostrar lista
3. Salir
La opción seleccionada es: 6
La opción no es válida


Menú:
1. Agregar número
2. Mostrar lista
3. Salir
La opción seleccionada es: 3

Adiós
Fin del programa
```

El usuario debe poder ejecutar varias opciones hasta que elija salir.