



Practica: 03
Nombre de la Practica: Ejercicios de
Pypass

Instituto Politécnico Nacional.
Escuela Superior de Cómputo.
Licenciatura en ciencia de datos.

Nombre de la materia: Desarrollo de Aplicaciones para el Análisis de Datos
Grupo: 4AV1
Profesora: Sandra Luz Morales Guitron

Lopez Mendez Emiliano

INDICE

INTRODUCCION	3
DESAROLLO	3
LISTAS.....	4
DICCIONARIOS	26
SETS	32
FICHEROS	32
CONCLUSIONES	40

INTRODUCCION

En esta práctica, me propuse aprender y aplicar una serie de conceptos y técnicas de programación utilizando Python, con el objetivo de manipular, analizar y procesar datos de manera efectiva. A lo largo de los ejercicios, trabajé con diferentes estructuras de datos como listas, diccionarios, conjuntos y archivos. También exploré cómo implementar funciones eficientes para resolver problemas prácticos que abarcan desde operaciones matemáticas básicas hasta el análisis de textos y procesamiento de matrices. Además, utilicé librerías clave de Python que me permitieron optimizar el manejo de datos y generar soluciones más robustas.

DESAROLLO

Durante la práctica, trabajé con diversas funciones que me permitieron resolver problemas específicos. Comencé por implementar operaciones básicas con listas, como el uso de comprensión de listas, sumas acumulativas y la manipulación de posiciones dentro de las listas. Además, profundicé en el uso de diccionarios para realizar tareas como contar ocurrencias de elementos, extraer subconjuntos de información y corregir estructuras de datos.

LISTAS

1. max-value El código encuentra y devuelve el valor máximo en una lista iterando a través de sus elementos y comparando cada uno con el máximo actual.

```
4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > listas-listo
1  def run(values: list) -> int:
2      max_value = values[0]
3      for i in range(0, len(values)):
4          if i == len(values) - 1:
5              break
6          if max_value >= values[i + 1]:
7              pass
8          else:
9              max_value = values[i + 1]
10     return max_value
11
12
13     # DO NOT TOUCH THE CODE BELOW
14     if __name__ == '__main__':
15         lista = [1, 2, 3, 4, 5]
16         print(f'La lista:{lista}\nSu elemento máximo:{run(lista)}') # 5
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Pyt
:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicacio
_Datos/Practica3/listas-listo/1-max-value/main.py"
La lista:[1, 2, 3, 4, 5]
Su elemento máximo:5
```

2. max-value-with-min El código encuentra el valor máximo de una lista convirtiendo todos los números a sus valores opuestos (negativos), y luego encuentra el mínimo de esa lista de números opuestos, devolviendo su equivalente positivo como el máximo original.

```

Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3
1 def run(values: list) -> int:
2     numeros_opuestos=[-x for x in values]
3     max_value = -min(numeros_opuestos)
4     return max_value
5
6
7 # DO NOT TOUCH THE CODE BELOW
8 if __name__ == '__main__':
9     lista = [1, 2, 3, 4, 5]
10    print(f'La lista:{lista}\nSu elemento máximo:{run(lista)}')

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```

PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python311/Python.exe -i
C:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas-listo/2-max-value-with-min/main.py
La lista:[1, 2, 3, 4, 5]
Su elemento máximo:5
PS C:\Users\HP\Documents\4to Semestre>

```

3. min-value El código encuentra y devuelve el valor mínimo en una lista iterando a través de sus elementos y comparando cada uno con el mínimo actual.

```

4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > listas-listo
1 def run(values: list) -> int:
2     min_value = values[0]
3     for i in range(0, len(values)):
4         if i == len(values) - 1:
5             break
6         if min_value <= values[i + 1]:
7             pass
8         else:
9             min_value = values[i + 1]
10    return min_value
11
12
13 # DO NOT TOUCH THE CODE BELOW
14 if __name__ == '__main__':
15     lista = [1, 2, 3, 4, 5]
16    print(f'La lista:{lista}\nSu elemento mínimo:{run(lista)}') # 1
17

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```

PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python311/Python.exe -i
C:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas-listo/3-min-value/main.py
La lista:[1, 2, 3, 4, 5]
Su elemento mínimo:1
PS C:\Users\HP\Documents\4to Semestre>

```

4. min-value-with-max El código encuentra el valor mínimo de una lista convirtiendo los números a sus valores opuestos (negativos) y luego encontrando el máximo de esa lista de opuestos, devolviendo su valor equivalente positivo como el mínimo original.

```

main.py
o-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > listas-listo > 4
1 def run(values: list) -> int:
2     lista_opuesta = [-x for x in values]
3     min_value = -max(lista_opuesta)
4     return min_value
5
6
7 # DO NOT TOUCH THE CODE BELOW
8 if __name__ == '__main__':
9     lista = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
10    print(f'La lista: {lista}\nEl valor mínimo: {run(lista)}') # 1
11

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```

PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python311/Python.exe -i
C:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas-listo/4-min-value-with-max/main.py
La lista: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
El valor mínimo: 1
PS C:\Users\HP\Documents\4to Semestre>

```

5. remove-dups El código elimina los elementos duplicados de una lista, manteniendo solo la primera aparición de cada número, y devuelve una nueva lista sin repeticiones.

```

4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > listas-listo > 5-remove-dups
1 def run(nums_dups: list) -> list:
2     nums_unique = []
3     for i in range(len(nums_dups)):
4         if nums_dups[i] not in nums_unique:
5             nums_unique.append(nums_dups[i])
6     return nums_unique
7
8
9 # DO NOT TOUCH THE CODE BELOW
10 if __name__ == '__main__':
11     list1 = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
12     print(f'La lista: {list1}\nLa lista sin repeticiones: {run(list1)}') # [1, 2, 3,
13
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS Python + -
PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe
:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas-listo/5-remove-dups/main.py"
La lista: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10, 1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
La lista sin repeticiones: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
PS C:\Users\HP\Documents\4to Semestre>

```

6. **flatten-list** El código aplanar una lista anidada, es decir, convierte cualquier sublista dentro de la lista original en una lista plana, manteniendo todos los elementos en un solo nivel.

```

4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > listas-listo > 6-flatten-list
1 def run(items: list) -> list:
2     flatten_items = []
3     for i in items:
4         if type(i) == list:
5             for j in i:
6                 flatten_items.append(j)
7         else:
8             flatten_items.append(i)
9     return flatten_items
10
11
12 # DO NOT TOUCH THE CODE BELOW
13 if __name__ == '__main__':
14     list1 = [1, 2, [3, 4, 5], 6, 7, [8, 9], 10]
15     print(f'La lista: {list1}\nLa lista aplanada: {run(list1)}') # [1, 2, 3,
16

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS Python
PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe
:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas-listo/6-flatten-list/main.py"
La lista: [1, 2, [3, 4, 5], 6, 7, [8, 9], 10]
La lista aplanada: [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
PS C:\Users\HP\Documents\4to Semestre>

```

7. **remove-consecutive-dups** El código elimina los elementos consecutivos duplicados de una lista, devolviendo una nueva lista donde solo se mantiene una aparición de cada elemento consecutivo repetido.

```

main.py U x
Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > listas-listo > 7-remove-consecutive-dups
1 def run(items: list) -> list:
2     if not items: # Verifica si la items está vacía
3         return []
4
5     resultado = [items[0]] # Inicia con el primer elemento de la items
6     for i in range(1, len(items)):
7         if items[i] != items[i - 1]: # Solo añade si el elemento actual es diferente
8             resultado.append(items[i])
9
10    return resultado
11
12 # DO NOT TOUCH THE CODE BELOW
13 if __name__ == '__main__':
14     list_items = [0, 0, 1, 2, 3, 4, 4, 5, 6, 6, 6, 7, 8, 9, 4, 4]
15     print(f'La lista normal:{list_items}\nLa lista sin elementos repetidos:{run(list_
16
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS Python + -
PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe
:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis
_Datos/Practica3/listas-listo/7-remove-consecutive-dups/main.py"
La lista normal:[0, 0, 1, 2, 3, 4, 4, 5, 6, 6, 6, 7, 8, 9, 4, 4]
La lista sin elementos repetidos:[0, 1, 2, 3, 4, 5, 6, 7, 8, 9, 4]
PS C:\Users\HP\Documents\4to Semestre>

```

8. all-same El código verifica si todos los elementos de una lista son iguales y devuelve `True` si lo son, o `False` si no lo son.

```

4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > listas-listo > 8-all-same >
1 def run(items: list) -> bool:
2     for i in items:
3         if i != items[0]:
4             return False
5     return True
6
7
8 # DO NOT TOUCH THE CODE BELOW
9 if __name__ == '__main__':
10     lista = [1, 1, 1, 1, 1, 1, 1, 1, 1]
11     print(f'La lista {lista},\ntiene todos los elementos iguales: {run(lista)}') # 0
12
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS Python + -
PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe
:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis
_Datos/Practica3/listas-listo/8-all-same/main.py"
La lista [1, 1, 1, 1, 1, 1, 1, 1, 1],
tiene todos los elementos iguales: True
PS C:\Users\HP\Documents\4to Semestre>

```

9. sum-diagonal El código calcula la suma de los elementos de la diagonal principal de una matriz cuadrada (misma cantidad de filas y columnas). Si la matriz no es cuadrada, devuelve `None`.

```

4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > listas-listo > 9-sum-diagonal >
1 def run(matrix: list) -> int | None:
2     sum_diagonal = 0
3     for i in range(0, len(matrix)):
4         if len(matrix) != len(matrix[i]):
5             return None
6         for j in range(0, len(matrix[i])):
7             if i == j:
8                 sum_diagonal += matrix[i][j]
9     return sum_diagonal
10
11
12 # DO NOT TOUCH THE CODE BELOW
13 if __name__ == '__main__':
14     matrix = [[1, 0, 0], [0, 1, 0], [0, 0, 1]]
15     print(f'La matriz:\n{matrix}\nLa suma de la diagonal es: {run(matrix)}') # Outpu
16     matrix = [[1, 0, 0], [0, 1, 0], [0, 0, 1], [0, 0, 0]]
17     print(f'La matriz:\n{matrix}\nLa suma de la diagonal es: {run(matrix)}') # Outpu
18     matrix = [[1, 0, 0], [0, 1, 0], [0, 0, 1], [0, 0, 0], [0, 0, 0]]
19

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS Python + -

```

PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe
:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis
_Datos/Practica3/listas-listo/9-sum-diagonal/main.py
La matriz:
[[1, 0, 0], [0, 1, 0], [0, 0, 1]]
La suma de la diagonal es: 3
La matriz:
[[1, 0, 0], [0, 1, 0], [0, 0, 1], [0, 0, 0]]
La suma de la diagonal es: None
PS C:\Users\HP\Documents\4to Semestre>

```

10. powers2 El código genera una lista de potencias de 2, desde (2^0) hasta (2^n) , donde (n) es un valor dado, y devuelve esa lista. En este caso, se genera una lista de potencias de 2 hasta (2^{10}) .

```

4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > listas-listo > 10-powers2 >
1 def run(n: int) -> list:
2     lista = [2 ** i for i in range(n+1)]
3     return lista
4
5
6 # DO NOT TOUCH THE CODE BELOW
7 if __name__ == '__main__':
8     list_items = 10
9     print(f'La lista de potencias de 2 hasta {list_items} es:\n{run(list_items)}') #
10

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS Python + -

```

PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.
:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Anali
_Datos/Practica3/listas-listo/10-powers2/main.py
La lista de potencias de 2 hasta 10 es:
[1, 2, 4, 8, 16, 32, 64, 128, 256, 512, 1024]
PS C:\Users\HP\Documents\4to Semestre>

```


11. dec2bin El código convierte un número decimal a su representación binaria y la devuelve como una cadena. En este caso, convierte el número decimal 56 en su equivalente binario "111000".

```
4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > listas-listo >
1 def run(n: int) -> str:
2     bin_repr = ''
3     if n == 0:
4         return '0'
5     while n > 0:
6         bin_repr = str(n % 2) + bin_repr
7         n = n // 2
8     return bin_repr
9
10
11 # DO NOT TOUCH THE CODE BELOW
12 if __name__ == '__main__':
13     num_decimal = 56
14     print(f'Numero decimal:{num_decimal}\n Binario: {run(num_decimal)}')
15
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS Python

```
PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python311/Python.exe -i
C:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas-listo/11-dec2bin/main.py
Numero decimal:56
Binario: 111000
PS C:\Users\HP\Documents\4to Semestre>
```

12. sum-mixed El código convierte los elementos de una lista (que pueden ser cadenas o enteros) a enteros y luego devuelve la suma de todos esos valores. En este caso, convierte los elementos de la lista `['1', '2', 3, '4', 5]` a enteros y calcula la suma, que es 15.

```
4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > listas-listo >
1 def run(items: list) -> int:
2     lista_enteros = []
3     for i in items:
4         lista_enteros.append(int(i))
5     return sum(lista_enteros)
6
7
8 # DO NOT TOUCH THE CODE BELOW
9 if __name__ == '__main__':
10     lista = ['1', '2', 3, '4', 5]
11     print(f'La suma de la lista {lista} es: {run(lista)}')
12
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS Python

```
PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python311/Python.exe -i
C:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas-listo/12-sum-mixed/main.py
La suma de la lista ['1', '2', 3, '4', 5] es: 15
PS C:\Users\HP\Documents\4to Semestre>
```

13. n-multiples El código genera y muestra una lista con los primeros (n) múltiplos de un número (x). En este caso, para (x = 2) y (n = 4), el resultado es la lista `[2, 4, 6, 8]`, que son los primeros 4 múltiplos de 2.

```
4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > listas-1
1  def run(x, n):
2      # Generar lista de múltiplos usando comprensión de listas
3      multiplos = [x * i for i in range(1, n + 1)]
4
5      # Imprimir el resultado
6      print(f'Los primeros {n} múltiplos de {x} son: {multiplos}')
7
8
9  # Llamar a la funcion
10 run(2, 4)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python
:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas-1/13-n_multiples.py
Los primeros 4 múltiplos de 2 son: [2, 4, 6, 8]
PS C:\Users\HP\Documents\4to Semestre>
```

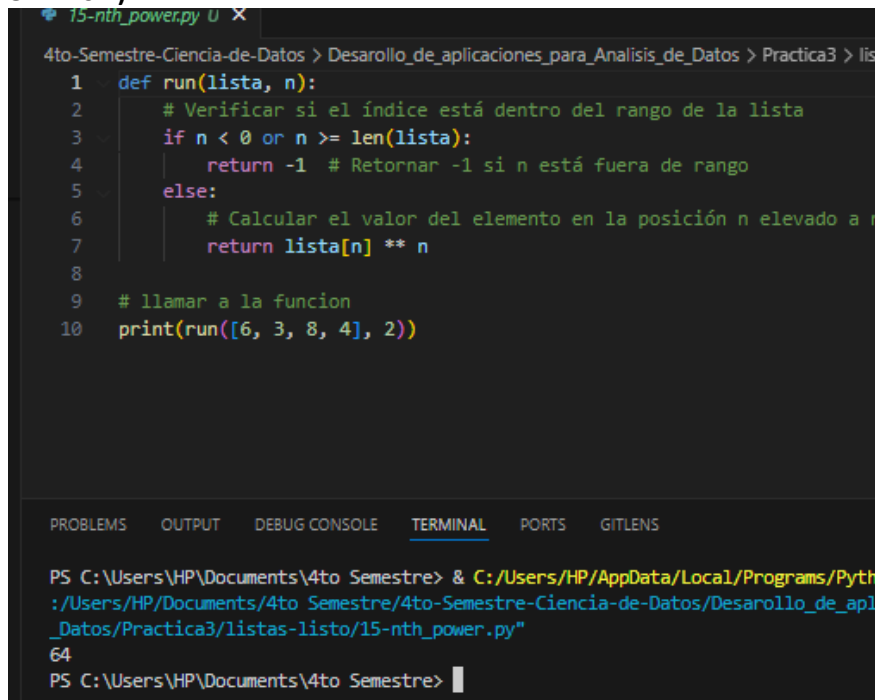
14. drop-even El código elimina los elementos en posiciones pares de una lista, manteniendo solo los que están en posiciones impares. En este ejemplo, para la lista `['U', 'V', 'W', 'X', 'Y']`, el resultado es `['V', 'X']`, que son los elementos en posiciones impares.

```
4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > listas-1 > 14-drop_
1  def run(lista):
2      # Usar comprensión de listas para seleccionar elementos en posiciones impares
3      resultado = [lista[i] for i in range(len(lista)) if i % 2 != 0]
4
5      # Imprimir el resultado
6      print(f'La lista eliminando los elementos en posiciones pares es: {resultado}')
7
8
9  # llamar a la funcion
10 run(['U', 'V', 'W', 'X', 'Y'])
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS Python + -

```
PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python
:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas-1/14-drop_even.py
La lista eliminando los elementos en posiciones pares es: ['V', 'X']
PS C:\Users\HP\Documents\4to Semestre>
```

15. nth-power El código toma una lista y un índice (n), y devuelve el valor del elemento en la posición (n) elevado a la potencia de (n). Si el índice (n) está fuera del rango de la lista, devuelve -1. En el ejemplo, para la lista `[6, 3, 8, 4]` y el índice 2, el resultado es ($8^2 = 64$).



```

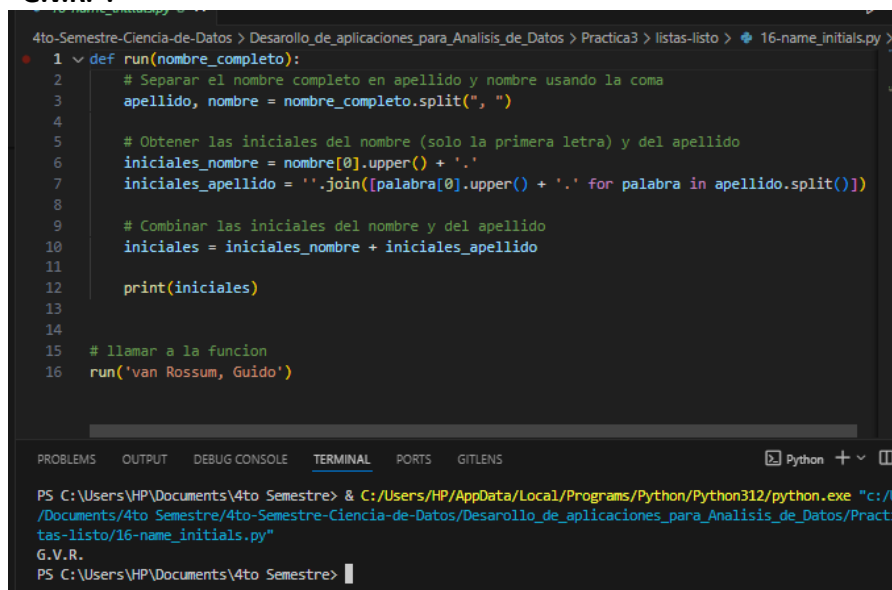
15-nth_power.py
4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > listas-listo
1 def run(lista, n):
2     # Verificar si el índice está dentro del rango de la lista
3     if n < 0 or n >= len(lista):
4         return -1 # Retornar -1 si n está fuera de rango
5     else:
6         # Calcular el valor del elemento en la posición n elevado a n
7         return lista[n] ** n
8
9 # llamar a la funcion
10 print(run([6, 3, 8, 4], 2))

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas-listo/15-nth_power.py"
64
PS C:\Users\HP\Documents\4to Semestre>

```

16. name-initials El código toma un nombre completo en formato "apellido, nombre", separa el apellido y el nombre, y luego genera las iniciales de ambos. Las iniciales del nombre son la primera letra del nombre en mayúscula seguida de un punto, y las iniciales del apellido se generan tomando la primera letra de cada palabra del apellido en mayúscula, también seguidas de un punto. En el ejemplo, para "van Rossum, Guido", el resultado es "G.V.R.".



```

16-name_initials.py
4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > listas-listo > 16-name_initials.py
1 def run(nombre_completo):
2     # Separar el nombre completo en apellido y nombre usando la coma
3     apellido, nombre = nombre_completo.split(", ")
4
5     # Obtener las iniciales del nombre (solo la primera letra) y del apellido
6     iniciales_nombre = nombre[0].upper() + '.'
7     iniciales_apellido = ''.join([palabra[0].upper() + '.' for palabra in apellido.split()])
8
9     # Combinar las iniciales del nombre y del apellido
10    iniciales = iniciales_nombre + iniciales_apellido
11
12    print(iniciales)
13
14
15 # llamar a la funcion
16 run('van Rossum, Guido')

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS Python +

PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas-listo/16-name_initials.py"
G.V.R.
PS C:\Users\HP\Documents\4to Semestre>

```

13 de octubre de 2024

17. non-consecutive El código revisa una lista de números para encontrar el primer número que no sea consecutivo respecto al anterior. Si encuentra un número no consecutivo, lo devuelve. Si todos los números son consecutivos, devuelve ``None``. En el ejemplo, para la lista ``[1, 2, 4, 5]``, el resultado es ``4`` porque 4 no sigue a 3. Para la lista ``[1, 2, 3, 4]``, devuelve ``None`` porque todos los números son consecutivos.

```
4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > listas-listo > 17-non_consecutive.py >
1 def run(lista):
2     # Recorrer la lista a partir del segundo elemento
3     for i in range(1, len(lista)):
4         # Si la diferencia entre el valor actual y el anterior no es 1, no son consecutivos
5         if lista[i] != lista[i - 1] + 1:
6             return lista[i] # Retornar el primer número no consecutivo
7     return None # Si todos los números son consecutivos
8
9 # llamar a la funcion
10 print(run([1, 2, 4, 5])) # Debe retornar 4
11 print(run([1, 2, 3, 4])) # Debe retornar None
```

18. **mult-reduce** El código utiliza la función ``reduce`` de la biblioteca ``functools`` para multiplicar todos los elementos de una lista. En este caso, toma la lista ``[9, 3, 4, 2]`` y calcula el producto de sus elementos, que es 216.

```

1  from functools import reduce
2
3
4  def run(lista):
5      # Usar reduce para multiplicar todos los elementos de la lista
6      resultado = reduce(lambda x, y: x * y, lista)
7
8      # Imprimir el resultado
9      print(resultado)
10
11
12 # llamar a la funcion
13 run([9, 3, 4, 2])

```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

GITLENS

PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python38-64/Scripts/python.exe C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis-de-Datos\listas-listo\18-mult_reduce.py"

216

PS C:\Users\HP\Documents\4to Semestre>

19. **digit-rev-list** El código toma un número entero, lo convierte en una cadena de texto, invierte esa cadena y luego convierte cada dígito de la cadena invertida de vuelta a un entero. Finalmente, imprime la lista de dígitos invertidos. En el ejemplo, el número `56442` se convierte en la lista `[2, 4, 4, 6, 5]`.

```
4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 >
1 def run(numero):
2     # Convertir el número a una cadena, invertirlo, luego volverlo a un número
3     resultado = [int(digito) for digito in str(numero)[::-1]]
4
5     # Imprimir el resultado
6     print(resultado)
7
8
9 # llamar a la función
10 run(56442)
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS GITLENS

```
PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python311/Python.exe C:\Users\HP\Documents\4to Semestre\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3\19-digit_rev_list.py
[2, 4, 4, 6, 5]
PS C:\Users\HP\Documents\4to Semestre>
```

20. **time-plus-minutes** El código toma una hora en formato de cadena (`hora_entrada`) y una cantidad de minutos, y luego suma esos minutos a la hora inicial utilizando la clase `timedelta` de la biblioteca `datetime`. Finalmente, convierte y muestra la nueva hora en formato `HH:MM`. En este ejemplo, a la hora `17:15` se le suman 240 minutos, resultando en la hora `21:15`.

```
4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 >
1 from datetime import datetime, timedelta
2
3
4 def run(hora_entrada, minutos):
5     # Convertir la hora de entrada en un objeto datetime
6     hora_obj = datetime.strptime(hora_entrada, '%H:%M')
7     # Sumar los minutos utilizando timedelta
8     nueva_hora = hora_obj + timedelta(minutes=minutos)
9     # Convertir la nueva hora a formato de cadena HH:MM
10    resultado = nueva_hora.strftime('%H:%M')
11    # Imprimir el resultado
12    print(resultado)
13
14
15 # llamar a la función
16 run('17:15', 240)
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS GITLENS

```
PS C:\Users\HP\Documents\4to Semestre>
```

13 de octubre de 2024

21. **add-positives** El código suma únicamente los valores positivos de una lista de números. En este ejemplo, la lista `[6, 3, 0, -1, -7, 5]` tiene los números positivos `6`, `3` y `5`, cuya suma es `14`.

```
4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Pra
1 def run(lista):
2     # Sumar solo los valores positivos de la lista
3     suma_positivos = sum([num for num in lista if num > 0])
4
5     # Imprimir el resultado
6     print(suma_positivos)
7
8
9 # llamar a la funcion
10 run([6, 3, 0, -1, -7, 5])
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python39-64/Scripts/python.exe C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones-listo\21-add_positives.py"
14
PS C:\Users\HP\Documents\4to Semestre> 
```

22. **add-opposites** El código calcula la suma de los valores opuestos (negativos) de los elementos de una lista. En este caso, para la lista `[6, 3, 0, -1, -7, 5]`, el código toma los valores opuestos, es decir, `[-6, -3, 0, 1, 7, -5]`, y luego suma esos valores, obteniendo `-6` como resultado.

```

1  def run(lista):
2      # Obtener la suma de los valores opuestos (negativos
3      suma_opuestos = sum([-num for num in lista])
4
5      # Imprimir el resultado
6      print(suma_opuestos)
7
8
9  # llamar a la funcion
10 run([6, 3, 0, -1, -7, 5])

```

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL

PORTS

GITLENS

```

PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/
/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de
-tas-listo/22-add_opposites.py"
-6
PS C:\Users\HP\Documents\4to Semestre>

```

13 de octubre de 2024

23. **descending-numbers** El código genera una lista de números en orden descendente desde un valor (n) hasta 1. En este ejemplo, para $(n = 5)$, la lista generada es `[5, 4, 3, 2, 1]`.

```
4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis
1 def run(n):
2     # Generar una lista con los números desde n hasta 0
3     conteo_regresivo = list(range(n, 0, -1))
4
5     # Imprimir el resultado
6     print(conteo_regresivo)
7
8
9 # llamar a la función
10 run(5)
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GIT LENS

```
PS C:\Users\HP\Documents\4to Semestre> & C:\Users\HP\AppData\Local\Programs\Python\Python38-64\Scripts\python.exe C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis\Desarrollo_de_aplicaciones_para_Analisis.py
[5, 4, 3, 2, 1]
```

24. merge-sorted El código combina dos listas ordenadas en una sola lista ordenada sin duplicados, comparando sus elementos y agregándolos a la nueva lista de manera eficiente.

```

1 def run(values1: list, values2: list) -> list:
2     merged = []
3     i, j = 0, 0
4
5     while i < len(values1) and j < len(values2):
6         if values1[i] < values2[j]:
7             if values1[i] not in merged:
8                 merged.append(values1[i])
9                 i += 1
10        elif values1[i] > values2[j]:
11            if values2[j] not in merged:
12                merged.append(values2[j])
13                j += 1
14        else:
15            if values1[i] not in merged:
16                merged.append(values1[i])
17                i += 1
18            j += 1
19
20    while i < len(values1):
21        if values1[i] not in merged:
22            merged.append(values1[i])
23            i += 1
24
25    while j < len(values2):
26        if values2[j] not in merged:
27            merged.append(values2[j])
28            j += 1
29
30    return merged
31
32 # Llamada de ejemplo
33 resultado = run([1, 3, 5, 7], [2, 3, 6, 8])
34 print(resultado)
35

```

25. **trimmed-add** El código calcula la suma de los elementos de una lista excluyendo el valor máximo y mínimo, devolviendo 0 si la lista tiene menos de 3 elementos.

```

You, 47 minutes ago | 2 authors (You and one other)
1 def run(values: list) -> int:
2     if len(values) < 3:
3         return 0
4     return sum(values) - max(values) - min(values)
5
6 # Llamada de ejemplo
7 resultado = run([7, 12, 4, 9, 3])
8 print(resultado)
9

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ...

```

PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas/25-trimmed-add/main.py"
20
PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3>

```

26. **wolves** El código evalúa la posición de un lobo en una lista de animales y, dependiendo de su ubicación respecto a las ovejas, emite una advertencia indicando si el lobo está al final o si se va a comer a la oveja más cercana.

```

4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > listas > 26-wolves > mail
You, 32 minutes ago | 2 authors (You and one other)
1 def run(farm: list) -> str:
2     wolf_index = farm.index('lobo')
3     if wolf_index == len(farm) - 1:
4         return "Oye lobo no te quiero ver más por aquí"
5     else:
6         sheep_index = len(farm) - wolf_index - 1
7         return f"Cuidado oveja {sheep_index}, el lobo te va a comer"
8
9 # Llamada de ejemplo
10 resultado1 = run(['oveja', 'oveja', 'lobo', 'oveja'])
11 print(resultado1) # Salida: Cuidado oveja 1, el lobo te va a comer
12
13 resultado2 = run(['oveja', 'oveja', 'oveja', 'lobo'])
14 print(resultado2) # Salida: Oye lobo no te quiero ver más por aquí
15

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ... Python + - [] [] ... ^

```

PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas/26-wolves/main.py"
Cuidado oveja 1, el lobo te va a comer
Oye lobo no te quiero ver más por aquí
PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3>

```

27. **minmax** El código encuentra y devuelve el valor mínimo y máximo de una lista de números, devolviendo `None` en ambos casos si la lista está vacía.


```

You, 33 minutes ago | 2 authors (You and one other)
1 def run(values: list) -> tuple:
2     if not values:
3         return None, None
4
5     min_value = values[0]
6     max_value = values[0]
7
8     for value in values[1:]:
9         if value < min_value:
10             min_value = value
11         if value > max_value:
12             max_value = value
13
14     return min_value, max_value
15
16 # Llamada de ejemplo
17 resultado = run([4, 2, 8, 11, 23, 8, 9])
18 print(f"min = {resultado[0]} y max = {resultado[1]}")
19

```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS ... Python + -

```

PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas/27-minmax/py"
min = 2 y max = 23
PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3>

```

28. cascading-subsets El código crea y devuelve una lista de tuplas que contienen subconjuntos consecutivos de la lista original, cada uno con el tamaño especificado ('size').

```

de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > listas > 28-cascading-subsets > ma
You, 34 minutes ago | 2 authors (You and one other)
1 def run(values: list, size: int) -> list:
2     cascading = [tuple(values[i:i+size]) for i in range(len(values) - size + 1)]
3     return cascading
4
5 # Llamada de ejemplo
6 resultado = run([1, 2, 3, 4], 2)
7 print(resultado) # Salida: [(1, 2), (2, 3), (3, 4)]
8

```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS ... Python + -

```

PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas/28-cascading-subsets/main.py"
[(1, 2), (2, 3), (3, 4)]
PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3>

```

29. diff-cuboid El código calcula la diferencia absoluta entre los volúmenes de dos cuboides representados por dos listas y devuelve dicha diferencia.

13 de octubre de 2024

```
encia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > listas > 29-diff-cuboid
You, 38 minutes ago | 2 authors (You and one other)
1 def run(cuboid1: list, cuboid2: list) -> float:
2     volume1 = cuboid1[0] * cuboid1[1] * cuboid1[2]
3     volume2 = cuboid2[0] * cuboid2[1] * cuboid2[2]
4     vol_diff = abs(volume1 - volume2)
5     return vol_diff
6
7 # Llamada de ejemplo
8 resultado = run([2, 2, 3], [5, 4, 1])
9 print(resultado) # Salida: 8
10
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ... Python + v

```
PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas/29-diff-cuboid/main.py"
8
PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3>
```

30. fl-strip El código elimina el primer y último número de una cadena de números separados por comas y devuelve los números restantes como una cadena con espacios en lugar de comas.

```
You, 38 minutes ago | 2 authors (You and one other)
1 def run(numbers: str) -> str:
2     elements = numbers.split(',')
3     if len(elements) <= 2:
4         return ''
5     strip_numbers = ' '.join(elements[1:-1])
6     return strip_numbers
7
8 # Llamada de ejemplo
9 resultado = run('1,2,3,4,5')
10 print(resultado) # Salida: '2 3 4'
11
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ... Python + v

```
PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas/30-fl-strip/main.py"
8
PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas/30-fl-strip/main.py"
2 3 4
PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3>
```

31. logical-chain El código aplica operaciones lógicas (`and` o `or`) a una lista de valores booleanos y devuelve el resultado final de la operación.

```

You, 39 minutes ago | 2 authors (You and one other)
1 def run(values: list, oper: str) -> bool:
2     result = values[0]
3     for value in values[1:]:
4         if oper == 'and':
5             result = result and value
6         elif oper == 'or':
7             result = result or value
8     return result
9
10 # Llamada de ejemplo
11 resultado = run([True, True, False], 'and')
12 print(resultado) # Salida: False
13

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ... Python

```

PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas/31/main.py"
False
PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3>

```

32. first-unused-id El código encuentra y devuelve el primer número entero positivo que no está presente en la lista dada.

```

You, 39 minutes ago | 2 authors (You and one other)
1 def run(ids: list) -> int:
2     ids_set = set(ids)
3     first_unused_id = 1
4     while first_unused_id in ids_set:
5         first_unused_id += 1
6     return first_unused_id
7
8 # Llamada de ejemplo
9 resultado = run([3, 1, 8, 4, 9])
10 print(resultado) # Salida: 2
11

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ... Python

```

PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas/32/main.py"
2
PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3>

```

33. find-odds El código filtra y devuelve una lista con solo los números impares de la lista original.

```

ica-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practicas > listas > 33-filtro-2
You, 45 minutes ago | 2 authors (You and one other)
1 def run(values: list) -> list:
2     return [num for num in values if num % 2 != 0]
3
4 # Llamada de ejemplo
5 resultado = run([7, 3, 4, 8, 12, 6, 9])
6 print(resultado) # Salida: [7, 3, 9]
7

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ... Python + v

```

PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas/33-filtro-2.in.py"
[7, 3, 9]
PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3>

```

34. chemistry El código verifica si una lista de números cumple con una serie de reglas lógicas específicas y devuelve `True` si todas las condiciones se cumplen o `False` en caso contrario.

```

You, 44 minutes ago | 2 authors (You and one other)
1 def run(formula: list) -> bool:
2     # Regla 1: El componente 1 y el componente 2 no pueden estar juntos
3     if 1 in formula and 2 in formula:
4         return False
5
6     # Regla 2: El componente 3 y el componente 4 no pueden estar juntos
7     if 3 in formula and 4 in formula:
8         return False
9
10    # Regla 3: El componente 5 y el componente 6 deben estar ambos o ninguno
11    if (5 in formula and 6 not in formula) or (6 in formula and 5 not in formula):
12        return False
13
14    # Regla 4: Al menos uno de los componentes 7 u 8 debe estar presente
15    if 7 not in formula and 8 not in formula:
16        return False
17
18    return True
19
20 # Llamada de ejemplo
21 resultado = run([1, 3, 5, 6, 8])
22 print(resultado) # Salida: True
23

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ... Python + v

```

PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas/34-chemistry/main.py"
True
PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3>

```

35. next-next El código busca un elemento específico en una lista y, si lo encuentra, devuelve el elemento siguiente a él; si no lo encuentra o está al final de la lista, devuelve `None`.

```

1  def run(items: list, ref_item: object) -> object:
2      if ref_item not in items:
3          return None
4
5      index = items.index(ref_item)
6      if index + 1 < len(items):
7          return items[index + 1]
8      else:
9          return None
10
11 # Llamada de ejemplo
12 resultado = run(['Ana', 'Sara', 'Noelia'], 'Sara')
13 print(resultado) # Salida: 'Noelia'
14

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ... Python + v

PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollaciones_para_Analisis_de_Datos\Practica3> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas/35-next-next.py"

Noelia

PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollaciones_para_Analisis_de_Datos\Practica3>

36. v-partition El código divide una lista de números en dos listas: una con los valores menores que un valor de referencia (`ref_value`) y otra con los valores mayores o iguales a ese valor de referencia.

```

1  def run(values: list, ref_value: int) -> list:
2      less_than = [x for x in values if x < ref_value]
3      greater_or_equal = [x for x in values if x >= ref_value]
4      return [less_than, greater_or_equal]
5
6 # Llamada de ejemplo
7 resultado = run([4, 3, 2, 9, 8, 5], 4)
8 print(resultado) # Salida: [[3, 2], [4, 9, 8, 5]]
9

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ... Python + v

PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollaciones_para_Analisis_de_Datos\Practica3> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas/36-v-partition.py"

[[3, 2], [4, 9, 8, 5]]

PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollaciones_para_Analisis_de_Datos\Practica3>

37. attach-len El código toma una cadena de texto, la divide en palabras, calcula la longitud de cada palabra, y devuelve una lista de cadenas donde cada palabra está seguida por su longitud. Por ejemplo, en el texto `"Hola a todos como estan"`, devuelve `['Hola:4', 'a:1', 'todos:5', 'como:4', 'estan:5']`.

```

>Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > listas-listo
1  def run(text: str) -> list:
2      list_text = text.split()
3      words_length = []
4      for word in list_text:
5          words_length.append(f"{word}:{len(word)}")
6      return words_length
7
8  text = 'H0la a todos como estan'
9  print(run(text))
10
11
12 # DO NOT TOUCH THE CODE BELOW
13 # if __name__ == '__main__':
14 #     import vendor
15 #
16 #     vendor.launch(run)
17
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python311/Scripts/python.exe C:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas-listo/37-attach-len/main.py
['H0la:4', 'a:1', 'todos:5', 'como:4', 'estan:5']
PS C:\Users\HP\Documents\4to Semestre>

```

38. reversing-words El código invierte el orden de las palabras en una cadena de texto. Por ejemplo, para la entrada "Hola como estan", devuelve "estan como Hola".

```

4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos
1
2  def run(text: str) -> str:
3      list_cadena = text.split()
4      reversing = list_cadena[::-1]
5      return ' '.join(reversing)
6
7  text = 'Hola como estan'
8  print(run(text))
9
10 # # DO NOT TOUCH THE CODE BELOW
11 # if __name__ == '__main__':
12 #     import vendor
13 #
14 #     vendor.launch(run)
15

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python311/Scripts/python.exe C:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas-listo/38-reversing-words/main.py
estan como Hola
PS C:\Users\HP\Documents\4to Semestre>

```

39. barycenter El código calcula el baricentro (centroide) de un triángulo dado por tres puntos (A), (B) y (C) en un plano. El baricentro se encuentra promediando las coordenadas (x) y (y) de los tres vértices. En el ejemplo, los puntos son (A = [4, 6]), (B = [12, 4]), y (C = [10, 10]), y el baricentro calculado es aproximadamente (8.67, 6.67).

```

main.py U X
4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Pr
1 def run(A: list, B: list, C: list) -> tuple:
2     xP = round((int(A[0])+int(B[0])+int(C[0]))/3,4)
3     yP = round((int(A[1])+int(B[1])+int(C[1]))/3,4)
4     return xP, yP
5
6 A = [4,6]
7 B = [12,4]
8 C = [10,10]
9 xP,yP = run(A,B,C)
10 print(f'El baricentro es: ({xP},{yP})')
11
12
13
14 # # DO NOT TOUCH THE CODE BELOW
15 # if __name__ == '__main__':
16 #     import vendor
17 #
18 #     vendor.launch(run)
19
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python311/Python.exe -i
:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas-listo/39-barycenter/main.py"
El baricentro es: (8.6667,6.6667)
PS C:\Users\HP\Documents\4to Semestre>

```

40. flatten-list-deep El código aplana una lista que puede contener sublistas de cualquier profundidad, es decir, convierte todas las sublistas anidadas en una única lista plana. En el ejemplo dado, la lista [3, [6, [4, 7], 9], 12] se convierte en [3, 6, 4, 7, 9, 12].

```

main.py U X
4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3
1 def run(items: list) -> list:
2     flatten_items = []
3     for elemento in items:
4         if isinstance(elemento, list):
5             flatten_items.extend(run(elemento))
6         else:
7             flatten_items.append(elemento)
8     return flatten_items
9
10 items = [3, [6, [4, 7], 9], 12]
11 print(run(items))
12
13 # # DO NOT TOUCH THE CODE BELOW
14 # if __name__ == '__main__':
15 #     import vendor
16 #
17 #     vendor.launch(run)
18
19
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python311/Python.exe -i
:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas-listo/41-flatten-list-deep/main.py"
[3, 6, 4, 7, 9, 12]
PS C:\Users\HP\Documents\4to Semestre>

```

41. first-duplicated El código encuentra y devuelve el primer valor duplicado en una lista. Utiliza un conjunto para rastrear los elementos que ya han aparecido y, en cuanto encuentra un valor repetido, lo devuelve. En el ejemplo dado, el primer valor repetido es `3`.

```
4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos
1  def run(items: list) -> int | None:
2      set_rep = set()
3      for i in range(len(items)):
4          if items[i] not in set_rep:
5              set_rep.add(items[i])
6          else:
7              fdup = items[i]
8              return fdup
9      return None
10
11 items = [3,2,3,1,5,6,7]
12 print(f"primer valor repetido: {run(items)}")
13
14
15 # DO NOT TOUCH THE CODE BELOW
16 # if __name__ == '__main__':
17 #     import vendor
18
19 #     vendor.launch(run)
20
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS GITLENS

```
PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python311/Scripts/python.exe C:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas-listo/42-first-duplicated/main.py
primer valor repetido: 3
PS C:\Users\HP\Documents\4to Semestre>
```

42. fill-values El código completa una lista insertando los números faltantes en secuencia. Si algún número en la secuencia no está en la lista, lo agrega en su lugar correspondiente. En el ejemplo dado, la lista `[0, 1, 3, 5]` se completa para incluir los números faltantes, resultando en `[0, 1, 2, 3, 4, 5]`.

```
4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos >
1  def run(values: list) -> list:
2      for i in range(len(values)+1):
3          if values[i] != i:
4              values.insert(i,i)
5      return values
6
7  values = [0,1,3,5]
8  print(f"lista completada: {run(values)}")
9
10
11 # DO NOT TOUCH THE CODE BELOW
12 # if __name__ == '__main__':
13 #     import vendor
14
15 #     vendor.launch(run)
16
```

PROBLEMS OUTPUT DEBUG CONSOLE **TERMINAL** PORTS GITLENS

```
PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python311/Scripts/python.exe C:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/listas-listo/43-fill-values/main.py
lista completada: [0, 1, 2, 3, 4, 5]
PS C:\Users\HP\Documents\4to Semestre>
```


43. **frange** El código genera una lista de números flotantes que empieza en un valor dado (`start`), termina en otro valor (`stop`), y avanza en incrementos definidos por un paso (`step`). En este ejemplo, la lista generada desde `0` hasta `1` con un paso de `0.21` es `[0, 0.21, 0.42, 0.63, 0.84]`.

```

4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > listas-lis
1  def run(start: float, stop: float, step: float) -> list:
2      values = list()
3      value = start
4      while value <= stop:
5          values.append(value)
6          value += step
7      return values
8
9  start = 0
10 stop = 1
11 step = 0.21
12
13 print(f'lista frange: {run(start,stop,step)}')
14
15 # # DO NOT TOUCH THE CODE BELOW
16 # if __name__ == '__main__':
17 #     import vendor
18 #
19 #     vendor.launch(run)
20
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/P
:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicac
_Datos/Practica3/listas-listo/44-frange/main.py"
lista frange: [0, 0.21, 0.42, 0.63, 0.84]
PS C:\Users\HP\Documents\4to Semestre>

```

44. **mul-matrix** El código realiza la multiplicación de dos matrices AAA y BBB. Primero verifica si el número de columnas de AAA coincide con el número de filas de BBB, ya que esta es una condición necesaria para realizar la multiplicación de matrices. Si la condición no se cumple, devuelve None. Si la condición se cumple, el código calcula el producto de matrices y devuelve el resultado.

```

4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica
1  def run(A: list, B: list) -> list | None:
2      if len(A[0]) != len(B):
3          return None
4      P = [[0 for _ in range(len(B[0]))] for _ in range(len(A))]
5
6      for i in range(len(A)):
7          for j in range(len(B[0])):
8              for k in range(len(B)): #
9                  P[i][j] += A[i][k] * B[k][j]
10     return P
11
12 A = [[1, 2, 3],
13      [4, 5, 6]]
14
15 B = [[7, 8],
16      [9, 10],
17      [11, 12]]
18
19 resultado = run(A, B)
20
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS  GITLENS

PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/P
:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de
_Datos/Practica3/listas-listo/46-mul-matrix/main.py"
Resultado de la multiplicación de matrices:
[58, 64]
[139, 154]
PS C:\Users\HP\Documents\4to Semestre>

```

DICCIONARIOS

1. **group-words** El código agrupa palabras de una lista según su primera letra y devuelve un diccionario donde las claves son las primeras letras y los valores son listas de palabras que comienzan con esa letra.

```

unmetro, 14 hours ago | 1 author (unmetro)
def run(words: list) -> dict:
    group_words = dict()
    for color in words:
        first_letter = color[0]
        if first_letter not in group_words:
            group_words[first_letter] = []
        group_words[first_letter].append(color)
    return group_words

list_colors = ['blue', 'black', 'red', 'green', 'white']
print(run(list_colors))

# # DO NOT TOUCH THE CODE BELOW
# if __name__ == '__main__':
#     import vendor
#     vendor.launch(run)

```

PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/diccionarios/1-group-words/main.py"

```

{'b': ['blue', 'black'], 'r': ['red'], 'g': ['green'], 'w': ['white']}
PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3>

```

2. **same-dict-values** El código verifica si todos los valores de un diccionario son iguales, devolviendo `True` si todos son iguales o `False` si hay diferencias.

```

unmetro, 14 hours ago | 1 author (unmetro)
def run(items: dict) -> bool:
    first_value = items.get('a')
    for value in items.values():
        if value != first_value:
            return False
    return True

diccionario = {'a':1, 'b':2, 'c':1}
print(f"El diccionario tiene los mismos valores? {run(diccionario)}")

# # DO NOT TOUCH THE CODE BELOW
# if __name__ == '__main__':
#     import vendor
#     vendor.launch(run)

```

PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/diccionarios/2-same-dict-values/main.py"

```

El diccionario tiene los mismos valores? False
PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3>

```


5. **fix-keys** El código limpia las claves de un diccionario eliminando los espacios en blanco de las claves originales y luego crea un nuevo diccionario con las claves modificadas y los valores originales. En el ejemplo, para el diccionario `{"key 1": "value 1", "key 2": "value 2"}`, el código produce un nuevo diccionario `{"key1": "value 1", "key2": "value 2"}`, eliminando los espacios de las claves.

```
4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > diccionarios >
1 def run(items: dict) -> dict:
2     fitems = {}
3     key_values = list(items.keys())
4     for i in range(len(key_values)):
5         key_values[i] = key_values[i].replace(" ", "")
6     for i in range(len(key_values)):
7         fitems[key_values[i]] = items[list(items.keys())[i]]
8     return fitems
9
10
11 # DO NOT TOUCH THE CODE BELOW
12 if __name__ == '__main__':
13     import vendor
14
15     vendor.launch(run)
16     print(run({"key 1": "value 1", "key 2": "value 2"})) # Expected output:
17
```

6. **order-stock** El código verifica si hay suficiente stock de un artículo en un diccionario de inventario. Si el artículo existe y la cantidad requerida está disponible, devuelve `True`; de lo contrario, devuelve `False`. En este ejemplo, el diccionario de inventario es `{'pen': 20, 'cup': 11, 'keyring': 40}`, y se verifica si hay 9 unidades de "cup" (taza) disponibles. El resultado es `True` porque hay 11 unidades y se requieren solo 9.

```
4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > diccionarios > 6-order-stock >
1 def run(stock: dict, merch: str, amount: int) -> bool:
2     if merch not in stock:
3         return False
4     if stock[merch] < amount:
5         return False
6     return True
7
8
9 # DO NOT TOUCH THE CODE BELOW
10 if __name__ == '__main__':
11     import vendor
12
13     vendor.launch(run)
14     dict1 = {'pen': 20, 'cup': 11, 'keyring': 40}
15     print(f'Stock:{dict1} \nArtículo:{'cup'}\nCantidad:{9}\nSalida: {run({'pen': 20, 'cup': 11, 'keyring': 40}, 'cup', 9)})
16
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS Python + v m

```
PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/HP/
Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/diccionarios/6-order-stock/main.py"
You have to create args.py with the main function arguments!
Stock:{'pen': 20, 'cup': 11, 'keyring': 40}
Artículo:cup
Cantidad:9
Salida: True
PS C:\Users\HP\Documents\4to Semestre>
```

13 de octubre de 2024

7. **sort-dict** El código ordena los elementos de un diccionario según sus valores y devuelve una lista de tuplas con las claves y valores ordenados.

```

1
2
3
4 def run(unsorted_items: dict) -> list[tuple]:
5     sorted_items = sorted(unsorted_items.items(),key = lambda item:
6         return sorted_items
7
8     input_dict = {'a': 'two', 'b': 'one', 'c': 'three'}
9     output = run(input_dict)
10    print(output)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ... Python + - [] [X] ... ^ X

```

PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_apli
caciones_para_Analisis_de_Datos\Practica3> & C:\Users\HP\AppData\Local\Programs\Pytho
n\Python312/python.exe "c:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Da
tos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/diccionarios/8-sort-di
ct/main.py"
[('b', 'one'), ('c', 'three'), ('a', 'two')]
PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_apli
caciones_para_Analisis_de_Datos\Practica3>

```

8. **money-back** El código calcula la cantidad de monedas necesarias para devolver un monto específico usando una lista de denominaciones disponibles y devuelve un diccionario con el número de cada denominación utilizada.

```

1 unmetro, 14 hours ago + tabs
2 def run(to_give_back: float, available_currencies: list) -> dict | None:
3     available_currencies.sort(reverse=True)
4     money_back = {}
5
6     for currency in available_currencies:
7         count = int(to_give_back // currency)
8         if count > 0:
9             money_back[currency] = count
10            to_give_back -= count * currency
11        if to_give_back == 0:
12            break
13
14    if to_give_back != 0:
15        return None
16
17    return money_back
18
19 # Ejemplo de uso
20 to_give_back = 7
21 available_currencies = [2, 1, 0.5]
22 result = run(to_give_back, available_currencies)
23 print(result)
24
25

```

9. money-back-max El código calcula el cambio necesario utilizando un número máximo permitido de cada denominación disponible y devuelve un diccionario con la cantidad de cada denominación utilizada para alcanzar el monto deseado.

```

1  unmetro, 14 hours ago | 1 author (unmetro)
2  unmetro, 14 hours ago • tareas
3  def run(to_give_back: float, available_currencies: dict) -> dict | None:
4      available_denominations = sorted(available_currencies.keys(), reverse=True)
5      money_back = {}
6
7      for currency in available_denominations:
8          max_units = available_currencies[currency]
9          count = int(min(to_give_back // currency, max_units))
10
11         if count > 0:
12             money_back[currency] = count
13             to_give_back -= count * currency
14
15         if to_give_back == 0:
16             break
17
18         if to_give_back != 0:
19             return None
20
21         if not money_back:
22             return {}
23
24         return money_back
25
26 to_give_back = 7
27 available_currencies = {2: 2, 1: 1, 0.5: 8}
28 result = run(to_give_back, available_currencies)
29 print(result)

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS ... Python + - [] ...

PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3> & C:\Users\HP\AppData\Local\Programs\Python\Python312\python.exe "c:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/diccionarios/10-money-back-max/main.py"

{2: 2, 1: 1, 0.5: 4}

PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3>

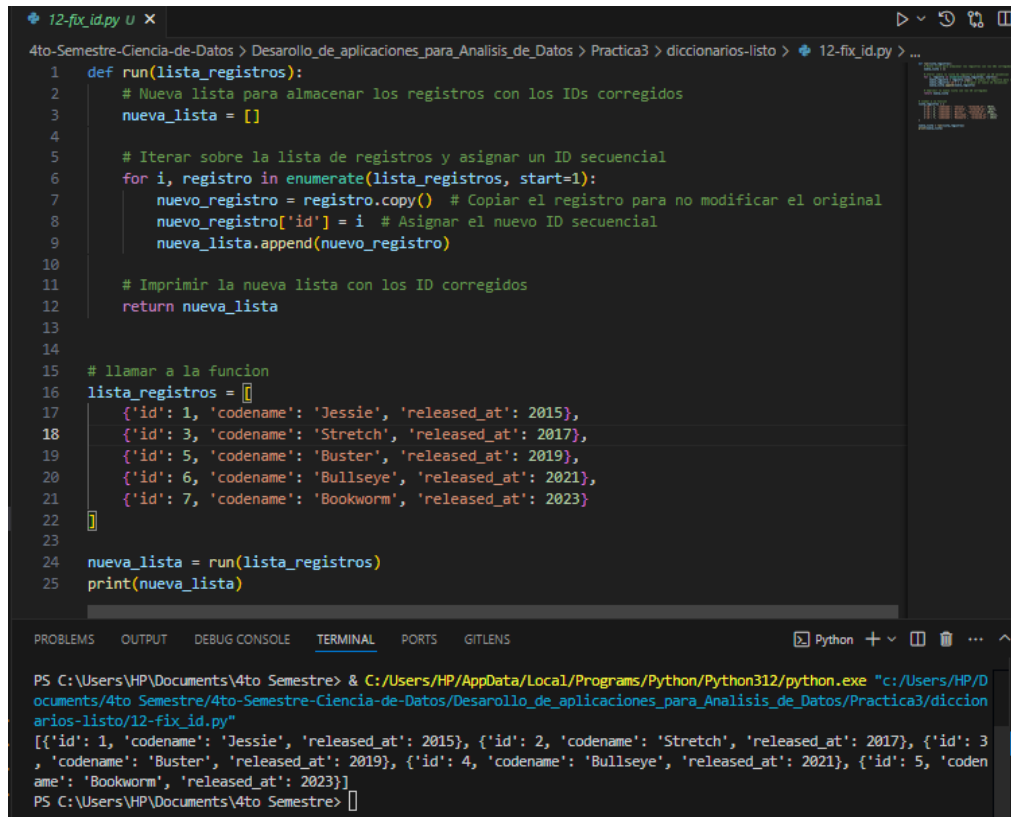
10. first-ntimes El código cuenta las ocurrencias de cada número en una lista y devuelve el primer número que aparece exactamente "n" veces; si no lo encuentra, devuelve -1.

```

1  def run(lista, n):
2      # Crear un diccionario para contar las ocurrencias
3      ocurrencias = {}
4
5      # Contar las ocurrencias de cada número
6      for num in lista:
7          if num in ocurrencias:
8              ocurrencias[num] += 1
9          else:
10             ocurrencias[num] = 1
11
12     # Buscar el primer número en la lista que aparece n veces
13     for num in lista:
14         if ocurrencias[num] == n:
15             return num
16
17     # Si no se encuentra ningún número que s
18     return -1
19
20
21 # llamar a la funcion
22 print(run([2, 3, 5, 3, 2, 1, 8, 2], 3))

```

11. **fix-id** El código reasigna un ID secuencial a cada registro en una lista de diccionarios, comenzando desde 1, y devuelve una nueva lista con los registros actualizados.



```

1  def run(lista_registros):
2      # Nueva lista para almacenar los registros con los IDs corregidos
3      nueva_lista = []
4
5      # Iterar sobre la lista de registros y asignar un ID secuencial
6      for i, registro in enumerate(lista_registros, start=1):
7          nuevo_registro = registro.copy() # Copiar el registro para no modificar el original
8          nuevo_registro['id'] = i # Asignar el nuevo ID secuencial
9          nueva_lista.append(nuevo_registro)
10
11     # Imprimir la nueva lista con los ID corregidos
12     return nueva_lista
13
14
15 # llamar a la funcion
16 lista_registros = [
17     {'id': 1, 'codename': 'Jessie', 'released_at': 2015},
18     {'id': 3, 'codename': 'Stretch', 'released_at': 2017},
19     {'id': 5, 'codename': 'Buster', 'released_at': 2019},
20     {'id': 6, 'codename': 'Bullseye', 'released_at': 2021},
21     {'id': 7, 'codename': 'Bookworm', 'released_at': 2023}
22 ]
23
24 nueva_lista = run(lista_registros)
25 print(nueva_lista)

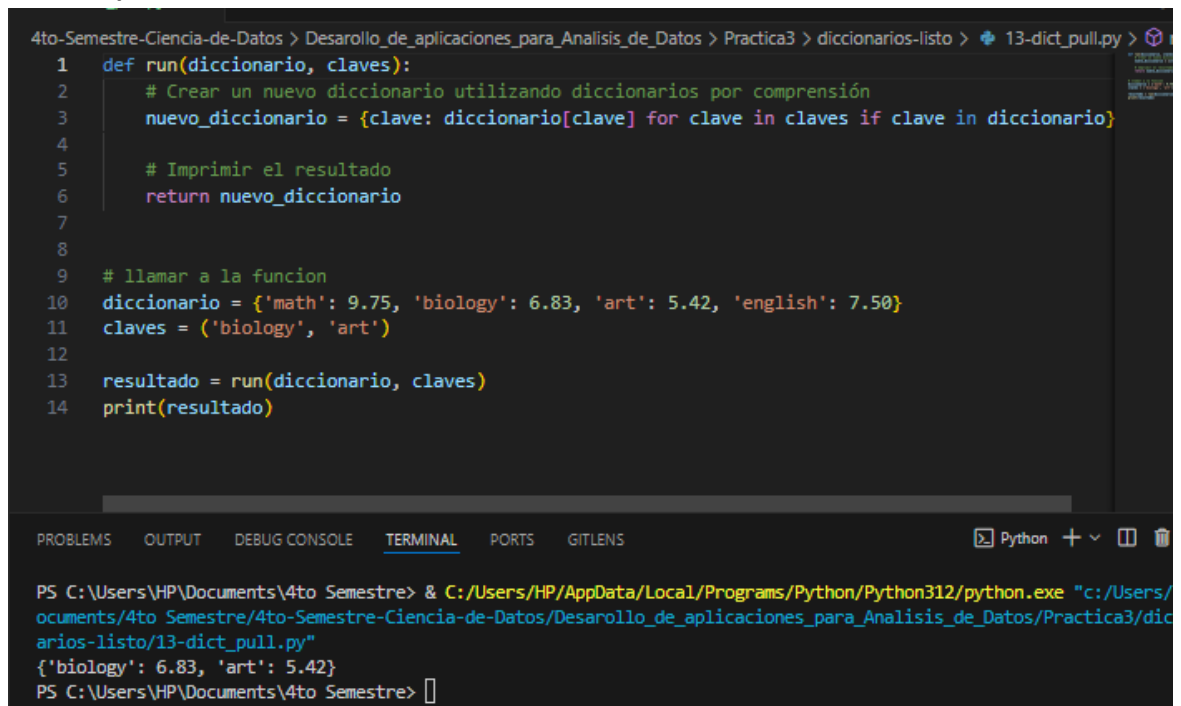
```

```

PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/HP/Doc
uments/4to Semestre/4to Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/diccio
narios-listo/12-fix_id.py"
[{'id': 1, 'codename': 'Jessie', 'released_at': 2015}, {'id': 2, 'codename': 'Stretch', 'released_at': 2017}, {'id': 3
, 'codename': 'Buster', 'released_at': 2019}, {'id': 4, 'codename': 'Bullseye', 'released_at': 2021}, {'id': 5, 'coden
ame': 'Bookworm', 'released_at': 2023}]
PS C:\Users\HP\Documents\4to Semestre>

```

12. **dict-pull** El código extrae un subconjunto de un diccionario original utilizando una lista de claves específicas, devolviendo un nuevo diccionario con solo esos elementos.



```

1  def run(diccionario, claves):
2      # Crear un nuevo diccionario utilizando diccionarios por comprensión
3      nuevo_diccionario = {clave: diccionario[clave] for clave in claves if clave in diccionario}
4
5      # Imprimir el resultado
6      return nuevo_diccionario
7
8
9 # llamar a la funcion
10 diccionario = {'math': 9.75, 'biology': 6.83, 'art': 5.42, 'english': 7.50}
11 claves = ('biology', 'art')
12
13 resultado = run(diccionario, claves)
14 print(resultado)

```

```

PS C:\Users\HP\Documents\4to Semestre> & C:/Users/HP/AppData/Local/Programs/Python/Python312/python.exe "c:/Users/
documents/4to Semestre/4to Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/dic
narios-listo/13-dict_pull.py"
{'biology': 6.83, 'art': 5.42}
PS C:\Users\HP\Documents\4to Semestre>

```

SETS

1. is-binary chequea si el numero dado es un numero binario o no

```
4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos
1  def run(number: str) -> bool:
2      number_set = {1,0}
3      for c in number:
4          if int(c) not in number_set:
5              return False
6      return True
7
8  number = input("introduzca un numero binario: ")
9  print(f"Es numero es binario?: {run(number)}")
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

```
on.exe "c:/Users/HP/Documents/4to Semestre/4to-Semestre-Ciencia-de-Datos/Practica3/sets/main.py"
introduzca un numero binario: 0110101
Es numero es binario?: True
PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3>
```

FICHEROS

1. wc El código abre un archivo de texto, cuenta el número de líneas, palabras y bytes, y muestra estos valores, manejando un error si el archivo no se encuentra.

```
4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > ficheros > wc
1  def run(input_path: str) -> tuple:
2      try:
3          with open(input_path, 'r', encoding='utf-8') as archivo:
4              lineas = archivo.readlines()
5              num_lines = len(lineas)
6              num_words = sum(len(linea.split()) for linea in lineas)
7              num_bytes = sum(len(linea.encode('utf-8')) for linea in lineas)
8
9              return num_lines, num_words, num_bytes
10     except FileNotFoundError:
11         print(f"El archivo {input_path} no fue encontrado")
12
13     ruta_archivo = 'C:/Users/Bjsan/OneDrive/Escritorio/4semestre/CAAD/Python/ficheros/ficheros.txt'
14     resultado = run(ruta_archivo)
15
16     if resultado is not None:
17         num_lineas, num_palabras, num_bytes = resultado
18         print(f"Número de líneas: {num_lineas}")
19         print(f"Número de palabras: {num_palabras}")
20         print(f"Número de bytes: {num_bytes}")
21
```


2. read-csv El código lee un archivo CSV, convierte los valores de cada fila a su tipo correspondiente (enteros, booleanos o `None`), y devuelve una lista de diccionarios donde cada fila del CSV es un diccionario con claves y valores convertidos según su tipo. Luego imprime cada fila convertida.

```
4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > ficheros > read-csv > main.py > ...
1  import csv
2  def convertir_valor(valor):
3      if valor == '':
4          return None
5      elif valor == 'True':
6          return True
7      elif valor == 'False':
8          return False
9      try:
10         return int(valor)
11     except ValueError:
12         return valor
13
14  def run(datafile: str) -> list:
15      with open(datafile, mode='r', encoding='utf-8') as archivo:
16          lector_csv = csv.DictReader(archivo)
17          lista_diccionarios = []
18          for fila in lector_csv:
19              fila_convertida = {clave: convertir_valor(valor) for clave, valor in fila.items()}
20              lista_diccionarios.append(fila_convertida)
21
22      return lista_diccionarios
23
24  ruta_csv = 'C:/Users/Bjsan/Downloads/AbandonoEmpleados.csv'
25  resultado = run(ruta_csv)
26
27  for fila in resultado:
28      print(fila)
29
30  # # DO NOT TOUCH THE CODE BELOW
```

3. txt2md El código convierte un archivo de texto con guiones y tabulaciones en formato de encabezados de Markdown. Los guiones o tabulaciones indican el nivel del encabezado en Markdown. El código lee el archivo de entrada, transforma las líneas según el formato de encabezado de Markdown, y luego escribe el resultado en un archivo de salida.
 - Los niveles de encabezado se determinan por la cantidad de tabulaciones o espacios al inicio de cada línea.
 - La función añade el símbolo `#` seguido del título correspondiente según el nivel de indentación detectado.El archivo `prueba.txt` es leído, transformado y luego guardado como `guion.txt`.

```

def convertir_guion_a_markdown(texto: str) -> str:
    lineas = texto.splitlines()
    markdown = []

    for linea in lineas:
        nivel = 0
        while linea.startswith('\t') or linea.startswith('  '):
            if linea.startswith('\t'):
                nivel += 1
                linea = linea[1:]
            elif linea.startswith('  '):
                nivel += 1
                linea = linea[4:]

        titulo = linea.strip()
        if titulo:
            encabezado = '#' * (nivel + 1) + ' ' + titulo
            markdown.append(encabezado)

    return '\n'.join(markdown)

def run(input_path: str, output_path: str) -> None:
    with open(input_path, 'r', encoding='utf-8') as f:
        texto = f.read()

    markdown = convertir_guion_a_markdown(texto)

    with open(output_path, 'w', encoding='utf-8') as f:
        f.write(markdown)

input_path = 'C:/Users/Bjsan/OneDrive/Escritorio/4semestre/CAAD/Python/ficheros/txt2md/prueba.t
output_path = 'C:/Users/Bjsan/OneDrive/Escritorio/4semestre/CAAD/Python/ficheros/txt2md/guion.t

run(input_path, output_path)

```

4. avg-temps El código lee un archivo de temperaturas mensuales, calcula el promedio de cada mes y guarda los resultados en un archivo de salida.

```

Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > ficheros > avg-temps > main.py >
def leer_temperaturas(input_path: str) -> list:
    with open(input_path, 'r', encoding='utf-8') as f:
        lineas = f.readlines()
        temperaturas_por_mes = []

    for linea in lineas:
        temperaturas = list(map(float, linea.strip().split(',')))
        temperaturas_por_mes.append(temperaturas)

    return temperaturas_por_mes

def calcular_media_mensual(temperaturas_por_mes: list) -> list:
    medias_mensuales = []

    for temperaturas in temperaturas_por_mes:
        media = sum(temperaturas) / len(temperaturas)
        medias_mensuales.append(media)

    return medias_mensuales

def escribir_medias(medias_mensuales: list, output_path: str) -> None:
    with open(output_path, 'w', encoding='utf-8') as f:
        for media in medias_mensuales:
            f.write(f"{media:.2f}\n")

def run(input_path: str, output_path: str) -> None:
    temperaturas_por_mes = leer_temperaturas(input_path)
    medias_mensuales = calcular_media_mensual(temperaturas_por_mes)
    escribir_medias(medias_mensuales, output_path)

input_path = 'C:/Users/Bjsan/OneDrive/Escritorio/4semestre/CAAD/Python/ficheros/avg-temps/prueb
output_path = 'C:/Users/Bjsan/OneDrive/Escritorio/4semestre/CAAD/Python/ficheros/avg-temps/fich
run(input_path, output_path)

```

5. **find-words** Este código busca una palabra objetivo en un archivo de texto, utilizando expresiones regulares para encontrar todas las ocurrencias sin importar las mayúsculas o minúsculas. Luego, devuelve las posiciones de cada coincidencia, indicando la línea y la columna donde se encuentra la palabra.

```

b-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > ficheros > find-words > main.py > ...
1 import re
2 def run(fichero, palabra_objetivo):
3     ocurrencias = []
4     palabra_objetivo = palabra_objetivo.lower() # Para hacer que la búsqueda no distinga entre mayúsculas
5
6     with open('help.txt', 'r') as archivo:
7         for num_linea, linea in enumerate(archivo, 1): # Enumerar las líneas, empezando por 1
8             palabras = re.finditer(r'\b' + re.escape(palabra_objetivo) + r'\b', linea, re.IGNORECASE)
9             for match in palabras:
10                 # Guardar la tupla (número de línea, índice de columna)
11                 ocurrencias.append((num_linea, match.start() + 1))
12
13     return ocurrencias
14
15 # DO NOT TOUCH THE CODE BELOW
16 if __name__ == '__main__':
17     import vendor
18
19     vendor.launch(run)
20     print(run("data.txt", "you"))

```

6. **sum-matrix** Este código suma dos matrices leídas desde archivos de texto, fila por fila, y luego escribe el resultado en un archivo de salida.

```

def run(matrix1_path: str, matrix2_path: str, result_path: str) -> None:
    """Lee dos matrices desde archivos, las suma y escribe el resultado en un archivo de salida."""

    # Leer las dos matrices desde los archivos
    with open(matrix1_path, 'r') as archivo1, open(matrix2_path, 'r') as archivo2:
        matriz1 = [list(map(int, linea.split())) for linea in archivo1]
        matriz2 = [list(map(int, linea.split())) for linea in archivo2]

    # Sumar las matrices
    matriz_resultado = []
    for fila1, fila2 in zip(matriz1, matriz2):
        fila_resultado = [x + y for x, y in zip(fila1, fila2)]
        matriz_resultado.append(fila_resultado)

    # Escribir la matriz resultante en el archivo de salida
    with open(result_path, 'w') as archivo_salida:
        for fila in matriz_resultado:
            archivo_salida.write(' '.join(map(str, fila)) + '\n')

    print(f"La suma de las matrices se ha guardado en {result_path}")

# Ejemplo de uso
matrix1_path = 'matriz1.txt' # Cambia esto por la ruta de tu archivo de la primera matriz
matrix2_path = 'matriz2.txt' # Cambia esto por la ruta de tu archivo de la segunda matriz
result_path = 'resultado.txt' # Archivo de salida

# Llamar a la función con los archivos correspondientes
run(matrix1_path, matrix2_path, result_path)

```

7. **longest-word** Este código lee un archivo de texto, elimina caracteres especiales, encuentra las palabras más largas, y luego selecciona la más frecuente entre ellas para devolverla como resultado.

```

semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > ficheros > longest-word > main.py > ...
import re
from collections import Counter

def run(input_path: str) -> str:
    """Encuentra la palabra más larga en un fichero de texto, considerando caracteres especiales."""

    # Definir los caracteres frontera de palabras que se deben eliminar: . , ; : ( )
    frontera = r"[.,;:()]"

    # Leer el contenido del archivo
    with open(input_path, 'r') as archivo:
        contenido = archivo.read()

    # Eliminar los caracteres especiales definidos en 'frontera'
    contenido_limpio = re.sub(frontera, '', contenido)

    # Dividir el texto en palabras
    palabras = contenido_limpio.split()

    # Crear un diccionario para contar las frecuencias de las palabras
    frecuencias = {}

    for palabra in palabras:
        if palabra in frecuencias:
            frecuencias[palabra] += 1
        else:
            frecuencias[palabra] = 1

    # Filtrar solo las palabras más largas
    max_longitud = max(len(palabra) for palabra in palabras)
    palabras_mas_largas = [palabra for palabra in palabras if len(palabra) == max_longitud]

    # Elegir la palabra más frecuente entre las palabras más largas
    palabra_mas_larga = palabras_mas_largas[0] # Inicializar con la primera palabra más larga
    max_frecuencia = frecuencias[palabra_mas_larga] # Frecuencia de la primera palabra

    for palabra in palabras_mas_largas:
        if frecuencias[palabra] > max_frecuencia:
            palabra_mas_larga = palabra
            max_frecuencia = frecuencias[palabra]

    return palabra_mas_larga

# DO NOT TOUCH THE CODE BELOW
if __name__ == '__main__':
    import vendor

    vendor.launch(run)
    input_path = 'texto.txt'
    resultado = run('java.txt')
    print(f"La palabra más larga es: {resultado}")

```

8. **word-freq** El código cuenta la frecuencia de cada palabra en un archivo de texto y devuelve un diccionario con las palabras que aparecen más veces que un umbral especificado (`lower_bound`).

13 de octubre de 2024

```

1
2
3 def run(input_path: str, lower_bound: int) -> dict:
4     freq = {}
5     try:
6         with open(input_path, 'r', encoding='utf-8') as file:
7             content = file.read().lower().split()
8             for word in content:
9                 if word in freq:
10                     freq[word] += 1
11                 else:
12                     freq[word] = 1
13             freq = {word: count for word, count in freq.items() if count >= lower_bound}
14     except FileNotFoundError:
15         print(f"El archivo {input_path} no se encontró.")
16     except Exception as e:
17         print(f"Ocurrió un error: {e}")
18     return {}
19
20
21 # llamada a la función con un archivo de ejemplo y un valor umbral
22 resultado = run("ruta_del_archivo.txt", 2)
23 print(resultado)
24

```

9. **get-line** El código lee una línea específica de un archivo de texto y devuelve su contenido o `None` si la línea no existe o el archivo no se encuentra.

```

1  def run(input_path: str, line_no: int) -> str | None:
2      try:
3          with open(input_path, 'r', encoding='utf-8') as file:
4              lines = file.readlines()
5              if 1 <= line_no <= len(lines):
6                  return lines[line_no - 1].rstrip()
7              else:
8                  return None
9          except FileNotFoundError:
10             return None
11
12 resultado = run("ruta/del/archivo.txt", 3)
13 print(resultado)
14
15

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS

PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3> C:\Users\HP\AppData\Local\Programs\Python\Python312\python.exe "c:/Users/HP/4to-Semestre-Ciencia-de-Datos/Desarrollo_de_aplicaciones_para_Analisis_de_Datos/Practica3/ficheros/main.py"

None

PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3>

10. **replace-chars** El código reemplaza caracteres específicos en el contenido de un archivo de texto y guarda el resultado en otro archivo de salida, manejando posibles errores si el archivo de entrada no se encuentra.

```

1
2
3 def run(input_path: str, replacements: str, output_path: str) -> None:
4     replacement_dict = {pair[0]: pair[1] for pair in replacements.split('|')}
5     try:
6         with open(input_path, 'r', encoding='utf-8') as infile:
7             content = infile.read()
8             for old_char, new_char in replacement_dict.items():
9                 content = content.replace(old_char, new_char)
10            with open(output_path, 'w', encoding='utf-8') as outfile:
11                outfile.write(content)
12    except FileNotFoundError:
13        print(f"El archivo {input_path} no se encontró.")
14    except Exception as e:
15        print(f"Ocurrió un error: {e}")
16
17 run('entrada.txt', 'aA|bB|cC', 'salida.txt')
18

```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL PORTS GITLENS Python + -

PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_Datos\Practica3> & C:\Users\HP\AppData\Local\Programs\Python\Python312\python.exe "c:\Users\HP\Documents\4to-Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_de_Datos\Practica3\archivos\replace-chars/main.py"

El archivo entrada.txt no se encontró.

PS C:\Users\HP\Documents\4to Semestre\4to-Semestre-Ciencia-de-Datos\Desarrollo_de_aplicaciones_para_Analisis_Datos\Practica3> █

11. **histogram-txt** El código lee un archivo de texto, cuenta la frecuencia de cada letra, y luego genera un histograma en un archivo de salida donde cada letra aparece seguida de un número de barras igual a su frecuencia.

```

4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > ficheros > 11-histogram
1 def run(fichero_entrada, fichero_salida):
2     # Diccionario para almacenar las frecuencias de cada letra
3     frecuencias = {}
4
5     # Leer el contenido del fichero de entrada
6     with open(fichero_entrada, 'r', encoding='utf-8', errors='ignore') as archivo:
7         contenido = archivo.read()
8
9     # Contar la frecuencia de cada letra
10    for letra in contenido:
11        if letra.isalpha(): # Solo contar letras
12            letra = letra.upper() # Convertir todo a mayúsculas
13            if letra in frecuencias:
14                frecuencias[letra] += 1
15            else:
16                frecuencias[letra] = 1
17
18    # Crear el fichero de salida con el histograma
19    with open(fichero_salida, 'w', encoding='utf-8') as archivo_salida:
20        for letra, frecuencia in sorted(frecuencias.items()):
21            # Escribir la letra, las barras "█" y el valor de la frecuencia
22            archivo_salida.write(f'{letra} {"█" * frecuencia} {frecuencia}\n')
23
24    # llamar a la funcion
25    run('entrada.txt', 'histograma.txt')
26

```

12. **submarine** Este código simula los movimientos de un submarino a partir de un archivo de entrada que contiene el combustible disponible y los movimientos del submarino. Calcula la distancia recorrida, la profundidad alcanzada y el combustible restante,

asegurándose de no superar ciertos límites de profundidad y verificando que haya suficiente combustible para cada movimiento.

```
4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > ficheros > 12-subma
1 def run(fichero_entrada):
2     # Inicializar variables
3     distancia = 0
4     profundidad = 0
5     with open(fichero_entrada, 'r') as archivo:
6         # Leer la primera línea para obtener los litros de combustible
7         combustible = int(archivo.readline().strip())
8
9         # Leer los movimientos del submarino
10        movimientos = archivo.readline().strip().split(',')
11
12    # Procesar cada movimiento
13    for movimiento in movimientos:
14        # Separar distancia y cambio de profundidad
15        x, y = map(int, movimiento.split(':'))
16
17        # Actualizar la distancia y la profundidad
18        nueva_distancia = distancia + x
19        nueva_profundidad = profundidad + y
20
21        # Validar límites de profundidad
22        if nueva_profundidad < 0:
23            nueva_profundidad = 0 # No puede ascender por encima de la superficie
24        if nueva_profundidad > 600:
25            nueva_profundidad = 600 # No puede descender por debajo de 600
26
27        # Calcular consumo de combustible
28        consumo = 3 * x + 2 * abs(y)
29        if combustible < consumo:
30            break # Si no hay suficiente combustible, se detiene el submarino
31
32        # Actualizar los valores si todo es válido
33        distancia = nueva_distancia
34        profundidad = nueva_profundidad
35        combustible -= consumo
36
37    # Imprimir resultados
38    print(f'Distancia: {distancia}')
39    print(f'Profundidad: {profundidad}')
40    print(f'Combustible restante: {combustible}')
41
42
43    # llamar a la funcion
44    run('entrada_submarino.txt')
```

13. common-words Este código toma dos archivos de entrada y salida. Lee todas las líneas del archivo de entrada, genera combinaciones de las líneas, encuentra las palabras comunes en cada combinación usando conjuntos y escribe la cantidad de palabras comunes en el archivo de salida.

```
4to-Semestre-Ciencia-de-Datos > Desarrollo_de_aplicaciones_para_Analisis_de_Datos > Practica3 > ficheros > 1
1  from itertools import combinations
2
3
4  def run(fichero_entrada, fichero_salida):
5      # Leer todas las líneas del fichero de entrada
6      with open(fichero_entrada, 'r', encoding='utf-8') as archivo:
7          lineas = [line.strip().lower() for line in archivo.readlines()] # Conv
8
9      # Abrir el fichero de salida para escribir los resultados
10     with open(fichero_salida, 'w', encoding='utf-8') as archivo_salida:
11         # Generar todas las combinaciones posibles de dos líneas
12         for (i, linea_a), (j, linea_b) in combinations(enumerate(lineas), 2):
13             # Dividir ambas líneas en palabras
14             palabras_a = set(linea_a.split())
15             palabras_b = set(linea_b.split())
16
17             # Encontrar palabras comunes
18             comunes = palabras_a & palabras_b # Intersección de conjuntos
19
20             # Escribir el número de palabras comunes en el fichero de salida
21             archivo_salida.write(f'{len(comunes)}\n')
22
23
24     # llamar a la funcion
25     run('entrada_palabras.txt', 'salida_palabras.txt')
```

CONCLUSIONES

En esta práctica, adquirí habilidades para utilizar diversas estructuras y funciones avanzadas de Python, tales como listas, diccionarios, sets y el manejo de archivos. Pude aplicar conceptos clave para resolver problemas, como el procesamiento y análisis de datos, desde operaciones básicas con listas, manipulación de cadenas y matrices, hasta la creación de histogramas y la lectura/escritura de archivos. Además, aprendí a utilizar librerías importantes como `csv` y `itertools`, lo que me permitió realizar tareas de manera más eficiente. A través de estos ejercicios, fortalecí mi capacidad para abordar problemas complejos en el ámbito del análisis de datos, mejorando mis competencias en programación y resolución de problemas con Python.