<!-- INF 129 | Paralelo 1 -->

Ayudantía(){

<Por="Mauricio Fernández"/>

}

Contenidos

Mención Ayudantía Pasada{

- Si tenemos listas de listas, al aplicar sort() a la lista que contiene sublistas, esta lista se ordenara tomando en cuenta el primer elemento de cada sublista

```
lista = [[1, 2], [0, 13], [20, 3, 3], [1, 3], [1], [5, 4], [True]]
lista.sort()
print(lista)
```

```
[[0, 13], [1], [True], [1, 2], [1, 3], [5, 4], [20, 3, 3]]
```

}

¡CUIDADO!

```
lista = [[1, 2], ["Patricio Estrella", 1], [3]]
lista.sort() #No puedo comparar un string con un entero, flotante o booleano
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
Cell In[18], line 2
      1 lista = [[1, 2], ["Patricio Estrella", 1], [3]]
----> 2 lista.sort()

TypeError: '<' not supported between instances of 'str' and 'int'
```

}

Diccionarios{

- Estructura de datos(como listas) no ordenada, estructurada en pares de clave-valor.

```
estudiante = {
    "nombre": None,
    "rol" : None,
    "nota_tareas": []
}

estudiante["nombre"] = "Sherlock"
estudiante["rol"] = "202581023-2"

for i in range(4):
    estudiante["nota_tareas"].append(91 + i * 3)
print(estudiante)
```

```
{'nombre': 'Sherlock', 'rol': '202581023-2', 'nota_tareas': [91, 94, 97, 100]}
```

}

Agregar pares clave-valor{

```python
estudiante["notas_TS"] = [100, 100, 100, 100, 100, 100]
print(estudiante)
```

```
{'nombre': 'Sherlock', 'rol': '202581023-2', 'nota_tareas': [91, 94, 97, 100], 'notas_TS': [100, 100, 100, 100, 100, 100], 2: ['ramo']}
```

```python
estudiante[2] = ["ramo"] #Salvedades
print(estudiante)
```

```
{'nombre': 'Sherlock', 'rol': '202581023-2', 'nota_tareas': [91, 94, 97, 100], 'notas_TS': [100, 100, 100, 100, 100, 100], 2: ['ramo']}
```

}

Modificar valores de una clave{

```
estudiante["nota_tareas"] = [100] * 4
print(estudiante)
```

```
{'nombre': 'Sherlock', 'rol': '202581023-2', 'nota_tareas': [100, 100, 100, 100], 'notas_TS': [100, 100, 100, 100, 100, 100], 2: ['ramo']}
```

}

Eliminar elementos (par clave-valor){

```
    del estudiante[2]
    del estudiante["notas_TS"]
    print(estudiante)
```

```
{'nombre': 'Sherlock', 'rol': '202581023-2', 'nota_tareas': [100, 100, 100, 100]}
```

}

Recorrido diccionario{

```
    for clave in estudiante: #Recorro el diccionario
        print(clave, ":", estudiante[clave])

    for nota in estudiante["nota_tareas"]: #recorro la lista asociada a la clave "notas_tareas" del diccionario
        print(nota)
]
```

```
nombre : Sherlock
rol : 202581023-2
nota_tareas : [100, 100, 100, 100]
100
100
100
100
```

}

Típico Caso Certamen{

```python
inscripciones = ["MAT071", "MAT021", "FIS120", "FIS130",
                 "INF253", "QUI010", "ELI015", "MAT021",
                 "MAT071", "QUI010", "FIS130", "MAT022", "MAT021",
                 "FIS130", "ELI015", "MAT061", "QUI010"]
pae = {}

for inscripcion_pae in inscripciones:
    if inscripcion_pae not in pae: #Consulto si la llave se encuentra o no en el diccionario
        pae[inscripcion_pae] = 0 #¿Por qué la inicializo en 0 y no en 1?
    pae[inscripcion_pae] += 1

print(pae)
```

```
{'MAT071': 2, 'MAT021': 3, 'FIS120': 1, 'FIS130': 3, 'INF253': 1, 'QUI010': 3, 'ELI015': 2, 'MAT022': 1, 'MAT061': 1}
```

}

Errores{

Pecado capital: agregar/eliminar un iterable mientras se recorre.

¿Una lista puede "parchar" este comportamiento de diccionarios?

```
    for ramo in pae: #Error no trivial, ¿Cual es la diferencia con listas?
        if pae[ramo] < 20:
            del pae[ramo]

    print(pae)
```

```
---------------------------------------------------------------------------
RuntimeError                              Traceback (most recent call last)
Cell In[71], line 1
----> 1 for ramo in pae:
      2     if pae[ramo] < 20:
      3         del pae[ramo]

RuntimeError: dictionary changed size during iteration
```

}

Errores {

- La respuesta es un si, pero con consideraciones. Si bien no tira error, no es recomendable, dado que pueden surgir comportamientos inesperados

```
    numeros = [1, 2, 2, 3, 4, 5, 5, 6]

    for num in numeros:
        if num % 2 == 0:
            numeros.remove(num)

    print(numeros)


[1, 2, 3, 5, 5]
```

}

¿Como filtrar entonces?

{

- La respuesta depende. Generalmente podemos crear una lista vacia y agregar los elementos del iterable con datos

```
boletas = [200, 120, 300, 70, 80, 21]
#Digamos que cualquier boleta sobre los 100 pesos, debe pagar + intereses
#Y queremos informar mediante código todas las boletas que pagan + intereses
aux = []

for boleta in boletas:
    if boleta > 100:
        aux.append(boleta)

print(aux)
```

```
[200, 120, 300]
```

}

Diccionario a lista {

```python
inf129y_asistencia = {
    "Malcom": 10,
    "Dui": 10,
    "Lois": 8,
    "Reese": 2,
}
lista_aux = []
for alumno in inf129y_asistencia: #recordar que la variable del for toma solo la clave del dict
    lista_aux.append([alumno, inf129y_asistencia[alumno]])

print(lista_aux)
```

```
[['Malcom', 10], ['Dui', 10], ['Lois', 8], ['Reese', 2]]
```

}

<!-- INF 129 | Paralelo 1 -->

Gracias {

<Por="Mauricio Fernández"/>

}