REPASO MÓDULO 3 IP

FELIPE

CONTEXTO

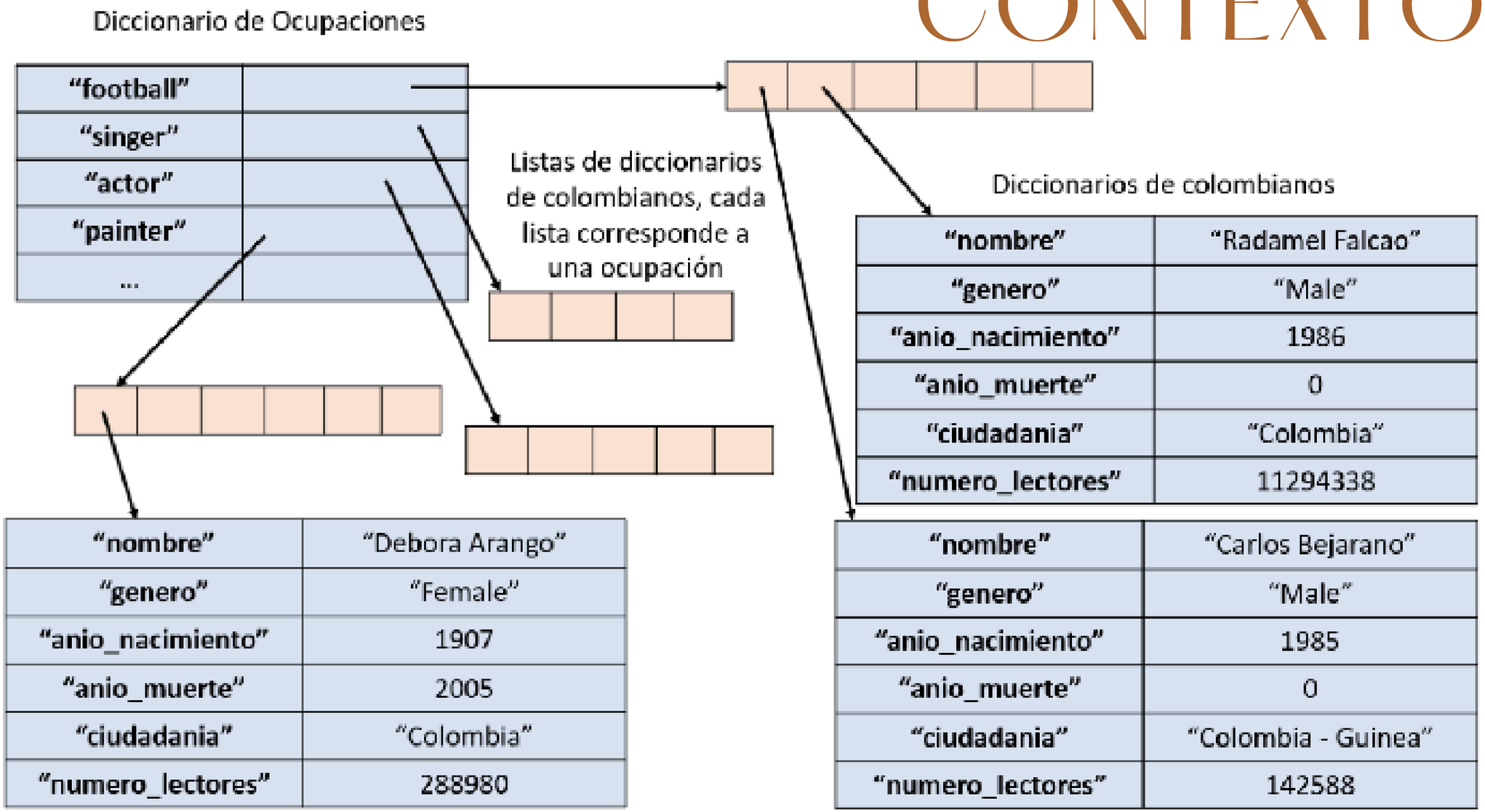

Figura 1. Ejemplo de la estructura deseada para el manejo de los datos del archivo *.csv*

01.

Función 1:

Implemente una función que reciba como parámetro el nombre de un archivo que contiene la información de los colombianos en Wikipedia y la cargue en el programa bajo la forma de **un diccionario de listas de diccionarios**. Cada colombiano se va a representar utilizando un diccionario que tiene las siguientes llaves:

Llave	Tipo de dato	Descripción de la llave
nombre	str	Nombre del colombiano.
genero	str	Género con el que se identifica el colombiano. En este caso ("Male" o "Female").
anio_nacimiento	int	Año de nacimiento.
anio_muerte	int	Año de muerte. En caso, de no haber muerto aún, este campo tendrá el valor cero (0).
ciudadania	str	Nombre del país en el que la persona tiene nacionalidad. En este caso, todas las personas tienen nacionalidad colombiana, pero algunas podrían tener también una segunda nacionalidad en algún otro país.
numero_lectores	int	Número de personas que han leído la información del colombiano en Wikipedia.

Esta función debe retornar el diccionario que tiene como llave cada ocupación y como valor asociado su respectiva lista de colombianos (de diccionarios) creada. La estructura objetivo se muestra en la Figura 1.

01.

Función 1:
Implemente una función que reciba como parámetro el nombre de un archivo que contiene la información de los colombianos en Wikipedia y la cargue en el programa bajo la forma de **un diccionario de listas de diccionarios**. Cada colombiano se va a representar utilizando un diccionario que tiene las siguientes llaves:

Llave	Tipo de dato	Descripción de la llave
nombre	str	Nombre del colombiano.
genero	str	Género con el que se identifica el colombiano. En este caso (“Male” o “Female”).
anio_nacimiento	int	Año de nacimiento.
anio_muerte	int	Año de muerte. En caso, de no haber muerto aún, este campo tendrá el valor cero (0).
ciudadania	str	Nombre del país en el que la persona tiene nacionalidad. En este caso, todas las personas tienen nacionalidad colombiana, pero algunas podrían tener también una segunda nacionalidad en algún otro país.
numero_lectores	int	Número de personas que han leído la información del colombiano en Wikipedia.

Esta función debe retornar el diccionario que tiene como llave cada ocupación y como valor asociado su respectiva lista de colombianos (de diccionarios) creada. La estructura objetivo se muestra en la Figura 1.

```python
def cargar_datos(ruta_archivo: str)->dict:
    ocupaciones={}
    archivo=open(ruta_archivo)
    titulos =archivo.readline().split(",")
    linea=archivo.readline()
    while len(linea) > 0:
        datos = linea.split(",")
        colombiano={}
        colombiano["nombre"] = datos[0]
        colombiano["genero"] = datos[1]
        colombiano["anio_nacimiento"] = int(datos[2])
        colombiano["anio_muerte"] = int(datos[3])
        colombiano["ciudadania"] = datos[5]
        colombiano["numero_lectores"] = int(datos[6])
        if datos[4] not in ocupaciones:
            ocupaciones[datos[4]] = []
        ocupaciones[datos[4]].append(colombiano)
        linea =archivo.readline()
    archivo.close()
    return ocupaciones
```

01.

Función 1:
Implemente una función que reciba como parámetro el nombre de un archivo que contiene la información de los colombianos en Wikipedia y la cargue en el programa bajo la forma de **un diccionario de listas de diccionarios**. Cada colombiano se va a representar utilizando un diccionario que tiene las siguientes llaves:

Llave	Tipo de dato	Descripción de la llave
nombre	str	Nombre del colombiano.
genero	str	Género con el que se identifica el colombiano. En este caso ("Male" o "Female").
anio_nacimiento	int	Año de nacimiento.
anio_muerte	int	Año de muerte. En caso, de no haber muerto aún, este campo tendrá el valor cero (0).
ciudadania	str	Nombre del país en el que la persona tiene nacionalidad. En este caso, todas las personas tienen nacionalidad colombiana, pero algunas podrían tener también una segunda nacionalidad en algún otro país.
numero_lectores	int	Número de personas que han leído la información del colombiano en Wikipedia.

Esta función debe retornar el diccionario que tiene como llave cada ocupación y como valor asociado su respectiva lista de colombianos (de diccionarios) creada. La estructura objetivo se muestra en la Figura 1.

```python
def carga_de_datos(nombre_archivo:str)->dict:
    ocupaciones = {}
    archivo = open(nombre_archivo, "r")
    titulos = archivo.readline()
    linea = archivo.readline()
    while len(linea) > 0:
        datos = linea.split(",")
        ocupacion = datos[4]
        datos_persona = {}
        datos_persona["nombre"] =  datos[0]
        datos_persona["genero"] =  datos[1]
        datos_persona["anio_nacimiento"] =  int(datos[2])
        datos_persona["anio_muerte"] =  int(datos[3])
        datos_persona["ciudadania"] =  datos[5]
        datos_persona["numero_lectores"] =  int(datos[6])
        personas = []
        if ocupacion not in ocupaciones:
            personas.append(datos_persona)
            ocupaciones[ocupacion] = personas
        if ocupacion in ocupaciones:
            ocupaciones[ocupacion].append(datos_persona)
        linea =  archivo.readline()

    archivo.close()
    return ocupaciones
```

02.

Función 2:
Implemente una función que reciba por parámetro el diccionario completo de colombianos y retorne el nombre del colombiano con mayor número de lectores en Wikipedia.

```
29
30    def mayor_lectores(diccionario:dict)->str:
31        mayor=0
32        maxi=""
33        for deporte in diccionario.keys():
34            for atleta in diccionario[deporte]:
35                if atleta["numero_lectores"] >=mayor:
36                    mayor=atleta["numero_lectores"]
37                    maxi=atleta["nombre"]
38        return maxi
39
```

02.

Función 2:
Implemente una función que reciba por parámetro el diccionario completo de colombianos y retorne el nombre del colombiano con mayor número de lectores en Wikipedia.

```python
def mayor_lectores (colombianos: dict) -> str:
    respuesta = ""
    mayor = 0
    for ocupaciones in colombianos:
        for personas in colombianos[ocupaciones]:
            if personas["numero_lectores"] > mayor:
                mayor = personas["numero_lectores"]
                respuesta = personas["nombre"]
    return respuesta
```

03.

Función 3:
Implemente una función que reciba por parámetro el diccionario completo de colombianos, el nombre de una ocupación, un género y un número de lectores, y retorne un booleano que indique si existen al menos 3 colombianos de dicha ocupación y género que superen ese número de lectores.

```
41
42  def hay_3_colombianos(diccionario:dict, ocupacion:str, genero:str, lectores:int)->bool:
43      numero=0
44      centinela=False
45      for colombiano in diccionario[ocupacion]:
46          if (colombiano["genero"]==genero) and (colombiano["numero_lectores"]>lectores):
47              numero+=1
48      if numero>=3:
49          centinela=True
50      return centinela
```

03.

Función 3:
Implemente una función que reciba por parámetro el diccionario completo de colombianos, el nombre de una ocupación, un género y un número de lectores, y retorne un booleano que indique si existen al menos 3 colombianos de dicha ocupación y género que superen ese número de lectores.

```python
def hay_3_colombianos (colombianos: dict, ocupacion: str, genero: str, num_lectores: int) -> bool:
    personas = 0
    llaves=list(colombianos.keys())
    i = 0
    while i < len(colombianos) and personas<3:
        ocu=llaves[i]
        if ocupacion == ocu:
            j=0
            while j < len(colombianos[ocu]) and personas<3:
                persona=colombianos[ocu][j]
                if persona["genero"] == genero and persona["numero_lectores"] > num_lectores :
                    personas += 1
                j+=1
        i+=1

    if personas >= 3:
        respuesta = True
    else:
        respuesta = False
    return respuesta
```

04.

Función 4:
Implemente una función que reciba por parámetro el diccionario completo de colombianos y el nombre de una ocupación y retorne el promedio de número de lectores de los colombianos con la ocupación dada por parámetro redondeado a dos cifras decimales.

```
def promedio_lectores(diccionario:dict, ocupacion:str)->float:
    promedio=0
    suma=0
    contador=0
    for llave,valor in diccionario.items():
        if llave==ocupacion:
            for x in valor:
                suma+=x["numero_lectores"]
                contador+=1
    promedio=suma/contador
    return round(promedio,2)
```

04.

Función 4:
Implemente una función que reciba por parámetro el diccionario completo de colombianos y el nombre de una ocupación y retorne el promedio de número de lectores de los colombianos con la ocupación dada por parámetro redondeado a dos cifras decimales.

```python
def promedio_lectores(colombianos: dict, ocupacion: str) -> float:
    suma = 0
    cantidad = 0
    if ocupacion not in colombianos:
        rta=0
    else:
        personas = colombianos[ocupacion]
        for persona in personas:
            suma += persona["numero_lectores"]
            cantidad += 1

        rta = suma/cantidad

    return round(rta,2)
```

05.

Función 5:
Implemente una función que reciba por parámetro el diccionario completo de colombianos y retorne qué ocupación tiene el mayor *número de lectores promedio*. Esta función debe hacer uso de la función 4.

```
def mayor_rating(colombianos: dict) -> str:
    rta = ""
    maximo = 0
    for ocupacion in colombianos:
        x = promedio_lectores(colombianos, ocupacion)
        if x > maximo:
            maximo = x
            rta = ocupacion

    return rta
```

06.

Función 6:
Implemente una función que reciba por parámetro el diccionario completo de colombianos, una ocupación y un rango de años, es decir, un límite inferior y uno superior y retorne una lista con los colombianos que tienen la ocupación dada por parámetro y que nacieron en ese rango de años. La lista esperada debe tener como elementos los diccionarios con la información de los colombianos que nacieron en ese rango de años y tienen la ocupación dada.

```python
def colombianos_rango(colombianos: dict, ocupacion: str, anio_min: int, anio_max: int) -> list:
    rta = []
    if ocupacion in colombianos:
        lista = colombianos[ocupacion]
        for persona in lista:
            anio = persona["anio_nacimiento"]
            if anio > anio_min and anio < anio_max:
                rta.append(persona)
    return rta
```

07.

Función 7:
Implemente una función que reciba por parámetro el diccionario completo de colombianos y retorne un diccionario con el número de personas de cada nacionalidad. El diccionario esperado debe tener como llaves las nacionalidades y como valor, el número de personas con esa nacionalidad.

```python
def nacionalidades(colombianos: dict) -> dict:
    persona = {}

    for ocupaciones in colombianos:
        for personas in colombianos[ocupaciones]:
            if personas["ciudadania"] not in persona:
                persona[personas["ciudadania"]] = 1
            else:
                persona[personas["ciudadania"]] += 1

    return persona
```

08.

Función 8:
Implemente una función que reciba por parámetro el diccionario completo de colombianos y retorne este mismo diccionario con una pareja llave-valor adicionada al diccionario de cada colombiano. La nueva llave debe ser “edad” y el valor de esta, es la edad del colombiano calculada. Para los colombianos que ya fallecieron, se calcula la edad que tenían cuando fallecieron.

```python
def calcular_edad(colombianos: dict) -> dict:
    edades={}
    for ocupaciones in colombianos:
        edades[ocupaciones]=[]
        for personas in colombianos[ocupaciones]:
            if personas["anio_muerte"] != 0:
                anios = personas["anio_muerte"] - personas["anio_nacimiento"]
                personas["edad"] = anios

            else:
                currentDateTime = datetime.datetime.now()
                date = currentDateTime.date()
                year = int(date.strftime("%Y"))
                anios = year - personas["anio_nacimiento"]
                personas["edad"] = anios
            edades[ocupaciones].append(personas)

    return edades
```

09.

Función 9:
Implemente una función que reciba por parámetro el diccionario completo de colombianos y retorne un diccionario de listas de diccionarios (tal como la estructura del diccionario construido en la función 1) únicamente con los colombianos que ya fallecieron.

```python
def colombianos_fallecidos (colombianos: dict) -> dict:
    fallecidos = {}
    for ocupaciones in colombianos:
        fallecidos[ocupaciones] = []
        for personas in colombianos[ocupaciones]:
            if personas["anio_muerte"] != 0 and personas not in fallecidos[ocupaciones]:
                fallecidos[ocupaciones].append(personas)

    return fallecidos
```

09.

Función 9:
Implemente una función que reciba por parámetro el diccionario completo de colombianos y retorne un diccionario de listas de diccionarios (tal como la estructura del diccionario construido en la función 1) únicamente con los colombianos que ya fallecieron.

```
104
105    def colombianos_fallecidos(diccionario:dict)->dict:
106        lista=[]
107        ocupaciones={}
108        for llave,valor in diccionario.items():
109                for y in valor:
110                    if y["anio_muerte"]!=0:
111                        lista.append(y)
112                        if llave not in ocupaciones:
113                            ocupaciones[llave] = []
114                        ocupaciones[llave].append(y)
115        return ocupaciones
```

MUCHAS GRACIAS