

NIVEL 2

DICCIONARIOS PARA MANEJAR ELEMENTOS
QUE TIENEN LAS MISMAS CARACTERÍSTICAS



OTRO USO INTERESANTE DE LOS DICCIONARIOS

Los diccionarios nos permiten manejar fácilmente elementos de la realidad que tienen las mismas características. Ejemplo:

- De todos los estudiantes de la universidad se guarda la misma información: nombre, código, género, carrera, promedio, semestre según créditos



INFORMACIÓN DE LOS ESTUDIANTES COMO DICCIONARIOS

```
In [36]: estudiante1 = {"nombre": "Juan Pérez", "código": "201824736", "género":  
"masculino", "carrera": "Biología", "promedio": 3.78, "ssc": 0.7}  
  
In [37]: estudiante2 = {"nombre": "Ana Gavalda", "código": "201724736", "género":  
"femenino", "carrera": "Ciencias Políticas", "promedio": 4.25, "ssc": 3.5}  
  
In [38]: estudiante3 = {"nombre": "Bastien Bosa", "código": "201815217", "género":  
"masculino", "carrera": "Economía", "promedio": 3.21, "ssc": 2.3}  
  
In [39]: estudiante4 = {"nombre": "Catalina Gómez", "código": "201715400", "género":  
"femenino", "carrera": "Arte", "promedio": 3.8, "ssc": 4}  
  
In [40]: print("Los estudiantes son:\n", "Estudiante 1:\n", estudiante1, "\nEstudiante 2:  
\n", estudiante2, "\nEstudiante 3:\n", estudiante3, "\nEstudiante 4:\n", estudiante4)  
Los estudiantes son:  
Estudiante 1:  
{'nombre': 'Juan Pérez', 'código': '201824736', 'género': 'masculino', 'carrera':  
'Biología', 'promedio': 3.78, 'ssc': 0.7}  
Estudiante 2:  
{'nombre': 'Ana Gavalda', 'código': '201724736', 'género': 'femenino', 'carrera':  
'Ciencias Políticas', 'promedio': 4.25, 'ssc': 3.5}  
Estudiante 3:  
{'nombre': 'Bastien Bosa', 'código': '201815217', 'género': 'masculino', 'carrera':  
'Economía', 'promedio': 3.21, 'ssc': 2.3}  
Estudiante 4:  
{'nombre': 'Catalina Gómez', 'código': '201715400', 'género': 'femenino', 'carrera':  
'Arte', 'promedio': 3.8, 'ssc': 4}
```

Un diccionario para
cada estudiante

¿Y SI HACEMOS UNA FUNCIÓN QUE NOS CREE UN ESTUDIANTE?

```
EjemploEstudiantes.py ✖
1 def crear_estudiante(nom: str, cod: str, gen: str, carr: str, prom: float, ssc: float)->dict:
2     dic_estudiante = { "nombre": nom,
3                         "código": cod,
4                         "género": gen,
5                         "carrera": carr,
6                         "promedio": prom,
7                         "ssc": ssc}
8     return dic_estudiante
9
10 #PROGRAMA PRINCIPAL
11 estudiante1 = crear_estudiante("Juan Pérez", "201824736", "masculino", "Biología", 3.78, 0.7)
12 estudiante2 = crear_estudiante("Ana Gavalda", "201724736", "femenino", "Ciencias Políticas", 4.25, 3.5)
13 estudiante3 = crear_estudiante("Bastien Bosa", "201815217", "masculino", "Economía", 3.21, 2.3)
14 estudiante4 = crear_estudiante("Catalina Gómez", "201715400", "femenino", "Arte", 3.8, 4)
15
16 print("Los estudiantes son:\n", "Estudiante 1:\n", estudiante1, \
17       "\nEstudiante 2:\n", estudiante2, \
18       "\nEstudiante 3:\n", estudiante3, \
19       "\nEstudiante 4:\n", estudiante4)
```

¿Cuál sería el resultado?

¿Y SI HACEMOS UNA FUNCIÓN QUE NOS CREE UN ESTUDIANTE?

Resultado de la ejecución

```
Terminal 6/A ✕  
  
In [42]: runfile('C:/Users/Mhernandez/Desktop/IP/N2-C5/EjemploEstudiantes.py', wdir='C:/Users/  
Mhernandez/Desktop/IP/N2-C5')  
Los estudiantes son:  
Estudiante 1:  
{'nombre': 'Juan Pérez', 'código': '201824736', 'género': 'masculino', 'carrera': 'Biología',  
'promedio': 3.78, 'ssc': 0.7}  
Estudiante 2:  
{'nombre': 'Ana Gavalda', 'código': '201724736', 'género': 'femenino', 'carrera': 'Ciencias  
Políticas', 'promedio': 4.25, 'ssc': 3.5}  
Estudiante 3:  
{'nombre': 'Bastien Bosa', 'código': '201815217', 'género': 'masculino', 'carrera': 'Economía',  
'promedio': 3.21, 'ssc': 2.3}  
Estudiante 4:  
{'nombre': 'Catalina Gómez', 'código': '201715400', 'género': 'femenino', 'carrera': 'Arte',  
'promedio': 3.8, 'ssc': 4}
```

EJEMPLOS

Teniendo en cuenta esos 4 estudiantes:



1. Averiguar quien es el estudiante con mayor promedio. Para esto tenemos la función **mayor_promedio** que recibe por parámetro los 4 estudiantes (diccionarios) y retorna el estudiante (diccionario) que tiene mayor promedio
2. Calcular cuántos estudiantes son mujeres. Para esto tenemos la función **cuantas_mujeres** que recibe por parámetro los 4 estudiantes y retorna cuántos son de género femenino
3. Decidir si hay al menos una mujer pila. Para esto tenemos la función **hay_mujer_pila** que recibe por parámetro los 4 estudiantes y dice (verdadero o falso) si hay al menos una mujer con promedio superior a 4

FUNCIÓN MAYOR_PROMEDIO



En la variable **mayor** se guarda el diccionario que contiene al estudiante de mayor promedio hasta el momento

```
def mayor_promedio(est1: dict, est2: dict, est3: dict, est4: dict)->dict:  
    mayor = est1  
  
    if (est2["promedio"] >= mayor["promedio"]):  
        mayor = est2  
  
    if (est3["promedio"] >= mayor["promedio"]):  
        mayor = est3  
  
    if (est4["promedio"] >= mayor["promedio"]):  
        mayor = est4  
  
    return mayor
```

Si se encuentra un estudiante con mayor promedio, se actualiza la variable **mayor** con ese estudiante

Note que la variable **mayor** guarda un diccionario (el estudiante completo), aunque la comparación se hace únicamente entre los valores de las claves «**promedio**»

LLAMADO DESDE EL PROGRAMA PRINCIPAL

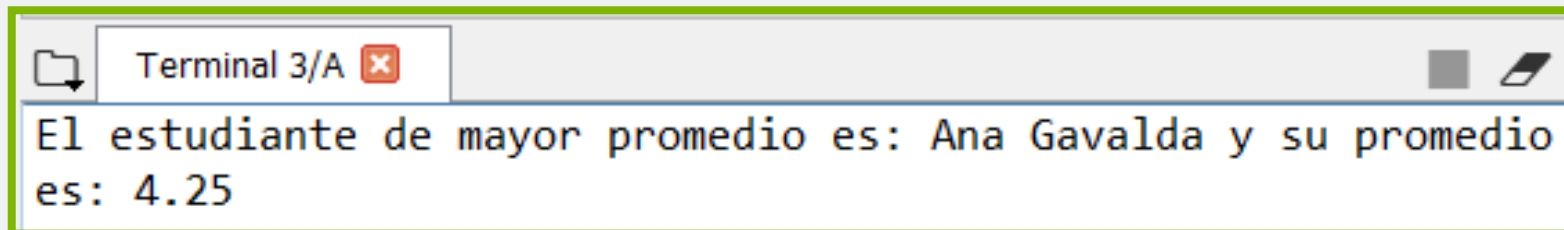
En la variable **mejor** se guarda el diccionario que contiene al estudiante de mayor promedio

```
mejor = mayor_promedio(estudiante1, estudiante2, estudiante3, estudiante4)

print("El estudiante de mayor promedio es: " + mejor["nombre"] + \
      " y su promedio es: " + str(mejor["promedio"]))
```

Se imprime el valor de la llave llamada «promedio»

Resultado de la ejecución



```
Terminal 3/A x
El estudiante de mayor promedio es: Ana Gavalda y su promedio
es: 4.25
```


FUNCIÓN CUANTAS_MUJERES



La variable **cuantas** es dónde se va a contar cuántas mujeres hay. Se inicializa con el valor cero

```
def cuantas_mujeres(est1: dict, est2: dict, est3: dict, est4: dict)->int:
    cuantas = 0

    if (est1["género"] == "femenino"):
        cuantas+=1

    if (est2["género"] == "femenino"):
        cuantas+=1

    if (est3["género"] == "femenino"):
        cuantas+=1

    if (est4["género"] == "femenino"):
        cuantas+=1

    return cuantas
```

A medida que se van encontrando mujeres, se incrementa el contador

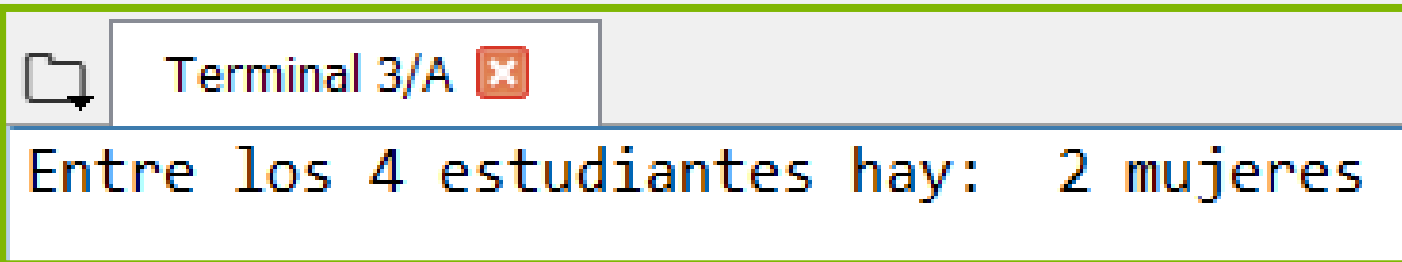
Note que son **ifs** independientes porque hay que preguntar el valor de la clave «género» a todos los 4 diccionarios (estudiantes)

LLAMADO DESDE EL PROGRAMA PRINCIPAL

En la variable **cuantas** se guarda el entero retornado por la función `cuantas_mujeres`

```
cuantas = cuantas_mujeres(estudiante1, estudiante2, estudiante3, estudiante4)  
print("Entre los 4 estudiantes hay: ", cuantas, "mujeres")
```

Resultado de la ejecución



The screenshot shows a terminal window titled "Terminal 3/A". The output displayed is "Entre los 4 estudiantes hay: 2 mujeres".

FUNCIÓN HAY_MUJER_PILA



La variable **hay** es dónde se va a guardar el valor booleano que se va a retornar. Se inicializa en **False** suponiendo que no hay mujer pila

```
def hay_mujer_pila(est1: dict, est2: dict, est3: dict, est4: dict)->bool:
    hay = False

    if (est1["género"] == "femenino" and est1["promedio"] >= 4):
        hay = True
    elif (est2["género"] == "femenino" and est2["promedio"] >= 4):
        hay = True
    elif (est3["género"] == "femenino" and est3["promedio"] >= 4):
        hay = True
    elif (est4["género"] == "femenino" and est4["promedio"] >= 4):
        hay = True

    return hay
```

Si se encuentra alguna mujer, se cambia el valor de la variable a **True**

Note que es una cascada de **if-elif** porque si se encuentra una mujer pila ya no debe continuar buscando

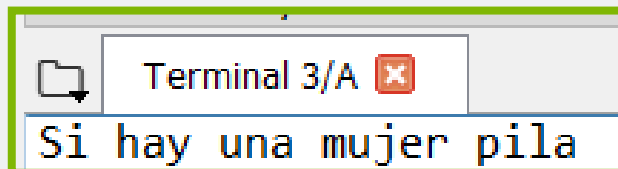
LLAMADO DESDE EL PROGRAMA PRINCIPAL (SIN COMPARACIÓN)

En la variable `var_hay` se guarda lo que devuelve la función `hay_mujer_pila` (que es `True` o `False`)

```
#Con variable sin ==  
var_hay = hay_mujer_pila (estudiante1, estudiante2, estudiante3, estudiante4)  
if (var_hay):  
    print("Si hay una mujer pila")  
else:  
    print("No hay una mujer pila")
```

Como `var_hay` es una variable booleana, se puede usar directamente en el `if` y es equivalente a preguntar si es igual a `True`. No es necesario poner `== True`

Resultado de la ejecución

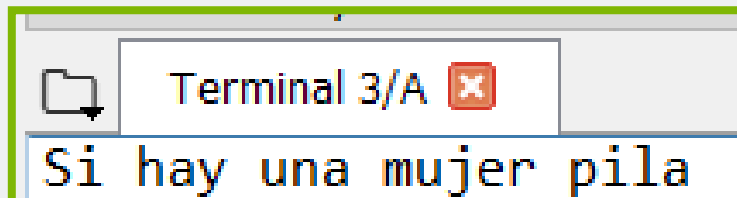


LLAMADO DESDE EL PROGRAMA PRINCIPAL (SIN VARIABLE)

Se hace el llamado directamente a la función `hay_mujer_pila` que devuelve `True` o `False` (sin necesidad de usar una variable)

```
#Sin variable
if (hay_mujer_pila (estudiante1, estudiante2, estudiante3, estudiante4) == True):
    print("Si hay una mujer pila")
else:
    print("No hay una mujer pila")
```

Resultado de la ejecución



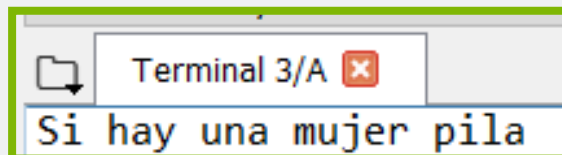
LLAMADO DESDE EL PROGRAMA PRINCIPAL (SIN VARIABLE Y SIN ==)

Se hace el llamado directamente a la función `hay_mujer_pila` que devuelve `True` o `False` (sin necesidad de usar una variable)

```
#Sin variable y sin ==  
if hay_mujer_pila (estudiante1, estudiante2, estudiante3, estudiante4):  
    print("Si hay una mujer pila")  
else:  
    print("No hay una mujer pila")
```

Como `hay_mujer_pila` es una función booleana, se puede usar directamente en el `if` y es equivalente a preguntar si es igual a `True`. No es necesario poner `== True`

Resultado de la ejecución



Trabajemos tres funciones un poco más complejas:

1. Escribe una función llamada **buscar_estudiante** que reciba por parámetro los 4 estudiantes y un nombre (str) y retorne el diccionario del estudiante que tiene ese mismo nombre (idéntico). Si el estudiante no existe, la función debe retornar **None**. Escribe el programa principal que informe al usuario el resultado con un mensaje apropiado (es decir, verificando si es **None** o no)
2. Escribe una función llamada **avanzar_semestre** que reciba por parámetro los 4 estudiantes y modifique todos los estudiantes para que avancen un semestre (es decir, que debe incrementar en 1 el semestre según créditos). Esta función NO retorna ningún valor, sólo modifica los 4 diccionarios
3. Escribe una función llamada **quienes_en_riesgo** que reciba por parámetro los 4 estudiantes y retorne un diccionario con los códigos (claves) y promedios (valores) de los estudiantes que tienen el promedio por debajo de 3.4

