


Estrategias: Dato mediante literal Intersección de contenedores Parámetro opcional



Fundamentos de Programación
Departamento de Lenguajes y Sistemas Informáticos



Estrategias (visualizar denominaciones que se corresponde con enumerados)

Supongamos que se dispone de un fichero csv con datos sobre nacimiento, entre los que se incluye la fecha y hora de cada nacimiento (*fecha_hora*) y se pide el número de nacimientos por meses del año. Pero con un “*detallito*” añadido que **los meses se identifiquen por su nombre**.

La solución inicial a plantear sería:

```
Nacimiento=NamedTuple("Nacimiento",[("sexo",str),("fecha_hora",datetime)])
nacimientos=[Nacimiento("Mujer",datetime(2024,12,1,22,30)),
              Nacimiento("Hombre",datetime(2024,11,28,22,30)),
              Nacimiento("Hombre",datetime(2024,10,1,22,30)),
              Nacimiento("Mujer",datetime(2023,12,12,22,30))]
res=DefaultDict(int)
for n in nacimientos:
    res[n.fecha_hora.month]+=1
return sorted(res.items())
```

→ [(10, 1), (11, 1), (12, 2)]



Estrategias (visualizar denominaciones que se corresponde con enumerados)

```
Nacimiento=NamedTuple("Nacimiento",[( "sexo",str),("fecha_hora",datetime)])
nacimientos=[Nacimiento("Mujer",datetime(2024,12,1,22,30)),
              Nacimiento("Hombre",datetime(2024,11,28,22,30)),
              Nacimiento("Hombre",datetime(2024,10,1,22,30)),
              Nacimiento("Mujer",datetime(2023,12,12,22,30))]
meses=['Enero','Febrero','Marzo','Abril','Mayo','Junio','Julio','Agosto','Septiembre',
       'Octubre','Noviembre','Diciembre']

aux=DefaultDict(int)
for n in nacimientos:
    aux[n.fecha_hora.month]+=1

res=dict()
for c,v in sorted(aux.items()):
    res[meses[c-1]]=v
return res
```

→ {'Octubre': 1, 'Noviembre': 1, 'Diciembre': 2}



Estrategias (visualizar denominaciones que se corresponde con enumerados)

Ahora se pide **el día** de la semana en que más nacimientos ha habido. Como ayuda hay que saber que el método **weekday()**, devuelve un número entre **0** y **6**. El **0** significa **lunes** y el **6** para **domingo**

La solución sería:

```
Nacimiento=NamedTuple("Nacimiento",[("sexo",str),("fecha_hora",datetime)])
nacimientos=[Nacimiento("Mujer",datetime(2024,12,1,22,30)),
              Nacimiento("Hombre",datetime(2024,11,28,22,30)),
              Nacimiento("Hombre",datetime(2024,10,1,22,30)),
              Nacimiento("Mujer",datetime(2023,12,12,22,30))]
días=['lunes','martes','miércoles','jueves','viernes','sábado','domingo']
```

```
aux=DefaultDict(int)
for n in nacimientos:
    aux[n.fecha_hora.weekday()]+=1
máximo=max(aux.items(),key=lambda e:e[1])
return máximo[0] → 1
Sin embargo:
return días[máximo[0]] → martes
```



Estrategias (encontrar si dos contenedores tienen elementos en común)

Encontrar si dos contenedores tienen elementos en común.

Basta hacer la *intersección* de los conjuntos generados a partir de ellos:

```
tupla1=(1,2,3,4,5)
tupla2=(3,5,6,7)
len(set(tupla1)&set(tupla2))>0
```

2

```
lista=[5,9,10]
len(set(lista)&set(tupla1))>0
```

1



Parámetros Optional

- Existen tres formas de indicar que un *parámetro* es opcional y puede tomar el valor *None*:

```
def función (... , parámetro:Optional[tipo]=None, ...)  
def función (... , parámetro:tipo|None=None, ....)  
def función (... , parámetro:Union[tipo,None]=None, ....)
```

- Las dos últimas también vale cuando el *parámetro* puede ser de más de un tipo o un tipo de dato:

```
def función (... , parámetro:tipo1|tipo2|tipo3..., ....)  
def función (... , parámetro:Union[tipo1,tipo2,tipo3,...], ....)
```

- De la misma manera se puede indicar que una función *devuelve* un valor o *None*:

```
def función (...)->Optional[tipo]  
def función (...)->tipo|None  
def función (...)->Union[tipo,None]
```

- Las dos últimas también vale cuando se puede *devolver* más de un tipo de dato:

```
def función (...)->tipo1|tipo2|tipo3|...  
def función (...)->Union[tipo1,tipo2,tipo3,...]
```



Ejercicio:

Proyecto T12_GitAcme:

- Descargar e Implementar el proyecto