Generación de listas por comprensión

Supongamos el siguiente tipo: `Par=namedtuple("pareja", "letra, número")`

y la lista: `lista=[Par("z",21),Par("a",62),Par("J",7),Par("b",56),Par("c",90),`
`Par("z",21),Par("a",1),Par("b",10),Par("b",2),Par("a",21)]`

Si quisiéramos obtener una nueva lista con sólo las letras haríamos los siguiente:

```
lista_letras=list()
for elto in lista:
  lista_letras.append(elto.letra)
```

Resultado: `lista_letras` → `['z','a','J','b','c','z','a','b','b','a']`

Esto mismo se puede *hacer por comprensión* con la siguiente sintaxis.

```
lista_letras=[elto.letra for elto in lista]
```

Generación de listas por comprensión

Supongamos el siguiente tipo: `Par=namedtuple("pareja", "letra, número")`

y la lista: `lista=[Par("z",21),Par("a",62),Par("J",7),Par("b",56),Par("c",90),`
`Par("z",21),Par("a",1),Par("b",10),Par("b",2),Par("a",21)]`

Si quisiéramos obtener una nueva lista de tuplas con el número y la letra de aquellas cuyo número es mayor de 20 haríamos lo siguiente:

```
lista_mayores=list()
for elto in lista:
    if elto.número>20:
       lista_mayores.append((elto.número,elto.letra))
```

Resultado: `lista_mayores` → `[(21,'z'),(62,'a'),(56,'b'),(90,'c'),(21,'z'),(21,'a')]`

Esto mismo se puede hacer por comprensión con la siguiente sintaxis.

```
lista_mayores=[(elto.número,elto.letra) for elto in lista if elto.número>20]
```

Generación de conjuntos por comprensión

Es análogo a las listas, pero con las funciones y métodos de conjunto

Supongamos el siguiente tipo: `Par=namedtuple("pareja", "letra, número")`

y la lista:
```
lista=[Par("z",21),Par("a",62),Par("J",7),Par("b",56),Par("c",90),
       Par("z",21),Par("a",1),Par("b",10),Par("b",2),Par("a",21)]
```

Si quisiéramos obtener un conjunto con sólo las letras haríamos los siguiente:

```
conj_letras=set()
for elto in lista:
  conj_letras.add(elto.letra)
```

Resultado: `conj_letras` → `{'J', 'a', 'b', 'z', 'c'}`

Esto mismo se puede *hacer por comprensión* con la siguiente sintaxis.

```
conj_letras={elto.letra for elto in lista}
```

Generación de conjuntos por comprensión

Supongamos el siguiente tipo: `Par=namedtuple("pareja", "letra, número")`

y la lista:
```
lista=[Par("z",21),Par("a",62),Par("J",7),Par("b",56),Par("c",90),
       Par("z",21),Par("a",1),Par("b",10),Par("b",2),Par("a",21)]
```

Si quisiéramos obtener un conjunto de tuplas con el número y la letra de aquellas cuyo número es mayor de 20 haríamos lo siguiente:

```
conj_mayores=set()
for elto in lista:
  if elto.número>20:
      conj_mayores.add((elto.número,elto.letra))
```

Resultado: `conj_mayores` → `{(21,'z'),(21,'a'),(56,'b'),(62,'a'),(90,'c')}`

Esto mismo se puede hacer por comprensión con la siguiente sintaxis.

```
conj_mayores={(elto.número,elto.letra) for elto in lista if elto.número>20}
```

Diccionarios

Algoritmo de construcción de diccionarios a partir de un contenedor (Esquema 1):

Los pasos son los siguientes:

1. *Se va recorriendo el contenedor (con un for)*
2. *Se consulta si el elemento que será la clave ya está en el diccionario (con un if … not in...)*
   a) *Si no está se inserta una nueva pareja clave-valor : diccionario[clave]=valor*
   b) *Si está se actualiza el valor: diccionario[clave]=nuevo_valor*

Ejemplo : Para la lista=['a','b','c','a','b','b','z','a','b','c'], contar cuantas veces aparece cada valor

```
dic=dict()
for letra in  lista :
    if letra not in dic:
        dic[letra]=1
    else:
        dic[letra]+=1          //esto es lo mismo que dic[letra]= dic[letra]+1
```

Resultado: dic → { 'a': 3, 'b': 4, 'c': 2, 'z': 1}

Diccionarios

Algoritmo de construcción de diccionarios a partir de un contenedor (Esquema 2):

Los pasos son los siguientes:

1. *Se va recorriendo el contenedor (con un for)*
2. *Se consulta si el elemento que será la clave ya está en el diccionario (con un if)*
   Si no está se inserta una nueva pareja clave-valor con el valor neutro de la operación
   diccionario[clave]=valor_neutro
3. *En todo caso, se actualiza el valor: diccionario[clave]=actualización del valor*

Ejemplo : Para la lista lista=['a','b','c','a','b','b','z','a','b','c'], contar cuantas veces aparece cada valor

```
dic=dict()
for letra in  lista :
    if letra not in dic:
        dic[letra]=0          //Elemento neutro de la suma
    dic[letra]+=1
```

Resultado: dic →{ 'a': 3, 'b': 4, 'c': 2, 'z': 1}

Diccionarios

Algoritmo de construcción de diccionarios a partir de un contenedor (Esquema 1):

Ejemplo:

Par=namedtuple("pareja", "letra, número")
lista=[("z",21),("a",62),("J",7),("b",56),("c",90),("z",21),("a",1),("b",10),("b",2),("a",21)], obtener un diccionario que a cada letra le haga corresponder una lista con los valores asociados

```
dic=dict()
for p in lista:
    if p.letra not in dic:
        dic[p.letra]=[p.número]
    else:
        dic[p.letra]+=[p.número]      // esto es lo mismo que dic[p.letra].append(p.número)
```

Resultado: dic →{'z': [21, 21], 'a': [62, 1, 21], 'J': [7], 'b': [56, 10, 2], 'c': [90]}

Diccionarios

Algoritmo de construcción de diccionarios a partir de un contenedor (Esquema 2):

Ejemplo:

Par=namedtuple("pareja", "letra, número")

lista=[("z",21),("a",62),("J",7),("b",56),("c",90),("z",21),("a",1),("b",10),("b",2),("a",21)], obtener un diccionario que a cada letra le haga corresponder una lista con los valores asociados

```
dic=dict()
for p in lista:
    if p.letra not in dic:
        dic[p.letra]=list()                 // o bien =[ ].Valor neutro de listas (una lista vacía)
    dic[p.letra]+=[p.número]                // esto es lo mismo que dic[p.letra].append(p.número)
```

Resultado: dic →{'z': [21, 21], 'a': [62, 1, 21], 'J': [7], 'b': [56, 10, 2], 'c': [90]}

Ejercicio:

Proyecto Vuelo:

Realice el enunciado ***T2023_11_07_Vuelos.pdf***

- Ejercicios del 6 al 11