Comprensiones en Python

¿Qué son las comprensiones de listas?

Las comprensiones en Python son una forma concisa y eficiente de crear listas, conjuntos y diccionarios. Permiten realizar operaciones de filtrado, mapeo y transformación de elementos en una sola línea de código, lo que facilita la lectura y escritura del código.

Siguiendo el ejemplo que vimos en clase las podemos observar en la nueva lista llamada alumnos_restamtes:

```
import random

lista_alumnos = ["salvador", "brisa", "cristian", "emanuel", "franco"]
alumnos_pasados = ["brisa"]

alumnos_restantes = [alumno for alumno in lista_alumnos if alumno not in alumnos_pasados]
# Utilizamos una comprensión de lista para filtrar y crear una nueva lista llamada
# alumnos_restantes a partir de la lista original lista_alumnos

lista_alumnos_seleccion = random.choice(alumnos_restantes)
print(lista_alumnos_seleccion)
```

¿Qué hace en este caso?

> Lo que esta haciendo nuestra comprension es generar una lista nueva donde en cada iteracion va comprobando si el alumno no esta en la lista de alumnos_pasados y lo agrega en la nueva lista de alumnos restantes. Siguiendo este caso, en la primer iteracion agregaria a salvador, en la segunda **NO** agregaria a brisa(ya que esta en alumnos_pasados) y despues de la tercera agregaria a todos los alumnos hasta finalizar la lista.

Vale aclarar que la lista nueva no se genera hasta que finaliza el ciclo for que nos encontramos en su interior, si nos paramos en la lista con el debbuger no la vamos a ver hasta que termine de ejecutarse la linea de codigo en su totalidad.

Filtros con comprensiones

Podemos filtrar los alumnos(en este caso) dependiendo la letra con la que empieza o termina. El metodo a utilizar es *.startswith() o .endswith()* donde en el parentesis le pasaremos como cadena de texto lo que tiene que buscar adelante o atras de cada iteracion.

Por ejemplo si colocamos *.startswith('s')*, filtrara todos los nombres que empiecen con S

```
nombres_con_s = [alumno for alumno in lista_alumnos if alumno.startswith('s')]
print(nombres_con_s)
#Salida: ['salvador']
terminado_con_o = [alumno for alumno in lista_alumnos si alumno termina con 'o']
print(terminado_con_o)
#Salida: ['franco']
```

Tambien podemos filtrar por contenido de una palabra, generando una nueva lista a partir de por ejemplo palabras que contengan tanto la letra "a" como la letra "o".

```
alumnos_con_ao = [alumno for alumno in lista_alumnos if "a" in
                    alumno or "o" in alumno]
print(alumno_con_ao)
#Salida: ['salvador','franco']
```

En este caso tambien podemos agregarle más condiciones ya que trabajamos con la estructura del if

Podemos Filtrar y aplicar una funcion!

```
nombres_restantes_mayuscula = [alumno.upper() for alumno in alumnos_restantes
                               if alumno not in alumnos_pasados]
```

Podemos crear Listas de listas

```
lista_alumnos_longitud = [[alumno, len(alumno)] for alumno in lista_alumnos]
```

Siendo un if podemos incluir un else: Aunque no es tan común, se puede incluir un bloque else en la comprensión de lista,pero esto suele hacerse más en comprensiones de listas que están aplicando una función o transformación a los elementos.

```
lista_transformada = [alumno.upper() if alumno[0].lower() >= 'g'
                      else alumno.lower() for alumno in lista_alumnos]
```

> Este caso es espectacular, convierte mediante el >= 'g' todas las palabras que comiencen con una letra que alfabeticamente se encuentre despues de la G en mayusculas y el resto en minusculas.

Podemos usar una comprensión de listas con la función range:

```
#range(Numero Inicial, Numero Final, Incremento)
numeros_con_salto = [numero for numero in range(2, 101, 2)]
#Salida: Genera una lista del 2 al 100 dando saltos de 2 en 2.
```

Otro ejemplos:

Obtener el índice y el elemento de una lista:

Queremos crear una lista de tuplas donde cada tupla contenga el índice y el elemento de la lista original. Podemos usar la función enumerate:

```
alumnos_con_indice = [(indice, alumno) for indice, alumno in enumerate(lista_alumnos)]
print(alumnos_con_indice)
```

Uso de la función pprint (pretty print):

es un módulo en Python que proporciona una forma de imprimir estructuras de datos complejas (como listas, diccionarios, etc.) de manera más legible y organizada. Al usar pprint, los datos se formatean con saltos de línea y sangrías apropiadas, facilitando su comprensión y análisis visual. Es especialmente útil cuando trabajas con datos anidados o grandes volúmenes de información.

```
 import pprint
lista_alumnos = ["Monica", "maricel", "carlos", "guillermo", "paula", "fermin", "gonzalo", "javier", "Jonathan", "florencia", "letic
alumnos_con_indice = [(indice, alumno) for indice, alumno in enumerate(lista_alumnos)]
# print(alumnos_con_indice)
pprint.pprint(alumnos_con_indice)

#Otro ejemplo:
data = [
    {'name': 'Alice', 'age': 30, 'job': 'Engineer'},
    {'name': 'Bob', 'age': 25, 'job': 'Designer'},
    {'name': 'Charlie', 'age': 35, 'job': 'Manager'}
]

pprint.pprint(data)
```