List Comprehension

Estructura de Python

Listas avanzadas en 1 linea de código

```
lista=[ expresion for elemento in iterable]
```

EJEMPLO

```python
# Método tradicional
lista = []
for numero in range(0,11):
    lista.append(numero**2)
print(lista)
```

```python
# Con comprensión de listas
lista = [numero**2 for numero in  range(0,11)]
print(lista)
```

EJEMPLO

```
# Método tradicional
lista = []
for numero in range(0,11):
    if numero % 2 == 0:
        lista.append(numero)
print(lista)
```

```
# Añadir los números del 0 al 10 cuando su módulo de 2 sea 0
[numero for numero in range(0,11) if numero % 2 == 0 ]
```

LAMBDA

Funciones anónimas

Simplificando

```
def doblar(num):
    resultado = num*2
    return resultado

doblar(2)
```

```
def doblar(num):
    return num*2
```

```
lambda num: num*2
```

```
doblar = lambda num: num*2

doblar(2)
```

filter() y map()

Las funciones preferidas del paradigma funcional

```python
def multiple(numero):     # Primero declaramos una función condicional
    if numero % 5 == 0:   # Comprobamos si un numero es múltiple de cinco
        return True       # Sólo devolvemos True si lo es

numeros = [2, 5, 10, 23, 50, 33]

filter(multiple, numeros)
```

```python
list( filter(multiple, numeros) )
```

```python
list( filter(lambda numero: numero%5 == 0, numeros) )
```

---

filter() : nos permite filtrar un iterable bajo una condición

```python
def doblar(numero):
    return numero*2

numeros = [2, 5, 10, 23, 50, 33]

map(doblar, numeros)
```

```python
list(map(doblar, numeros))
```

```python
list( map(lambda x: x*2, numeros) )
```

---

map() : nos permite iterar y aplicar una funcion