06.1.Ejercicios

August 5, 2024

1 Ejercicios

1.1 map, filter, lambda

1. Escribe una función lambda que tome dos números y devuelva su producto.

[ ]:
```
product = lambda x, y: x * y
print(product(4, 5))
```

```
20
```

2. Usa una función lambda dentro de map para sumar 10 a cada número en una lista.

[ ]:
```
numbers = [1, 2, 3, 4, 5]
result = list(map(lambda x: x + 10, numbers))
print(result)
```

```
[11, 12, 13, 14, 15]
```

3. Usa una función lambda para ordenar una lista de tuplas en función del segundo elemento de cada tupla.

[ ]:
```
tuples = [(1, 2), (3, 1), (5, 4)]
sorted_tuples = sorted(tuples, key=lambda x: x[1])
print(sorted_tuples)  # Salida esperada: [(3, 1), (1, 2), (5, 4)]
```

```
[(3, 1), (1, 2), (5, 4)]
```

4. Usa una función lambda dentro de sorted para ordenar una lista de cadenas en función de su longitud.

[ ]:
```
strings = ["Hola", "Mundo", "Python", "es", "genial"]
sorted_strings = sorted(strings, key=lambda s: len(s))
print(sorted_strings)
```

```
['es', 'Hola', 'Mundo', 'Python', 'genial']
```

5. Filtrar y transformar una lista de diccionarios. Escribe un programa que tome una lista de diccionarios, donde cada diccionario representa a una persona con llaves nombre y edad. Devuelve una nueva lista de nombres de personas que tengan al menos 18 años.

[ ]:
```python
def mayores_de_edad(personas):
    return list(
                map(
                    lambda persona: persona['nombre'],
                    filter(
                        lambda persona: persona['edad'] >= 18,
                        personas
                    )
                )
            )

# Ejemplo de uso
personas = [
    {"nombre": "Ana", "edad": 22},
    {"nombre": "Luis", "edad": 17},
    {"nombre": "Marta", "edad": 19},
    {"nombre": "Carlos", "edad": 15}
]
nombres_mayores = mayores_de_edad(personas)
print(nombres_mayores)  # Output: ['Ana', 'Marta']
```

```
['Ana', 'Marta']
```

6. Aplicar múltiples funciones a una lista de números. Escribe un programa que tome una lista de números y aplique dos funciones diferentes a cada número: una que calcule el cuadrado y otra que calcule el cubo. Devuelve una lista de tuplas donde cada tupla contiene el resultado de ambas funciones.

[ ]:
```python
def cuadrado_y_cubo(lista):
    return list(
        map(
            lambda x: (x ** 2, x ** 3),
            lista
        )
    )

# Ejemplo de uso
numeros = [1, 2, 3, 4]
resultados = cuadrado_y_cubo(numeros)
print(resultados)  # Output: [(1, 1), (4, 8), (9, 27), (16, 64)]
```

```
[(1, 1), (4, 8), (9, 27), (16, 64)]
```

7. Ordenar una lista de tuplas basada en la suma de sus elementos. Escribe un programa que tome una lista de tuplas, donde cada tupla contiene dos números. Devuelve una nueva lista de tuplas ordenadas por la suma de sus elementos.

[ ]:
```python
def ordenar_por_suma(lista):
    return list(
```

```
        map(
            lambda x: x,
            sorted(
                lista,
                key = lambda x: x[0] + x[1]
            )
        )
    )

# Ejemplo de uso
tuplas = [(1, 2), (3, 4), (1, 1), (2, 2)]
tuplas_ordenadas = ordenar_por_suma(tuplas)
print(tuplas_ordenadas)  # Output: [(1, 1), (1, 2), (2, 2), (3, 4)]
```

```
[(1, 1), (1, 2), (2, 2), (3, 4)]
```

8. Convertir una lista de tuplas a un diccionario. Escribe un programa que tome una lista de tuplas, donde cada tupla contiene una llave y un valor. Devuelve un diccionario construido a partir de estas tuplas.

[ ]:
```
def tuplas_a_diccionario(lista):
    return dict(map(lambda x: (x[0], x[1]), lista))

# Ejemplo de uso
tuplas = [("llave1", "valor1"), ("llave2", "valor2"), ("llave3", "valor3")]
diccionario = tuplas_a_diccionario(tuplas)
print(diccionario)  # Output: {'llave1': 'valor1', 'llave2': 'valor2', 'llave3':
 ↪ 'valor3'}
```

```
{'llave1': 'valor1', 'llave2': 'valor2', 'llave3': 'valor3'}
```

9. Aplicar una función a una lista de listas. Escribe un programa que tome una lista de listas de números y aplique una función que calcule el promedio de cada lista interna. Devuelve una lista de promedios.

[ ]:
```
def promedios_de_listas(lista):
    return list(
        map(
            lambda x: sum(x) / len(x) if len(x) > 0 else 0,
            lista
        )
    )

# Ejemplo de uso
listas = [[1, 2, 3], [4, 5, 6, 7], [8, 9], []]
promedios = promedios_de_listas(listas)
print(promedios)  # Output: [2.0, 5.5, 8.5, 0]
```

```
[2.0, 5.5, 8.5, 0]
```

1.2 Reduce

10. Usa una función lambda dentro de reduce para calcular el producto de todos los números en una lista.

[ ]:
```
from functools import reduce

numbers = [1, 2, 3, 4, 5]
product = reduce(lambda x, y: x * y, numbers)
print(product)  # Salida esperada: 120
```

```
120
```

11. Suma de una lista de números. Escribe un programa que tome una lista de números y devuelva la suma de todos los números en la lista.

[ ]:
```
from functools import reduce

def suma(lista):
    return reduce(lambda x, y: x + y, lista)

# Ejemplo de uso
numeros = [1, 2, 3, 4, 5]
resultado = suma(numeros)
print(resultado)  # Output: 15
```

```
15
```

12. Encontrar el máximo en una lista de números. Escribe un programa que tome una lista de números y devuelva el número más grande en la lista.

[ ]:
```
from functools import reduce

def maximo(lista):
    return reduce(lambda x, y: x if x > y else y, lista)

# Ejemplo de uso
numeros = [1, 7, 3, 9, 5]
resultado = maximo(numeros)
print(resultado)  # Output: 9
```

```
9
```

13. Contar la frecuencia de elementos en una lista. Escribe un programa que tome una lista de elementos y devuelva un diccionario con la frecuencia de cada elemento en la lista.

[ ]:
```
from functools import reduce

def frecuencia(lista):
    return reduce(lambda acc, x: {**acc, x: acc.get(x, 0) + 1}, lista, {})
```

```python
# Ejemplo de uso
elementos = ['a', 'b', 'a', 'c', 'b', 'a']
resultado = frecuencia(elementos)
print(resultado)  # Output: {'a': 3, 'b': 2, 'c': 1}
```

```
{'a': 3, 'b': 2, 'c': 1}
```

14. Calcular la diferencia entre el número más grande y el más pequeño en una lista. Escribe un programa que tome una lista de números y devuelva la diferencia entre el número más grande y el más pequeño en la lista.

[ ]:
```python
from functools import reduce

def diferencia_max_min(lista):
    maximo = reduce(lambda x, y: x if x > y else y, lista)
    minimo = reduce(lambda x, y: x if x < y else y, lista)
    return maximo - minimo

# Ejemplo de uso
numeros = [1, 7, 3, 9, 5]
resultado = diferencia_max_min(numeros)
print(resultado)  # Output: 8
```

```
8
```

15. Encontrar el número de palabras en una lista de cadenas. Escribe un programa que tome una lista de cadenas y devuelva el número total de palabras en todas las cadenas.

[ ]:
```python
from functools import reduce

def contar_palabras(lista):
    return reduce(lambda acc, x: acc + len(x.split()), lista, 0)

# Ejemplo de uso
cadenas = ["hola mundo", "python es genial", "reduce es útil"]
resultado = contar_palabras(cadenas)
print(resultado)  # Output: 8
```

```
8
```