# apuntes_sge

# UT04: El lenguaje de programación Python

## PR0405: Ejercicios de programación funcional

**NOTA**: para poder evaluar esta práctica como APTA debes haber realizado, por lo menos, 6 ejercicios de los propuestos.

### 1.- Filtrado de una lista de números

Usa `filter` para obtener solo los números pares de una lista de enteros.

```
# Ejemplo de lista de entrada
numeros = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
# Resultado esperado: [2, 4, 6, 8, 10]
```

### 2.- Mapeo de temperaturas

Dada una lista de temperaturas en Celsius, usa `map` para convertirlas a Fahrenheit.

```
# Ejemplo de lista de entrada
celsius = [0, 20, 37, 100]
# Resultado esperado: [32.0, 68.0, 98.6, 212.0]
```

### 3.- Suma acumulativa

Utiliza `reduce` para obtener la suma acumulativa de una lista de números.

```
# Ejemplo de lista de entrada
numeros = [1, 2, 3, 4, 5]
# Resultado esperado: 15
```

### 4.- Palabras con cierta longitud

Dada una lista de palabras, usa `filter` para encontrar las que tienen más de cinco letras y luego `map` para convertirlas en mayúsculas.

```
# Ejemplo de lista de entrada
palabras = ["perro", "gato", "elefante", "oso", "jirafa"]
# Resultado esperado: ['ELEFANTE', 'JIRAFA']
```

## 5.- Multiplicación de pares

Dada una lista de números, utiliza `filter`, `map` y `reduce` para obtener el producto de los números pares.

```
# Ejemplo de lista de entrada
numeros = [1, 2, 3, 4, 5, 6]
# Resultado esperado: 48 (producto de 2, 4 y 6)
```

## 6.- Combinar operaciones en listas anidadas

Dada una lista de listas de enteros, usa `map`, `filter`, y `reduce` para obtener la suma de todos los números positivos.

```
# Ejemplo de lista de entrada
numeros = [[-3, 2, 7], [10, -5, 3], [0, 8, -2]]
# Resultado esperado: 30 (suma de 2, 7, 10, 3, y 8)
```

## 7.- Frecuencia de palabras

Dado un texto, crea una función que use `map` y `reduce` para obtener la frecuencia de cada palabra. Ignora las mayúsculas y minúsculas y supón que no hay símbolos de puntuación.

```
# Ejemplo de texto de entrada
texto = "Hola hola mundo mundo cruel"
# Resultado esperado: {'hola': 2, 'mundo': 2, 'cruel': 1}
```

## 8.- Intersección de conjuntos

Dadas dos listas de números, usa filter para obtener una lista de los elementos que están en ambas listas (sin usar conjuntos).

```
# Ejemplo de listas de entrada
lista1 = [1, 2, 3, 4, 5]
lista2 = [3, 4, 5, 6, 7]
# Resultado esperado: [3, 4, 5]
```

## 9.- Agrupación de palabras por longitud

Dada una lista de palabras, crea un diccionario donde la clave es la longitud de la palabra y el valor es una lista de palabras de esa longitud. Usa `map` y `filter`.

```
# Ejemplo de lista de entrada
palabras = ["sol", "luna", "estrella", "cielo", "mar"]
# Resultado esperado: {3: ['sol', 'mar'], 4: ['luna', 'cielo'], 8: ['estrella']}
```

## 10.- Concatenación de listas de caracteres

Dadas dos listas de caracteres, usa `reduce` para concatenarlas en una sola lista sin utilizar + o métodos de concatenación.

```
# Ejemplo de listas de entrada
lista1 = ['a', 'b', 'c']
lista2 = ['x', 'y', 'z']
# Resultado esperado: ['a', 'b', 'c', 'x', 'y', 'z']
```

## 11.- Calificación de alumnos

Dada una lista de tuplas con el nombre del alumno y su calificación, utiliza `map` y `filter` para obtener una lista con los nombres de los alumnos que han aprobado (nota >= 5).

```
# Ejemplo de lista de entrada
alumnos = [("Ana", 4), ("Bruno", 7), ("Clara", 5), ("David", 8)]
# Resultado esperado: ['Bruno', 'Clara', 'David']
```

## 12.- Calcular producto cartesiano

Dadas dos listas de números, usa `map` para obtener el producto cartesiano de ambas listas, devolviendo una lista de tuplas.

```
# Ejemplo de listas de entrada
```

```
lista1 = [1, 2]
lista2 = [3, 4]
# Resultado esperado: [(1, 3), (1, 4), (2, 3), (2, 4)]
```

## 13.- Pipe de transformaciones de listas

Implementa una función que tome una lista de funciones y una lista de números, y aplique cada función de la lista en secuencia sobre la lista de números usando `reduce`.

```
# Ejemplo de funciones y lista de entrada
funciones = [lambda x: x*2, lambda x: x+3, lambda x: x-1]
numeros = [1, 2, 3]
# Resultado esperado: [((1*2)+3)-1, ((2*2)+3)-1, ((3*2)+3)-1] -> [4, 6, 8]
```

## 14.- Aplicar operaciones de cadena

Dada una lista de cadenas, usa `map` y `filter` para crear una nueva lista con las cadenas que tengan más de tres letras y en las que todas las letras sean mayúsculas. Además, convierte el primer carácter en minúscula.

```
# Ejemplo de lista de entrada
palabras = ["HOLA", "MUNDO", "SOL", "CIELO", "mar"]
# Resultado esperado: ['hOLA', 'mUNDO', 'cIELO']
```