m11-julihocc

March 31, 2024

1 Módulo 11 | Funciones Avanzadas

Juliho Castillo

1.1 Instrucciones

1. Generar un archivo tipo Notebook de Python que contenga el código fuente de varios ejercicios aplicados a los conceptos vistos:
    1. Función lambda para obtener la raíz cuadrada de un número.
    2. Función map, para obtener el largo de una cadena de palabras.
    3. Tip! La cadena tiene que ser dividida en palabras antes de empezar.
    4. Función reduce, que sirva para calcular el producto de una lista
    5. Función filter que sirva para encontrar palabras que contengan mayúsculas o números en un listado
2. Por cada ejercicio es necesario realizar el código en Python sin utilizar la función estudiada, y además generar el código utilizando la función estudiada. Incluir un ejemplo por cada una.

[ ]:
```python
# Importando las librerías necesarias
from functools import reduce

# Ejercicio 1: Función lambda para obtener la raíz cuadrada de un número.
# Tradicional
def raiz_cuadrada_tradicional(numero):
    return numero ** 0.5

# Con función lambda
raiz_cuadrada_lambda = lambda numero: numero ** 0.5

# Ejemplo de uso
numero = 9
print(f"Raíz cuadrada de {numero} (tradicional):␣
 ↪{raiz_cuadrada_tradicional(numero)}")
print(f"Raíz cuadrada de {numero} (lambda): {raiz_cuadrada_lambda(numero)}")
```

```
Raíz cuadrada de 9 (tradicional): 3.0
Raíz cuadrada de 9 (lambda): 3.0
```

[ ]:
```python
# Ejercicio 2: Función map para obtener el largo de una cadena de palabras.
cadena = "Hola mundo Python y programación"
# Tradicional
def largo_palabras_tradicional(cadena):
    palabras = cadena.split()
    largo = []
    for palabra in palabras:
        largo.append(len(palabra))
    return largo

# Con función map
largo_palabras_map = list(map(len, cadena.split()))

# Ejemplo de uso
print(f"Largo de palabras (tradicional): {largo_palabras_tradicional(cadena)}")
print(f"Largo de palabras (map): {largo_palabras_map}")
```

```
Largo de palabras (tradicional): [4, 5, 6, 1, 12]
Largo de palabras (map): [4, 5, 6, 1, 12]
```

[ ]:
```python
# Ejercicio 3: Función reduce para calcular el producto de una lista.
lista_numeros = [1, 2, 3, 4, 5]
# Tradicional
def producto_tradicional(lista):
    resultado = 1
    for numero in lista:
        resultado *= numero
    return resultado

# Con función reduce
producto_reduce = reduce(lambda x, y: x*y, lista_numeros)

# Ejemplo de uso
print(f"Producto de la lista (tradicional): ␣
 ↪{producto_tradicional(lista_numeros)}")
print(f"Producto de la lista (reduce): {producto_reduce}")
```

```
Producto de la lista (tradicional): 120
Producto de la lista (reduce): 120
```

[ ]:
```python
# Ejercicio 4: Función filter para encontrar palabras que contengan mayúsculas␣
 ↪o números en un listado.
listado_palabras = ["Hola", "Mundo", "python3", "Programación", "2pac"]
# Tradicional
def palabras_con_mayusculas_o_numeros_tradicional(listado):
    resultado = []
    for palabra in listado:
```

```
        if any(c.isupper() for c in palabra) or any(c.isdigit() for c in␣
 ↪palabra):
            resultado.append(palabra)
    return resultado

# Con función filter
palabras_con_mayusculas_o_numeros_filter = list(filter(lambda palabra: any(c.
 ↪isupper() for c in palabra) or any(c.isdigit() for c in palabra),␣
 ↪listado_palabras))

# Ejemplo de uso
print(f"Palabras con mayúsculas o números (tradicional):␣
 ↪{palabras_con_mayusculas_o_numeros_tradicional(listado_palabras)}")
print(f"Palabras con mayúsculas o números (filter):␣
 ↪{palabras_con_mayusculas_o_numeros_filter}")
```

```
Palabras con mayúsculas o números (tradicional): ['Hola', 'Mundo', 'python3',
'Programación', '2pac']
Palabras con mayúsculas o números (filter): ['Hola', 'Mundo', 'python3',
'Programación', '2pac']
```