EXERCISE 1: TRANSFORM AND CLEAN DATA

Objective:

You are given a list of product names that contain extra spaces and inconsistent capitalization.

Code:

```
# Exercise 1: Transform and Clean Data

products = [" LAPTOP ", "phone ", " Tablet", "CAMERA "]

# Removing extra spaces and Converting to title case
cleaned_products = list(map(lambda x: x.strip().title(), products))

print("Original products:")
print(products)
print("\nCleaned products:")
print(cleaned_products)
print("\nExpected Output: ['Laptop', 'Phone', 'Tablet', 'Camera']")

```

Explanation:

We use *map()* with a lambda to remove spaces and fix capitalization.
strip() cleans the spaces, title() fixes the formatting.

EXERCISE 2: CONVERT TEMPERATURES (CELSIUS TO FAHRENHEIT)

Objective:

Given a list of temperatures in Celsius, convert them to Fahrenheit using the formula $F = (9/5) \times C + 32$.

Code:

```
# Exercise 2: Convert Temperatures (Celsius -> Fahrenheit)

celsius = [0, 10, 20, 30, 40]

# Formula: F = (9/5) * C + 32
fahrenheit = list(map(lambda c: (9/5) * c + 32, celsius))

print("Temperatures in Celsius:")
print(celsius)
print("\nTemperatures in Fahrenheit:")
print(fahrenheit)
print("\nExpected Output: [32.0, 50.0, 68.0, 86.0, 104.0]")
```

Explanation:

Each Celsius value is converted to Fahrenheit with the formula (9/5) * C + 32.
map() applies this formula to every number in the list.

EXERCISE 3: APPLY MULTIPLE TRANSFORMATIONS

Objective:

You have a list of integers. You need to:

- Square each number.
- Then add 10 to each result.
  All using one *map()* and one lambda.

Code:

```
1. # Exercise 3: Apply Multiple Transformations
2.
3. nums = [1, 2, 3, 4, 5]
4.
5. # Square each number, then add 10
6. transformed = list(map(lambda x: x**2 + 10, nums))
7.
8. print("Original numbers:")
9. print(nums)
10. print("\nSquared + 10:")
11. print(transformed)
12. print("\nExpected Output: [11, 14, 19, 26, 35]")
```

Explanation:

The lambda takes each number, squares it, and adds 10.
map() runs this transformation on the entire list at once.

EXERCISE 4: EXTRACT FIRST AND LAST CHARACTERS

Objective:

Given a list of words, create a new list of tuples (first_char, last_char)

Code:

```
1. # Exercise 4: Extract First and Last Characters
2.
3. words = ["python", "lambda", "programming", "map", "function"]
4.
5. # Create tuples of (first_char, last_char) for each word
6. char_tuples = list(map(lambda c: (c[0], c[-1]), words))
7.
8. print("Words:")
9. print(words)
10. print("\n(First, Last) character tuples:")
11. print(char_tuples)
12. print("\nExpected Output: [('p', 'n'), ('l', 'a'), ('p', 'g'), ('m', 'p'),
('f', 'n')]")
```

Explanation:

The lambda takes each number, squares it, and adds 10.
map() runs this transformation on the entire list at once.

EXERCISE 5: NESTED MAP TRANSFORMATION (CHALLENGE)

Objective:

Use nested *map()* and lambda functions to increase each student's marks by 5% and round the results to the nearest integer.

Code:

```
1. # Exercise 5: Nested Map Transformation (Challenge 🌟)
2.
3. marks = [[45, 80, 70], [90, 60, 100], [88, 76, 92]]
4.
5. # Increase each mark by 5% and round to nearest integer
6. incMarks = list(
7.               map(lambda row: list(
8.                 map(lambda x: round(x * 1.05), row)
9.               ), marks)
10.            )
11.
12. print("Original marks:")
13. print(marks)
14. print("\nMarks increased by 5% (rounded):")
15. print(incMarks)
16. print("\nExpected Output: [[47, 84, 74], [95, 63, 105], [92, 80, 97]]")
17.
```

Explanation:

We use nested map() functions:

- The inner one increases each mark by 5% and rounds it.
- The outer one applies this to every row of marks.

EXERCISE 6: NORMALIZE NUMBERS BETWEEN 0 AND 1

Objective:

Create a program that normalizes a list of numbers between 0 and 1 using *map()* and lambda.

Code:

```
1. # Exercise 6: Normalize Numbers Between 0 and 1
2.
3. nums = [10, 20, 30, 40, 50]
4.
5. # Find min and max values
6. minVal = min(nums)
7. maxVal = max(nums)
8.
9. # Formula: (x - min) / (max - min)
10. normalized = list(map(lambda x: (x - minVal) / (maxVal - minVal), nums))
11.
12. print("Original numbers:")
13. print(nums)
```

```
print(f"\nMin value: {minVal}")
print(f"Max value: {maxVal}")
print("\nNormalized (0 to 1):")
print(normalized)

```

Explanation:

Using min-max normalization, we scale numbers to the range 0–1.
The lambda applies the formula (x - min) / (max - min) to each value.

EXERCISE 7: EXTRACT LENGTH OF EACH WORD IN SENTENCES

Objective:

Given a list of sentences, extract the length of each word in every sentence using nested *map()*.

Code:

```
# Exercise 7: Extract Length of Each Word in Sentences

sentences = [
    "Python is awesome",
    "Lambda functions are powerful",
    "Map applies transformations"
]

# extract the length of each word in every sentence using nested map()
word_lengths = list(map(
    lambda sentence: list(map(lambda word: len(word), sentence.split())),
    sentences
))

print("Sentences and their word lengths:\n")
for i in range(len(sentences)):
    print(f"Sentence {i+1}: '{sentences[i]}'")
    print(f"Word lengths: {word_lengths[i]}\n")
```

Explanation:

Each sentence is split into words.
Then a nested *map()* calculates the length of every word.
This gives a list of word-length lists.

All source code for these exercises is available in my GitHub repository:

https://github.com/Basem3sam/python-college-exercises