Lab 4 - Functional Programming (Section 2)

Course: Principles of Programming Languages (Code: IT092IU)

HCMIU-CSE, Summer 2024

Instructor: Le Thi Ngoc Hanh, PhD

1 Higher - order Functions: Overview

any() - is used to test whether any elements of an iterable meet a certain condition
Syntax: *any()* returns “True” if any element in the iterable is a true value.

all() - is used to test whether all elements of an iterable meet a certain condition
Syntax: *all()* returns “True” if all of the elements are true values.

Compose - combines multiple functions into a single function. This combined function applies the given functions in sequence, where the output of one function becomes the input of the next.

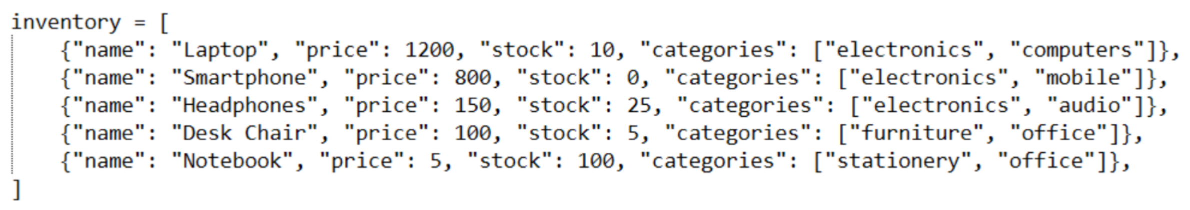
Currying - We can use higher-order functions to convert a function that takes multiple arguments into a chain of functions that each take a single argument. More specifically, given a function $f(x, y)$, we can define a function g such that $g(x)(y)$ is equivalent to $f(x, y)$. Here, g is a higher-order function that takes in a single argument x and returns another function that takes in a single argument y. This transformation is called currying. Currying can make function composition and partial application more convenient.

2 Exercises

Exercise 1

You are managing an inventory system for an e-commerce platform. Each product in the inventory has the following properties:
- name: The name of the product.
- price: The price of the product.
- stock: The number of items available in stock.
- categories: A list of categories the product belongs to.

The dictionary of products is as below:



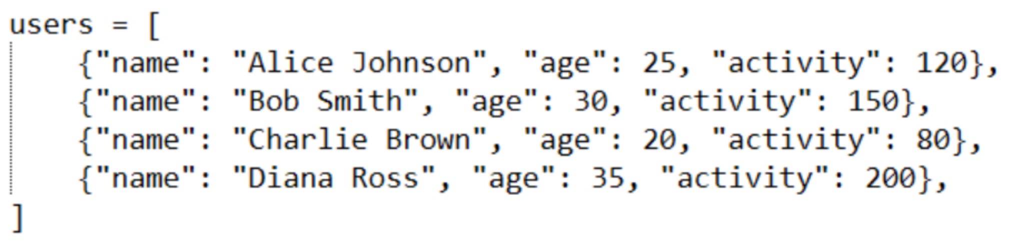
```
inventory = [
    {"name": "Laptop", "price": 1200, "stock": 10, "categories": ["electronics", "computers"]},
    {"name": "Smartphone", "price": 800, "stock": 0, "categories": ["electronics", "mobile"]},
    {"name": "Headphones", "price": 150, "stock": 25, "categories": ["electronics", "audio"]},
    {"name": "Desk Chair", "price": 100, "stock": 5, "categories": ["furniture", "office"]},
    {"name": "Notebook", "price": 5, "stock": 100, "categories": ["stationery", "office"]},
]
```

Please implement a program/function, by using *map()*, *lambda()*, *filter()* nested inside the usage of functions *any()* and *all()*, that checks the following conditions for the inventory :

- Determine if there are any products that are out of stock.
- Verify if all products in a specific category are available and have a price greater than a specified threshold δ, where δ is input by users.

Exercise 2

Given a dictionary of users as:



```
users = [
    {"name": "Alice Johnson", "age": 25, "activity": 120},
    {"name": "Bob Smith", "age": 30, "activity": 150},
    {"name": "Charlie Brown", "age": 20, "activity": 80},
    {"name": "Diana Ross", "age": 35, "activity": 200},
]
```

Define the **curried functions** with the below instruction:

- filter_by_min_age(min_age): Filters users who are above the given minimum age.
- format_name(format_style): Formats the user's name according to the specified style.
- calculate_score(criteria): Calculates the user's score based on the given criteria.
- Combine the curried functions into a data processing pipeline.

Exercise 3 Create a series of text processing functions and **use compose()** to build a text processing pipeline. The pipeline should perform the following transformations on a given text:

- Remove punctuation.
- Convert the text to lowercase.

- Split the text into a list of words.
- Filter out common stop words (like “the”, “is”, “in”, etc.).

Hint: Define individual text processing functions for each transformation. Use the compose() function to combine these transformations into a single processing pipeline. Apply the composed function to a sample text and output the segmented result.