EE6310

Problem Set 4

1. You are given two lists of integers. (a) Write a python code that calculate the sum of the products of corresponding elements from two lists, list1 and list2. Use zip in iteration. (b) Write the code in the function form that takes two lists as inputs

def sum\_of\_products(list1, list2):

result = 0

for x, y in zip(list1, list2):

result += x \* y

return result

# Test the function

list1 = [1, 2, 3]

list2 = [4, 5, 6]

print(sum\_of\_products(list1, list2))

1. Given two lists of integers, write a function multiply\_odd\_indices that takes two lists as input and returns a new list where each element is the product of the corresponding elements from two lists, list1 and lsit2, but only for the odd indices of the two lists. Use zip and enumerate in the iteration.

def multiply\_odd\_indices(list1, list2):

result = []

for i, (x, y) in enumerate(zip(list1, list2), start=1):

if i % 2 != 0:

result.append(x \* y)

return result

# Test the function

list1 = [1, 2, 3, 4, 5]

list2 = [2, 3, 4, 5, 6]

print(multiply\_odd\_indices(list1, list2))

1. Suppose you are managing a small bookstore and you want to create a program to keep track of your inventory. Write a Python function called update\_inventory that takes in two dictionaries as input parameters:

The first dictionary represents the current inventory of books in your store. The keys are the titles of the books (strings), and the values are the quantities of each book in stock (integers). The example is below:

current\_inventory = {

"Python Programming": 15,

"Introduction to Algorithms": 10,

"Data Structures and Algorithms in Python": 20

}

The second dictionary represents a shipment of new books that have arrived. The keys are also book titles (strings), and the values are the quantities of each book that have been delivered (integers). The example is below:

shipment = {

"Python Programming": 5,

"Introduction to Python": 8,

"Introduction to Algorithms": 5,

"Machine Learning Basics": 12

}

The function should update the current inventory dictionary to reflect the arrival of the new books. If a book title already exists in the current inventory, you should add the quantity from the shipment to the existing quantity. If a book title is not already in the current inventory, you should add it with its corresponding quantity from the shipment.

Your function should return the updated inventory dictionary.

def update\_inventory(current\_inventory, shipment):

updated\_inventory = current\_inventory.copy() # Create a copy of the current inventory

for book\_title, quantity in shipment.items():

if book\_title in updated\_inventory:

# Book already exists in inventory, so update the quantity

updated\_inventory[book\_title] += quantity

else:

# Book is not in inventory, so add it with the quantity from the shipment

updated\_inventory[book\_title] = quantity

return updated\_inventory

1. You are tasked with creating a Python class to represent a basic bank account. Your class, called BankAccount, should have the following attributes and methods:

Attributes:

account\_number: A unique identifier for each bank account (an integer).

account\_holder: The name of the account holder (a string).

balance: The current balance in the account (a float).

Methods:

\_\_init\_\_(): A constructor method that initializes the account\_number, account\_holder, and balance attributes.

deposit(amount): A method that takes in a parameter amount (a float) and adds it to the account's balance.

withdraw(amount): A method that takes in a parameter amount (a float) and subtracts it from the account's balance. Ensure that the withdrawal amount does not exceed the current balance. If the withdrawal amount is greater than the balance, print a message indicating insufficient funds.

get\_balance(): A method that returns the current balance of the account.

Create an instance of the BankAccount class and demonstrate its usage by performing the following operations:

* Initialize the bank account with an account number, account holder's name, and initial balance.
* Deposit some amount into the account.
* Withdraw some amount from the account.
* Display the current balance after each transaction..

class BankAccount:

def \_\_init\_\_(self, account\_number, account\_holder, balance):

self.account\_number = account\_number

self.account\_holder = account\_holder

self.balance = balance

def deposit(self, amount):

self.balance += amount

def withdraw(self, amount):

if amount <= self.balance:

self.balance -= amount

else:

print("Insufficient funds. Balance remains:", self.balance)

def get\_balance(self):

return self.balance