Ex No.04	Dictionary Manipulation	Reg. No: URK24CS9068
14.02.25		

Q1. Given the following dictionary:

```
inventory = {
    'gold' : 500,
    'pouch' : ['flint', 'twine', 'gemstone'],
    'backpack' : ['xylophone', 'dagger', 'bedroll', 'bread loaf']
}
```

Try to do the followings:

- **Add a key to inventory called 'pocket'.**
- **Set the value of 'pocket' to be a list consisting of the strings 'seashell', 'strange berry', and 'lint'.**
- **sort() the items in the list stored under the 'backpack' key.**
- **Then .remove('dagger') from the list of items stored under the 'backpack' key.**
- **Add 50 to the number stored under the 'gold' key.**

Aim: To perform dictionary manipulation operations on a given inventory using Python.

Objective: To understand and implement dictionary operations in Python such as adding keys, modifying list elements, and updating numerical values.

Algorithm:

Step 1: Start

Step 2: Define the dictionary inventory with keys 'gold', 'pouch', and 'backpack'

Step 3: Add a new key 'pocket' with the value ['seashell', 'strange berry', 'lint']

Step 4: Sort the list of items under the 'backpack' key using .sort()

Step 5: Remove 'dagger' from the 'backpack' list using .remove()

Step 6: Add 50 to the existing value of the 'gold' key

Step 7: Print the updated inventory dictionary

Step 8: End

Program:

```
print("D. Brian Gabriel")
print("URK24CS9068")
print("------------------")
inventory = {
'gold' : 500,
'pouch' : ['flint', 'twine', 'gemstone'],
'backpack' : ['xylophone', 'dagger', 'bedroll', 'bread loaf']
}
inventory['pocket']=['seashell','strange berry','lint']
inventory['backpack'].sort()
inventory['backpack'].remove('dagger')
inventory['gold']+=50
print(inventory)
```

Output:

```
D. Brian Gabriel
URK24CS9068
------------------
{'gold': 550, 'pouch': ['flint', 'twine', 'gemstone'], 'backpack': ['bedroll', 'bread loaf', 'xylophone'], 'pocket': ['seashell', 'strange berry', 'lint']}
```

Result: The program successfully performs all the required operations on the dictionary.

Q2. Follow the steps bellow: -Create a new dictionary called prices using {} format like the example above.

● Put these values in your prices dictionary:

"banana": 4,

"apple": 2,

"orange": 1.5,

"pear": 3

● Loop through each key in prices. For each key, print out the key along with its price and stock information. Print the answer in the following format:

apple

price: 2

stock: 0

● Let's determine how much money you would make if you sold all of your food.

o Create a variable called total and set it to zero.

o Loop through the prices dictionaries. For each key in prices, multiply the number in prices by the number in stock. Print that value into the console and then add it to total.

o Finally, outside your loop, print total.

Aim: To calculate the total revenue from selling all items using dictionaries in Python.

Objective: To understand and implement dictionary creation, iteration, and value calculations using loops in Python.

Algorithm:

Step 1: Start
Step 2: Create a dictionary called prices with items and their prices
Step 3: Create another dictionary called stock with the same items and their stock values
Step 4: Loop through each key in prices
Step 5: For each item, print the item name, its price, and its stock
Step 6: Initialize a variable total to 0
Step 7: Loop again through the prices dictionary
Step 8: For each item, multiply its price with the stock and print the result
Step 9: Add the result to the total variable
Step 10: After the loop, print the total
Step 11: End

Program:

```
print("D. Brian Gabriel")
print("URK24CS9068")
print("------------------")
prices={"banana": [4,50],
"apple": [2,60],
"orange": [1.5,70],
"pear": [3,50]}
total=0
for fruits in prices:
    a=prices[fruits][1]*prices[fruits][0]
    print(fruits)
    print("price :",prices[fruits][0])
    print("stock:",prices[fruits][1])
    print(a)
    total+=a
print("Total money you would make if you sold all of your food is",total)
```

Output:

```
D. Brian Gabriel
URK24CS9068
------------------
banana
price : 4
stock: 50
200
apple
price : 2
stock: 60
120
orange
price : 1.5
stock: 70
105.0
pear
price : 3
stock: 50
150
Total money you would make if you sold all of your food is 575.0
```

Result: The program successfully loops through the prices dictionary, prints each item's price and stock, and calculates the total money that would be earned if all items were sold. The total amount is computed accurately based on price × stock for each item.

Q3. Follow the steps:

- **First, make a list called groceries with the values "banana", "orange", and "apple".**
- **Define this two dictionaries:**

  stock = {"banana": 6, "apple": 0, "orange": 32, "pear": 15}

  prices = {"banana": 4, "apple": 2, "orange": 1.5, "pear": 3}
- **Define a function compute_bill that takes one argument food as input. In the function, create a variable total with an initial value of zero.For each item in the food list, add the price of that item to total. Finally, return the total. Ignore whether or not the item you're billing for is in stock.Note that your function should work for any food list.**
- **Make the following changes to your compute_bill function:**
  - **While you loop through each item of food, only add the price of the item to total if the item's stock count is greater than zero.**
  - **If the item is in stock and after you add the price to the total, subtract one from the item's stock count.**

Aim: To create a function that calculates the total bill of selected groceries while updating the stock accordingly.

Objective: To understand and implement function creation, conditional logic, and dictionary manipulation in Python for billing purposes.

Algorithm:

Step 1: Start
Step 2: Create a list called groceries with values "banana", "orange", and "apple"
Step 3: Define two dictionaries stock and prices with respective values
Step 4: Define a function compute_bill(food)
Step 5: Initialize a variable total to 0
Step 6: Loop through each item in the food list
Step 7: Check if the item is in stock (i.e., stock[item] > 0)
Step 8: If in stock, add the price to total and reduce the stock by 1
Step 9: After the loop, return the total value
Step 10: Call the function with the groceries list and print the result
Step 11: End

Program:

```
print("D. Brian Gabriel")
print("URK24CS9068")
print("------------------")
def compute_bill(food):
    global prices
    global stock
    total=0
    for i in food:
        if i in prices and stock[i]>0:
            total+=prices[i]
            stock[i]-=1
    return total
L1=["banana", "orange","apple"]
stock = {"banana": 6,"apple": 0,"orange": 32,"pear": 15}
prices = {"banana": 4,"apple": 2,"orange": 1.5,"pear": 3}
total=compute_bill(L1)
print(total)
print(stock)
print(prices)
```

Output:

```
D. Brian Gabriel
URK24CS9068
------------------
5.5
{'banana': 5, 'apple': 0, 'orange': 31, 'pear': 15}
{'banana': 4, 'apple': 2, 'orange': 1.5, 'pear': 3}
```

Result: The program accurately computes the total bill for a given grocery list. It adds the item's price to the total only if it is in stock, and decreases the stock count after billing. The updated stock and total amount are displayed correctly.

Q4. The aim of this exercise is to make a gradebook for teacher's students. Try to follow the steps:

- **Create three dictionaries: lloyd, alice, and tyler.**
- **Give each dictionary the keys "name", "homework", "quizzes", and "tests".**

  Have the "name" key be the name of the student (that is, lloyd's name should be "Lloyd") and the other keys should be an empty list.

  Now copy this code:

  lloyd = {"name": "Lloyd", "homework": [90.0,97.0,75.0,92.0], "quizzes": [88.0,40.0,94.0], "tests": [75.0,90.0]}

  alice = {"name": "Alice", "homework": [100.0, 92.0, 98.0, 100.0], "quizzes": [82.0, 83.0, 91.0], "tests": [89.0, 97.0]}

  tyler = {"name": "Tyler", "homework": [0.0, 87.0, 75.0, 22.0], "quizzes": [0.0, 75.0, 78.0], "tests": [100.0, 100.0]}

- **Below your code, create a list called students that contains lloyd, alice, and tyler.**
- **For each student in your students list, print out that student's data, as follows:**
  - **o print the student's name**
  - **o print the student's homework**
  - **o print the student's quizzes**
  - **o print the student's tests**
- **Write a function average that takes a list of numbers and returns the average.**
  - **o Define a function called average that has one argument, numbers.**
  - **o Inside that function, call the built-in sum() function with the numbers list as a parameter. Store the result in a variable called total.**
  - **o Use float() to convert total and store the result in total.**
  - **o Divide total by the length of the numbers list. Use the built-in len() function to calculate that.**
  - **o Return that result.**

● Write a function called get_average that takes a student dictionary (like lloyd, alice, or tyler) as input and returns his/her weighted average.

- **o Define a function called get_average that takes one argument called student.**
- **o Make a variable homework that stores the average() of student["homework"].**
- **o Repeat step 2 for "quizzes" and "tests".**
- **o Multiply the 3 averages by their weights and return the sum of those three. Homework is 10%, quizzes are 30% and tests are 60%.**

● Define a new function called get_letter_grade that has one argument called score. Expect score to be a number.

- **o Inside your function, test score using a chain of if: / elif: / else: statements, like so:**
  - **▪ If score is 90 or above: return "A"**
  - **▪ Else if score is 80 or above: return "B"**
  - **▪ Else if score is 70 or above: return "C"**
  - **▪ Else if score is 60 or above: return "D"**
  - **▪ Otherwise: return "F"**
- **o Finally, test your function. Call your get_letter_grade function with the result of get_average(lloyd). Print the resulting letter grade.**

● Define a function called get_class_average that has one argument, students. You can expect students to be a list containing your three students.

- **o First, make an empty list called results.**
- **o For each student item in the class list, calculate get_average(student) and then call results.append() with that result.**
- **o Finally, return the result of calling average() with results.**

● Finally, print out the result of calling get_class_averagewith your students list. Your students should be [lloyd, alice, tyler].

● Then, print the result of get_letter_grade for the class's average.

Aim: To create a gradebook system using dictionaries and functions in Python to compute student and class averages, as well as assign letter grades.

Objective: To implement dictionary and list operations in Python to manage student records and compute weighted averages and letter grades.

Algorithm:

Step 1: Start
Step 2: Create dictionaries for each student with keys "name", "homework", "quizzes", and "tests"
Step 3: Populate the dictionaries with sample data
Step 4: Store all students in a list called students
Step 5: Print each student's name and their homework, quizzes, and test scores
Step 6: Define average() function to return average of a list
Step 7: Define get_average() to calculate weighted average for a student
Step 8: Define get_letter_grade() to return grade based on average
Step 9: Print letter grade for an individual student (e.g., Lloyd)
Step 10: Define get_class_average() to compute class average
Step 11: Print class average and its letter grade
Step 12: End

Program:

```
print("D. Brian Gabriel")
print("URK24CS9068")
print("------------------")
lloyd = {"name": "Lloyd","homework": [90.0,97.0,75.0,92.0],"quizzes": [88.0,40.0,94.0],"tests": [75.0,90.0]}
alice={"name": "Alice","homework": [100.0, 92.0, 98.0, 100.0],"quizzes": [82.0, 83.0, 91.0],"tests": [89.0, 97.0]}
tyler = {"name": "Tyler","homework": [0.0, 87.0, 75.0, 22.0],"quizzes": [0.0, 75.0, 78.0],"tests": [100.0, 100.0]}
students = [lloyd, alice, tyler]
for student in students:
    print("Name:", student["name"])
    print("Homework:", student["homework"])
    print("Quizzes:", student["quizzes"])
    print("Tests:", student["tests"])
    print("----------------------")
def average(numbers):
    total = sum(numbers)
    total = float(total)
    return total / len(numbers)
def get_average(student):
    homework = average(student["homework"])
    quizzes = average(student["quizzes"])
    tests = average(student["tests"])
    return 0.1 * homework + 0.3 * quizzes + 0.6 * tests
def get_letter_grade(score):
    if score >= 90:
        return "A"
    elif score >= 80:
        return "B"
    elif score >= 70:
        return "C"
    elif score >= 60:
        return "D"
    else:
        return "F"
```

```
print("Lloyd's letter grade:", get_letter_grade(get_average(lloyd)))
def get_class_average(students):
    results = []
    for student in students:
        results.append(get_average(student))
    return average(results)
class_avg = get_class_average(students)
print("Class average:", class_avg)
print("Class letter grade:", get_letter_grade(class_avg))
```

Output:

```
D. Brian Gabriel
URK24CS9068
------------------
Name: Lloyd
Homework: [90.0, 97.0, 75.0, 92.0]
Quizzes: [88.0, 40.0, 94.0]
Tests: [75.0, 90.0]
----------------------
Name: Alice
Homework: [100.0, 92.0, 98.0, 100.0]
Quizzes: [82.0, 83.0, 91.0]
Tests: [89.0, 97.0]
----------------------
Name: Tyler
Homework: [0.0, 87.0, 75.0, 22.0]
Quizzes: [0.0, 75.0, 78.0]
Tests: [100.0, 100.0]
----------------------
Lloyd's letter grade: B
Class average: 83.86666666666666
Class letter grade: B
```

Result: The program correctly calculates averages and assigns letter grades for students and the class.