Programming Assignment Unit 7

**Problem statement:**

In a program, a dictionary contains lists of students and their courses. The teacher is interested to have a dictionary that has the courses as key and the students enrolled in each course as values. Each key has three different values.

To address this requirement, write a function to invert the dictionary and implement a solution that satisfies the teacher’s need. In particular, the function will need to turn each of the list items into separate keys in the inverted dictionary. Also provide a technical explanation for the code and its output in minimum 200 words.

Sample input:

{

    'Stud1: ['CS1101', 'CS2402', 'CS2001'],

    'Stud2: ['CS2402’,’CS2001’,’CS1102’]

    }

Inverted Output:

{

‘CS1101’: [‘Stud1’],

‘CS2402’:['Stud1’,’Stud2’],

‘CS2001’: ['Stud1’,’Stud2’]

‘CS 1102’[‘Stud2’]

}

**Programming Instructions:**

* Print the original dictionary as well as the inverted dictionary.
* Include your the Python program and the output in your submission.
* The code and its output must be explained technically. The explanation can be provided before or after the code, or in the form of comments within the code.

:: Solution ::

This Python script provides functionality to invert a dictionary representing students and their enrolled courses into a dictionary where courses are keys and students are values, with each course having at most three different students. The `invert\_dictionary` function achieves this inversion by iterating through the original dictionary and populating the inverted dictionary accordingly. Additionally, the script includes a `print\_courses\_and\_students` function to display the courses and their enrolled students. Comments throughout the code provide clarity on the purpose and functionality of each component. The script demonstrates the usage of these functions with an example original dictionary, printing both the original and inverted dictionaries along with their respective courses and students enrolled.

:: Code ::

def invert\_dictionary(original\_dict):

"""

Here we define a function called invert\_dictionary that takes

in a dictionary called original\_dict as input. It creates an empty

dictionary called inverted\_dict to store the inverted mapping.

The function then iterates over the key-value pairs in original\_dict.

For each pair, it checks if the course is already present in inverted\_dict.

If not, it adds an empty list as the value for that course.

Then it appends the student to the list of students for that course in

inverted\_dict. The loop breaks if the list of students for a course reaches

a length of 3.

In the end, the function returns the inverted\_dict containing the inverted mapping.

"""

inverted\_dict = {}

for students, courses in original\_dict.items():

for course in courses:

# If course is not already in inverted dictionary, add it with an empty list

if course not in inverted\_dict:

inverted\_dict[course] = []

# Append the student to the list of students for this course

inverted\_dict[course].append(students)

# Break if each key has three different values

if len(inverted\_dict[course]) == 3:

break

return inverted\_dict

def print\_courses\_and\_students(courses):

"""

Here we define a function called print\_courses\_and\_students that takes

a dictionary courses as input. The function iterates over the key-value

pairs, where each key represents a course and each value is a list of

students enrolled in that course.

For each course, the function prints the course name, followed by a list of

the enrolled students. The function uses f-strings to format the output.

"""

for course, students in courses.items(): # Iterate over the key-value pairs

print(f"Course: {course}") # Print the course

print("Students Enrolled:")

for student in students: # Print the list of students

print(f"- {student}") # Print each student

print()

# Original dictionary

original\_dict = {

"Arfan": ["Computer Science", "Mathematics", "Statistics"],

"Adnan": ["Computer Science", "Data Structures", "UML"],

"Aadil": ["Mathematics", "Statistics", "Data Structures"],

}

# Inverted dictionary

print("Original Dictionary:")

for student, courses in original\_dict.items(): # Iterate over the key-value pairs

print(f"Student: {student}") # Print the student

print("Courses Enrolled:")

for course in courses: # Print the list of courses

print(f"- {course}") # Print each course

print() # Print an empty line

inverted\_dict = invert\_dictionary(original\_dict) # Call the invert\_dictionary function

print("\nInverted Dictionary:") # Print the inverted dictionary

print\_courses\_and\_students(inverted\_dict) # Call the print\_courses\_and\_students function

# :: Code snapshot frocode.pngm VSCode editor ::

# :: Explanation ::

**Function Definition: `invert\_dictionary(original\_dict)`**

def invert\_dictionary(original\_dict):

This line defines a function named `invert\_dictionary` that takes one argument, `original\_dict`, which is the dictionary to be inverted.

inverted\_dict = {}

Here, an empty dictionary `inverted\_dict` is initialized. This dictionary will hold the inverted data structure where courses are keys and lists of students are values.

for students, courses in original\_dict.items():

This loop iterates over each key-value pair in the `original\_dict`. Each key represents a student, and the corresponding value is a list of courses that the student is enrolled in.

for course in courses:

Within the loop, it iterates over each course in the list of courses for the current student.

if course not in inverted\_dict:

inverted\_dict[course] = []

For each course, it checks if it already exists as a key in the `inverted\_dict`. If not, it initializes it with an empty list. This ensures that each course has an entry in the inverted dictionary.

inverted\_dict[course].append(students)

It appends the current student to the list of students associated with the current course in the `inverted\_dict`.

if len(inverted\_dict[course]) == 3:

break

It checks if the number of students enrolled in the current course has reached three. If so, it breaks out of the loop to ensure that each course in the inverted dictionary has at most three different students.

return inverted\_dict

Finally, it returns the inverted dictionary containing courses as keys and lists of students as values.

**Function Definition: `print\_courses\_and\_students(courses)`**

def print\_courses\_and\_students(courses):

This line defines a function named `print\_courses\_and\_students` that takes one argument, `courses`, which is the dictionary containing courses as keys and lists of students as values.

for course, students in courses.items(): # Iterate over the key-value pairs

Within the function, it iterates over the key-value pairs in the `courses` dictionary, where each key is a course, and the corresponding value is a list of students enrolled in that course.

print(f"Course: {course}") # Print the course

print("Students Enrolled:")

for student in students: # Print the list of students

print(f"- {student}") # Print each student

print()

For each course, it prints the course name and then iterates over the list of students enrolled in that course, printing each student's name preceded by a dash ("-"). After printing all students enrolled in a particular course, it prints a blank line to separate each course's information.

**The original dictionary**

original\_dict = {

"Arfan": ["Computer Science", "Mathematics", "Statistics"],

"Adnan": ["Computer Science", "Data Structures", "UML"],

"Aadil": ["Mathematics", "Statistics", "Data Structures"],

}

In this block we initialize the original\_dict dictionary with some student and courses data. Each key in the dictionary represents a student's name, and the corresponding value is a list of courses that the student is enrolled in.

print("Original Dictionary:")

for student, courses in original\_dict.items():

print(f"Student: {student}")

print("Courses Enrolled:")

for course in courses:

print(f"- {course}")

print()

Next we print the contents of the original\_dict. It iterates over each key-value pair in the dictionary using items(). For each student, it prints the student's name and then iterates over the list of courses they are enrolled in. For each course, it prints the course name preceded by a dash ("-"). After printing all courses for a student, it prints a blank line to separate each student's information.

inverted\_dict = invert\_dictionary(original\_dict)

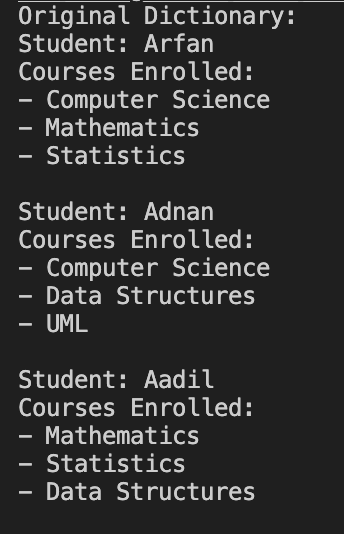
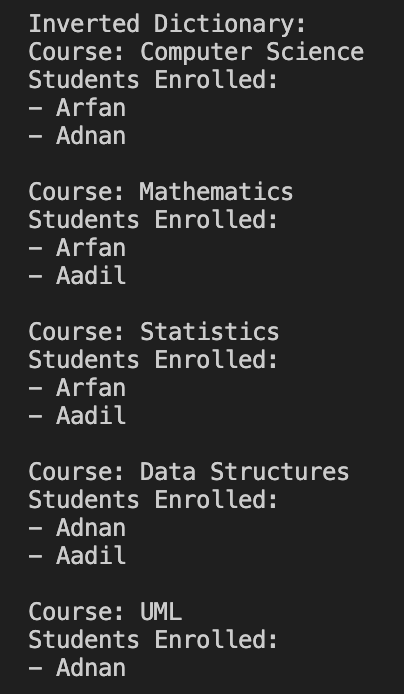
This calls the invert\_dictionary function, passing the original\_dict as an argument. It stores the result, which is the inverted dictionary, in the variable inverted\_dict.

print("\nInverted Dictionary:")

print\_courses\_and\_students(inverted\_dict)

Finally, it prints the contents of the inverted\_dict. It first prints a header indicating that it's printing the inverted dictionary. Then, it calls the print\_courses\_and\_students function, passing the inverted\_dict as an argument. This function prints each course and the students enrolled in that course. Each course's information is printed with the course name followed by the list of enrolled students.

# :: Output ::



# References:

1. docs.python.org. *Python Official Documentation*. Python 2.12.2 Documentation. <https://docs.python.org/>
2. Python Official Documentation. *Dictionaries*. Data Structures. <https://docs.python.org/3/tutorial/datastructures.html#dictionaries>
3. Python Official Documentation. *Control Flow Tools*. if-for Tools. <https://docs.python.org/3/tutorial/controlflow.html>
4. Matthes, Eric, 2022, *Python Crash Course*. No Starch Press
5. Sweigart, Al, 2019, *Automate the Boring Stuff with Python*. No Starch Press
6. Ramalho, Luciano, 2015, *Fluent Python.* O’Reilly Media