New York University

CSC-101 Intro To Programming Language

Tutor: Raiyan Reza
Student Name: Ameen

Final Prep
August 1, 2022

Dictionary

Question modified from Gaddis 9.1

(a) Write a function courseInfo(courseNum:str)→tuple that creates a dictionary containing course numbers as a key. The information has values should be room number, insturctor name, and meeting time.

Course Number	Room Number	Instructor	Meeting Time
CS101	3004	Haynes	8:00 am
CS102	4501	Alvarado	9:00 am
CS103	6755	Rich	10:00 am
NT110	1244	Burke	11:00 am
CM241	1411	Lee	1:00 pm

Therefore, when an input is a valid course number the function should return the room number, instructor name, and meeting time as a tuple. Otherwise, it should display a error message saying: " The course ⟨course number⟩ does not exist."

Examples of input and output

Input: CS101

Output: (3004, Haynes, 8:00 am)

Input: PHY101

Output: "The course PHY101 does not exist"

(b) Imagine you are given a list, courses=[CS101,CS102,CS103,NT110,CM241]. You this lista and the function you have defined in part a) to generate an output on the console where each line will correspond to one row in the table above.

Output:

CS101, 3004, Haynes, 8:00 am

CS102, 4501, Alvarado, 9:00 am

CS103, 6755, Rich, 10:00 am

NT110, 1244, Burke, 11:00 am

CM241, 1411, Lee, 1:00 PM

Question modified from Gaddis 9.2

(a) You are given two lists. The list provinces is a list of provinces of China. The list is as follows:

provinces=

[Anhui, Fujian, Gansu, Guangdong, Guizhou, Hainan, Hebei, Heilongjiang, Henan, Hubei, Hunan,

Jiangsu, Jiangxi, Jilin, Liaoning, Qinghai, Shaanxi, Shandong, Shanxi, Sichuan, Yunnan, Zhejiang]

The list capitals is a list of capitals of the provinces of China respectively.

capitals=

[Hefei, Fujian, Lanzhou, Guangzhou

Guiyang, Haikou, Shijiazhuang, Harbin, Zhengzhou, Wuhan, Changsha,

Nanjin, Nanchang, Changchun, Shenyang, Xining, Xian

Jinan, Taiyuan, Chengdu, Kumming, Hangzhou]

The relationship between the two lists is for a valid index, i, capitals[i] is the capital of the provinces[i].

Given this property create a dictionary, capitalOfProvinces, such that the provinces are keys, and capitals values.

(b) Using the dictionary in part a, capitalOfProvinces, create a simple game on I/O console. At the start of the game, the game asks for the name of the user, and displays the following message on the console:"Welcome ⟨user name⟩!" Unless, the user types in "EXIT", the game will continue. In each round, the user has to state the capital of a province selected on random basis. If the player provides the right answer, the game must say, "CORRECT!", otherwise, "WRONG! Better luck next time!". In addition, in each round the number of round, the correct guesses the user has taken, the incorrect guesses made, and his score should be displayed. The first round will be round 1. If he gets a the right answer, his score will be incremented by one. When the user presses "EXIT", you will display the score and produce an exit message:

"Thank you for playing ⟨name of user⟩! Your score is ⟨score⟩"

(c) Assume the dictionary, capitalOfProvinces, in part a) is properly created with the right keys and values. Now, produce an output on the console where the information is displayed in a tabular format, where each row would correspond to one province-capital pair. The exact empty space is not important but the console output must be tabbed.

Output:

PROVINCE	CAPITAL
Anhui	Hefei
Fujian	Fujian
Gansu	Lanzhou
Guangdong	Guangzhou .
.	
.	

2D Lists

Using 2D List to redo Gaddis 9.1

(a) We can use 2D list to do redo the question from Gaddis 9.1. The following table can be encoded in the form of a 2D list:

Course Number	Room Number	Instructor	Meeting Time
CS101	3004	Haynes	8:00 am
CS102	4501	Alvarado	9:00 am
CS103	6755	Rich	10:00 am
NT110	1244	Burke	11:00 am
CM241	1411	Lee	1:00 pm

So put this table in the form of a 2D list where each internal list corresponds to one row of the table.

(b) Now, use the 2D list representation of the table in part a to define a function

courseInfo2(courseNum:str)→list[str]

So given an input of valid course number it should return a list containing information about room number, instructor, and meeting time. if the course number is invalid, the function should display a message:

" The course ⟨course number⟩ does not exist."

Example

Input: CS101

Output: [3004, Haynes, 8:00 am]

Input: PHY101

Output: "The course PHY101 does not exist"

(c) Based on your solution in part b) and previous material taught about dictionaries can you guess why dictionary is a more time efficient way of storing and retrieving the relevant information for this question?

Matrices

(a) In theory, one of the simple applications of using 2D lists in Python is for representing matrices. In practice, there are highly optimized Python libraries, such as NumPy, for implementing and using matrices. So the naive implementations of matrices are useless in all but simple case of demonstration. For example, take the following 3x3 matrix:

$$\begin{pmatrix} 2 & 55 & -1 \\ 50 & 9 & 150 \\ -3 & 8 & -200 \end{pmatrix}$$

This can be represented in the format of list in a list or in other words 2D list:

$[[2, 55, -1], [50, 9, 150], [-3, 8, -200]]$.

Your task is is define a function:

printMatrix(A:list[list[int]])→None

It will take a matrix represented as a 2D list, and print out each row. For example, let:

A=$[[2, 55, -1], [50, 9, 150], [-3, 8, -200]]$,

Then, calling printMatrix(A) will print out:

Row 1: 2, 55, -1

Row 2: 50, 9, 150

Row 3:-3, 8, -200

(b) Redo part a) but the output format has to be print out each inner list on separate line on the console. For example if printMatrix(A) is called, where A is the same matrix as in part a) you will get:

$[2, 55, -1]$

$[50, 9, 150]$

$[-3, 8, -200]$

(c) Your task is now to define a function:

addMatrices(A:list[list[int]],B:list[list[int])→list[list[int]]

Parameter A and B are matrices represented as 2D list. Your function should generate a matrix C, such that C=A+B. A and B will have the same dimensions so the addition operation on A and B is defined. You, therefore, do not have to worry about inputs for which the addition operation is not defined. If you have forgotten matrix addition, the following example should help you remember matrix addition:

$$\begin{pmatrix} 2 & 50 \\ -5 & 10 \end{pmatrix} + \begin{pmatrix} 2 & 10 \\ 5 & 25 \end{pmatrix} = \begin{pmatrix} 4 & 60 \\ 0 & 35 \end{pmatrix}$$

More explicitly C[i][j]=A[i][j] + B[i][j], where i and j are valid row and column numbers respectively.

(d) You now have a significantly more difficult task of defining the following function:

mulMatrix(A:list[list[int]],B:list[list[int])→list[list[int]

Parameter A and Parameter B are once again matrices represented as 2D list. Your function should generate a matrix C,such that C= AxB. If A is a mxn matrix, then B will be a pxm matrix. You, therefore, do not have to worry about any argument being passed for which the matrix operation is not defined. If you have forgotten matrix multiplication, check out the following example:

$$\begin{pmatrix} 2 & 50 \\ -5 & 10 \end{pmatrix} \text{x} \begin{pmatrix} 2 & 10 \\ 5 & 25 \end{pmatrix} = \begin{pmatrix} (2)(2)+(50)(5) & (2)(10)+(50)(25) \\ (-5)(2)+(10)(25) & (-5)(10)+(10)(25) \end{pmatrix}$$

$$= \begin{pmatrix} 254 & 1270 \\ 240 & 200 \end{pmatrix}$$