First Semester – Assignment (1)

DMC-6102: Python Programming

Date: 17-Apr-23

Register Number: 2232MCA0058

Question 2

Syntax used in this program:

Here is the description of the syntax used in the program:

- def stud_rank(details): - This line defines a function called stud_rank that takes a single input parameter called details. This function will calculate the total internal marks for each student, rank them based on their marks, and return a sorted dictionary.
- for key, value in details.items(): - This line loops through each key-value pair in the details dictionary.
- total_marks = sum(value['ass1'].values()) + sum(value['ass2'].values()) - This line calculates the total internal marks for each student by adding up the values in the ass1 and ass2 dictionaries.
- value['imark'] = total_marks - This line adds the total marks to the dictionary for each student using the key imark.
- sorted_details = sorted(details.items(), key=lambda x: x[1]['imark'], reverse=True) - This line sorts the dictionary items in descending order based on the imark value.
- for key, value in sorted_details: - This line loops through each key-value pair in the sorted dictionary.
- value['rank'] = rank - This line adds the rank to the dictionary for each student using the key rank.
- return sorted_details - This line returns the sorted dictionary with ranks added to it.

Problem statement:

Write a function stud_rank(details) which performs the following actions: ¬ Details is a dictionary given as input argument. ¬ Calculates the total internal marks for each student. Total internal marks is simply the cumulative summation of all subject marks obtained in ass1 and ass2. ¬ Calculates the rank for each student based on the total internal marks. ¬ Sorts the elements in ascending order based on the computed rank. ¬ Test the function by creating a dictionary as given below

For example,

Input: details={20201010:
{'name':'Khan','age':18,‘ass1':{'phy':88,'chem':99,'mat':99,'py':99},'ass2':{'phy':88,'chem':99,'mat':99,'py':99},'imark':0,'rank':0}, 20201011:
{'name':'Sam','age':18,'ass1':{'phy':81,'chem':79,'mat':99,'py':89},'ass2':{'phy':80,'chem':89,'mat':79,'

py':79},'imark':0,'rank':0}, 20201012: {'name':'Ram','age':18,'ass1':{'phy':68,'chem':79,'mat':89,'py':99},'ass2':{'phy':58,'chem':69,'mat':79,'py':99},'imark':0,'rank':0}}

OUTPUT: [(20201012, {'name': 'Ram', 'age': 18, 'ass1': {'phy': 98, 'chem': 99, 'mat': 99, 'py': 99},'ass2': {'phy': 98, 'chem': 99, 'mat': 99, 'py': 99}, 'imark': 790,'rank': 3}), (20201010, {'name': 'Khan', 'age': 18,'ass1': {'phy': 88, 'chem': 99, 'mat': 99, 'py': 99}, 'ass2': {'phy': 88, 'chem': 99, 'mat': 99, 'py': 99}, 'imark': 770, 'rank': 1}), (20201011,{'name': 'Sam', 'age': 18,'ass1': {'phy': 81, 'chem': 79, 'mat': 99, 'py': 89}, 'ass2': {'phy': 80, 'chem': 89, 'mat': 79, 'py': 79}, 'imark': 675, 'rank': 2})]

Solution:

```
# -*- coding: utf-8 -*-
"""
Created on Sat Mar 25 12:43:05 2023

@author: Haryish Elangumaran
"""
import json #<-- Optional line

def stud_rank(details):
    # iterate over the dictionary to calculate total internal marks for
each student
    for key, value in details.items():
        total = sum(value['ass1'].values()) +
sum(value['ass2'].values())
        details[key]['imark'] = total

    # sort the dictionary based on total internal marks in ascending
order
    sorted_dict = sorted(details.items(), key=lambda x: x[1]['imark'],
reverse=True)

    # calculate rank for each student based on the sorted dictionary
    rank = 1
    for key, value in sorted_dict:
        details[key]['rank'] = rank
        rank += 1

    # return the sorted dictionary with updated ranks and internal marks
    return sorted_dict

details = {
  20201010: {
    'name': 'Khan', 'age': 18,
    'ass1': {
      'phy': 88,'chem': 99,'mat': 99, 'py': 99
    },
    'ass2': {
      'phy': 88,'chem': 99,'mat': 99,'py': 99
    },
    'imark': 0,'rank': 0
  },
  20201011: {
    'name': 'Sam','age': 18,
    'ass1': {
      'phy': 81,'chem': 79,'mat': 99,'py': 89
    },
    'ass2': {
      'phy': 80,'chem': 89,'mat': 79,'py': 79
    },
    'imark': 0,'rank': 0
  },
  20201012: {
    'name': 'Ram','age': 18,
    'ass1': {
      'phy': 68,'chem': 79,
      'mat': 89,'py': 99
    },
    'ass2': {
      'phy': 58,'chem': 69,'mat': 79,'py': 99
    },
    'imark': 0,'rank': 0
  }
}

sorted_details=stud_rank(details)
json_sorted_details = json.dumps(sorted_details, indent=4) #<---
Optional line if the output you decire should be in JSON format
print(json_sorted_details) #<--- To print in Json Format, Place
sorted_details with json_sorted_details
```

Possible Outputs:

From the Code:

```
[
    [
        20201010,
        {
            "name": "Khan",
            "age": 18,
            "ass1": {
                "phy": 88,
                "chem": 99,
                "mat": 99,
                "py": 99
            },
            "ass2": {
                "phy": 88,
                "chem": 99,
                "mat": 99,
                "py": 99
            },
            "imark": 770,
            "rank": 1
        }
    ],
    [
        20201011,
        {
            "name": "Sam",
            "age": 18,
            "ass1": {
                "phy": 81,
                "chem": 79,
                "mat": 99,
                "py": 89
            },
            "ass2": {
                "phy": 80,
                "chem": 89,
                "mat": 79,
                "py": 79
            },
            "imark": 675,
            "rank": 2
        }
    ],
    [
        20201012,
        {
            "name": "Ram",
            "age": 18,
            "ass1": {
                "phy": 68,
                "chem": 79,
                "mat": 89,
                "py": 99
            },
            "ass2": {
                "phy": 58,
                "chem": 69,
                "mat": 79,
                "py": 99
            },
            "imark": 640,
            "rank": 3
        }
    ]
]
```

Possible Test Inputs and Outputs:

Test Summary	Input	Output

A dictionary with only one student	details = {
	20201010: {
	'name': 'Khan', 'age': 18,
	'ass1': {
	'phy': 88,'chem': 99,'mat': 99, 'py': 99
		[(20201010, {'name': 'Khan', 'age': 18, 'ass1': {'phy': 88, 'chem': 99, 'mat': 99, 'py': 99}, 'ass2': {'phy': 88, 'chem': 99, 'mat': 99, 'py': 99}, 'imark': 792, 'rank': 1})]

	},
	'ass2': {
	'phy': 88,'chem': 99,'mat': 99,'py': 99
	},
	'imark': 0,'rank': 0
	}
	}	

A dictionary with two students	details = {
	20201010: {
	'name': 'Khan', 'age': 18,
	'ass1': {
	'phy': 88,'chem': 99,'mat': 99, 'py': 99
	},
	'ass2': {
	'phy': 88,'chem': 99,'mat': 99,'py': 99
	},
	'imark': 0,'rank': 0
	},
	20201011: {
	'name': 'Sam','age': 18,
	'ass1': {
	'phy': 81,'chem': 79,'mat': 99,'py': 89
	},
	'ass2': {
	'phy': 80,'chem': 89,'mat': 79,'py': 79
	},
	'imark': 0,'rank': 0
	}
	}
		[(20201010, {'name': 'Khan', 'age': 18, 'ass1': {'phy': 88, 'chem': 99, 'mat': 99, 'py': 99}, 'ass2': {'phy': 88, 'chem': 99, 'mat': 99, 'py': 99}, 'imark': 792, 'rank': 1}), (20201011, {'name': 'Sam', 'age': 18, 'ass1': {'phy': 81, 'chem': 79, 'mat': 99, 'py': 89}, 'ass2': {'phy': 80, 'chem': 89, 'mat': 79, 'py': 79}, 'imark': 328, 'rank': 2})]

A dictionary with three students, where two students have the same internal marks	details = {
	20201010: {
	'name': 'Khan', 'age': 18,
	'ass1': {
	'phy': 88,'chem': 99,'mat': 99, 'py': 99
	},
	'ass2': {
	'phy': 88,'chem': 99,'mat': 99,'py': 99
	},
	'imark': 0,'rank': 0
	},
	20201011: {
	'name': 'Sam','age': 18,
	'ass1': {
		[(20201010, {'name': 'Khan', 'age': 18, 'ass1': {'phy': 88, 'chem': 99, 'mat': 99, 'py': 99}, 'ass2': {'phy': 88, 'chem': 99, 'mat': 99, 'py': 99}, 'imark': 880, 'rank': 1}), (20201011, {'name': 'Sam', 'age': 18, 'ass1': {'phy': 81, 'chem': 79, 'mat': 99, 'py': 89}, 'ass2': {'phy': 80, 'chem': 89, 'mat': 79, 'py': 79}, 'imark': 797, 'rank': 2}), (20201012, {'name': 'Ram', 'age': 18, 'ass1': {'phy': 68, 'chem': 79, 'mat': 89, 'py': 99}, 'ass2': {'phy': 58, 'chem': 69, 'mat': 79, 'py': 99}, 'imark': 741, 'rank': 3})]

	'phy': 81,'chem': 79,'mat': 99,'py': 89
	},
	'ass2': {
	'phy': 80,'chem': 89,'mat': 79,'py': 79
	},
	'imark': 0,'rank': 0
	},
	20201012: {
	'name': 'Ram','age': 18,
	'ass1': {
	'phy': 68,'chem': 79, 'mat': 89,'py': 99
	},
	'ass2': {
	'phy': 58,'chem': 69,'mat': 79,'py': 99
	},
	'imark': 0,'rank': 0
	}
	}	

If the input is not a dictionary, the code will raise a TypeError.	details = '20201010, Khan, 18, 88, 99, 99, 99, 88, 99, 99, 99, 0, 0'
	
	sorted_details = stud_rank(details)
		TypeError: 'str' object is not callable

Question 3: COPRIME

Syntax used in this program:

1. Function definition: def coprimenumber(a, b):
2. Variable assignment: x = set(factors(a)); y = set(factors(b)); hcf = x.intersection(y)
3. Conditional statement: if hcf == {1}:
4. String formatting: print('{} and {} are coprime'.format(a, b)) print('{} and {} are not coprime'.format(a, b))
5. Function definition: def factors(number):
6. Looping statement: for j in range(2, number + 1):

Problem statement:

Write a function coprimenumber(a,b),when given a pair of numbers returns, if the pair of number are coprime or not. Co-Prime Numbers are a set of Numbers where the Common factor among them is 1. It implies that the HCF or the Highest Common Factor should be 1 for those Numbers.

Example: 14 and 15
Factors of 14 are 1, 2, 7 and 14.
Factors of 15 are 1, 3, 5 and 15.
The Common factor of 14 and 15 is only 1.
So, 14 and 15 are Co-Prime Numbers.

Solution:

```
# -*- coding: utf-8 -*-
"""
Created on Sat Mar 25 13:13:43 2023

@author: Haryish Elangumaran
"""

def coprimenumber(a, b):
  x = set(factors(a))
  y = set(factors(b))
  hcf = x.intersection(y)
  print("The common factor of {} and {} is {}".format(a,b,hcf))
  if hcf == {1}:
    print('{} and {} are coprime'.format(a, b))
  else:
    print('{} and {} are not coprime'.format(a, b))

def factors(number):
  factor = [1]
  for j in range(2, number + 1):
    if number % j == 0:
      factor.append(j)
  return factor
```

```
a,b=14,15
print("Inputs: {},{}".format(a,b))
print("Factor of {} is {} and {}".format(a,factors(a)[:len(factors(a))-
1],factors(a)[len(factors(a))-1]))
print("Factor of {} is {} and {}".format(b,factors(b)[:len(factors(b))-
1],factors(b)[len(factors(b))-1]))
coprimenumber(a,b)
```

Possible Outputs:

From code:

```
Inputs: 14,15
Factor of 14 is [1, 2, 7] and 14
Factor of 15 is [1, 3, 5] and 15
The common factor of 14 and 15 is {1}
14 and 15 are coprime
```

Possible test inputs and outputs:

Test name	Input	Output
1: Positive Test Case Expected Output: 7 and 10 are coprime	a=7, b=10	7 and 10 are coprime
2: Positive Test Case Expected Output: 15 and 16 are coprime	a=15, b=16	15 and 16 are coprime
3: Negative Test Case Expected Output: 18 and 24 are not coprime	a=18, b=24	18 and 24 are not coprime
4: Edge Case Expected Output: 1 and 1 are coprime	a=1, b=1	1 and 1 are coprime
5: Edge Case Expected Output: 0 and 5 are not coprime	a=0, b=5	0 and 5 are not coprime
6: Edge Case Expected Output: -7 and 10 are coprime	a=-7, b=10	-7 and 10 are coprime