กิจกรรมที่ 2 : Python

1. ให้เขียน function ชื่อ day_of_year(day, month ,year)
   โดยมีการคืนค่า คือ day_of_years เป็นวันที่ลำดับที่เท่าใดของปีคริสตศักราช year
   - ปีที่เป็น Leap Year เดือนกุมภาพันธ์จะมี 29 วัน
   - ให้สร้างฟังก์ชัน is_leap เพื่อตรวจสอบ leap year แยกออกมา และให้ฟังก์ชัน day_of_year เรียกใช้ is_leap อีกที
2. จากโปรแกรมในข้อ 7 ให้เขียนฟังก์ชัน เพิ่มเติมเป็น date_diff
   - รับข้อมูลในรูปแบบ “dd-mm-yyyy” เช่น

     date_diff(“1-1-2018”, “1-1-2020”) จะได้ 731 วัน
     date_diff(“25-12-1999”, “9-3-2000”) จะได้ 76 วัน
   - ให้เขียนฟังก์ชัน day_in_year โดยจะส่งค่าจำนวนวันของปี (365 หรือ 366) โดยรับข้อมูลเป็น ปี
   - ส่งคืนข้อมูลเป็นจำนวนวันตั้งแต่วันที่แรก จนถึงวันที่สอง โดยรวมทั้ง 2 วันนั้นเข้าไปด้วย
   - ให้สมมติว่าวันแรก จะต้องมาก่อนวันที่สองเสมอ ดังนั้นไม่ต้องตรวจสอบ
3. เขียนฟังก์ชัน add_score(subject_score, subject, score) โดยมีพารามิเตอร์ 3 ตัว ได้แก่ subject_score เป็น dictionary ที่มีคู่ key : value เป็น subject : score พารามิเตอร์ตัวที่ 2 และ 3 เป็น subject และ score โดย subject เป็น string และ score เป็น integer โดยให้นำ subject และ score ไปเพิ่มใน dictionary เช่น

   Input : subject_score = { }, subject = ‘python’, score = 50
   return : { ‘python’ : 50 }
   input : subject_score = { ‘python’ : 50 }, subject = ‘calculus’, score = 60
   return : { ‘python’ : 50, ‘calculus : 60 }

   จากนั้นให้เขียนฟังก์ชัน calc_average_score หาค่าเฉลี่ยของคะแนนในทุกรายวิชาใน dictionary ที่ได้จากฟังก์ชัน add_score โดยให้ส่งค่าคืนมาเป็น string ที่มีทศนิยม 2 ตำแหน่ง
4. ให้นำโปรแกรมตามข้อ 3 มาขยายความสามารถให้รองรับนักศึกษาหลายคน โดยให้ refactor ฟังก์ชัน add_score ให้รับพารามิเตอร์เป็น add_score(subject_score, student, subject, score) โดย student เป็นข้อมูลของนักศึกษาเป็น string (ในที่นี้เป็น id) และ return เป็น dictionary

   Input : subject_score = { }, student = '65010001', subject = 'python', score = 50

   return : { '65010001' : { 'python' : 50 } }

input : subject_score = { '65010001' : { 'python' : 50 } },

student = '65010001', subject = 'calculus', score = 60

return : {'65010001': {'python': 50, 'calculus', 60} }

โดยหากชื่อมีข้อมูล key ใดที่มีใน dictionary อยู่แล้ว ให้ถือเป็นการ update ข้อมูลนั้น

ให้ refactor ฟังก์ชัน calc_average_score โดยให้ส่งคืนเป็น dictionary ของนักศึกษาและคะแนนเฉลี่ยของนักศึกษาคนนั้น เช่น **{'65010001': '55.00' }**

5. ข้อมูลต่อไปนี้แทน music album แต่ละ album เก็บใน dictionary ซึ่งมีตัวเลข id เป็น key โดยแต่ละ album ไม่จำเป็นต้องมีข้อมูลครบ

```
record_collection = {
  2548: {
    albumTitle: 'Slippery When Wet',
    artist: 'Bon Jovi',
    tracks: ['Let It Rock', 'You Give Love a Bad Name']
  },
  2468: {
    albumTitle: '1999',
    artist: 'Prince',
    tracks: ['1999', 'Little Red Corvette']
  },
  1245: {
    artist: 'Robert Palmer',
    tracks: []
  },
  5439: {
    albumTitle: 'ABBA Gold'
  }
```

ให้เขียนฟังก์ชัน update_records โดยรับพารามิเตอร์ 4 ตัว คือ 1) dictionary record 2) id 3) property (เช่น artist หรือ tracks) 4) value โดยหน้าที่ของฟังก์ชัน คือ ให้เพิ่ม/เปลี่ยน ค่า property และ value ของ album ของ id ที่ส่งค่าไปในฟังก์ชัน โดยมีรายละเอียดดังนี้

- ฟังก์ชันจะต้องส่งคืนข้อมูล record ทั้งหมดกลับมา
- ถ้า property ไม่ใช่ tracks และ value ไม่ใช่ empty string ให้ update หรือ set ข้อมูล property กับ value ใน album นั้น
- ถ้า property เป็น tracks แต่ album นั้นไม่มี tracks property ให้สร้าง List ใหม่และเพิ่มข้อมูลเข้าไปใน List นั้น
- ถ้า property เป็น tracks และ value ไม่ใช่ empty string ให้เพิ่ม value ต่อท้ายใน List ของ tracks
- ถ้า value เป็น empty string ให้ลบข้อมูล property นั้นออกจาก album

```
record_collection = {
  2548: {
    'albumTitle': 'Slippery When Wet',
    'artist': 'Bon Jovi',
    'tracks': ['Let It Rock', 'You Give Love a Bad Name']
  },
  2468: {
    'albumTitle': '1999',
    'artist': 'Prince',
    'tracks': ['1999', 'Little Red Corvette']
  },
  1245: {
    'artist': 'Robert Palmer',
    'tracks': []
  },
  5439: {
    'albumTitle': 'ABBA Gold'
  }
}
```