กิจกรรมที่ 3 : Python

1. 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

2. ให้นำโปรแกรมตามข้อ 1 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' }**

3. ให้เขียนฟังก์ชัน is_plusone_dictionary(d) โดยรับพารามิเตอร์ 1 ตัว เป็นข้อมูลชนิด dictionary และให้ทดสอบว่า dictionary ที่รับเข้ามาเป็น plusone dictionary หรือไม่ ผลลัพธ์ให้ return เป็น True หรือ False โดย plusone dictionary คือ dictionary ที่ key และ value จะบวก 1 ต่อกันไปเรื่อยๆ

Input : {1:2, 3:4, 5:6, 7:8}

return : True

อธิบาย : เพราะ key : value ต่างกันเป็น +1 ต่อกันไป

Input : {1:2, 3:10, 5:6, 7:8}

return : False

อธิบาย : เพราะ key, value ไม่เป็นไปตามลำดับ

4. เขียนฟังก์ชัน char_count(str) โดยรับพารามิเตอร์ 1 ตัว เป็นข้อมูลชนิด string และให้ส่งคืนเป็น dictionary ที่มี key เป็นตัวอักษรแต่ละตัวของ string นั้น และ value คือ จำนวนครั้งที่ตัวอักษรนั้นปรากฏใน string เช่น

   Input : 'language'
   return : {'l': 1, 'a': 2, 'n': 1, 'g': 2, 'e': 1}

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