Programming Fundamentals I Lab.

4. สตริง และทูเปิล

ชื่อ_____ รหัสนิสิต_____

ในปฏิบัติการนี้ คุณจะได้รู้จักการเข้าถึงรายละเอียดต่าง ๆ ของสตริง การประมวลผลข้อความเบื้องต้น และ ทูเปิลที่เป็นชนิดข้อมูลที่ซับซ้อนมากขึ้น

4.1 ลำดับในสตริง

สร้างไฟล์ `.py` ใหม่ และพิมพ์โค้ดโปรแกรมต่อไปนี้

```
word = 'Hi Bell'
```

หลังจากนั้นให้เซฟและทดลองรันโปรแกรม ทดลองเรียกคำสั่งต่อไปนี้ใน Python Shell แล้วบันทึกผลที่เกิดขึ้น

`word[0]`	
`word[1]`	
`word[2]`	
`word[3]`	
`word[4]`	
`word[5]`	
`word[6]`	
`word[7]`	

จากการทดลอง ตัวอักษรตัวแรกในสตริง ถูกนับเป็นลำดับที่เท่าไหร่ในภาษา Python

ให้คุณลองใช้ฟังก์ชัน `type()` หาว่าผลจากการเรียก `word[i]` ที่ค่า `i` ต่าง ๆ นั้น ได้ผลลัพท์เป็นชนิดข้อมูลประเภทใด เขียนคำตอบในช่องต่อไปนี้

ทดลองเรียกคำสั่งต่อไปนี้ใน Python Shell แล้วบันทึกผลที่เกิดขึ้น

`word[-1]`	
`word[-2]`	
`word[-3]`	
`word[-4]`	
`word[-5]`	
`word[-6]`	
`word[-7]`	
`word[-8]`	

จากการทดลอง การเข้าถึงตัวอักษรในสตริงด้วยเลขลำดับที่เป็นจำนวนเต็มลบ มีความหมายอย่างไร

ในการคำนวณเกี่ยวกับสตริง มีฟังก์ชันหนึ่งที่มีประโยชน์มาก คือ ฟังก์ชัน `len()` ให้คุณทดลองเรียกใช้ฟังก์ชัน `len()` โดยใส่พารามิเตอร์เป็นสตริงหลาย ๆ แบบ สังเกตและบันทึกว่าฟังก์ชัน `len()` มีหน้าที่การทำงานอย่างไร

นอกจากฟังก์ชัน `len()` ยังมีค่าคงที่อีกค่าหนึ่งที่ใช้มากในการคำนวณเกี่ยวกับสตริง เรียกว่า *สตริงว่าง* ให้คุณทดลองสร้างสตริงว่างใส่ตัวแปรชื่อ `empty_string` ด้วยคำสั่ง `empty_string = ""` ทดลองใช้คำสั่งหรือฟังก์ชันต่าง ๆ เพื่อตอบคำถามต่อไปนี้

สตริงว่างมีความยาวเท่าไหร่	
สตริงว่างมีชนิดข้อมูลเป็นอะไร	
สตริงว่างมีค่าเท่ากับ `None` หรือไม่	
สตริงว่างมีค่าเท่ากับสตริงที่มีช่องว่างหนึ่งช่อง (`" "`) หรือไม่	

ทดลองสร้างฟังก์ชันในไฟล์ `.py` เป็นดังนี้

```
def reverse1(s):
    result = ''
    l = len(s)
    i = 0
    while i < l:
        result += s[l-i-1]
        print('i =', i, 'result =',result)
        i += 1
    return result
```

ทดลองเรียกฟังก์ชันด้วยคำสั่ง `reverse1('Hi Bell')` บันทึกผลที่ได้

ทดลองเรียกฟังก์ชัน `reverse1()` ด้วยสตริงต่าง ๆ ทำความเข้าใจ และบันทึกว่าฟังก์ชันนี้คืนค่าเป็นอะไร

สร้างฟังก์ชัน `reverse2()` ให้มีโค้ดดังนี้

```
def reverse2(s):
    result = ''
    l = len(s)
    i = 0
    while i < l:
        result = s[ <1> ] + result
        print('i =', i, 'result =',result)
        i += 1
    return result
```

จงหาว่าเราต้องแทนที่ <1> ด้วยอะไร จึงทำให้ฟังก์ชัน `reverse2()` คืนค่าเหมือนกับฟังก์ชัน `reverse1()` เสมอ (ไม่จำเป็นต้องให้ข้อมูลที่ print ออกจากฟังก์ชันเหมือนกัน แต่ขอให้ค่าที่ return ออกจากฟังก์ชันเหมือนกันเสมอ)

ทดลองสร้างตัวแปร `word = 'Hi Bell'` และลองเรียกคำสั่งต่อไปนี้ บันทึกผลที่ได้

`word[1:4]`	
`word[2:4]`	
`word[:4]`	
`word[3:]`	
`word[1:-3]`	
`word[-6:-3]`	
`word[:-2]`	
`word[-2:]`	
`word[:]`	
`word[2:100]`	
`word[80:100]`	
`word[100:]`	
`word[-1:0]`	

จากการทดลอง จงสรุปและอธิบายว่า ผลที่ได้จากการเรียก `word[a:b]` คืออะไร

ถ้าส่วนของสตริงในช่วง `[a:b]` ที่ต้องการไม่สามารถหาได้ จะได้ค่าอะไรคืนกลับมา

สร้างฟังก์ชัน `reverse3()` ให้มีโค้ดดังนี้

```
def reverse3(s):
    if s == '':
        return s
    else:
        return s[ <1> ] + reverse3(s[: <2> ])
```

จงหาว่าเราต้องแทนที่ <1> และ <2> ด้วยอะไร จึงทำให้ฟังก์ชัน `reverse3()` คืนค่าเหมือนกับฟังก์ชัน `reverse1()` เสมอ

<1>	
<2>	

สร้างฟังก์ชัน `reverse4()` ให้มีโค้ดดังนี้

```
def reverse4(s):
    if s == '':
        return s
    else:
        return reverse4(s[ <1> :]) + s[ <2> ]
```

จงหาว่าเราต้องแทนที่ <1> และ <2> ด้วยอะไร จึงทำให้ฟังก์ชัน `reverse4()` คืนค่าเหมือนกับฟังก์ชัน `reverse1()` เสมอ

<1>	
<2>	

4.2 คำสั่ง in และคำสั่งวนรอบ for

ทดลองสร้างตัวแปร `word = 'Hi Bell'` และลองเรียกคำสั่งต่อไปนี้ บันทึกผลที่ได้

`'h' in word`	
`'H' in word`	
`'l' in word`	
`' ' in word`	
`'' in word`	
`'Bell' in word`	
`'HiBell' in word`	
`'Hi Bell' in word`	
`word in word`	
`10 in word`	

จากการทดลอง คำสั่ง `in` มีไว้ใช้อย่างไร

ทดลองแก้ไขโค้ดในไฟล์ `.py` เป็นดังนี้

```
word = 'Hi Bell'
for c in word:
    print("writing: " + c)
```

เซฟและรัน บันทึกผลที่ได้

ลองเปลี่ยนค่าของตัวแปร `word` เป็นสตริงต่าง ๆ สังเกตและสรุปการทำงานของคำสั่ง `for ... in ...:` เมื่อใช้กับสตริง

สร้างฟังก์ชันดังนี้

```
def reverse5(s):
    result = ''
    for c in s:
        <1>
    return result
```

จงเติมคำสั่งในตำแหน่ง <1> เพื่อให้ฟังก์ชันนี้คืนค่าเหมือนกับฟังก์ชัน `reverse1()` เสมอ

4.3 คำสั่งเฉพาะของสตริง

ให้คุณสร้างตัวแปร `word = "you look, but you don't see. ok."` จากนั้นทดลองเรียกคำสั่งต่อไปนี้ บันทึกผลที่ได้

`word.find('u')`	
`word.find('d')`	
`word.find('a')`	
`word.find('')`	
`word.find('you')`	
`word.find('you', 1)`	
`word.find('ok')`	
`word.find('ok', 1)`	
`word.find('ok', 7)`	
`word.find('ok', 14)`	
`word.find('ok', 30)`	
`word.find('ok', 50)`	

จากการทดลอง หากเรามีสตริง `s` ฟังก์ชัน `s.find()` ทำหน้าที่อะไร

สังเกตว่าฟังก์ชัน `find()` ของสตริงนั้น มี default parameter ที่เป็นจำนวนเต็มอยู่ตัวหนึ่ง default parameter ตัวนี้ใช้ทำอะไร

คุณคิดว่าถ้าเราเรียก `find()` โดยไม่ใส่จำนวนเต็มตามเข้าไป default parameter ตัวนั้นจะมีค่าเป็นเท่าไหร่

ในการเรียก `s.find(w)` ถ้าในสตริง `s` ไม่มีสตริงย่อย `w` อยู่เลย จะได้ค่าอะไรคืนกลับมา

ให้คุณสร้างตัวแปร `s = "And we're counting 1, 2, 3."` จากนั้นทดลองเรียกคำสั่งต่อไปนี้ บันทึกผลที่ได้

`s.lower()`	
`s.upper()`	

คุณคิดว่าฟังก์ชัน `lower()` มีหน้าที่อย่างไร

คุณคิดว่าฟังก์ชัน `upper()` มีหน้าที่อย่างไร

4.4 ทูเปิล

หากเราต้องการข้อมูลหลาย ๆ อย่างรวมกันอยู่ในตัวแปรเดียวในภาษา Python เราสามารถใช้ชนิดข้อมูลทูเปิล (tuple) ได้

ทดลองสร้างตัวแปร `point = (2, 4)` และเรียกใช้คำสั่งต่อไปนี้ บันทึกผลที่ได้

`type(point)`	
`len(point)`	
`point[0]`	
`point[1]`	
`point[2]`	
`point[-1]`	
`type(point[0])`	

ทดลองสร้างตัวแปร `info = (1, 'Anna', (26, 4, 1992))` และเรียกใช้คำสั่งต่อไปนี้ บันทึกผลที่ได้

`type(info)`	
`len(info)`	
`info[0]`	
`info[1]`	
`info[2]`	
`type(info[-1])`	
`info[-1][0]`	
`info[:-1]`	

จากการทดลอง สมาชิกในทูเปิลจำเป็นต้องมีชนิดข้อมูลเหมือนกันหมดหรือไม่

ทดลองเขียนโค้ดต่อไปนี้ในไฟล์ `.py`

```
info = (1,  'Anna', (26, 4, 1992))
num, name, birth_date = info
print('num =', num)
print('name =', name)
print('birth_date =', birth_date)
```

เซฟและรัน บันทึกผลที่ได้

จากนั้นลองพิมพ์คำสั่ง `x = num, birth_date` ใน Python Shell แล้วทำการทดลองเพื่อตอบคำถามต่อไปนี้

`x` มีชนิดข้อมูลเป็นอะไร	
`x` มีสมาชิกทั้งหมดกี่ตัว	
ถ้าสั่ง `a, b = x` จะได้ตัวแปร `a` มีค่าเป็นอะไร	
ถ้าสั่ง `a, b = b, a` ต่อจะได้ตัวแปร `a` มีค่าเป็นอะไร	

ทดลองสร้างตัวแปร `x = (2)` จะได้ตัวแปร `x` มีชนิดข้อมูลเป็นอะไร

ถ้าสั่งด้วยคำสั่ง `x = (2,)` จะได้ตัวแปร `x` มีชนิดข้อมูลเป็นอะไร และมีจำนวนสมาชิกเป็นเท่าไหร่

ถ้าสั่งด้วยคำสั่ง `x = ()` จะได้ตัวแปร `x` มีชนิดข้อมูลเป็นอะไร และมีจำนวนสมาชิกเป็นเท่าไหร่

4.5 โจทย์ปัญหา

1. จงสร้างฟังก์ชันชื่อ `remove_spaces()` ที่รับสตริงแล้วคืนค่าเป็นสตริงตัวเดิมที่ทำการลบช่องว่างออกไป ตัวอย่างการใช้งานเช่น

```
>>> remove_spaces('Hi Bell, how are you?')
'HiBell,howareyou?'
```

2. จงสร้างฟังก์ชันชื่อ `binary_string()` ที่รับจำนวนเต็มไม่ติดลบ และคืนค่าเป็นสตริงที่แสดงค่าของจำนวนเต็มนั้นในรูปแบบตัวเลขฐาน 2 ตัวอย่างเช่น

```
>>> binary_string(489)
'111101001'
>>> binary_string(0)
'0'
```

3. จงสร้างฟังก์ชันชื่อ `use_all_aeiou()` ที่รับสตริงหนึ่งตัว และคืนค่าเป็น `True` ถ้าสตริงนั้นมีสระ a, e, i, o, u ครบทุกตัว (สมมติให้สตริงที่รับมาใช้ตัวอักษรภาษาอังกฤษตัวเล็กทั้งหมด) ตัวอย่างการทำงานเช่น

```
>>> use_all_aeiou('Hi Bell')
False
>>> use_all_aeiou('Hi Bell, how are you?')
True
```

4. จงสร้างฟังก์ชันชื่อ `count_word()` ที่รับสตริงสองตัวได้แก่ `w` และ `s` และคืนจำนวนครั้งที่ `w` ปรากฏใน `s` ตัวอย่างเช่น

```
>>> count_word('you', 'Hi Bell. Nice to meet you. How are you?')
2
>>> count_word('yyy', 'I am very happyyyyy.')
3
>>> count_word('you', 'I am very happyyyyy.')
0
```

5. จงสร้างฟังก์ชันชื่อ `older_name()` ที่รับทูเปิลสองตัว โดยแต่ละตัวอยู่ในรูป `(name, (d, m, y))` โดย `name` เป็นชื่อ มีชนิดเป็นสตริง และ `(d, m, y)` เป็นทูเปิลของจำนวนเต็มสามตัว แทนวันเดือนปีเกิดของเจ้าของชื่อ ให้ฟังก์ชัน `older_name()` คืนค่าเป็นชื่อของคนที่มีอายุมากกว่า ถ้าทั้งสองคนมีวันเกิดตรงกัน ให้คืนค่าเป็นชื่อของคนที่รับมาเป็นลำดับแรก ตัวอย่างเช่น

```
>>> older_name(('Betty', (4, 11, 1977)), ('Frank', (22, 10, 1990)))
'Betty'
```

```
>>> older_name(('Frank', (22, 10, 1990)), ('Carl', (22, 10, 1990)))
'Frank'
```