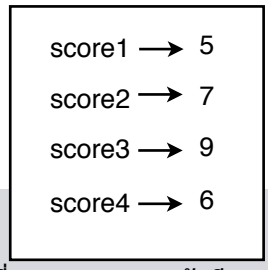
ชื่อ_____รหัสนิสิต_____หมู่_____

Lab 5 - รายการ (list) และ คำสั่ง for

1. รายการ (list)



ภาพที่ 1.1 : คะแนนของนักเรียน 4 คน

คะแนนของนักเรียน 4 คนประกอบด้วย นาย ก. นาย ข. นาย ค. และนาย ง. ซึ่งมีคะแนนเป็น 5 7 9 และ 6 ตามลำดับ อาจเก็บข้อมูลได้ดังนี้

```
score1 = 5
score2 = 7
score3 = 9
score4 = 6
```

จากโปรแกรมดังกล่าวจะได้ตัวแปร 4 ตัวเก็บข้อมูลคะแนนนักเรียน 4 คนดังภาพที่ 1.1

จะเห็นว่าการประกาศตัวแปร score สำหรับเก็บคะแนน ของนักเรียน 4 คนเป็นสิ่งที่ทำได้โดยง่าย แต่หากเพิ่มจำนวนนักเรียนจาก 4 คนเป็น 1,000 คน การใช้วิธีการแบบเดิมคงเกิดความไม่สะดวกอย่างยิ่ง ดังนั้นเราจึงควรนำรายการ (list) เข้ามาช่วย โดยรายการ (list) เป็นการรวบรวมข้อมูลหลาย ๆ ข้อมูลไว้ด้วยกัน ดังตัวอย่างต่อไปนี้

ข้อมูล	รายการ
รายการของวันในหนึ่งสัปดาห์	['Sunday','Monday','Tuesday','Wednesday',Thursday','Friday','Saturday']
รายการของจำนวนวันในแต่ละเดือน	[31,28,31,30,31,30,31,31,30,31,30,31]

รายการสามารถเขียนได้โดยใส่ข้อมูลแต่ละตัวในเครื่องหมายวงเล็บเหลี่ยม [ ] ขั้นข้อมูลแต่ละตัวด้วยเครื่องหมายลูกน้ำ (,) เช่นรายการของคะแนนของนาย ก. นาย ข. นาย ค. และ นาย ง. ข้างต้นสามารถเขียนได้ดังนี้

```
[5, 7, 9, 6]
```

ทั้งนี้ข้อมูลในรายการไม่จำเป็นต้องเป็นชนิดเดียวกันเสมอไปอาทิเช่นรายการของข้อมูลนักเรียน ประกอบไปด้วยชื่อ Jack Johnson รหัสนิสิต 999 สาขา CPE เกรดเฉลี่ยสะสม 3.87 อาจเขียนเป็นรายการได้ดังนี้

```
["Jack Johnson", 999, "CPE", 3.87]
```

นอกจากนี้รายการอาจไม่มีข้อมูลเลยก็ได้ (รายการว่าง) โดยเขียนได้ดังนี้

```
[ ]
```

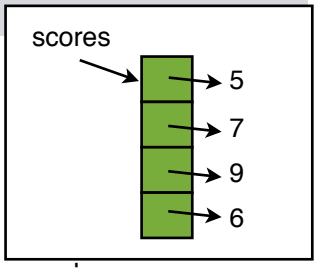
แบบฝึกหัดที่ 1.1 : จงเขียนรายการที่สอดคล้องกับข้อมูลต่อไปนี้

ข้อที่	ข้อมูล	รายการที่เขียนได้
1	ชื่อประเทศที่เป็นเจ้าภาพในการจัดการแข่งขันฟุตบอลโลก 4 ครั้งแรกตามลำดับ	
2	ขนาดของผลรวมของมุมภายใน (องศา) ของรูปสามเหลี่ยม สี่เหลี่ยม และห้าเหลี่ยมตามลำดับ	

เราสามารถตั้งชื่อให้รายการได้ (นำตัวแปรไปอ้างถึงรายการ) เพื่อให้ง่ายต่อการนำรายการนั้นไปใช้งานต่อไป เช่น หากต้องการให้รายการของคะแนนของ นาย ก. นาย ข. นาย ค. และ นาย ง. ข้างต้นมีชื่อรายการว่า scores ก็ทำได้ดังนี้

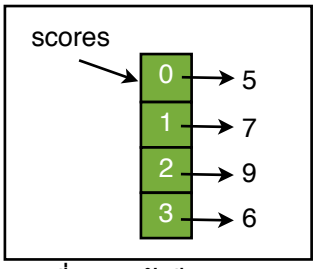
```
scores = [5, 7, 9, 6]
```

ลักษณะของรายการ score ที่ได้เป็นดังภาพที่ 1.2



ภาพที่ 1.2 : รายการ score

2. การอ้างถึงข้อมูลในรายการ



ภาพที่ 2.1 : ดัชนีของรายการ

ในการเขียนโปรแกรมหลายครั้งเราจำเป็นต้องใช้ข้อมูลในรายการเช่นจากภาพที่ 1.2 เราอาจต้องการนำคะแนนของนาย ค. (ซึ่งมีค่าเท่ากับ 9) ไปตัดเกรด หรือนำคะแนนของนาย ง. (ซึ่งมีค่าเท่ากับ 6) ไปใช้หาค่าเฉลี่ย เราจึงจำเป็นต้องรู้วิธีการอ้างถึงข้อมูลในรายการ

เราสามารถอ้างถึงข้อมูลในรายการใด ๆ ได้โดยใช้ดัชนีของแต่ละข้อมูลในรายการ โดยดัชนีแรกของรายการนั้นจะเริ่มที่ศูนย์เสมอดังภาพที่ 2.1 ดังนั้นหากต้องการอ้างถึงข้อมูลคะแนนของนาย ค. ก็ใช้ชื่อรายการและดัชนีประกอบกันเป็น scores[2] เช่นเดียวกันกับข้อมูลคะแนนของนาย ง. ใช้ scores[3] ในการอ้างถึง

ตัวอย่างที่ 2.1 : การใช้งานข้อมูลต่าง ๆ ในรายการ

จากรายการ scores = [5, 7, 9, 6] ในภาพที่ 1.2 และ 2.1 เราสามารถอ้างถึงข้อมูลต่าง ๆ ในรายการได้อาทิเช่น

การทำงาน	นิพจน์ที่ใช้
แสดงข้อมูลคะแนนของนาย ก. (ข้อมูลตัวแรกของรายการ scores)	print(score[0])
เปลี่ยนคะแนนของนาย ข. เป็น 8 คะแนน	scores[1] = 8
เพิ่มคะแนนให้นาย ง. อีก 2 คะแนน	scores[3] = scores[3] + 2
หาผลรวมของคะแนนทุกคนเก็บใส่ตัวแปร total	total = scores[0] + scores[1] + scores[2] + scores[3]

แบบฝึกหัดที่ 2.1 : อ้างข้อมูลในรายการ

กำหนดให้รายการ a เป็นรายการที่มีข้อมูลเป็นจำนวนเต็ม 5 จำนวน ให้เขียนนิพจน์ที่อ้างถึงข้อมูลในรายการนี้ตามที่ระบุ

ข้อที่	การทำงาน	นิพจน์ที่ใช้
1	ให้ตัวแปร w มีค่าเท่ากับข้อมูลตัวแรกในรายการ a	
2	ให้ตัวแปร x มีค่าเท่ากับข้อมูลตัวสุดท้ายในรายการ	
3	ให้ตัวแปร y มีค่าเท่ากับผลบวกของข้อมูลตัวแรกและตัวที่สอง	
4	ให้ตัวแปร z มีค่าเท่ากับผลคูณของข้อมูลทุกตัวในรายการ	

3. รายการกับคำสั่ง while

เราสามารถใช้คำสั่ง while ประกอบกับรายการได้ปกติอาทิเช่น หากต้องการแสดงข้อมูลในรายการ scores = [5, 7, 9, 6] สามารถทำได้โดยใช้คำสั่งดังตัวอย่างโปรแกรมที่ 3.1

หรือหากต้องการแสดงข้อมูลเฉพาะที่เป็นจำนวนเต็มคู่ สามารถทำได้โดยใช้คำสั่งดังตัวอย่างโปรแกรมที่ 3.2 หรือหากต้องการหาผลบวกของข้อมูลทั้งหมดทำได้โดยใช้คำสั่งดังตัวอย่างโปรแกรมที่ 3.3

ตัวอย่างโปรแกรมที่ 3.1	ตัวอย่างโปรแกรมที่ 3.2	ตัวอย่างโปรแกรมที่ 3.3

`i = 0`
`while (i<4) :`
`  print(score[i])`
`  i = i + 1`
	`i = 0`
	`while (i<4) :`
	`  if (score[i]%2 == 0):`
	`  print(score[i])`
	`  i = i + 1`
		`total = 0`
		`i = 0`
		`while (i<4) :`
		`  total = total + score[i]`
		`  i = i + 1`

แบบฝึกหัดที่ 3.1 : ล้างข้อมูล

กำหนดเติมโปรแกรมในช่องว่างเพื่อให้โปรแกรมทำหน้าที่เปลี่ยนข้อมูลในรายการ data ทั้งหมดให้เป็นจำนวนที่ผู้ใช้ต้องการ

โปรแกรมที่กำหนดให้ (เติมโปรแกรมในช่องว่างให้สมบูรณ์)	ตัวอย่างการทำงาน

`data = [1, 2, 3, 4, 5]`
`n = int(input("Change data to: "))`
`print(data[0], data[1], data[2], data[3], data[4])`
`_____`
`while(_____):`
`  _____`
`  _____`
`print(data[0], data[1], data[2], data[3], data[4])`
	ตัวอย่างที่ 1
	`Change data to: 0`
	`1 2 3 4 5`
	`0 0 0 0 0`
	ตัวอย่างที่ 2
	`Change data to: 4`
	`1 2 3 4 5`
	`4 4 4 4 4`

แบบฝึกหัดที่ 3.2 : แสดงจำนวนเต็มบวกในรายการ

กำหนดให้รายการ data เป็นรายการที่มีข้อมูลทั้งหมด 100 ตัว (ทุกตัวเป็นเลขจำนวนเต็มแต่ไม่บอกว่ามีข้อมูลอะไรบ้าง) ให้เติมโปรแกรมในช่องว่างเพื่อให้โปรแกรมทำหน้าที่พิมพ์ข้อมูลเฉพาะที่เป็นจำนวนเต็มบวกทั้งหมดของรายการ data

โปรแกรมที่กำหนดให้ (เติมโปรแกรมในช่องว่างให้สมบูรณ์)

`_____`
`while _____:`
`  if _____:`
`_____`
`_____`

แบบฝึกหัดที่ 3.3 : พิมพ์ข้อมูลตัวเว้นตัว

กำหนดให้รายการ data เป็นรายการที่มีข้อมูลทั้งหมด 100 ตัวเช่นเดียวกันกับแบบฝึกหัดที่ 3.2 (ทุกตัวเป็นเลขจำนวนเต็มแต่ไม่บอกว่ามีข้อมูลอะไรบ้าง) ให้เติมโปรแกรมในช่องว่างเพื่อให้โปรแกรมทำหน้าที่พิมพ์ข้อมูลตัวเว้นตัวโดยเริ่มที่ตัวแรกของข้อมูล

โปรแกรมที่กำหนดให้ (เติมโปรแกรมในช่องว่างให้สมบูรณ์)

`_____`
`while _____:`
`  _____`
`  _____`

4. รายการกับคำสั่ง for

นอกจากคำสั่ง while ที่เราสามารถใช้ในการไล่พิจารณาข้อมูลทุกตัวแล้ว ภาษาไพธอนยังมีอีกคำสั่งหนึ่งที่สามารถใช้ในการไล่พิจารณาข้อมูลทุกตัวในรายการได้ นั่นก็คือคำสั่ง "**for**" นั่นเองซึ่งมีรูปแบบการใช้งานดังนี้

```
for ตัวแปร in รายการ :
        คำสั่งในความควบคุมของ for คำสั่งที่ 1
        คำสั่งในความควบคุมของ for คำสั่งที่ 2
                :       :       :
        คำสั่งในความควบคุมของ for คำสั่งที่ n
```

โดยหลักการทำงานของคำสั่ง for ก็คือรอบแรก "ตัวแปร" จะทำหน้าที่อ้างถึงข้อมูลใน "รายการ" ตามด้วยทำคำสั่งในความควบคุมของ for คำสั่งที่ 1 ไล่ไปเรื่อย ๆ จนถึงคำสั่งที่ n จากนั้น "ตัวแปร" ก็จะเปลี่ยนไปอ้างถึงข้อมูลในลำดับถัดไปของรายการและไล่ทำคำสั่งในความควบคุมของ for คำสั่งที่ 1 ไล่ไปเรื่อย ๆ จนถึงคำสั่งที่ n อีกครั้งเป็นเช่นนี้ไปเรื่อย ๆ จนกว่า "ตัวแปร" จะอ้างถึงข้อมูลในรายการครบทุกตัว

จากตัวอย่างโปรแกรมที่ 3.1, 3.2 และ 3.3 สามารถเปลี่ยนจากการใช้คำสั่ง while เป็นคำสั่ง for ได้ดังตัวอย่างโปรแกรมที่ 4.1, 4.2 และ 4.3 ตามลำดับ

ตัวอย่างโปรแกรมที่ 4.1	ตัวอย่างโปรแกรมที่ 4.2	ตัวอย่างโปรแกรมที่ 4.3

`for x in score :`
`  print(x)`
	`for x in score :`
	`  if (x % 2 == 0):`
	`  print(x)`
		`total = 0`
		`for x in score :`
		`  total = total + x`

แบบฝึกหัดที่ 4.1 : พิมพ์ค่าสัมบูรณ์ของข้อมูลในรายการ

กำหนดให้รายการ data = [-3, 6, 8, 0, -12, -9] ให้เติมคำในช่องว่างให้โปรแกรมแสดงค่าสัมบูรณ์ของข้อมูลแต่ละตัวของรายการ data

โปรแกรมที่กำหนดให้ (เติมโปรแกรมในช่องว่างให้สมบูรณ์)	ตัวอย่างการทำงาน

`import math`
`data = [-3, 6, 8, 0, -12, -9]`
`for _____:`
`  _____`
	3
	6
	8
	0
	12
	9

แบบฝึกหัดที่ 4.2 : นับจำนวนข้อมูลที่มีค่าระหว่าง A และ B

กำหนดให้รายการ data = [-3, 6, 8, 0, -12, -9] จงเขียนโปรแกรมโดยใช้คำสั่ง for เพื่อตรวจสอบว่าข้อมูลในรายการ data มีข้อมูลที่อยู่ระหว่าง A และ B ที่ถูกระบุโดยผู้ใช้อยู่กี่จำนวน

โปรแกรมที่กำหนดให้ (เติมโปรแกรมในช่องว่างให้สมบูรณ์)	ตัวอย่างการทำงาน

`data = [-3, 6, 8, 0, -12, -9]`	**ตัวอย่างที่ 1**
	`Input A: -2.4`
	`Input B: 6.8`
	`Numbers of data between -2.4 and 6.8 is 2`
	_____
	ตัวอย่างที่ 2
	`Input A: -20`
	`Input B: 20`
	`Numbers of data between -20.0 and 20.0 is 5`

5. การดำเนินการและฟังก์ชันที่เกี่ยวข้องกับรายการ

ตัวดำเนินการที่เกี่ยวข้องกับรายการมีดังนี้

ตัวดำเนินการ	การทำงาน	ตัวอย่าง	ผลลัพธ์ที่ได้จากตัวอย่าง
+ (บวก)	นำรายการสองรายการมารวมกัน (ต่อกัน)	["4", 5, 6.7] + [0, "M"]	["4", 5, 6.7, 0, "M"]
* (คูณ)	คัดลอกรายการเดิมอีกให้ครบตามจำนวนที่คูณ	[1, 3] * 4	[1, 3, 1, 3, 1, 3, 1, 3]

ฟังก์ชันที่เกี่ยวข้องกับรายการมีดังนี้

ฟังก์ชัน	การทำงาน	ตัวอย่าง	ผลลัพธ์ที่ได้จากตัวอย่าง
len	คืนจำนวนสมาชิกของรายการ	len(["4", 5, 6.7])	3
sum	คืนผลรวมทั้งหมดของรายการ	sum([5, 3, –6])	2
max	คืนค่าของสมาชิกที่มากที่สุดในรายการ	max([5, 3, –6])	5
min	คืนค่าของสมาชิกที่น้อยที่สุดในรายการ	min([5, 3, –6])	–6

<u>ฟังก์ชัน append</u>

นอกจากฟังก์ชันที่กล่าวมาข้างต้นแล้ว เรายังสามารถเพิ่มข้อมูลในรายการใด ๆ ได้ด้วย**ฟังก์ชัน append** ดังตัวอย่างโปรแกรมที่ 5.1

ตัวอย่างโปรแกรมที่ 5.1	ผลลัพธ์ที่ได้จากการรันโปรแกรม

`s = [4, 8, 9, 11]`
`print("Before append: s=",s)`
`# เรียกใช้ฟังก์ชัน append โดยเอาเลข 5 ไปต่อท้ายรายการ s`
`s.append(5)`
`# หลังเรียนใช้ฟังก์ชัน append(5)`
`# รายการ s จะมีข้อมูลเป็น [4, 8, 9, 11, 5]`
`print("After append: s=",s)`
	`Before append: s= [4, 8, 9, 11]`
	`After append: s= [4, 8, 9, 11, 5]`

แบบฝึกหัดที่ 5.1 : ฟังก์ชันอ่านรายการ

โปรแกรมด้านล่างมีการเรียกใช้ฟังก์ชัน read_list() เพื่ออ่านรายการของจำนวนเต็มที่ผู้ใช้ป้อน โดยสิ้นสุดการป้อนเมื่อป้อน -1 จากนั้นนำรายการดังกล่าวแสดงออกทางหน้าจออีกครั้งหนึ่ง

ให้นักเรียนเขียนฟังก์ชัน read_list() ข้างต้นโดยฟังก์ชัน read_list() จะคือรายการของจำนวนเต็มที่ผู้ใช้พิมพ์เข้ามา (ไม่รวม -1)

* หมายเหตุ เราจะนำฟังก์ชัน read_list() นี้ไปใช้อีกในแบบฝึกหัดข้อต่อ ๆ ไป

โปรแกรมที่กำหนดให้ (เติมโปรแกรมในช่องว่างให้สมบูรณ์)	ตัวอย่างการทำงาน

`# นิยามฟังก์ชัน read_list()`
`def read_list():`
`  numbers = []`
`  x = int(input())`
`  while _____:`
`  _____`
`  x = int(input())`
`  return _____`
 	
`# ส่วนของโปรแกรมหลัก`
`num = read_list()`
`for a in num:`
`  print(a)`
	<u>10</u>
	<u>20</u>
	<u>15</u>
	<u>3</u>
	<u>-1</u>
	10
	20
	15
	3

แบบฝึกหัดที่ 5.2 : ผลรวมกำลังสอง

เขียนโปรแกรมรับค่าจำนวนเต็ม จากนั้นคำนวณหาผลรวมของจำนวนเต็มในรายการยกกำลังสองยกตัวอย่างเช่น ถ้าจำนวนเต็มในรายการเป็น 10 20 15 และ 3 ผลรวมคือ $10^2 + 20^2 + 15^2 + 3^2 = 734$ ทั้งนี้ผู้ใช้จะป้อนข้อมูลทีละจำนวนและจะจบการป้อนโดยการป้อน -1

*หมายเหตุ แนะนำให้ใช้ฟังก์ชัน read_list() ที่เขียนในข้อที่แล้ว

โปรแกรมที่เขียนได้คือ	ตัวอย่างการทำงาน
	<u>10</u> <u>20</u> <u>15</u> <u>3</u> <u>-1</u> Answer = 734

แบบฝึกหัดที่ 5.3 : ฝากเงิน

Bella Swan ฝากเงินใส่กระปุกทุก ๆ เดือน โดยไม่ถอนออก ให้นักเรียนเขียนโปรแกรมรับรายการเงินที่ Bella ฝากโดย Bella จะป้อน -1 เมื่อข้อมูลหมด หลังจากรับข้อมูลการฝากเงินโปรแกรมจะพิมพ์เงินรวมทั้งหมด และเงินฝากสะสมในทุก ๆ สิ้นเดือนภายหลังจากที่ Bella ฝากไป

โปรแกรมที่เขียนได้คือ	ตัวอย่างการทำงาน
	10 20 100 15 -1 Total = 145 End Month1 = 10 End Month2 = 30 End Month3 = 130 End Month4 = 145

แบบฝึกหัดที่ 5.4 : ฝากเงินแบบมีดอกเบี้ยรายเดือน

Kaka ฝากเงินใส่ธนาคารทุก ๆ เดือน เงินที่ฝากเข้าไปจะสะสมไปเรื่อย ๆ ในแต่ละเดือนก่อนที่จะรับเงินฝาก ธนาคารจะให้ดอกเบี้ย 1% จากเงินฝากที่มีอยู่ ยกตัวอย่างเช่น ถ้าฝากเงินสองเดือนแรกเท่ากับ 10 และ 20 บาท ในเดือนแรกเนื่องจากยังไม่มีเงินต้น เงินรวมคือ 10 บาท ในเดือนที่สอง ก่อนจะได้เงินฝากจะได้รับดอกเบี้ย 1% คือ 0.1 บาทรวมเงินต้นเป็น 10.1 รวมกับเงินฝาก ได้เป็น 30.1 บาท

ให้เขียนโปรแกรมรับรายการเงินที่ Kaka ฝาก โดย Kaka จะป้อน -1 เมื่อข้อมูลหมด จากนั้นให้โปรแกรมพิมพ์เงินรวมทั้งหมด และเงินฝากรวมในทุก ๆ เดือนภายหลังจากที่ Kaka ฝากไป ให้แสดงด้วยทศนิยม 3 ตำแหน่ง

โปรแกรมที่เขียนได้คือ	ตัวอย่างการทำงาน
	10 20 100 15 -1 Total = 146.705 End Month1 = 10.000 End Month2 = 230.100 End Month3 = 130.401 End Month4 = 146.705

แบบฝึกหัดที่ 5.5 : เลขกำลังสอง ภาค 1

ให้เขียนฟังก์ชัน is_square(x) เพื่อตรวจสอบว่าจำนวนเต็ม x เป็นจำนวนเต็มที่เป็นกำลังสองของจำนวนเต็มบางจำนวนหรือไม่

* หมายเหตุ ฟังก์ชัน is_square(x) นี้จะนำไปใช้ในแบบฝึกหัดที่ 5.6 ต่อไป

โปรแกรมที่เขียนได้คือ	ตัวอย่างการเรียกใช้ใน Python Shell

import math
def is_square(number):
	>>> is_square(4)
	True
	>>> is_square(5)
	False
	>>> is_square(121)
	True
	>>> is_square(200)
	False
	>>> is_square(80)
	False

แบบฝึกหัดที่ 5.6 : เลขกำลังสอง ภาค 2

เขียนฟังก์ชัน filter_square(ls) ที่รับรายการ ls และคืนค่าเป็นรายการที่ประกอบด้วยจำนวนเต็มที่เป็นเลขกำลังสอง ทั้งนี้ให้นำฟังก์ชัน is_square ที่เขียนในแบบฝึกหัดที่ 5.5 มาใช้

โปรแกรมที่เขียนได้คือ	ตัวอย่างการเรียกใช้ใน Python Shell

`import math`
`def is_square(number):`
 	
`def filter_square(ls):`
	`>>> filter_square([1,2,3,4,5,6,7,8,9,10])`
	`[1,4,9]`
	`>>> filter_square([10,20,30])`
	`[]`
	`>>> filter_square([80,100,200,400,9,80])`
	`[100,400,9]`

ฟังก์ชัน range

ฟังก์ชัน range ก็เป็นอีกฟังก์ชันหนึ่งที่มีประโยชน์ในการเขียนโปรแกรมโดยมีการทำงานดังนี้

ฟังก์ชัน	การทำงาน	ตัวอย่าง	ผลลัพธ์ที่ได้จากตัวอย่าง
range(r)	คืนรายการที่มีสมาชิกเป็นจำนวนเต็มเริ่มตั้งแต่ 0 ถึง r - 1	range(4)	[0, 1, 2, 3]
range(a, b)	คืนรายการที่มีสมาชิกเป็นจำนวนเต็มเริ่มตั้งแต่ a ถึง b - 1	range(8, 15)	[8, 9, 10, 11, 12, 13, 14]

แบบฝึกหัดที่ 5.7 : ซิกม่า ภาค 1

จงเติมข้อความที่เหมาะสมในช่องว่างต่อไปนี้เพื่อให้โปรแกรมทำหน้าที่คำนวณหาค่าของ y ตามสูตรต่อไปนี้

$$y=\sum_{i=0}^{n} i = 0+1+2+\ldots+n$$

โปรแกรมที่เขียนได้คือ	ตัวอย่างการทำงาน

`n = int(input("Enter n: "))`
`total = 0`
`for i in _____:`
`  total += _____`
`print("Sumation of 0 to",n,"is",total)`
	ตัวอย่างที่ 1
	`Enter n: 5`
	`Sumation of 0 to 5 is 15`
	ตัวอย่างที่ 2
	`Enter n: 100`
	`Sumation of 0 to 100 is 5050`

แบบฝึกหัดที่ 5.8 : ซิกม่า ภาค 2

จงเติมข้อความที่เหมาะสมในช่องว่างต่อไปนี้เพื่อให้โปรแกรมทำหน้าที่คำนวณหาค่าของ y ตามสูตรต่อไปนี้

$$y=\sum_{i=a}^{b} \sqrt{i+2}$$

โปรแกรมที่เขียนได้คือ	ตัวอย่างการทำงาน

`import math`
`a = int(input("Enter a: "))`
`b = int(input("Enter b: "))`
`total = 0`
`for i in _____:`
`  total += _____`
`print("y = %.2f" %total)`
	ตัวอย่างที่ 1
	`Enter a: 0`
	`Enter b: 1`
	`y = 3.15`
	ตัวอย่างที่ 2
	`Enter a: 10`
	`Enter b: 300`
	`y = 3481.50`

แบบฝึกหัดที่ 5.9 : พิมพ์กลับหลัง

เขียนโปรแกรมรับจำนวนเต็ม จนกระทั่งผู้ใช้ป้อน -1 จากนั้นพิมพ์จำนวนเต็มที่รับ จากหน้าไปหลัง บรรทัดละ 1 ตัว

โปรแกรมที่เขียนได้คือ	ตัวอย่างการทำงาน
	10 20 15 3 -1 3 15 20 10

6. ฝึกเขียนโปรแกรมแสนสนุก - ร้านขายของ

ในโจทย์ชุดร้านขายของ เราจะพัฒนาโปรแกรมสำหรับเป็นหน้าร้านซื้อขายเครื่องเขียน เราจะเก็บข้อมูลของเครื่องเขียนในรายการโดยข้อมูลที่จะเก็บเป็นดังนี้ คือ ชื่อของ ราคา และจำนวนที่เหลือในร้าน ข้อมูลทั้งสามจะเก็บในรายการสามรายการ ตัวอย่างดังด้านล่างนี้

names = ['pen', 'pencil', 'ruler', 'cutter', 'ink']
prices = [5, 1, 10, 15, 20]
counts = [100, 200, 50, 30, 50]

จากข้อมูลข้างต้น ปากกา (pen) มีราคา 5 บาท มีจำนวนของเหลือ 100 ชิ้น

แบบฝึกหัดที่ 6.1 : ร้านขายของ - ค้นหาสินค้า

จากข้อมูลข้างต้นสังเกตได้ว่าข้อมูลด้านราคาและจำนวนสินค้าจะถูกเก็บในรายการในตำแหน่งที่สอดคล้องกับข้อมูลชื่อสินค้าดังนั้นเราจะประมวลผลข้อมูลดังกล่าวได้สะดวกถ้าเราทราบดัชนีของสินค้านั้น ด้วยเหตุนี้จึงให้นักเรียนเขียนฟังก์ชัน find_product(plist, name) ที่รับรายการของชื่อสินค้า plist และชื่อสินค้า name จากนั้นคืน ดัชนี (หรือตำแหน่ง) ของ name ในรายการ plist ในกรณีที่ไม่มีสินค้านั้นในรายการให้คืนค่า -1

ฟังก์ชันที่เขียนได้คือ	ตัวอย่างการเรียกใช้ใน Python Shell

`def find_product(plist, name):`	`>>> find_product(['pen','ink','pencil'],'pen')`
	`0`
	`>>> find_product(['pen','ink','pencil'],'pencil')`
	`2`
	`>>> find_product(['pen','ink','pencil'],'ruler')`
	`-1`

แบบฝึกหัดที่ 6.2 : ร้านขายของ - พิมพ์รายการสินค้าที่เหลือ

สำหรับข้อนี้ จงเขียนฟังก์ชัน show_stock(products, counts) ที่พิมพ์รายการสินค้าที่เหลือ โดยใช้ข้อมูลรายชื่อสินค้าในรายการ products และรายการจำนวนที่เหลือในรายการ counts ดังตัวอย่าง

ตัวอย่างโปรแกรมหลักที่เรียกใช้งานฟังก์ชันดังกล่าว	ตัวอย่างผลลัพธ์ของโปรแกรม

`products = ['pen', 'pencil', 'ruler', 'cutter', 'ink']`
`counts = [100, 200, 50, 30, 50]`
`show_stock(products, counts)`
	`pen 100`
	`pencil 200`
	`ruler 50`
	`cutter 30`
	`ink 50`

ฟังก์ชันที่เขียนได้คือ
`def show_stock(products, counts):`

แบบฝึกหัดที่ 6.3 : ร้านขายของ - ขายสินค้า

สำหรับข้อนี้ ให้เขียนฟังก์ชัน sale(products, prices, counts) ที่รับชื่อสินค้า และจำนวนที่ต้องการซื้อจากผู้ใช้ จากนั้นระบุราคาที่ต้องจ่าย พร้อมทั้งตัดจำนวนสินค้าที่เหลือ โดยฟังก์ชันดังกล่าวจะต้องตรวจสอบว่าสินค้าที่ซื้อมีหรือไม่ และจำนวนมากพอที่จะขายหรือไม่ด้วย ถ้าผู้ใช้ป้อนจำนวนซื้อน้อยกว่าศูนย์ ให้ระบุว่าเป็นจำนวนที่ผิดพลาดด้วย ตัวอย่างโปรแกรมหลักที่เรียกใช้งานฟังก์ชันดังกล่าวเป็นดังนี้

* หมายเหตุ: ในการเขียนฟังก์ชันนั้น ถ้าต้องการให้ฟังก์ชันจบการทำงานและคืนกลับไปทำงานที่โปรแกรมหลักเลย สามารถทำได้โดยใช้คำสั่ง return (สามารถสั่งโดยไม่ต้องตามด้วยค่าที่จะคืน)

ตัวอย่างโปรแกรมหลักที่เรียกใช้งานฟังก์ชันดังกล่าว	ฟังก์ชันที่เขียนได้คือ

`products = ['pen', 'pencil']`
`prices = [5, 1]`
`counts = [100, 200]`
`sale(products, prices, counts)`
`show_stock(products, counts)`
	`def sale(products, prices, counts):`

ตัวอย่างผลลัพธ์ของโปรแกรม

ตัวอย่างที่ 1	ตัวอย่างที่ 2

`Enter product name: pen`
`Enter amount: 3`
`You have to pay 15 baths.`
`pen 97`
`pencil 200`
# ***หมายเหตุ** บรรทัดผลลัพธ์สองบรรทัดสุดท้ายพิมพ์โดย show_stock ฟังก์ชันที่คุณต้องเขียนไม่ต้องพิมพ์ค่านี้ อย่างไรก็ตาม ให้สังเกตว่าฟังก์ชัน sale จะต้องปรับค่าในรายการ counts ด้วย 	
	`Enter product name: ruler`
	`Sorry, we do not have that product.`
	`pen 100`
	`pencil 200`

ตัวอย่างที่ 3	**ตัวอย่างที่ 4**

`Enter product name: pencil`
`Enter amount: 210`
`Sorry, we do not have enough.`
`pen 100`
`pencil 200`
	`Enter product name: pencil`
	`Enter amount: -20`
	`Sorry, that is an invalid amount.`
	`pen 100`
	`pencil 200`

แบบฝึกหัดที่ 6.4 : ร้านขายของ - รวมเป็นโปรแกรม

ในข้อนี้ ให้นำฟังก์ชันที่เขียนไว้จากข้อก่อน ๆ มารวมเป็นโปรแกรมร้านขายของ ที่มีการทำงานดังนี้

- เริ่มต้นโดยการแสดงรายการสินค้าที่มี
- จากนั้นถามผู้ใช้ว่าต้องการจบการทำงานหรือซื้อสินค้า
- ถ้าผู้ใช้ซื้อสินค้า ให้ทำงานในลักษณะเดียวกับโจทย์ข้างต้น เมื่อซื้อเสร็จให้พิมพ์รายการสินค้าที่เหลือ
- ถ้าผู้ใช้สั่งให้จบการทำงาน ให้พิมพ์รายการสินค้าที่เหลือและจบการทำงาน

ให้โปรแกรมเริ่มต้นด้วยสินค้าที่มีตามรายการ products, prices และ counts ตามที่กำหนดในตอนแรก (ก่อนทำแบบฝึกหัด 6.1) (ให้ประกาศตัวแปร products, prices, และ counts ตามด้านบนในส่วนของโปรแกรมหลัก)

ตัวอย่างผลลัพธ์ของโปรแแกรม

```
Current stock:
pen 100
pencil 200
ruler 50
cutter 30
ink 50
Buy or Exit (enter B or E): B
Enter product name: pen
Enter amount: 15
You have to pay 75 bath.
pen 85
pencil 200
ruler 50
cutter 30
ink 50
Buy or Exit (enter B or E): B
Enter product name: cutter
Enter amount: 50
Sorry, we do not have enough.
pen 85
pencil 200
ruler 50
cutter 30
ink 50
Buy or Exit (enter B or E): E
At the end:
pen 85
pencil 200
ruler 50
cutter 30
ink 50
```

โปรแกรมที่เขียนได้คือ

7. ฝึกเขียนโปรแกรมแสนสนุก - พหุนาม

แบบฝึกหัดที่ 7.1 : การอ่านพหุนาม

ในโจทย์หลายข้อถัดไป เราจะอ่านพหุนามจากผู้ใช้ จึงให้นักเรียนเขียนฟังก์ชัน read_poly() สำหรับอ่านเลขสัมประสิทธิ์ของพหุนามในแต่ละพจน์ โดยการอ่านจะมีรูปแบบดังนี้

- ผู้ใช้จะป้อนข้อมูลตัวแรกเป็นจำนวนเต็ม d แทนกำลังสูงจุดของตัวแปร x ในพหุนามดังกล่าว
- จากนั้นจะป้อนจำนวนจริงอีก d + 1 จำนวน แทนสัมประสิทธิ์ของตัวแปร x โดยเริ่มจาก a_0, a_1, ไปจนถึง a_d พหุนามที่ผู้ใช้ป้อนคือ

$$f(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \cdots + a_d x^d$$

ให้ฟังก์ชันดังกล่าวคืนค่าเป็นรายการของสัมประสิทธิ์ [a_0,a_1,a_2,...,a_d] ยกตัวอย่างเช่น ถ้าพหุนามเป็น $f(x) = 2x^3 - x^2 + 3x + 10$ ผู้ใช้ป้อนข้อมูลดังนี้

```
3
10
3
-1
2
```

ฟังก์ชันดังกล่าวจะคืนค่าเป็น [10,3,-1,2]

ตัวอย่างของโปรแกรมหลักที่เรียกใช้ฟังก์ชันดังกล่าว	ตัวอย่างผลลัพธ์ของโปรแกรม	ฟังก์ชัน read_poly() ที่เขียนได้คือ

`print("Enter polynomial:")`
`p = read_poly()`
`print("Coefficients:")`
`for c in p:`
`  print(c)`
	`Enter polynomial:`
	3
	10
	3
	-1
	2
	`Coeeficients:`
	`10`
	`3`
	`-1`
	`2`
		`def read_poly():`

แบบฝึกหัดที่ 7.2 : หาค่าพหุนาม

เขียนโปรแกรมรับพหุนาม f(x) จากผู้ใช้ จากนั้นรับค่า a แล้วคำนวณค่า f(a)

ข้อมูลป้อนเข้า

- ผู้ใช้จะป้อนข้อมูลตัวแรกเป็นจำนวนเต็ม d แทนกำลังสูงจุดของตัวแปร x ในพหุนามดังกล่าว
- จากนั้นจะป้อนจำนวนจริงอีก d + 1 จำนวน แทนสัมประสิทธิ์ของตัวแปร x โดยเริ่มจาก a_0, a_1, ไปจนถึง a_d พหุนามที่ผู้ใช้ป้อนคือ

$$f(x) = a_0 + a_1 \cdot x + a_2 \cdot x^2 + \cdots + a_d x^d$$

- สุดท้ายผู้ใช้จะป้อนจำนวนจริง a

ให้โปรแกรมตอบค่า f(a) เป็นทศนิยม 2 ตำแหน่ง ตัวอย่างเช่น ถ้าพหุนามเป็น $f(x) = 2x^3 - x^2 + 3x + 10$ และผู้ใช้ต้องการคำนวณค่า f(3)

โปรแกรมที่เขียนได้คือ	ตัวอย่างผลลัพธ์ของโปรแกรม
	3 10 3 -1 2 3 `64.00`

แบบฝึกหัดที่ 7.3 : บวกพหุนาม

เขียนโปรแกรมรับพหุนามสองพหุนาม จากนั้นหาผลบวก แล้วพิมพ์รายการของสัมประสิทธิ์ให้กับผู้ใช้

โปรแกรมที่เขียนได้คือ	ตัวอย่างผลลัพธ์ของโปรแกรม
	Enter first polynomial: 2 1 2 3 Enter second polynomial: 5 10 -1 0 2 3 4 Result is: 11 1 3 2 3 4

แบบฝึกหัดที่ 7.4 : หาอนุพันธ์ของพหุนาม

สูตรการหาอนุพันธ์ของฟังก์ชันพหุนามคือ $\frac{d}{dx}x^k = kx^{k-1}$

ให้เขียนโปรแกรมรับพหุนาม จากนั้นคำนวนหาอนุพันธ์และแสดงกับผู้ใช้

*หมายเหตุ: ตัวอย่างจากตัวอย่างต่อไปนี้แสดง $\frac{d}{dx}(1x^3 + 2x^2 + 3x + 4) = 3x^2 + 4x + 3$

โปรแกรมที่เขียนได้คือ	ตัวอย่างผลลัพธ์ของโปรแกรม
	Enter polynomial: 3 4 3 2 1 Result: 3 4 3

แบบฝึกหัดที่ 7.5 : หาผลคูณของพหุนาม (ข้อนี้ยากจัด ทำได้เจ๋งมาก)

เขียนโปรแกรมรับพหุนามสองพหุนาม จากนั้นคำนวณหาผลคูณของพหุนาม แล้วแสดงผล

*หมายเหตุ: ตัวอย่างจากตัวอย่างต่อไปนี้แสดง $(3x^2 + 2x + 1) \times (x + 2) = 3x^3 + 8x^2 + 5x + 2$

โปรแกรมที่เขียนได้คือ	ตัวอย่างผลลัพธ์ของโปรแกรม
	Enter first polynomial: 2 1 2 3 Enter second polynomial: 1 2 1 Result: 2 5 8 3