ลิสต์ (List)

Outlines:

1. Overview
2. Elements in List
3. Length of a List
4. Iterate over a List
5. Operators for Lists
6. List Methods
7. Making Copies of Lists
8. Exercise

Overview

ลิสต์ (List) เป็นโครงสร้างข้อมูลชนิดหนึ่ง

- ใช้เก็บข้อมูลแบบลำดับ (Sequence) โดยมี Index เป็นตัวระบุตำแหน่งในการเข้าถึงข้อมูล
- สามารถเก็บข้อมูลจำนวนมากในลิสต์เดียว
- สามารถเก็บข้อมูลหลากหลายชนิด (Data type) ในลิสต์เดียว

Overview

ลิสต์มีประโยชน์อย่างไร?

จงเขียนโปรแกรมที่เก็บคะแนนสอบของนักศึกษา 100 คน

ถ้าไม่มีลิสต์

จะต้องใช้ตัวแปรถึง 100 ตัวแปร ในการเก็บคะแนนสอบของนักศึกษาแต่ละคน เช่น score_1 score_2 ….. score_100

ถ้ามีลิสต์

ใช้เพียงลิสต์ 1 ลิสต์ เก็บคะแนนของนักศึกษาแต่ละคนลงในแต่ละตำแหน่งของลิสต์

ลิสต์ในภาษาไพทอน

- **ลิสต์** เป็นตัวแปรประเภทหนึ่งในภาษาไพทอน
- การประกาศลิสต์ ข้อมูล (สมาชิก) จะอยู่ภายในเครื่องหมาย [ ] และคั่นสมาชิกแต่ละตัวด้วยเครื่องหมายคอมมา ,

```
score = [98, 87, 82, 82, 99]
names = ['Mateo', 'Danny', 'James', 'Thomas', 'Luke']
mixed_type = [-2, 5, 84.2, "Mountain", "Python"]
```

เราสามารถสร้างลิสต์ว่าง (Empty List) ได้โดย

```
init_score = []
```

การเข้าถึงสมาชิกในลิสต์

- เราเข้าถึงสมาชิกแต่ละตัวในลิสต์ได้ โดยการระบุหมายเลขตำแหน่งในเครื่องหมาย [ ] ท้ายตัวแปรลิสต์

ชื่อตัวแปรลิสต์

หมายเลขตำแหน่งสมาชิก ต้องเป็นจำนวนเต็มที่มีค่ามากกว่าหรือเท่ากับ 0 และน้อยกว่าจำนวนสมาชิกของลิสต์

การเข้าถึงสมาชิกในลิสต์

- ตำแหน่งสมาชิกของตัวแปรลิสต์ในภาษาไพทอน
  - สมาชิกตัวแรก มีหมายเลขตำแหน่ง คือ 0 และเพิ่มขึ้นทีละหนึ่งลำดับเรื่อยๆ
  - ถ้านับจากท้ายลิสต์ สมาชิกตัวสุดท้าย มีหมายเลขตำแหน่ง คือ -1 และลดลงทีละหนึ่งลำดับเรื่อยๆ

	[0]	[1]	[2]	[3]	[4]
score	**98**	**87**	**82**	**82**	**99**
	[-5]	[-4]	[-3]	[-2]	[-1]

```
>> score = [98, 87, 82, 82, 99]
>> print(score[1])
81
>> print(score[-1])
99
```

Elements in List

การเข้าถึงสมาชิกในลิสต์

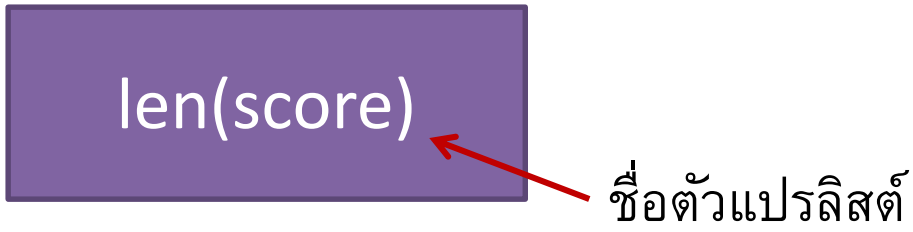
- เราแก้ไขข้อมูลสมาชิกแต่ละตัวในลิสต์ได้ โดยใช้ตัวดำเนินการ assignment (=)

```
>> score = [98, 87, 82, 82, 99]
>> score[1] = 90
>> print(score)
[98, 90, 82, 82, 99]
>> score[-1] =95
>> print(score)
[98, 90, 82, 82, 95]
```

Length of a List

จำนวนสมาชิกในลิสต์

- เราสามารถทราบจำนวนสมาชิกในตัวแปรได้โดยใช้ฟังก์ชัน **len**
- ฟังก์ชัน **len**(*ชื่อตัวแปรลิสต์*) มีพารามิเตอร์ คือ *ชื่อตัวแปรลิสต์*ที่ต้องการทราบจำนวนสมาชิก
- ฟังก์ชัน **len()** ส่งจำนวนสมาชิกในลิสต์กลับออกมา




```
>> score = [98, 87, 82, 82, 99]
>> print(len(score))
5
```

Iterate over a List

วนดำเนินการกับสมาชิกในลิสต์ ทำได้ 2 วิธีตามตัวอย่างต่อไปนี้

```
>> L = [0, 1, 2, 3, 4, 5]
>> for i in range(0,len(L)):
>>                 print(L[i])
0
1
2
3
4
5
```

เหมาะสำหรับเมื่อต้องการใช้ตัวแปรชี้ตำแหน่ง i ในดำเนินการต่อ

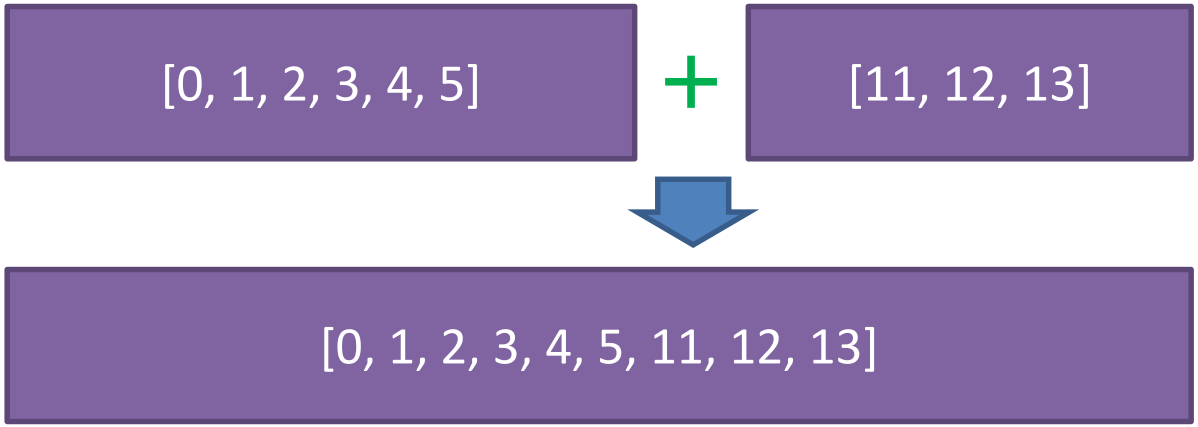
```
>> L = [0, 1, 2, 3, 4, 5]
>> for item in L:
>>                 print(item)
0
1
2
3
4
5
```

ถ้าไม่ต้องการติดตามตำแหน่งสมาชิกที่กำลังดำเนินการอยู่ ใช้วิธีนี้จะเหมาะสมกว่า

Operators for Lists

การเชื่อมลิสต์เข้าด้วยกัน

- ใช้ตัวดำเนินการ **+** ในการเชื่อมลิสต์ 2 ลิสต์เข้าด้วยกัน

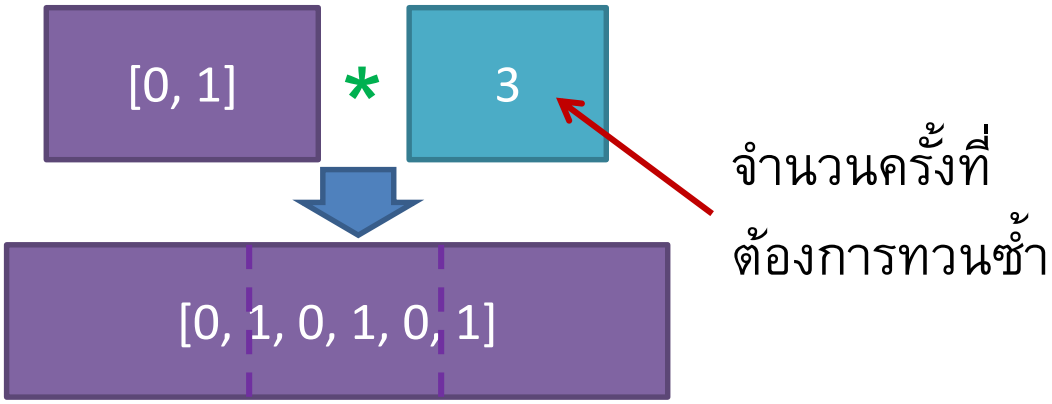



```
>> L = [0, 1, 2, 3, 4, 5]
>> M = [11, 12, 13]
>> A = L+M
>> print(A)
[0, 1, 2, 3, 4, 5, 11, 12, 13]
```

Operators for Lists

การทวนซ้ำสมาชิกในลิสต์

- ใช้ตัวดำเนินการ * ในการทำซ้ำสมาชิกทั้งหมดในลิสต์

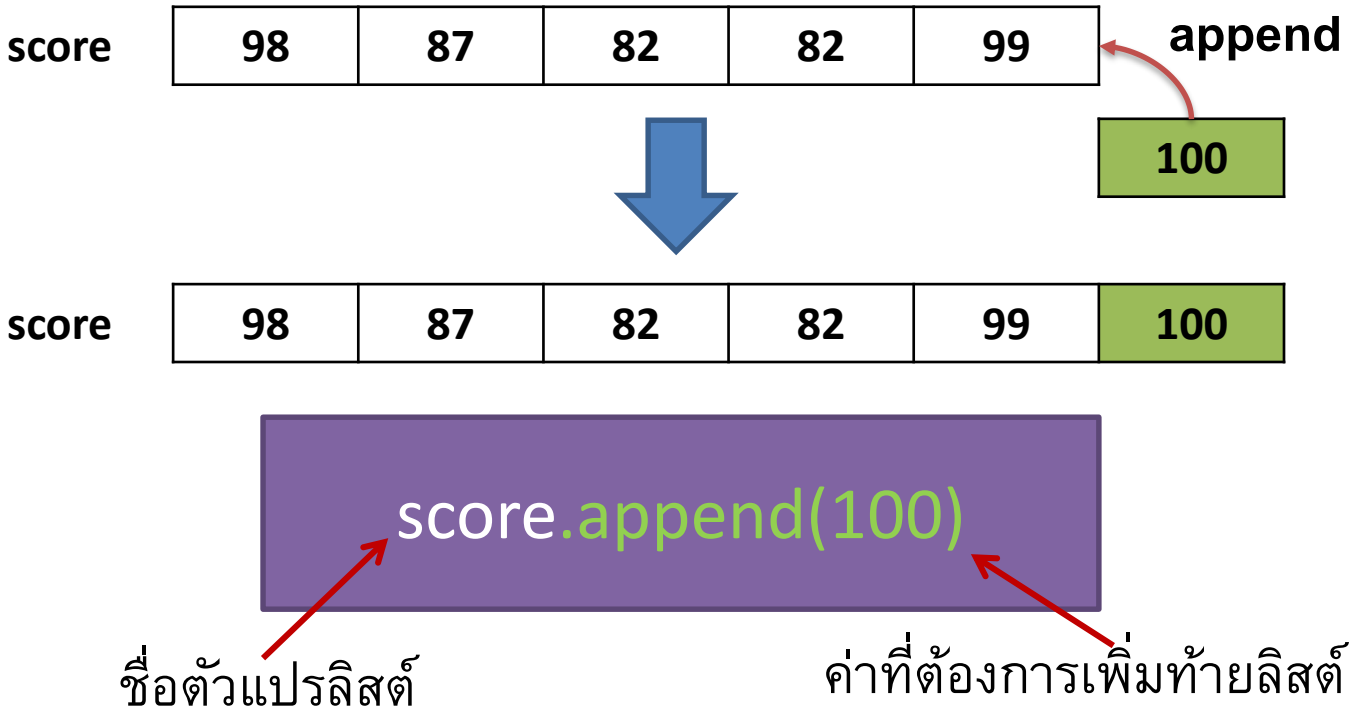



```
>> L = [0, 1]
>> A = L*3
>> print(A)
[0, 1, 0, 1, 0, 1]
```

List Methods

การเพิ่มสมาชิก

- การเพิ่มสมาชิกโดยการเติมลงท้ายลิสต์ ใช้การดำเนินการ append(x) เมื่อ x คือ ค่าที่ต้องการเพิ่มในลิสต์



List Methods

การเพิ่มสมาชิก

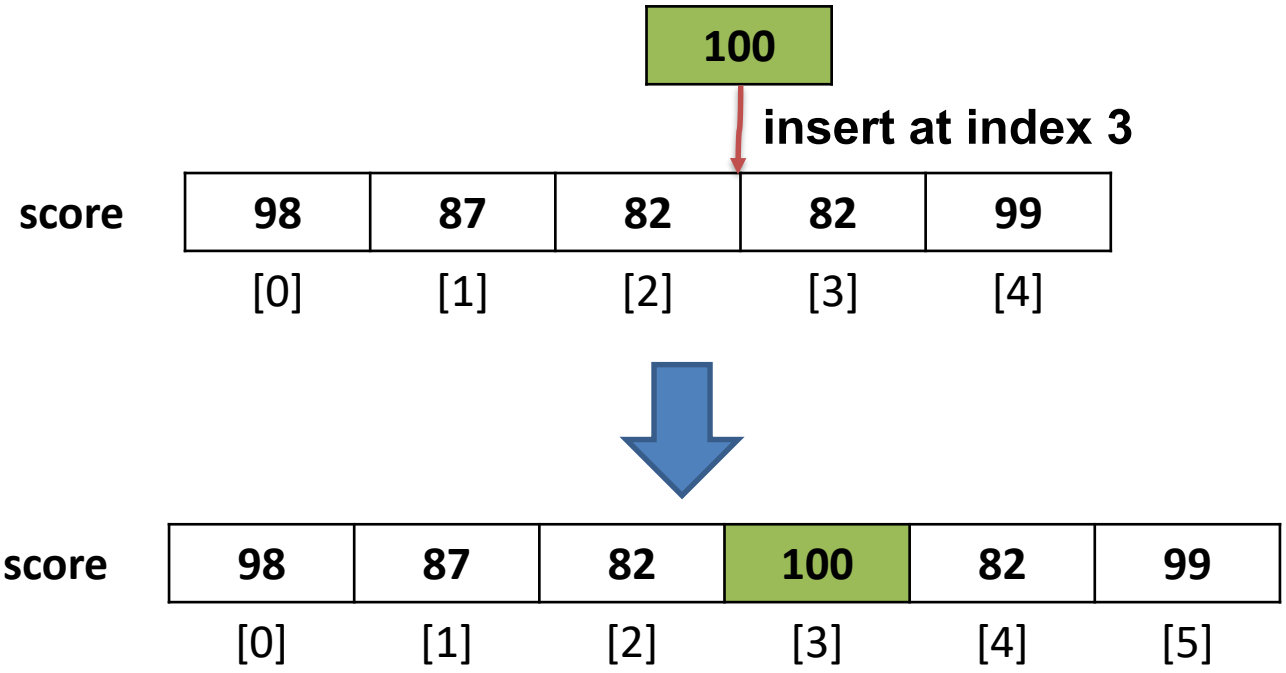
- การเพิ่มสมาชิกโดยการเติมลงท้ายลิสต์ ใช้การดำเนินการ append(x)

```
>> score = [98, 87, 82, 82, 99]
>> print(score)
[98, 87, 82, 82, 99]
>> score.append(100)
>> print(score)
[98, 87, 82, 82, 99, 100]
```

List Methods

การเพิ่มสมาชิก

- การแทรกสมาชิก ณ ตำแหน่งที่ต้องการ ใช้การดำเนินการ insert(p,x) เมื่อ p คือ ตำแหน่งที่ต้องการแทรก และ x คือ ค่าที่ต้องการแทรกในลิสต์



List Methods

การเพิ่มสมาชิก

- การแทรกสมาชิก ณ ตำแหน่งที่ต้องการ ใช้การดำเนินการ insert(p,x)


```
>> score = [98, 87, 82, 82, 99]
>> print(score)
[98, 87, 82, 82, 99]
>> score.insert(3,100)
>> print(score)
[98, 87, 82, 100, 82, 99]
```

List Methods

การนับจำนวนครั้งที่ค่าที่สนใจปรากฏในลิสต์

- สามารถนับจำนวนครั้งการปรากฏของค่าที่สนใจในลิสต์ได้ โดยใช้การดำเนินการ count(x) เมื่อ x คือ ค่าที่สนใจ

score	98	87	82	100	82	99

List Methods

การนับจำนวนครั้งที่ค่าที่สนใจปรากฏในลิสต์

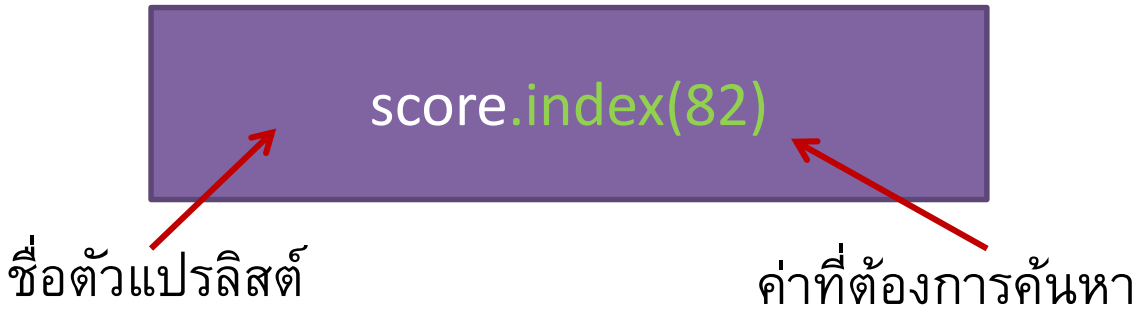
- สามารถนับจำนวนครั้งการปรากฏของค่าที่สนใจในลิสต์ได้ โดยใช้การดำเนินการ count(x) เมื่อ x คือ ค่าที่สนใจ

```
>> score = [98, 87, 82, 100, 82, 99]
>> print(score.count(82))
2
>> print(score.count(100))
1
>> print(score.count(101))
0
```

List Methods

การค้นหาค่าที่สนใจปรากฏในลิสต์

- ค้นหาตำแหน่งแรกที่ค่าที่สนใจปรากฏในลิสต์ โดยใช้การดำเนินการ index(x) เมื่อ x คือ ค่าที่สนใจ
  - ถ้าค้นหาเจอ จะได้ค่าตำแหน่งของค่าที่สนใจตัวแรกกลับมา ในกรณีที่มีค่าเดียวกันในหลายตำแหน่ง จะได้เฉพาะตำแหน่งแรกสุด
  - ถ้าค้นหาไม่เจอ จะเกิด ValueError เช่น ValueError: 101 is not in list



List Methods

การค้นหาค่าที่สนใจปรากฏในลิสต์

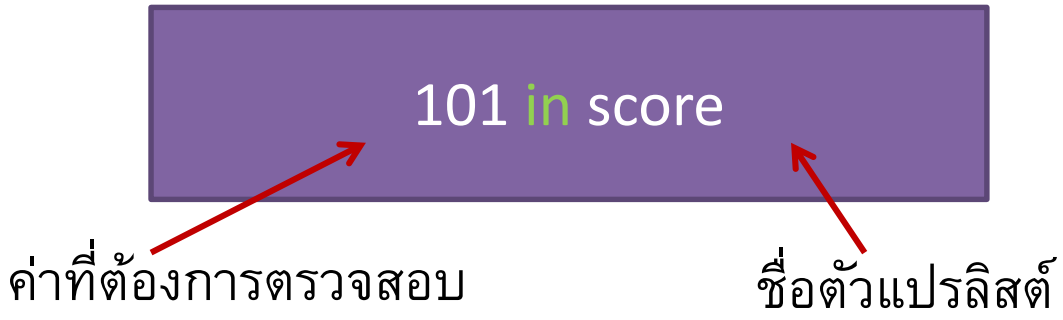
```
>> score = [98, 87, 82, 100, 82, 99]
>> print(score.index(82))
2
>> print(score.index(99))
5
>> print(score.count(101))
-----------------------------------------------
ValueError          Traceback (most recent call last)
<ipython-input-10-bd4c26e70894> in <module>()
      1 score = [98,87,82,82,99]
      2 score.insert(3,100)
----> 3 print(score.index(101))

ValueError: 101 is not in list
```

List Methods

การค้นหาค่าที่สนใจปรากฏในลิสต์

- เพื่อป้องการ Error ที่เกิดจากการค้นหาค่าที่ไม่มีอยู่ลิสต์ ควรตรวจสอบก่อนว่ามีค่าที่ต้องการเป็นสมาชิกในลิสต์หรือไม่?
- ใช้ตัวดำเนินการ in ในการตรวจสอบ
  - ถ้า ค่าที่สนใจ อยู่ในลิสต์ จะส่งค่า True กลับมา
  - ถ้า ค่าที่สนใจ ไม่อยู่ในลิสต์ จะส่งค่า False กลับมา



List Methods

```
>> score = [98, 87, 82, 100, 82, 99]
>> print(101 in score)
False
>> print(82 in score)
True
```

ตัวอย่างการประยุกต์ใช้

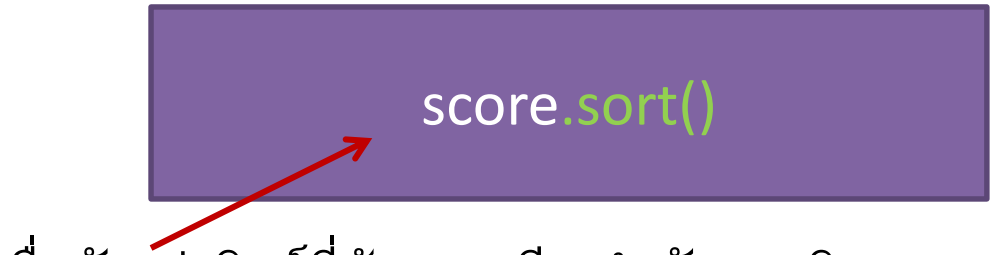
```
>> score = [98, 87, 82, 100, 82, 99]
>> x = int(input("Enter a score:"))
>> if x in score:
>>         print(x,"is at index",score.index(x))
>> else:
>>         print(x,"is not in list")
```

List Methods

การเรียงลำดับสมาชิกในลิสต์

- สามารถเรียงลำดับสมาชิกในลิสต์ได้ โดยใช้การดำเนินการ sort()
  - ถ้าสมาชิกเป็นตัวเลข จะเรียงลำดับตามค่าตัวเลข
  - ถ้าสมาชิกเป็นข้อความ จะเรียงตามลำดับอักษรอย่างพจนานุกรม
  - ปกติแล้ว sort() จะเรียงสมาชิกจากค่าน้อยไปค่ามาก
  - ถ้าต้องการเรียงลำดับจากมากไปน้อย ให้กำหนดพารามิเตอร์ reverse=True

    → sort(reverse=True)



List Methods

การเรียงลำดับสมาชิกในลิสต์

```
>> score = [98, 87, 82, 100, 82, 99]
>> score.sort()
>> print(score)
[82, 82, 87, 98, 99, 100]
```

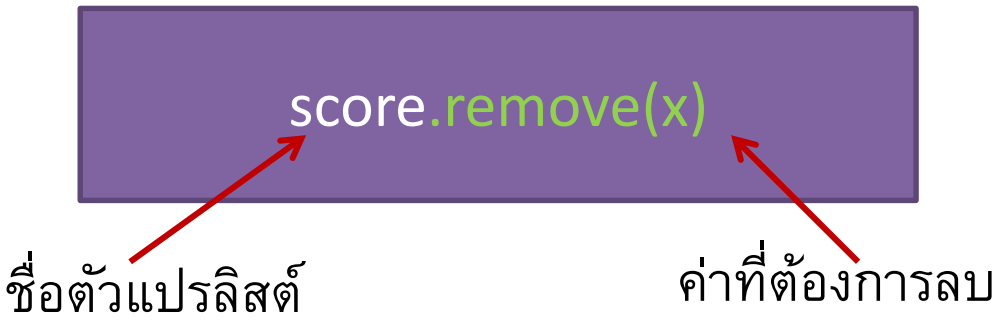
```
>> score = [98, 87, 82, 100, 82, 99]
>> score.sort(reverse=True)
>> print(score)
[100, 99, 98, 87, 82, 82]
```

```
>> words = ['zoo','small','apple','day']
>> words.sort()
>> print(words)
['apple', 'day', 'small', 'zoo']
```

List Methods

การลบสมาชิก

- ลบสมาชิกตัวแรกที่มีค่าเท่ากับค่าที่สนใจ โดยใช้การดำเนินการ remove(x) เมื่อ x คือ ค่าที่สนใจ
  - ถ้ามีสมาชิกที่มีค่า เท่ากับ x หลายตัว จะลบเฉพาะตัวแรกที่เจอที่นั้น



List Methods

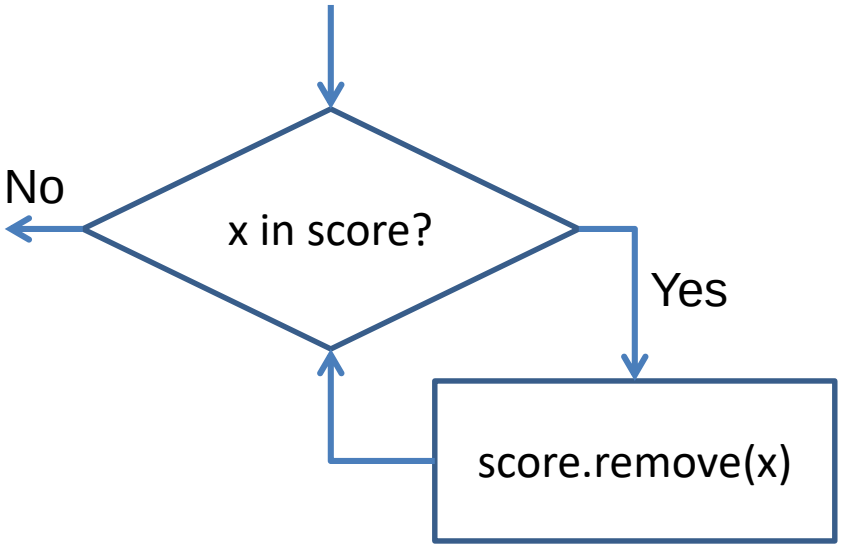
การลบสมาชิก

- ลบสมาชิกตัวแรกที่มีค่าเท่ากับค่าที่สนใจ โดยใช้การดำเนินการ remove(x) เมื่อ x คือ ค่าที่สนใจ
  - ถ้ามีสมาชิกที่มีค่า เท่ากับ x หลายตัว จะลบเฉพาะตัวแรกที่เจอที่นั้น

```
>> score = [98, 87, 82, 100, 82, 99]
>> score.remove(87)
>> print(score)
[98, 82, 100, 82, 99]
>> score.remove(82)
>> print(score)
[98, 100, 82, 99]
```

การลบสมาชิก

- ถ้าต้องการลบสมาชิกทุกตัวที่มีค่าเท่ากับ x จะทำอย่างไร?




```
>> score = [98, 87, 82, 100, 82, 99]
>> x = 82
>> while x in score:
>>                 score.remove(x)
>> print(score)
[98, 87, 100, 99]
```

List Methods

การลบสมาชิก

- ลบสมาชิกโดยการระบุตำแหน่ง ใช้การดำเนินการ pop(p) เมื่อ p คือตำแหน่งของสมาชิกที่ต้องการลบ
  - ข้อควรระวัง การลบสมาชิกในตำแหน่งที่ไม่มีอยู่จริง จะเกิด IndexError ดังนี้

    IndexError: pop index out of range

List Methods

การลบสมาชิก

- ลบสมาชิกโดยการระบุตำแหน่ง ใช้การดำเนินการ pop(p) เมื่อ p คือตำแหน่งของสมาชิกที่ต้องการลบ

```
>> score = [98, 87, 82, 100, 82, 99]
>> score.pop(3)
>> print(score)
[98, 87, 82, 82, 99]
```

List Methods

การลบสมาชิก

- การลบโดยใช้ตัวดำเนินการ **del**

`del score`	ลบตัวแปรลิสต์ทิ้ง
`del score[2]`	ลบสมาชิกในตำแหน่งที่ 2
`del score[1:3]`	ลบสมาชิกในตำแหน่งที่ 1 จนถึงก่อนหน้าตำแหน่งที่ 3

List Methods

การลบสมาชิก

- การลบโดยใช้ตัวดำเนินการ **del**

del score

```
>> score = [98, 87, 82, 100, 82, 99]
>> del score
>> print(score)
NameError: name 'score' is not defined
```

del score[2]

```
>> score = [98, 87, 82, 100, 82, 99]
>> del score[2]
>> print(score)
[98, 87, 100, 82, 99]
```

del score[1:3]

```
>> score = [98, 87, 82, 100, 82, 99]
>> del score[1:3]
>> print(score)
[98, 100, 82, 99]
```

Making Copies of Lists

การคัดลอกตัวลิสต์

- ถ้ามีตัวแปรลิสต์ L แล้วต้องการคัดลอก L ไปอีกตัวแปร M

`M = L`	✗	ไม่ควรทำ เพราะจะถือว่า M และ L เป็นลิสต์เดียวกันแต่มี 2 ชื่อ
`M = L[:]`	✓	คัดลอกทั้งลิสต์
`M = L[p:q]`	✓	คัดลองบางส่วน จากสมาชิกตำแหน่ง p จนถึงสมาชิกตำแหน่งก่อนหน้า q

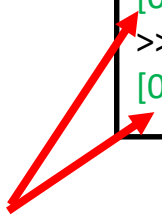
Making Copies of Lists

การคัดลอกตัวแปรลิสต์

M = L

ทำอะไรกับอีกตัวแปรหนึ่ง จะส่งผลกระทบไปยังอีกตัวแปรหนึ่ง

```
>> L = [0, 1, 2, 3, 4, 5]
>> M = L
>> M.remove(4)
>> print(M)
[0, 1, 2, 3, 5]
>> print(L)
[0, 1, 2, 3, 5]
```



Making Copies of Lists

การคัดลอกตัวแปรลิสต์

M = L[:]

```
>> L = [0, 1, 2, 3, 4, 5]
>> M = L[:]
>> print(L)
[0, 1, 2, 3, 4, 5]
>> print(M)
[0, 1, 2, 3, 4, 5]
>> L.remove(4)
>> print(L)
[0, 1, 2, 3, 5]
>> print(M)
[0, 1, 2, 3, 4, 5]
```

M = L[p:q]

```
>> L = [0, 1, 2, 3, 4, 5]
>> M = L[2:5]
>> print(M)
[2, 3, 4]
```

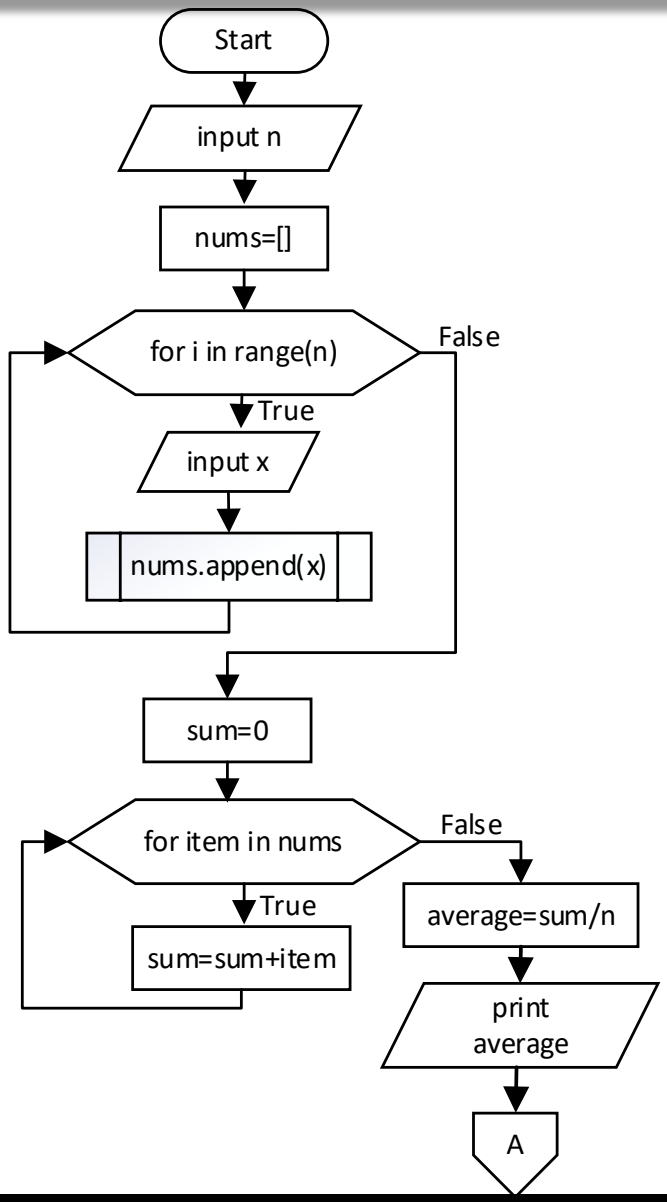
Exercise

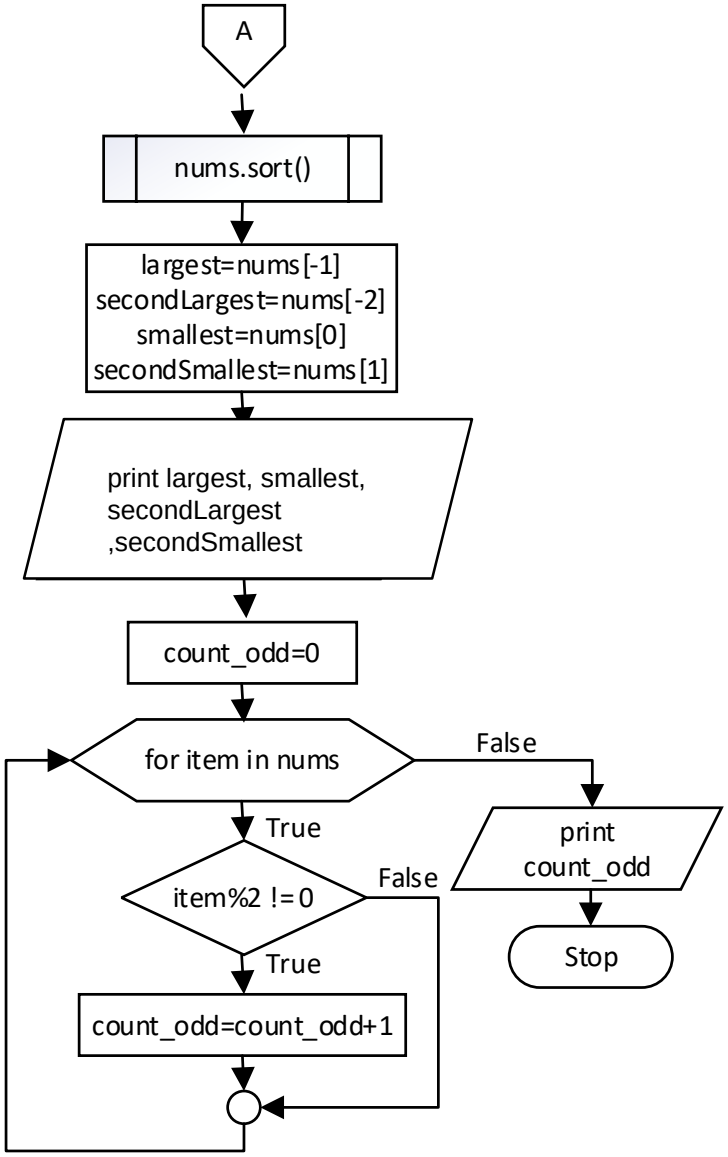
แบบฝึกหัด

จงเขียนโปรแกรมเพื่อรับตัวเลขจำนวน n ตัว (ค่า n รับจากผู้ใช้) เก็บไว้ในตัวแปรลิสต์ nums และทำงานดังต่อไปนี้

- แสดงตัวแปรลิสต์ nums
- คำนวณค่าเฉลี่ยและแสดงค่าเฉลี่ยของสมาชิกในลิสต์ nums
- หาค่ามากที่สุดและค่าน้อยที่สุดของตัวเลขในลิสต์ nums
- หาค่ามากที่สุดอันดับ 2 และค่าน้อยที่สุดอันดับ 2 ของตัวเลขในลิสต์ nums
- นับจำนวนตัวเลขที่เป็นเลขคี่ในลิสต์ nums และแสดงจำนวนที่นับได้

Exercise


Start
input n
nums=[]
for i in range(n)
False
True
input x
nums.append(x)
sum=0
for item in nums
False
True
sum=sum+item
average=sum/n
print
average
A


A
nums.sort()
largest=nums[-1]
secondLargest=nums[-2]
smallest=nums[0]
secondSmallest=nums[1]
print largest, smallest,
secondLargest
,secondSmallest
count_odd=0
for item in nums
False
True
item%2 != 0
False
True
count_odd=count_odd+1
print
count_odd
Stop