



01076103, 01076104
Programming Fundamental
Programming Project

Boolean, If statement



Boolean

- Boolean เป็นชนิดข้อมูลหนึ่งของ Python
- มีค่าเพียง 2 ค่า คือ True และ False
- ค่า True สามารถตีความเป็นตัวเลขได้ โดยมีค่า = 1
 - เช่น ถ้ามีประโยค `a = 1 == True` ; a จะมีค่า = True
 - ประโยค `a = "1" == True` มีค่าเป็นอะไร
- สำหรับค่า False ถ้าตีความเป็นตัวเลข มีค่า = 0
 - เช่น ถ้ามีประโยค `a = 0 == False` ; a จะมีค่า = True



การกระทำแบบเปรียบเทียบ

- การกระทำแบบเปรียบเทียบจะให้ผลลัพธ์เป็น Boolean

Operator	Meaning
==	Equal to
!=	Not equal to
>	Greater than
<	Less than
>=	Greater than or equal to
<=	Less than or equal to



การกระทำแบบเปรียบเทียบ

main.py ×

```
1 spam = True
2 print(spam)
3 print(42 == 42)
4 print(42 == 99)
5 print(2 != 3)
6 print(2 != 2, end='\n\n')
7
8 print('hello' == 'hello')
9 print('hello' == 'Hello')
10 print('dog' != 'cat')
11 print(True == True)
12 print(True != False)
13 print(42 == 42.0)
14 print(42 == '42')
15
```

Console Shell

```
True
True
False
True
False

True
False
True
True
True
True
False
> 
```



การกระทำแบบเปรียบเทียบ

```
main.py x
1 print(42 < 100)
2 print(42 > 100)
3 print(42 < 42)
4
5 eggCount = 42
6 print(eggCount <= 42)
7 myAge = 29
8 print(myAge >= 10)
9
```

Console Shell

```
True
False
False
True
True
>
```



การกระทำแบบเปรียบเทียบ

- การกระทำแบบเปรียบเทียบยังมีอีก 1 ตัว โดยใช้ตรวจสอบการเป็นสมาชิก ได้แก่ operator in
- เช่น 'a' in 'abc' จะได้ผลลัพธ์เป็น True
- การใช้ in จะมีบทบาทมากในการเขียนโปรแกรมต่อไปในอนาคต
- นอกจากนั้นยังมีการใช้แบบตรงกันข้าม คือ not in อีกด้วย



การกระทำแบบเปรียบเทียบ

- การกระทำแบบเปรียบเทียบ สามารถเขียนในรูปแบบ “ลูกโซ่” หรือ Chained Comparison ได้
 - โดย $a == b == c \Rightarrow a == b \text{ and } b == c$
 - และ $a < b < c \Rightarrow a < b \text{ and } b < c$
- การเขียนในลักษณะ “ลูกโซ่” แบบนี้ จะทำให้ นิพจน์ สั้นลง
- เช่น หากต้องการทราบว่าตัวแปร a อยู่ระหว่าง 1 – 10 หรือไม่ ก็สามารถเขียนเป็น $1 < a < 10$




การกระทำแบบเปรียบเทียบ

- ให้กระจายและบอกผลของ Expression ต่อไปนี้
 - $5 < 6 > 2 \quad \Rightarrow \quad 5 < 6 \text{ and } 6 > 2 \quad \Rightarrow \text{True}$
 - $5 < 6 > 10 \quad \Rightarrow \quad 5 < 6 \text{ and } 6 > 10 \quad \Rightarrow \text{False}$
 - $1 < 2 < 3 < 4 \quad \Rightarrow \quad 1 < 2 \text{ and } 2 < 3 \text{ and } 3 < 4$

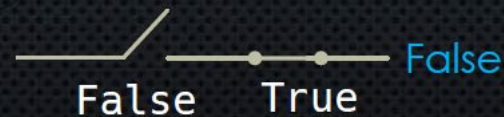
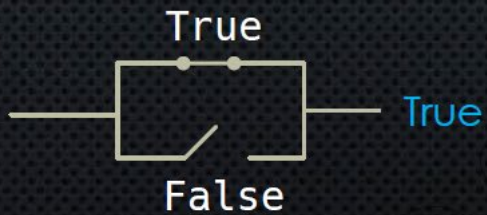
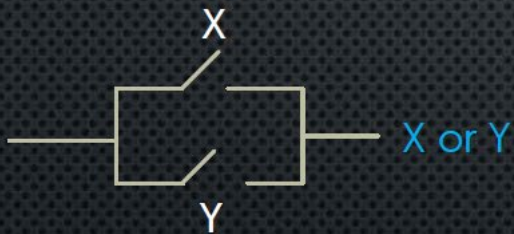


การกระทำทางลอจิก (Logical)

X	Y	not X	X and Y	X or Y
0	0	1	0	0
0	1	1	0	1
1	0	0	0	1
1	1	0	1	1

 open False

 closed True





การกระทำทางลอจิก (Logical)

- Operator AND จะ True กรณีเดียว คือ True and True
- Operator OR จะ False กรณีเดียว คือ False or False

```
main.py x
1 print(True and True)
2 print(True and False)
3 print(False or True)
4 print(False or False)
5
6 print(not True)
7 print(not not not not True)
8
```

Console Shell

```
True
False
True
False
False
True
>
```



การกระทำทางลอจิก (Logical)

- การทำงานของ AND คือ

`X and Y`: If X is falsy, returns X, otherwise evaluates and returns Y

- ถ้า X เป็น False จะคืนค่า X โดยไม่ประเมินค่า Y เลย แต่ถ้าเป็น True จึงประเมินค่า Y

```
main.py x
1 s1 = None
2 s2 = ''
3 s3 = 'abc'
4
5 print(s1 and s1[0])
6 print(s2 and s2[0])
7 print(s3 and s3[0])
```

Console

```
None
a
>
```



การกระทำทางลอจิก (Logical)

- การทำงานของ OR คือ

`X or Y` : If X is falsy, returns Y, otherwise evaluates and returns X

- ถ้า X เป็น False จะประเมินค่า Y แต่ถ้าเป็น True จะประเมิน X

```
main.py x Console Shell
1 print('' or 'abc')
2 print(0 or 100)
3 print(1 or 1/0)
4
```

abc
100
1
>

- ในบรรทัดที่ 3 ปกติ 1/0 จะ Error แต่กรณีนี้ไม่ Error เพราะไม่ทำงาน



การกระทำทางลอจิก (Logical)

- กรณีที่มีการทำงานทางลอจิกหลายๆ ตัวอยู่ด้วยกัน จะมีลำดับการทำงานดังนี้
 - () ถ้ามีวงเล็บ ทำในวงเล็บก่อน
 - < > <= >= != in ลำดับที่ 2
 - not ลำดับที่ 3
 - and ลำดับที่ 4
 - or ลำดับที่ 5

(4 < 5) and (5 < 6)
↓
True and (5 < 6)
↓
True and True
↓
True



การกระทำทางลอจิก (Logical)

- ลองพิจารณานิพจน์ต่อไปนี้
 - True or True and False
 - True or (True and False)
 - True or True) and False
- $a < b$ or $a > c$ and not x or y มีลำดับการทำงานอย่างไร
- $2 + 2 == 4$ and not $2 + 2 == 5$ and $2 * 2 == 2 + 2$ ได้ผลลัพธ์เป็นอะไร



Exercise

- ให้บอกผลลัพธ์ของการทำงานต่อไปนี้

$(5 > 4) \text{ and } (3 == 5)$

$\text{not } (5 > 4)$

$(5 > 4) \text{ or } (3 == 5)$

$\text{not } ((5 > 4) \text{ or } (3 == 5))$

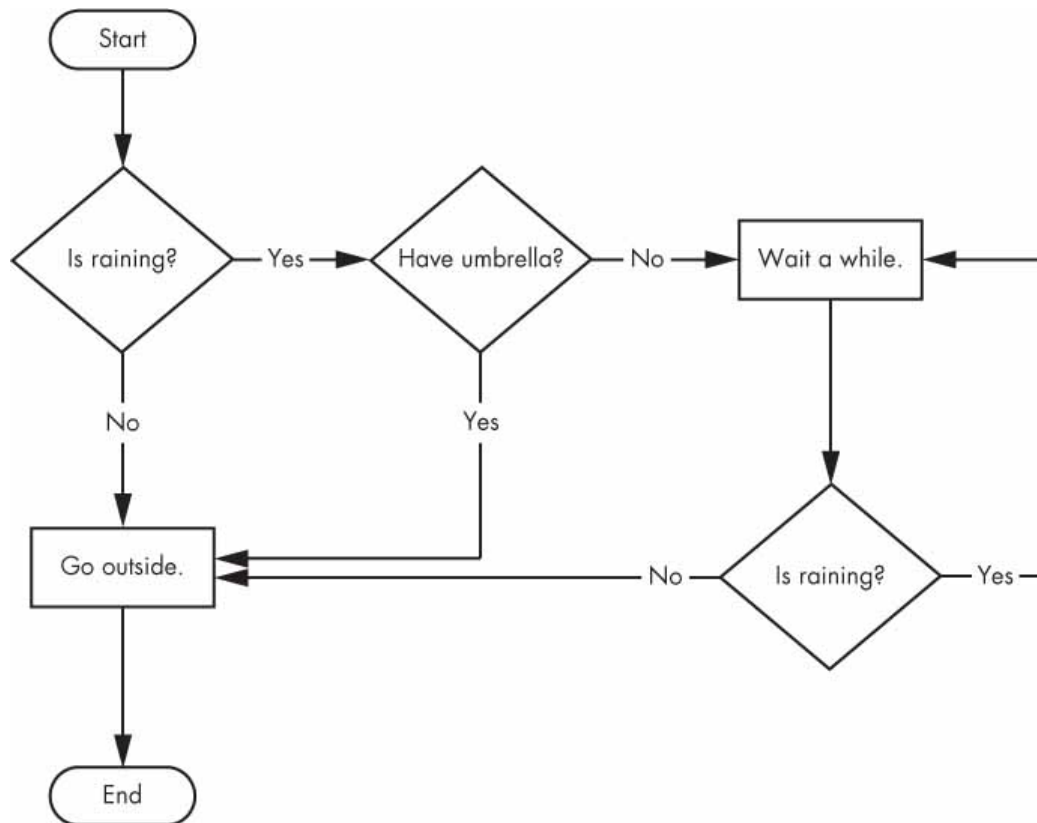
$(\text{True and True}) \text{ and } (\text{True} == \text{False})$

$(\text{not False}) \text{ or } (\text{not True})$



Flowchart

- Flowchart เป็นเครื่องมือที่ช่วยในการคิด เพื่อเขียนโปรแกรม สัญลักษณ์ที่ใช้ใน Flowchart ไม่ตายตัว โดยทั่วไป จะมีลักษณะดังรูป





Pseudo code

- Pseudo code หรือ โค้ดเทียม เป็นอีกเครื่องมือที่ช่วยในการคิด
- วิธีเขียน ไม่มีรูปแบบตายตัว แต่เขียนให้ใกล้เคียงภาษามนุษย์ แต่มีรูปแบบที่สามารถแปลงเป็นโปรแกรมได้ **เป็นเครื่องมือที่ใช้อธิบายความคิด** เช่น ถ้าเป็นงานเดียวกับ slide ก่อนหน้าจะเขียนเป็น

ฝนตกหรือไม่

มีร่มหรือไม่

มี -> ออกไปนอกบ้านได้

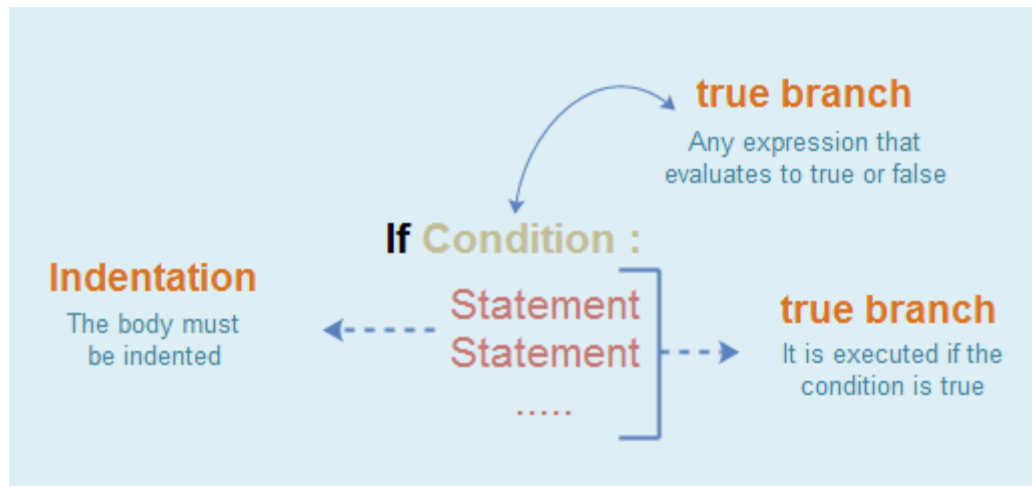
ไม่มี -> รอจนกว่าฝนจะหยุด แล้วออกนอกบ้านได้

ฝนไม่ตก -> ออกนอกบ้านได้



If statement

- If statement เป็นการทำงานที่พบได้บ่อยในโปรแกรม โดยมีรูปแบบดังนี้
 - เริ่มด้วย if
 - ต่อด้วย expression ซึ่งต้องให้ผลเป็น True หรือ False
 - ปิดท้าย expression ด้วยเครื่องหมาย :
 - Block code ส่วนของการทำงานเมื่อเป็น True (ต้อง Indent)



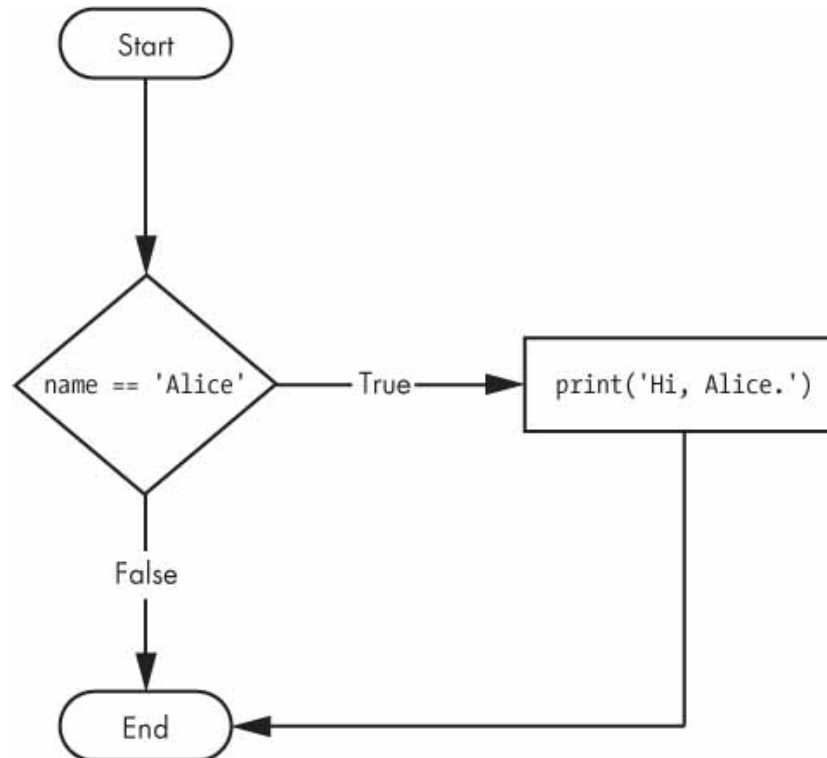


If statement

- การทำงานของ

```
if name == 'Alice':  
    print('Hi, Alice.')
```

- เป็นตามรูป





If statement

- สมมติว่าค่าเข้าชมสวนสนุก เด็กจะได้ลด 50% ถ้าสูง < 120
- สามารถเขียนเป็นโปรแกรมได้ดังนี้

main.py ×

```
1  TICKET_PRICE = 100
2
3  payment = TICKET_PRICE
4  height = int(input("What is your height in cm. :"))
5  ▼ if height < 120:
6      payment *= 0.5
7
8  print("Your payment = ", payment)
9
```

Console Shell

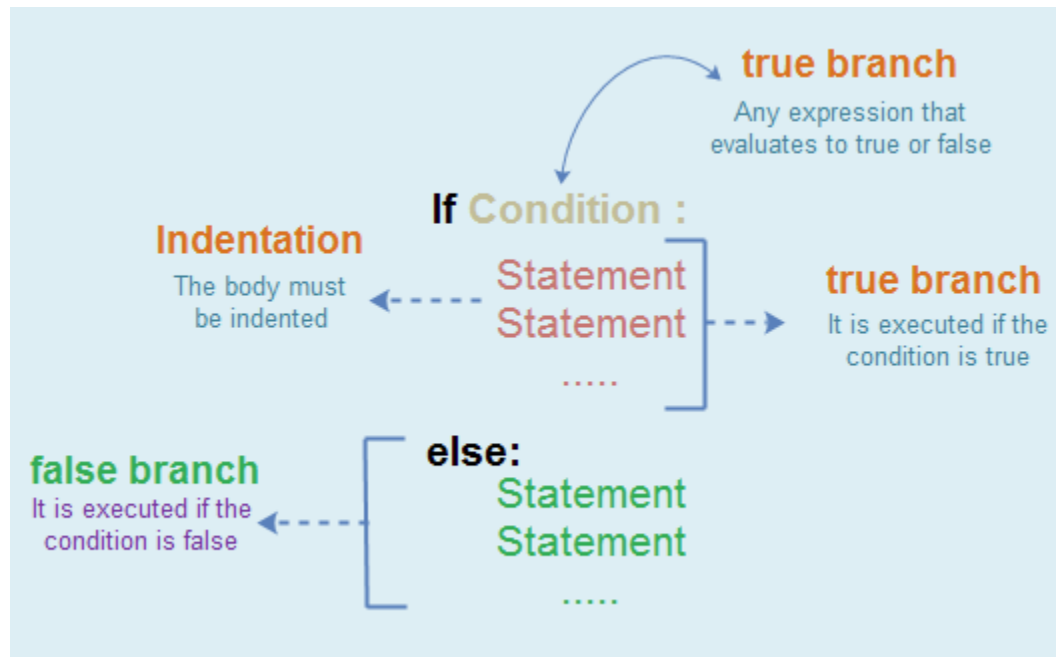
```
What is your height in cm. :100
Your payment = 50.0
> □
```

- กรณีที่เป็นข้อมูล ลักษณะที่เป็นค่าคงที่ ให้เขียนเป็นตัวใหญ่ทั้งหมด



If else statement

- บางครั้งจะมีเงื่อนไขที่ถ้า ใช่ ให้ทำแบบหนึ่ง แต่ถ้าไม่ใช่ให้ทำอีกแบบหนึ่ง
- จากรูปจะเห็นว่าถ้า Expression เป็น True จะทำ Block code แรก แต่ถ้าเป็น False ก็จะทำ Block code ที่ 2
- จะมีคำว่า else: เพิ่มเข้ามา



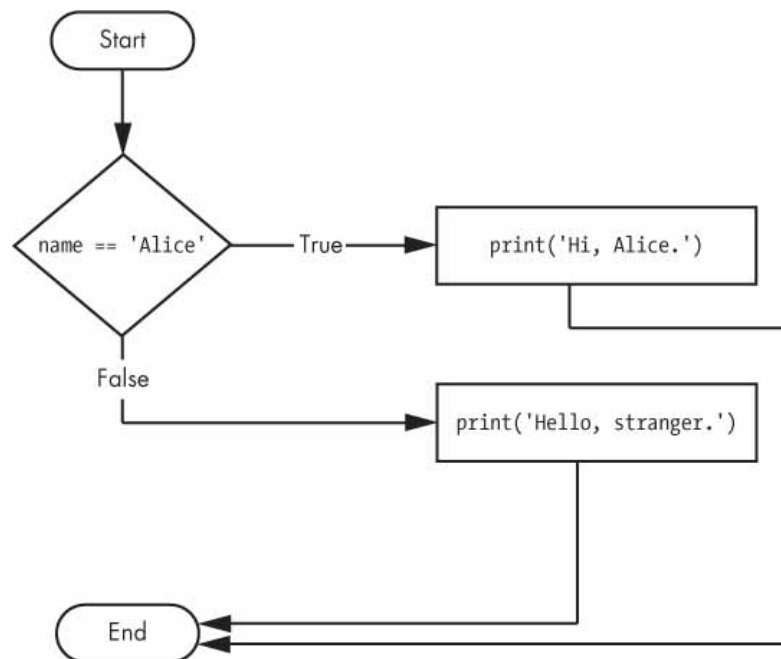


If else statement

- การทำงานของ

```
if name == 'Alice':  
    print('Hi, Alice.')  
  
else:  
    print('Hello, stranger.')
```

- เป็นตามรูป





If else statement

- สมมติว่าจะเขียนโปรแกรมเพื่อบอกว่าตัวเลขที่ป้อนเข้ามาเป็นเลขคู่หรือเลขคี่
- สามารถเขียนโค้ดเทียมดังนี้

รับข้อมูลตัวเลข

ถ้าหารด้วย 2 ลงตัว ให้ print even มิฉะนั้นให้ print odd

- และเขียนโปรแกรมได้ดังนี้

main.py ×

```
1 num = int(input("Enter your number (integer): "))
2 ▼ if num % 2 == 0:
3     print(num, "is an even number")
4 ▼ else:
5     print(num, "is an odd number")
6
```



Console

Shell

```
Enter your number (integer): 23
23 is an odd number
> █
```



If else statement

- **Exercise 2.1** จงเขียนโปรแกรมรับค่าของตัวเลข 1 ค่า (x) จากคีย์บอร์ด และทดสอบว่าเป็นเลขที่หารด้วย 5 ลงตัว หรือไม่
- ตัวอย่าง

Enter x: 10

10 is divisible by 5

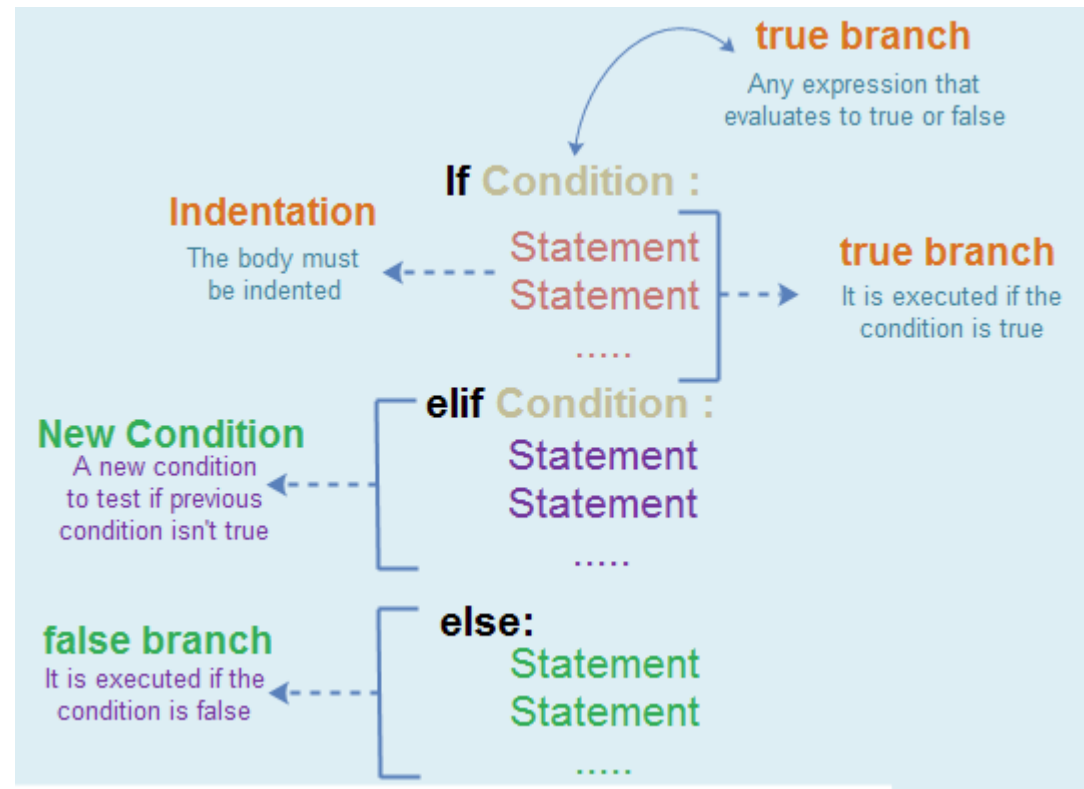
Enter x: 11

11 is not divisible by 5



elif statement

- บางครั้งจะมีเงื่อนไขที่เป็นหลายเงื่อนไข ถ้าไม่ตรงกับเงื่อนไขที่ 1 ก็จะตรวจสอบเงื่อนไขที่ 2 และต่อไป
 - ในเงื่อนไขแรก จะเขียนเหมือน if else ตามปกติ
 - ในเงื่อนไขที่ 2 จะเขียนเป็น elif ซึ่งสามารถใช้ได้หลายครั้ง
 - และหากไม่ตรงกับเงื่อนไขใดเลย จะทำงาน ใน Block else สุดท้าย
 - ในส่วนของ else สุดท้าย จะมีหรือไม่ก็ได้



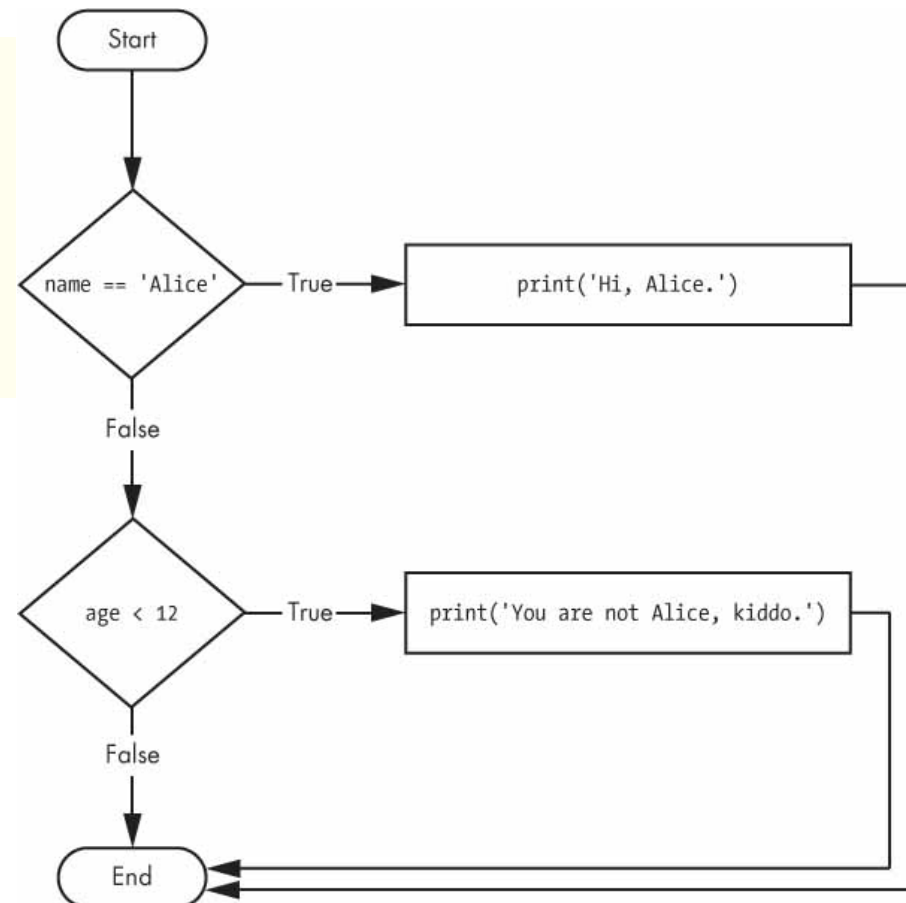


elif statement

- การทำงานของ

```
if name == 'Alice':  
    print('Hi, Alice.')  
elif age < 12:  
    print('You are not Alice, kiddo.')
```

เป็นตามรูป





elif statement

- ตัวอย่าง ในสวนสนุกแห่งหนึ่งมีอัตราการเข้าชมต่างกันตามอายุ
 - ถ้าต่ำกว่า 4 ขวบ ฟรี
 - อายุ 4-18 = 25 บาท
 - อายุ 18-65 = 40 บาท
 - อายุ > 65 = 20 บาท
 - จงเขียนโปรแกรมรับอายุแล้วคำนวณจำนวนเงิน



elif statement

- วิเคราะห์โจทย์ สามารถเขียนโค้ดเทียมได้ดังนี้

รับข้อมูลตัวเลขอายุ

ถ้าอายุน้อยกว่า 4 fee = 0

แต่ถ้าอายุ 5-18 fee = 25

แต่ถ้าอายุมากกว่า 18-64 fee = 40

แต่ถ้าอายุมากกว่า 65 fee = 20



elif statement

- โจทย์ลักษณะนี้ ควรระวังปัญหาค่าขอบ หรือ รอยต่อระหว่างเงื่อนไข

```
main.py x
1 age = int(input("Enter your age : "))
2
3 ▼ if age <= 4:
4     price = 0
5 ▼ elif age <= 18:
6     price = 25
7 ▼ elif age < 65:
8     price = 40
9 ▼ elif age >= 65:
10     price = 20
11
12 print(f"Your admission cost is {price}.")
```

```
Console Shell
Enter your age : 66
Your admission cost is 20.
> 
```



elif statement

- โจทย์ข้อที่แล้ว เขียนได้อีกแบบหนึ่ง

```
main.py x
1 age = int(input("Enter your age : "))
2
3 ▼ if age <= 4:
4     price = 0
5 ▼ if 4 < age <= 18:
6     price = 25
7 ▼ if 18 < age <= 65:
8     price = 40
9 ▼ if age >= 65:
10     price = 20
11
12 print(f"Your admission cost is {price}.")
13
```

Console Shell

```
Enter your age : 45
Your admission cost is 40.
>
```



elif statement

- ตัวอย่าง สมมติว่าการโอนเงินไปต่างประเทศมีค่าธรรมเนียม ดังนี้
 - 0-5000 ไม่มีค่าธรรมเนียม
 - 5000-30,000 2 บาท
 - 30,000-100,000 5 บาท
 - > 100,000 10 บาทต่อ 1 แสน
 - จงหาจำนวนเงินที่ต้องจ่ายในการโอน



elif statement

- โจทย์ข้อนี้ หากเขียนแบบนี้จะใช้ if หรือ elif ก็ไม่แตกต่างกัน ข้อควรระวัง คือ จะต้องมียกเว้นเพียง if เดียวที่ตรงตามเงื่อนไขเท่านั้น

main.py ×

```
1 amount = int(input("Enter amount of transfer money: "))
2 fee = 0
3 ▼ if amount <= 5000:
4     fee=0
5 ▼ elif 5000 < amount <= 30000:
6     fee=2
7 ▼ elif 30000 < amount <= 100000:
8     fee=5
9 ▼ else:
10     fee=10*(amount//100000)
11
12 print(f"Transfer fee = {fee}.")
13
```



Console

Shell

```
Enter amount of transfer money: 1000000
Transfer fee = 100.
> 
```




elif statement

- **Exercise 2.2** จงเขียนโปรแกรมรับข้อมูลจำนวน 2 ค่า คือ คะแนน midterm และ final ถ้าคะแนนรวมกัน > 50 แสดง pass ถ้าต่ำกว่าแสดง fail
 - `m_score = int(input("Enter midterm score:"))`
 - `f_score = int(input("Enter final score:"))`



elif statement

- **Exercise 2.3** จงเขียนโปรแกรมรับข้อมูลความเร็ว
 - ถ้าน้อยกว่า 45 แสดงผล slow speed
 - ถ้าระหว่าง 45 -> 90 แสดงผล moderate speed
 - ถ้ามากกว่า 90 ให้แสดงผล fast speed
 - `speed = input("Enter speed:")`



elif statement

- ตัวอย่าง จงเขียนโปรแกรมรับค่าคะแนนเพื่อตัดเกรด ข้อมูลที่รับมาเป็น float ให้ปรับเป็นจำนวนเต็มก่อน โดย หาก 0.5 ขึ้นไปให้ปัดขึ้น เช่น 50.7 -> 51 หากน้อยกว่า 50 ให้ปัดลง 50.49 -> 50 จากนั้นค่อยเอามาตัดเกรดดังนี้
 - < 50 : F
 - 50 -> 55 : D
 - 55 -> 60 : D+
 - 60 -> 65 : C
 - 65 -> 70 : C+
 - 70 -> 75 : B
 - 75 -> 80 : B+
 - > 80 : A



elif statement

main.py x



Console

Shell

```
1 score = float(input("Enter score = "))
2
3 ▼ if (score % 1) >= 0.5:
4     Before_point = score // 1
5     score = Before_point + 1
6     print("Round up score =", score)
7
8 ▼ if(score < 50):
9     print("Your grade is 0.0")
10 ▼ elif (score >= 50 and score < 55):
11     print("Your grade is 1.0")
12 ▼ elif (score >= 55 and score < 60):
13     print("Your grade is 1.5")
14 ▼ elif (score >= 60 and score < 65):
15     print("Your grade is 2.0")
16 ▼ elif (score >= 65 and score < 70):
17     print("Your grade is 2.5")
18 ▼ elif (score >= 70 and score < 75):
19     print("Your grade is 3.0")
20 ▼ elif (score >= 75 and score < 80):
21     print("Your grade is 3.5")
22 ▼ elif (score >= 80):
23     print("Your grade is 4.0")
24 ▼ else:
25     print("Your score is incorrect")
26
```

```
Enter score = 75
Your grade is 3.5
>
```



elif statement

- การปัดเศษ เป็นปัญหาอย่างหนึ่งในการเขียนโปรแกรม การปัดเศษ สามารถทำได้หลายวิธี
 - ปัดเศษทิ้ง จะใช้คำสั่ง `floor(x)` เช่น `floor(3.17) = 3` หรือ `floor(3.7) = 3`
 - การปัดเศษทิ้งจะใช้ floor division ก็ได้ เช่น `3.17 // 1` หรือ `3.7 // 1`
 - ปัดเศษขึ้น จะใช้คำสั่ง `ceil(x)` เช่น `ceil(3.1) = 4`
 - `round` เป็นคำสั่งที่ปัดเข้าหาค่า `int` ที่ใกล้ที่สุด เช่น `round(3.1) = 3`, `round(3.7)=4`
 - `round` สามารถกำหนดทศนิยมที่จะปัดได้เช่น `round(3.14,1) = 3.1`
 - ปกติ ถ้า 3.5 จะปัดขึ้น แต่ `round(3.15,1) = 3.1`, `round(3.25,1) = 3.2` แต่ `round(3.35,1) = 3.4` ???
 - เนื่องจาก python ออกแบบให้ปัดลงผสมปัดขึ้น เพื่อให้ค่าเฉลี่ยไม่เพี้ยนมากเกินไป
 - ถ้าจะหลีกเลี่ยง ให้ใช้วิธีบวก 0.5 เข้าไปแล้วค่อยใช้ `round`



elif statement

- **Exercise 2.4** จากโปรแกรม BMI Calculator ในบทที่แล้ว $BMI = \text{weight(kg)} / \text{height(m)}^2$ ให้เขียนโปรแกรมเพิ่มเติม ดังนี้
 - Input:
 - Enter weight : 80
 - Enter height : 1.75
 - Output:
 - BMI = 26
- นอกจากจะบอกค่า BMI แล้ว ยังให้คำแนะนำดังนี้
 - น้อยกว่า 18.5 : they are underweight
 - มากกว่า 18.5 แต่น้อยกว่า 25 : they have a normal weight
 - มากกว่า 25 แต่น้อยกว่า 30 : they are slightly overweight
 - มากกว่า 30 แต่น้อยกว่า 35 : they are obese
 - มากกว่า 35 : they are clinically obese.



elif statement

- การเปรียบเทียบที่ใช้บ่อย
 - if $a \% 2 == 0$: a เป็นเลขคู่หรือไม่
 - if $a \% 100 == 0$: a หารด้วย 100 ลงตัวหรือไม่
 - if $(a // 100) \% 10 == 9$: เลขหลักร้อยของ a คือ 9 หรือไม่ หรือ
 - if $(a \% 1000) // 100 == 9$: ก็เหมือนกัน
 - if $a \leq x \leq b$: x มีค่าในช่วงตั้งแต่ a ถึง b หรือไม่



Logical Operation

- คุณสมบัติบางประการของ Boolean
 - คุณสมบัติการสลับที่ (Commutativity)

$$A \text{ or } B == B \text{ or } A$$
$$A \text{ and } B == B \text{ and } A$$

- คุณสมบัติของการกระจาย

$$A \text{ and } (B \text{ or } C) == (A \text{ and } B) \text{ or } (A \text{ and } C)$$
$$A \text{ or } (B \text{ and } C) == (A \text{ or } B) \text{ and } (A \text{ or } C)$$

- คุณสมบัติของการจัดกลุ่ม

$$A \text{ or } (B \text{ or } C) == (A \text{ or } B) \text{ or } C$$
$$A \text{ and } (B \text{ and } C) == (A \text{ and } B) \text{ and } C$$



Logical Operation

- คุณสมบัติบางประการของ Boolean
 - De Morgan's Theorem

```
not(A or B) == (not A) and (not B)
```

```
not(A and B) == (not A) or (not B)
```

- อื่นๆ

```
not(x < y) == x >= y    not(x <= y) == x > y
```

```
not(x > y) == x <= y    not(x >= y) == x < y
```

```
not(not A) == A
```



Short Circuit Evaluation ใน Logical Operation

- จากที่กล่าวว่า การทำงานของ AND คือ

X and Y: If X is falsy, returns X, otherwise evaluates and returns Y

— ถ้า X เป็น False จะคืนค่า X โดยไม่ประเมินค่า Y เลย แต่ถ้าเป็น True จึงประเมินค่า Y

- และการทำงานของ OR คือ

X or Y: If X is falsy, returns Y, otherwise evaluates and returns X

- ถ้า X เป็น False จะประเมินค่า Y แต่ถ้าเป็น True จะประเมิน X



Short Circuit Evaluation ใน Logical Operation

- จะมีบางกรณีที่เราใช้ประโยชน์จากการทำงานข้างต้น
- สมมติว่าเรามีระบบหนึ่ง เก็บข้อมูลจาก sensor จำนวนหนึ่ง โดยมีเงื่อนไขว่าหาก sensor ที่เราสนใจ ถ้ามีระดับสูงกว่าที่กำหนดให้แจ้งเตือน สามารถเขียนเป็นโปรแกรม

```
if sensor in watch_list:  
    if value(sensor) > threshold:  
        # do something
```

- จากเงื่อนไข AND จะเห็นว่า ถ้าเงื่อนไขแรกเป็น False มันจะไม่ทำงานในเงื่อนไขหลังเลย ดังนั้นหากเราเขียนแบบนี้ ก็ยังทำงานเหมือนเดิมทุกประการ

```
if sensor in watch_list and value(sensor) > threshold:  
    # do something
```



Short Circuit Evaluation ใน Logical Operation

- ดูอีกตัวอย่าง โปรแกรมนี้ จะมีปัญหาหาก b เป็น 0 (ตัวหารเป็น 0)

```
a = 10
b = 2

if a/b > 2:
    print('a is at least double b')
```

- แต่หากเขียนแบบนี้ จะสามารถแก้ปัญหาก็ได้

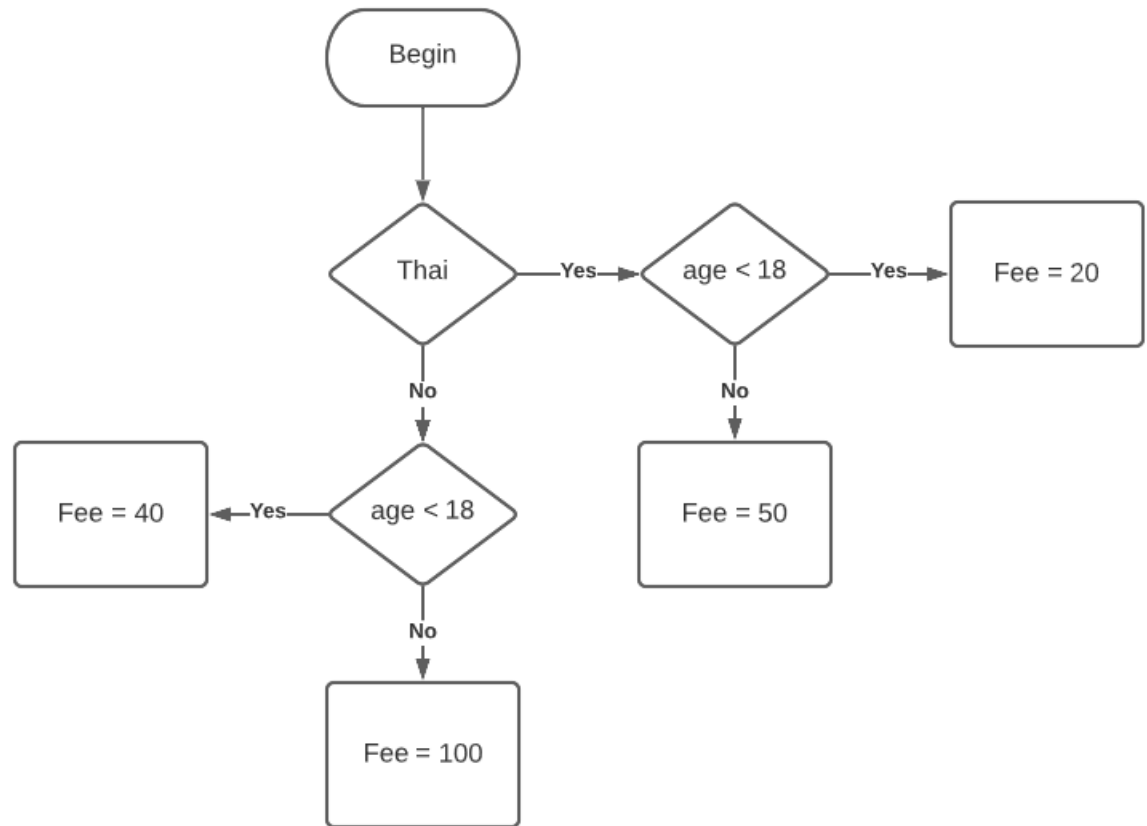
```
a = 10
b = 0

if b and a/b > 2:
    print('a is at least double b')
```



Nest if

- บางครั้งในการเขียนโปรแกรม อาจมีเงื่อนไขที่ต้องพิจารณามากกว่า 1 เงื่อนไข
- เช่น สมมติว่า การเข้าอุทยานแห่งชาติ มีการแบ่งออกเป็น
 - คนไทย ผู้ใหญ่ 50 บาท
 - เด็ก 20 บาท
 - ต่างชาติ ผู้ใหญ่ 100 บาท
 - เด็ก 40 บาท
- จะเห็นว่ามี 2 เงื่อนไข คือ คนไทย หรือ ต่างชาติ และ เด็ก หรือ ผู้ใหญ่





Nest if

main.py x

```
1 thai_citizen = input("Thai citizen (Y/N) :")
2 age = int(input("Enter your age : "))
3
4 fee=0
5 ▼ if (thai_citizen=='Y'):
6 ▼     if (age < 18):
7         fee = 20
8 ▼     else:
9         fee = 50
10 ▼ else:
11 ▼     if (age < 18):
12         fee = 40
13 ▼     else:
14         fee = 100
15
16 print(f"Your admission cost is {fee}.")
17
```



Console

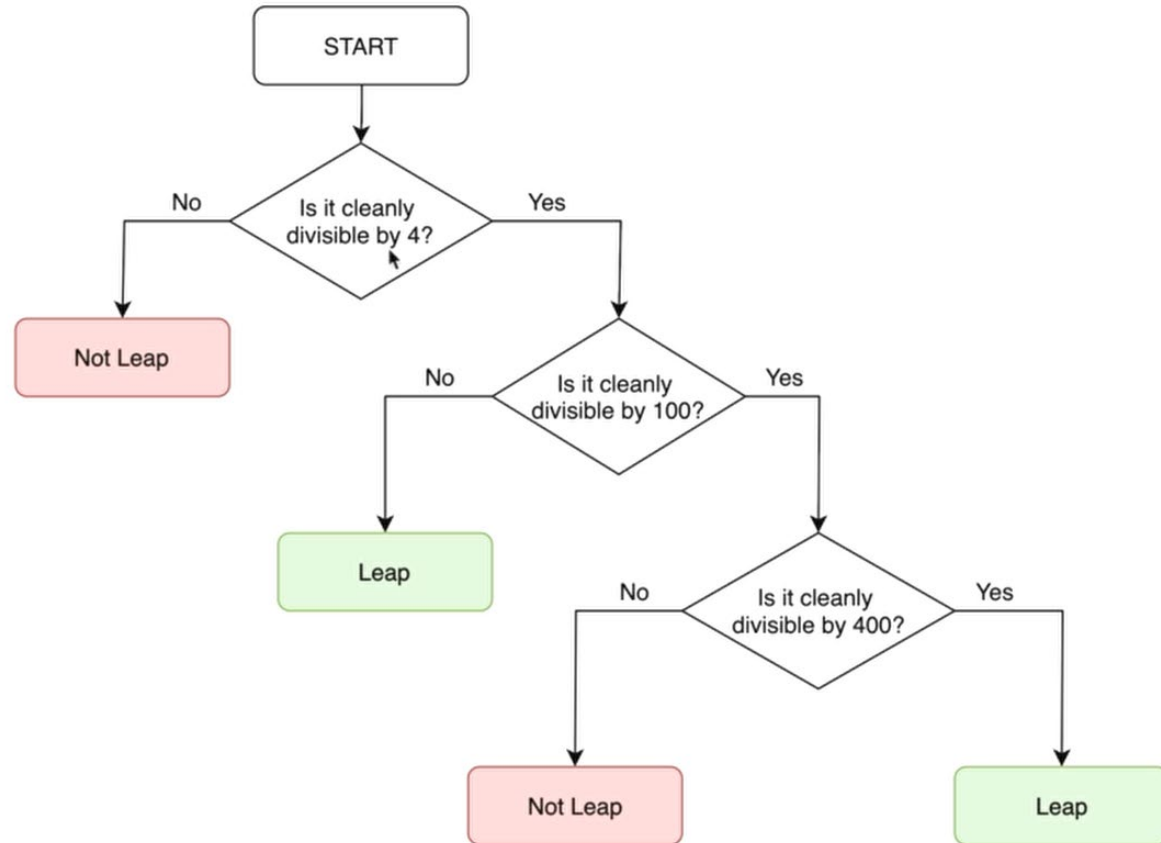
Shell

```
Thai citizen (Y/N) :N
Enter your age : 40
Your admission cost is 100.
> |
```



if statement

- ปีอธิกสุรทิน หมายถึง ปีที่เดือนกุมภาพันธ์มี 29 วัน คือปีหารด้วย 4 ลงตัว แต่ปีหารด้วย 100 ลงตัวจะมีไม่ใช่ปีอธิกสุรทิน แต่ยกเว้นปีหารด้วย 400 ลงตัวจะเป็น เช่น ค.ศ. 1600 และ 2000 เป็นปีอธิกสุรทิน แต่ ค.ศ. 1700, 1800 และ 1900 ไม่ใช่





if statement

- สามารถเขียนเป็นโปรแกรมได้ดังนี้

main.py ×

```
1 year = int(input("Which year do you want to check? "))
2
3 ▼ if year % 4 != 0:
4     print("Not leap year")
5 ▼ elif year % 100 != 0:
6     print("Leap year")
7 ▼ elif year % 400 != 0:
8     print("Not leap year")
9 ▼ else:
10    print("Leap year")
11
```

- จะเห็นว่าหากเราเขียน Flowchart แล้วค่อยนำมาเขียนโปรแกรมจะทำให้เขียนง่ายขึ้น



if statement

- มีการทำงานหนึ่งในการเขียนโปรแกรม เรียกว่า Refactor คือการปรับปรุงโปรแกรมให้กะทัดรัดขึ้น เช่น โปรแกรมที่ผ่านมา จะมีคำสั่ง print() หลายครั้ง
- ให้นักศึกษาหาทางปรับปรุงโปรแกรมให้สั้นลง



if statement

- **Exercise 2.5** จงเขียนโปรแกรมตรวจสอบสถานะการโปร
— Probation = GPA ก่อนหน้า > 2.0 และ GPS เทอมปัจจุบัน < 2.0
— หรือ GPA ก่อนหน้า < 2.0 แต่ GPS เทอมปัจจุบัน ≥ 2.0 แต่ GPA ปัจจุบัน ก็ยัง < 2.0
— ให้ลองเขียน if statement ที่บอกว่าโปรหรือไม่ หรือ ผ่าน หรือ retire
มี 3 คำตอบ คือ pass probation retire



if statement

- **Exercise 2.6** จงเขียนโปรแกรม ทอนเงิน เพื่อให้มีจำนวนน้อยที่สุด
 - กำหนดจำนวนเงินทอน ไม่เกิน 500 บาท
 - มีธนบัตร 100 50 20
 - เหรียญ 10 5 1
 - ให้แสดงว่าเงินทอนที่ Input ต้องทอนธนบัตร หรือ เหรียญ จำนวนเท่าไร
- ตัวอย่าง

Enter price: 123

Enter money: 500

Change : 100 x 3 ,50 x 1 ,20 x 1 ,5 x 1 ,1 x 2



if statement

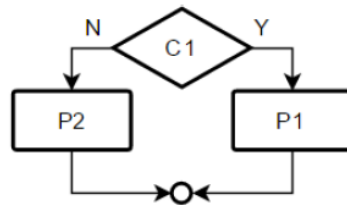
- สรุปตัวอย่างรูปแบบของ if เทียบกับ flowchart

<pre>graph TD; C1{C1} -- Y --> P1[P1]; C1 -- N --> Exit(()); P1 --> Exit;</pre>		<pre>if C1 : P1</pre>	
<pre>graph TD; C1{C1} -- Y --> Exit(()); C1 -- N --> P1[P1]; P1 --> Exit;</pre>	<pre>graph TD; C1{not C1} -- Y --> P1[P1]; C1 -- N --> Exit(()); P1 --> Exit;</pre>	<pre>if C1 : pass else : P1</pre>	<pre>if not C1 : P1</pre>

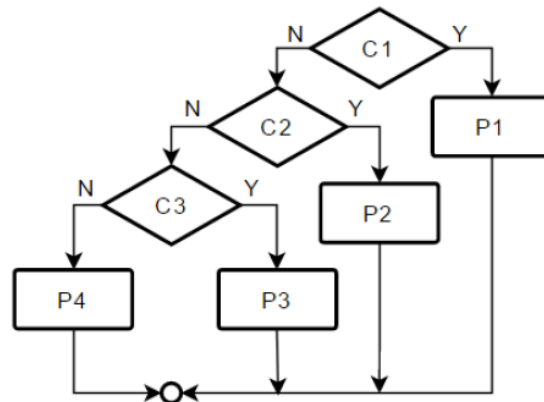
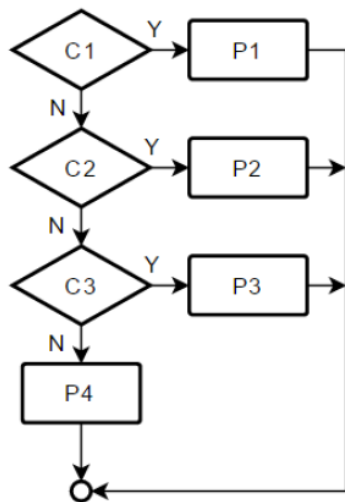


if statement

- สรุปรูปแบบอย่างรูปแบบของ if เทียบกับ flowchart



```
if C1 :  
    P1  
else :  
    P2
```

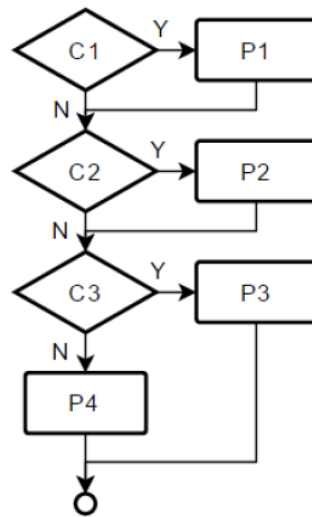


```
if C1 :  
    P1  
elif C2 :  
    P2  
elif C3 :  
    P3  
else :  
    P4
```



if statement

- สรุปตัวอย่างรูปแบบของ if เทียบกับ flowchart

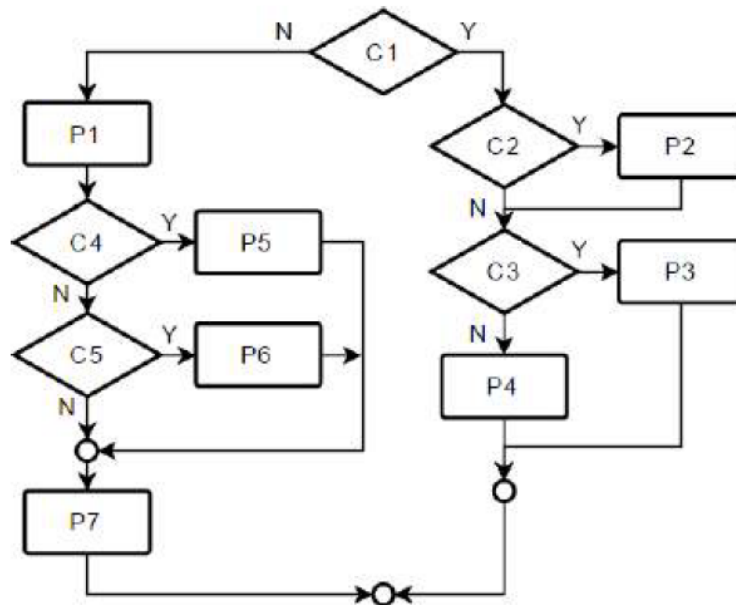


```
if C1 :  
    P1  
if C2 :  
    P2  
if C3 :  
    P3  
else :  
    P4
```



if statement

- สรุปตัวอย่างรูปแบบของ if เทียบกับ flowchart



```
if C1 :  
    if C2 :  
        P2  
    if C3 :  
        P3  
    else :  
        P4  
else :  
    P1  
    if C4 :  
        P5  
    elif C5 :  
        P6  
    P7
```



การรับ Input ครึ่งละหลายค่า v.2

- ในการรับ Input ครึ่งละหลายค่า มีวิธีการเพิ่มอีก 1 วิธี วิธีนี้เรียกว่า List Comprehension แต่จะอธิบายรายละเอียดภายหลัง
- ให้จำรูปแบบการใช้งานไปก่อน ดังนี้

`p1,p2,p3,n1,n2 = [int(e) for e in input("Input : ").split()]`

- จากตัวอย่างเป็นการ Input ข้อมูลจำนวน 5 ตัว โดยป้อนในบรรทัดเดียว และ คั่นด้วยช่องว่าง



Comment

- บางครั้งเราต้องการจะใส่ข้อความอื่นๆ ที่ไม่ใช่โปรแกรม เราสามารถทำได้โดยใช้ Comment
- comment แบบ 1 บรรทัด ขึ้นต้นด้วย #
Say hello to everyone.
print("Hello Python people!")
- comment แบบหลายบรรทัด ขึ้นต้นด้วย """



if statement

- โจทย์ประเภท if บางครั้งอาจมีความซับซ้อน เช่น โจทย์นี้
 - หากเราซื้อสลากกินแบ่งเรียงหมายเลขตั้งแต่หมายเลข $n1$ ต่อเนื่องไปจนถึงหมายเลข $n2$ (เช่น หมายเลข 10300 ถึง 13999) และงวดนี้รางวัลที่ 1 คือหมายเลข $p1$ เลขท้ายสองตัวคือหมายเลข $p2$ และ เลขท้ายสามตัวคือหมายเลข $p3$ เราจะได้รางวัลรวมเป็นเงินเท่าไร
 - กำหนดให้สลากกินแบ่งที่ขายนี้เป็นรุ่นพิเศษ เป็นเลข 5 หลัก รางวัลที่หนึ่ง 10,000 บาท หนึ่งรางวัล รางวัลเลขท้ายสองตัวหนึ่งหมายเลข 25 บาท และรางวัลเลขท้ายสามตัวหนึ่งหมายเลข 100 บาท
 - Input เป็น จำนวนเต็ม 5 จำนวน $p1$ $p2$ $p3$ $n1$ $n2$ คั่นด้วยช่องว่าง โดย $p1$ คือรางวัลที่ 1 $p2$ คือ เลขท้าย 2 ตัว $p3$ คือ เลขท้าย 3 ตัว



if statement

- วิเคราะห์โจทย์

- รางวัลที่ 1 มีหมายเลขเดียว วิธีการตรวจสอบ คือ ตรวจสอบว่า รางวัลที่ 1 อยู่ในช่วงหมายเลขที่ซื้อหรือไม่
- ในการทดสอบ เราจะต้องทำ Test Case เพื่อให้แน่ใจว่าโปรแกรมที่เขียนขึ้นครอบคลุมทุกกรณี โดยข้อมูลที่จะตรวจสอบมีดังนี้
 - Input เป็น จำนวนเต็ม 5 จำนวน p1 p2 p3 n1 n2 คั่นด้วยช่องว่าง
 - กรณี 1 รางวัลอยู่ในช่วง ใบที่ซื้อ 12345 00 000 12000 13000 -> 10000
 - กรณี 2 รางวัลอยู่ที่ขอบบน 12345 00 000 12345 13000 -> 10000
 - กรณี 3 รางวัลอยู่ที่ขอบล่าง 12345 00 000 12000 12345 -> 10000
 - กรณี 4 กรณีซื้อใบเดียวและถูก 12345 00 000 12345 12345 -> 10000
 - กรณี 5 กรณีไม่ถูก นอกช่วงซ้าย 12345 00 000 12346 14000 -> 0
 - กรณี 6 กรณีไม่ถูก นอกช่วงขวา 12345 00 000 12000 12344 -> 0



if statement

- ในการเขียนโปรแกรม กรณีที่โจทย์สามารถแบ่งเป็นส่วนๆ ได้ ให้นำทยอยเขียนทีละส่วน แล้วทดสอบให้ถูกต้องก่อน กรณีนี้จะเริ่มจากทดสอบรางวัลที่ 1 เขียนดังนี้

main.py ×

```
1  ## สนใจเฉพาะรางวัลที่ 1
2  ## ข้อมูลที่จะตรวจสอบ
3  # 12345 00 000 12000 13000 -> 10000
4  # 12345 00 000 12345 13000 -> 10000   ขอบบน
5  # 12345 00 000 12000 12345 -> 10000   ขอบล่าง
6  # 12345 00 000 12345 12345 -> 10000   กรณีมีใบเดียว
7  # 12345 00 000 12346 14000 -> 0       นอกช่วงซ้าย
8  # 12345 00 000 12000 12344 -> 0       นอกช่วงขวา
9
10 p1,p2,p3,n1,n2 = [int(e) for e in input("Input : ").split()]
11 s = 0
12 ▼ if n1 <= p1 <= n2 :
13     s += 10000
14 print(s)
```



if statement

- วิเคราะห์โจทย์ กรณีเลขท้าย 2 ตัว จะมี 2 กรณี
 - กรณีที่น้อยกว่า 100 ใบ จะมีโอกาสถูกได้ 1 รางวัล
 - กรณีที่มากกว่า 100 ใบ จะมีโอกาสถูกได้มากกว่า 1 รางวัล
- เราจะเริ่มจากสร้าง test case
 - กรณี ซื้อ 100 ใบ 00000 50 000 10000 10099 -> 25
 - กรณี ซื้อ 300 ใบ 00000 50 000 10000 10299 -> 75
 - กรณี > 100 ใบ แต่เลขที่ออกอยู่ก่อนใบแรก
00000 50 000 10060 10680 -> 150
 - กรณี > 100 ใบ แต่เลขที่ออกอยู่หลังใบแรก
00000 50 000 10060 10680 -> 175



if statement

- เฉพาะกรณีน้อยกว่า 100 ไบ

main.py ×

```
1  """
2  ## สนใจเฉพาะเลขท้าย 2 ตัว
3  ## ข้อมูลที่จะตรวจสอบ
4  # 00000 50 000 10000 10099 -> 25
5  ! 00000 50 000 10000 10199 -> 50
6  ! 00000 50 000 10000 10299 -> 75
7  """
8  p1,p2,p3,n1,n2 = [int(e) for e in input("Input : ").split()]
9  s = 0
10 #if n1 <= p1 <= n2 :
11 #     s += 10000
12 ▼ if n1%100 <= p2 <= n2%100 :
13     s += 25
14 print(s)
```



if statement

- เลขท้าย 2 ตัวทุกกรณี พบว่ากรณีสุดท้ายจะผิด

main.py ×

```
1  """
2  # สนใจเฉพาะเลขท้าย 2 ตัว
3  # ข้อมูลที่จะตรวจสอบ
4  00000 50 000 10000 10099 -> 25      100 ใบ
5  00000 50 000 10000 10199 -> 50      200 ใบ
6  00000 50 000 10000 10299 -> 75      300 ใบ
7  00000 50 000 10060 10680 -> 150
8  """
9  p1,p2,p3,n1,n2 = \
10 [int(e) for e in input("Input : ").split()]
11 s = 0
12 #if n1 <= p1 <= n2 :
13 #    s += 10000
14 ▼ if n1//100 == n2//100 :                # กรณี 3 หลักแรกเท่ากัน
15 ▼     if n1%100 <= p2 <= n2%100 :        # ถ้าเลขอยู่ในช่วงจะถูก 1 ครั้ง
16         s += 25
17 ▼ else:                                # กรณี 3 หลักแรกไม่เท่ากัน
18     s += 25*(n2//100 - n1//100 + 1)
19 print(s)
```

Input : 00000 50 000 10060 10680
175
➤



if statement

- เลขท้าย 2 ตัวทุกกรณี version 2 โค้ดทำงานถูกต้อง แต่ค่อนข้างยาว

main.py ×

```
1  """
2  ## สนใจเฉพาะเลขท้าย 2 ตัว
3  ## ข้อมูลที่จะตรวจสอบ
4  # 00000 50 000 10000 10099 -> 25
5  # 00000 50 000 10000 10199 -> 50
6  # 00000 50 000 10000 10299 -> 75
7  # 00000 50 000 10060 10680 -> 150
8  """
9  p1,p2,p3,n1,n2 = [int(e) for e in input("Input : ").split()]
10 s = 0
11 #if n1 <= p1 <= n2 :
12 #     s += 10000
13▼ if n1//100 == n2//100 :                                # กรณี 3 หลักแรกเท่ากัน
14▼     if n1%100 <= p2 <= n2%100 :                        # ถ้าเลขอยู่ในช่วงจะถูก 1 ครั้ง
15         s += 25
16▼ else:                                                  # กรณี 3 หลักแรกไม่เท่ากัน
17▼     if n1%100 <= p2 :                                    # 1) เลข 2 หลักหลังใบแรก <= เลขท้าย 2 ตัว
18         s += 25                                         # บวก 25
19▼     if p2 <= n2%100 :                                    # 2) เลข 2 หลักหลังใบสุดท้าย >= เลขท้าย 2 ตัว
20         s += 25                                         # บวก 25
21         s += 25*((n2//100-1) - \
22                 (n1//100+1) + 1)                        # ส่วนที่อยู่เต็ม 100 ใบ
23 print(s)
```




if statement

- เลขท้าย 2 ตัวทุกกรณี version 3 ทำการ refactor ให้ code สั้นลง

main.py ×

```
1  """
2  ## สนใจเฉพาะเลขท้าย 2 ตัว
3  ## ข้อมูลที่จะตรวจสอบ
4  # 00000 50 000 10000 10099 -> 25
5  # 00000 50 000 10000 10199 -> 50
6  # 00000 50 000 10000 10299 -> 75
7  # 00000 50 000 10060 10680 -> 150
8  """
9  p1,p2,p3,n1,n2 = [int(e) for e in input("Input : ").split()]
10 s = 0
11 #if n1 <= p1 <= n2 :
12 #     s += 10000
13 ▼ if n1%100 <= p2 :                                # 1) เลข 2 หลักหลังใบแรก <= เลขท้าย 2 ตัว
14     s += 25                                         # บวก 25
15 ▼ if p2 <= n2%100 :                                # 2) เลข 2 หลักหลังใบสุดท้าย >= เลขท้าย 2 ตัว
16     s += 25                                         # บวก 25
17 s += 25*((n2//100-1) - \
18         (n1//100+1) + 1)                            # ส่วนที่อยู่เต็ม 100 ใบ
19 print(s)
20
```



if statement

- วิเคราะห์โจทย์ กรณีเลขท้าย 3 ตัว จะคล้ายกับเลขท้าย 2 ตัว

```
main.py x
1  """
2  ## สนใจเฉพาะเลขท้าย 3 ตัว
3  ## ข้อมูลที่จะตรวจสอบ
4  01234 11 811 01000 01250 -> 10075
5  99999 99 999 99950 99999 -> 10125
6  19999 13 001 09015 13000 -> 1275
7  """
8  p1,p2,p3,n1,n2 = [int(e) for e in input("Input : ").split()]
9  s = 0
10 ▼ if n1 <= p1 <= n2 :
11     s += 10000          # รางวัลที่ 1
12 ▼ if n1%100 <= p2 :      # เลขท้าย 2 ตัว
13     s += 25
14 ▼ if p2 <= n2%100 :
15     s += 25
16 s += 25*(n2//100 - n1//100 - 1)
17 ▼ if n1%1000 <= p3 :      # เลขท้าย 3 ตัว
18     s += 100
19 ▼ if p3 <= n2%1000 :
20     s += 100
21 s += 100*(n2//1000 - n1//1000 - 1)
22 print(s)
23
```



Random

- บางครั้งเรามีความจำเป็นต้องนำเข้าโปรแกรมจากไฟล์อื่น ที่ผู้อื่นเขียน หรือตนเองเขียน
- ภาษา Python มี Library จำนวนมาก <https://docs.python.org/3/library/>
- ในการเรียกใช้ Library เหล่านั้น เราจะใช้คำสั่ง import
- เช่น หากเราต้องการสุ่ม (random) หัว ก้อย เราต้องเขียนโปรแกรมดังนี้

```
main.py x
1 import random
2
3 random_side = random.randint(0,1)
4 # คำสั่ง randint(start,stop) จะสุ่มจำนวนเต็มตั้งแต่ start - stop
5 ▼ if random_side == 1:
6     print("Heads")
7 ▼ else:
8     print("Tails")
9
```

Console Shell

Tails



Import

- การ Import มี 2 รูปแบบดังนี้
- **Import มาทั้งหมด** วิธีการเขียน ก็จะเหมือนกับ Slide ก่อนหน้า โดยวิธีการอ้าง function ที่จะเรียกใช้ จะต้องขึ้นต้นด้วยชื่อ Library. เสมอ เช่น random.randint()
- Import เฉพาะ function วิธีการเขียนจะมีดังนี้ (ไม่ต้องอ้างชื่อ Library)

```
main.py ×
1 from random import randint
2
3 random_side = randint(0,1)
4 # คำสั่ง randint(start,stop) จะสุ่มจำนวนเต็มตั้งแต่ start - stop
5 ▼ if random_side == 1:
6     print("Heads")
7 ▼ else:
8     print("Tails")
9
```

Console Shell

Heads
▶



Ternary Operation

- ในภาษา Python มีรูปแบบย่อของ if else เรียกว่า Ternary Operation

```
[on_true] if [expression] else [on_false]
```

- การทำงานก็เหมือนกับ if else ปกติ แต่ข้อดี คือ สามารถเขียนได้ใน 1 บรรทัด

```
# Program to demonstrate conditional operator
a, b = 10, 20

# Copy value of a in min if a < b else copy b
min = a if a < b else b

print(min)
```

- เหมาะสำหรับการใช้กับเงื่อนไขง่ายๆ



if statement

- **Exercise 2.7** จงเขียนโปรแกรมเป่ายิงฉุบ

โดยให้รับข้อมูลจาก User จำนวน 1 ตัวเป็น 0) Rock, 1) Paper, 2) Scissors (ให้ป้อน 0 หรือ 1 หรือ 2) จากนั้นให้คอมพิวเตอร์สุ่มเลือกอีกข้าง แล้วแสดง character art ออกมา จากนั้น ดูว่าใครชนะ ถ้าผู้เล่นชนะ ให้แสดง **You win!!!** ถ้าแพ้ให้แสดง **You lose TT** ถ้าเสมอ ให้แสดง **It's a draw**

ปล. เนื่องจากระบบไม่สามารถตรวจสอบผลการสุ่มได้ ดังนั้นในการส่งให้เขียนโปรแกรมรับค่าแทนการสุ่มของคอมพิวเตอร์ด้วย



if statement

```
What do you choose? Type 0 for Rock, 1 for Paper or 2 for Scissors.  
1
```

```
-----  
--- '      )  
      )  
      )  
      )  
      )  
--- .      )
```

```
Computer choose:
```

```
-----  
--- '      )  
      (      )  
      (      )  
      (      )  
      (      )  
--- .__ (      )
```

```
You win!
```



For your attention