

w04-Lab2

# Conditionals

## Part II

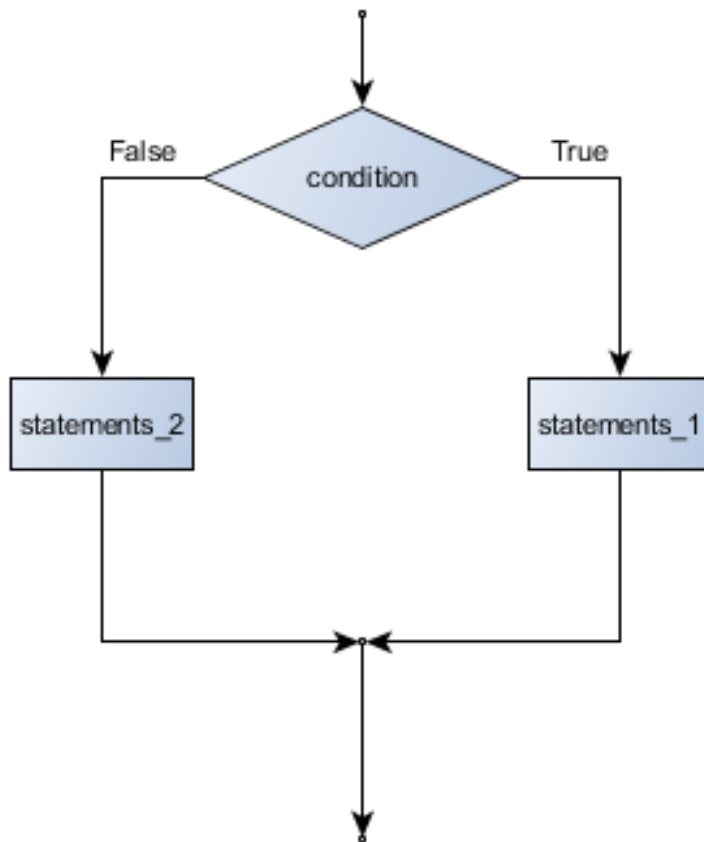
for 204111

Kittipitch Kuptavanich

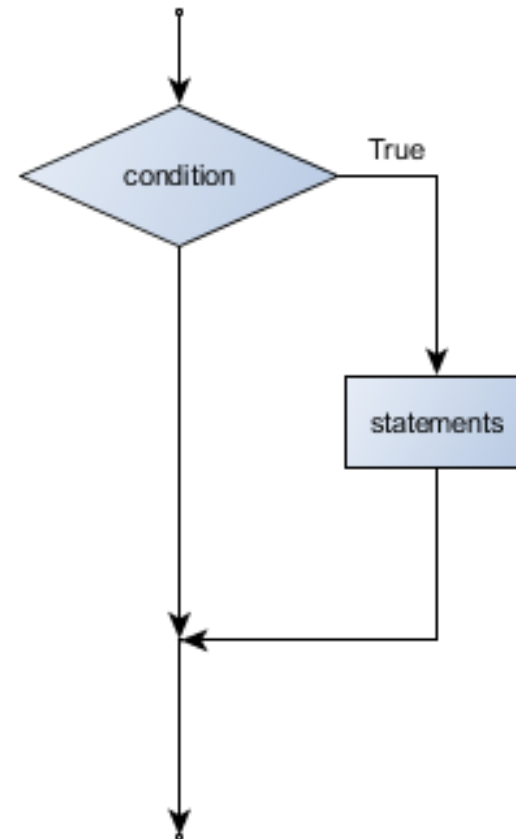
# Conditional Executions

- Basic form:

(if - else)

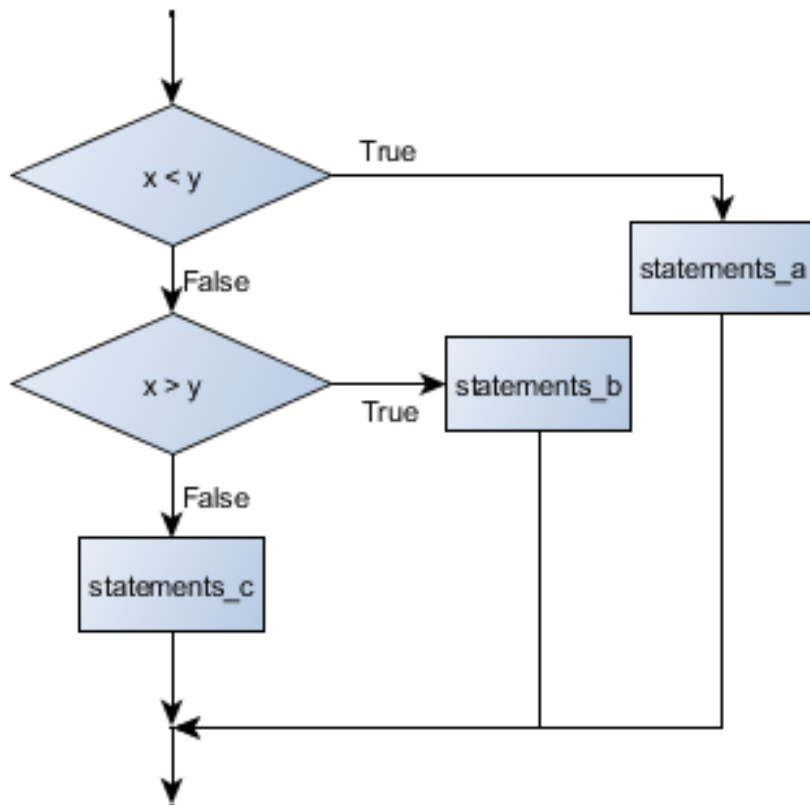


- Omitting else

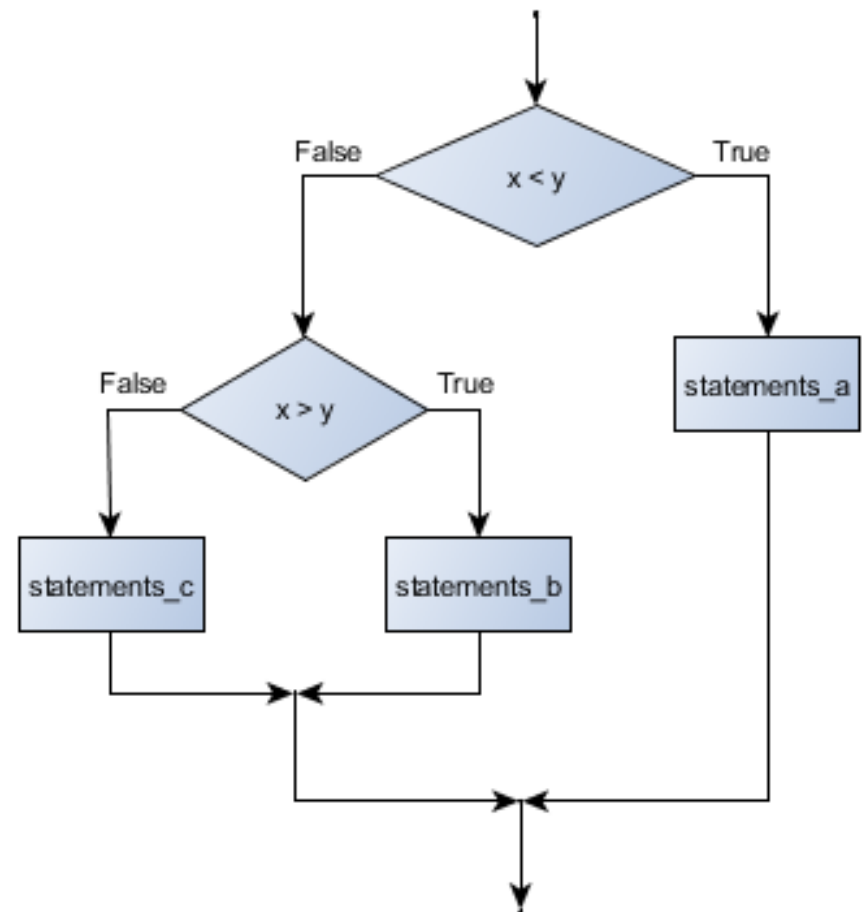


# Conditional Executions [2]

- Chain Conditionals:  
(if - elif - else)



- Nested Conditionals:

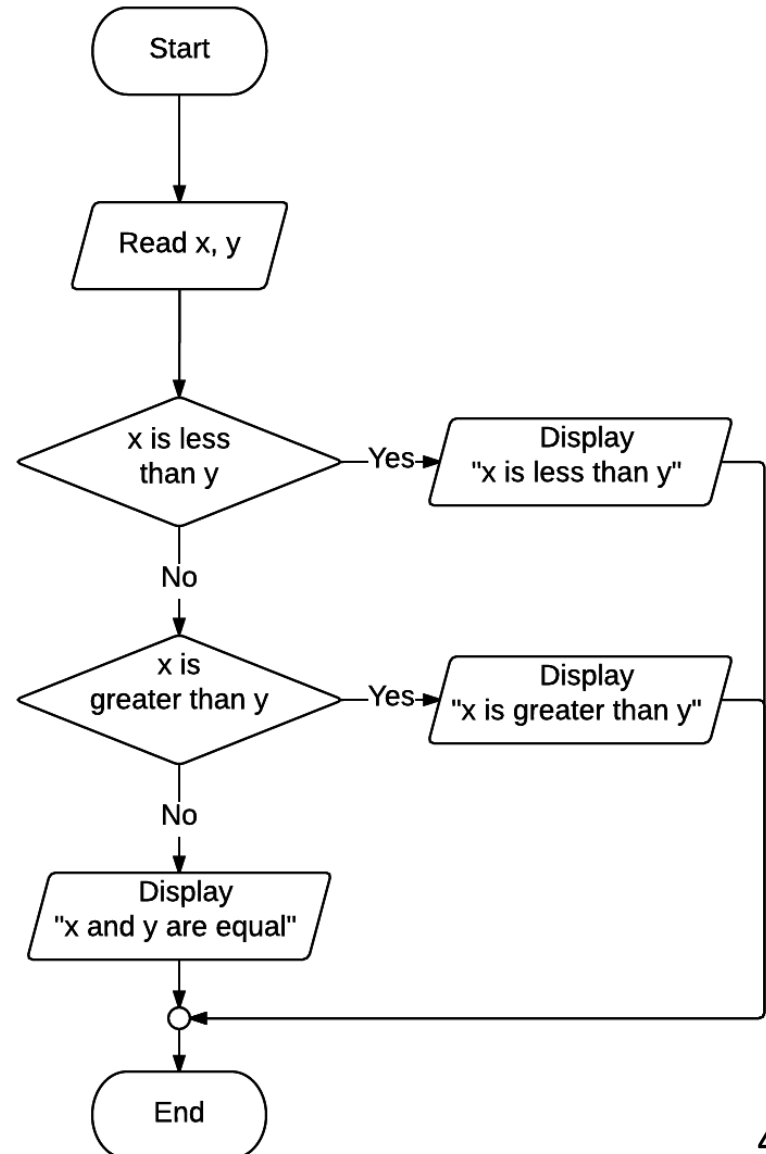


# Chained Conditionals [Recap]

- ในบางกรณี ทางเลือกที่เป็นไปได้ อาจมีมากกว่า 2 ทาง เราสามารถใช้ **chained condition** (**if** - **elif** - **else**) เพื่อรองรับเงื่อนไขการตัดสินใจในลักษณะนี้
- **elif** คือตัวย่อของ "else if"
- จากตัวเลือกที่เป็นไปได้ ทั้งหมด ชุดคำสั่งเพียง 1 ชุดเท่านั้น ที่จะถูกดำเนินการ

```

if Boolean expression:
    block of code
elif Boolean expression:
    block of code
else:
    block of code
  
```



# Example 1: Commission

- **Problem Statement:**

- ในบริษัทแห่งหนึ่งมีการคำนวณเงินค่าตอบแทนพนักงานขายต่อเดือนตามตารางด้านล่าง ให้เขียน function `cal_output(sales)` เพื่อคำนวณรายได้ตามตาราง

Monthly Sales	Income
greater than or equal to \$50,000	\$575 plus 16% of sales
less than \$50,000 but greater than or equal to \$40,000	\$550 plus 14% of sales
less than \$40,000 but greater than or equal to \$30,000	\$525 plus 12% of sales
less than \$30,000 but greater than or equal to \$20,000	\$500 plus 9% of sales
less than \$20,000 but greater than or equal to \$10,000	\$450 plus 5% of sales
less than \$10,000	\$400 plus 3% of sales

# Example 1: Commission [2]

```
def calc_output(sales):
    if sales >= 50000:
        pass
    elif 50000 > sales >= 40000:
        pass
    elif 40000 > sales >= 30000:
        pass
    elif 30000 > sales >= 20000:
        pass
    elif 20000 > sales >= 10000:
        pass
    elif 10000 > sales:
        pass
```



```
def calc_output(sales):
    if sales >= 50000:
        pass
    elif sales >= 40000:
        pass
    elif sales >= 30000:
        pass
    elif sales >= 20000:
        pass
    elif sales >= 10000:
        pass
    else:
        pass
```

- Some part of the Boolean expressions can be omitted to form simpler expressions.
- Why?

# Example 1: Commission [2]

```
def calc_output(sales):  
    if sales >= 50000:  
        output = _____  
    elif sales >= 40000:  
        output = _____  
    elif sales >= 30000:  
        output = _____  
    elif sales >= 20000:  
        output = _____  
    elif sales >= 10000:  
        output = _____  
    else:  
        output = _____  
  
    return output
```

# Example 2: Letter Grades

```
score = 85
```

```
if score >= 90:
```

```
    grade = 'A'
```

```
if score >= 80:
```

```
    grade = 'B'
```

```
if score >= 70:
```

```
    grade = 'C'
```

```
if score >= 60:
```

```
    grade = 'D'
```

```
else:
```

```
    grade = 'F'
```

Correct?

```
print("Score::0, Grade::1".format(score, grade))
```



# Example 2: Letter Grades [2]

```
score = 85
```

```
if score >= 90:  
    grade = 'A'  
elif score >= 80:  
    grade = 'B'  
elif score >= 70:  
    grade = 'C'  
elif score >= 60:  
    grade = 'D'  
else:  
    grade = 'F'
```

```
print("Score::0, Grade::1".format(score, grade))
```

# Nested Conditionals

- ในบางกรณี เราสามารถใช้ **Basic Conditionals**, **Nested Conditionals** หรือ **Chained Conditionals** เพื่อแก้ปัญหาเดียวกันได้ (เช่น max-mid-min)
- อย่างไรก็ตามการเขียนโปรแกรม **Nested Condition** (**if** ซ้อน **if**) นั้น มีลักษณะโครงสร้างที่อ่านและเข้าใจได้ยากกว่า **Conditional Statement** ลักษณะอื่น
  - ควรหลีกเลี่ยงหากเป็นไปได้

# Nested Conditionals [2]

- เราสามารถนำ **Logical Operators** มาประยุกต์ใช้เพื่อหลีกเลี่ยงการใช้ **Nested Condition** ได้เช่น

```
if 0 < x:
    if x < 10:
        print("x is a positive single digit number.")
```

- จะเห็นได้ว่าฟังก์ชัน `print()` จะถูกเรียกใช้ก็ต่อเมื่อเงื่อนไขใน `if` เป็นจริงทั้ง 2 กรณี
- เราสามารถใช้ **and Operator** เพื่อสร้าง **Expression** ใหม่

```
if (0 < x) and (x < 10):      # 0 < x < 10
    print("x is a positive single digit number.")
```

# Logical Opposites

- ใน Relational Operator แต่ละตัว จะมีเครื่องหมายตรงกันข้ามอยู่ ตัวอย่างเช่น
- ในประเทศไทย ผู้ที่ต้องการทำใบขับขี่ จะต้องมีความมากกว่าหรือเท่ากับ 18 ปี ( $\geq 18$ )
  - ไม่สามารถทำใบขับขี่ได้หากมีอายุต่ำกว่า 18 ปี ( $< 18$ )
  - สังเกตว่า เครื่องหมายตรงกันข้ามของ  $\geq$  คือ  $<$

operator	logical opposite
$==$	$!=$
$!=$	$==$
$<$	$>=$
$<=$	$>$
$>$	$<=$
$>=$	$<$

# Logical Opposites [2]

- พิจารณาชุดคำสั่ง

```
if not (age >= 18):  
    print("You're too young to get a driving license!")
```

- สามารถ simplify ได้เป็น

```
if age < 18:                                # อ่านและเข้าใจได้ง่ายกว่า  
    print("You're too young to get a driving license!")
```

# De Morgan's Laws

- เช่นเดียวกันกับใน Boolean Algebra เราสามารถนำ De Morgan's Laws มาใช้เพื่อ Simplify ประโยคเงื่อนไข (Conditional Statements) ได้ดังนี้

$$\sim(a \wedge b) = \sim a \vee \sim b$$

$$\sim(a \vee b) = \sim a \wedge \sim b$$



$$\text{not } (x \text{ and } y) == (\text{not } x \text{ or } \text{not } y)$$

$$\text{not } (x \text{ or } y) == (\text{not } x \text{ and } \text{not } y)$$

# De Morgan's Laws [2]

## Example:

- วัยเรียน (หรือนักศึกษา) คือคนที่มีอายุในช่วง 6 ปีขึ้นไป จนถึง 21 ปี
  - $6 \leq \text{age} \leq 21 \rightarrow \text{student}$
  - $(\text{age} \geq 6) \text{ and } (\text{age} \leq 21) \rightarrow \text{student}$
- คนที่มีอายุ น้อยกว่า 6 ปี หรือ มากกว่า 75 ปี ไม่ถือเป็นวัยทำงาน
  - $\text{age} < 6 \text{ หรือ } \text{age} > 75 \rightarrow \text{not professional}$
  - $(\text{age} < 6) \text{ or } (\text{age} > 75) \rightarrow \text{not professional}$
  - $\text{not } ((\text{age} < 6) \text{ or } (\text{age} > 75)) \rightarrow \text{professional}$

# De Morgan's Laws [3]

## Example (contd.):

```
if (age >= 6) and (age <= 21):  
    print("student")  
  
elif not ((age < 6) or (age > 75)):  
    print("professional")
```

- เมื่อใช้ De Morgan's Law จะได้

```
if (age >= 6) and (age <= 21):  
    print("student")  
  
elif (age >= 6) and (age <= 75):  
    print("professional")
```





# Refactoring Conditions

```
if (age >= 6) and (age <= 21):
    print("student")

elif (age >= 6) and (age <= 75):
    print("professional")
```

- สามารถใช้กฎการกระจายเพื่อดึง (age >= 6) ออกมา

```
if age >= 6:
    if age <= 21:
        print("student")

    elif age <= 75:
        print("professional")
```

Readability?

**Note:** ตัวอย่างนี้เป็นไปเพื่อการแสดงการใช้สมบัติของ Boolean Algebra

# Refactoring Conditionals [2]

## Love6 Game Revisited:

- กำหนด integer 2 ตัว **first** และ **second** โปรแกรมจะแสดงผล **True** ก็ต่อเมื่อ
  - ตัวใดตัวหนึ่งมีค่าเท่ากับ 6
  - ผลบวกของทั้งสองตัวมีค่าเท่ากับ 6
  - ผลต่างของทั้งสองตัวมีค่าเท่ากับ 6

```
if ((first == 6) or (second == 6) or
    (first + second == 6) or
    abs(first - second) == 6):
    print("True")
else:
    print("False")
```

สังเกตการใช้วงเล็บ **()** กรณี Boolean Expression มีความยาวมากกว่า 1 บรรทัด

# Refactoring Conditionals [3]

## Love6 Game Revisited (2):

ในกรณีที่เงื่อนไขมีลักษณะเป็น **Mutually Exclusive** (เหตุการณ์ไม่เกิดร่วม) เราสามารถเปลี่ยน **or Statement** ให้เป็น **elif** ได้

```
if first == 6:
    print("True")
elif second == 6:
    print("True")
elif first + second == 6:
    print("True")
elif abs(first - second) == 6:
    print("True")
else:
    print("False")
```




# Tips

- ในกรณีโครงสร้างแบบ Chain (**elif**) ที่ Boolean Expression สามารถ evaluate เป็น **True** ได้มากกว่าหนึ่งเงื่อนไข เงื่อนไขแรกที่เป็น **True** เท่านั้นที่จะถูกดำเนินการ

```
x = 360

if x % 2 == 0:      # กรณีนี้เท่านั้นที่ถูกแสดงผลเมื่อ x มีค่าเท่ากับ 360
    print("2 divides x")
elif x % 3 == 0:
    print("3 divides x")
elif x % 5 == 0:
    print("5 divides x")
else:
    print("x is not divisible by 2 3 or 5")
```

# Example 1: Leap Year

- ปีอธิกสุรทินโดยปรกติแล้ว จะเป็นปีคริสต์ศักราช ที่หารด้วย 4 ลงตัว เช่นปี 2012, 2016 และ 2020 
- ข้อยกเว้น (ในกรณีปีหาร 4 ลงตัว)
  - ☒ ปีที่หารด้วย 100 ลงตัว ไม่ใช่ปีอธิกสุรทิน (1700, 1800, 1900) 
  - ☑ แต่ปีที่หารด้วย 400 ลงตัว เป็นปีอธิกสุรทิน (1600, 2000) 
- STEP1: สร้าง test case

	Test Case	output
Case 1	2012 2016 2020	YES
Case 2	1700 1800 1900	NO
Case 3	1600 2000 2400	YES
Case 4	2013 2014 2015	NO

# Example 1: Leap Year

- **STEP2:** พิจารณา condition แรก (case 1 และ 4) ก่อน
  - ปีคริสตศักราช ที่หารด้วย 4 ลงตัว เช่นปี 2012, 2016 และ 2020

```
if year % 4 == 0:
    print("YES")
else:
    print("NO")
```

	Test Case	output
<b>Case 1</b>	2012 2016 2020	YES
<b>Case 2</b>	1700 1800 1900	NO
<b>Case 3</b>	1600 2000 2400	YES
<b>Case 4</b>	2013 2014 2015	NO

# Example 1: Leap Year [2]

- **STEP3:** พิจารณา ใส่ข้อยกเว้นแรก (Case 2)
  - ข้อยกเว้น (ในกรณีปีหาร 4 ลงตัว)
    - ☒ ปีที่หารด้วย 100 ลงตัว ไม่ใช่ปีอธิกสุรทิน (1700, 1800, 1900)

```
if year % 4 == 0:
    print("YES")
```

```
else:
    print("NO")
```

	Test Case	output
Case 1	2012 2016 2020	YES
Case 2	1700 1800 1900	NO
Case 3	1600 2000 2400	YES
Case 4	2013 2014 2015	NO

# Example 1: Leap Year [3]

- **STEP3:** พิจารณา ใส่ข้อยกเว้นแรก (Case 2)
  - ข้อยกเว้น (ในกรณีปีหาร 4 ลงตัว)
    - ☒ ปีที่หารด้วย 100 ลงตัว ไม่ใช่ปีอธิกสุรทิน (1700, 1800, 1900)

```

if year % 4 == 0:    //ต้องเพิ่มเป็น nested เพราะอยู่เป็นกรณีย่อยของ case 1
    if year % 100 == 0:
        print("NO")
    else:
        print("YES")
else:
    print("NO")

```

	Test Case	output
Case 1	2012 2016 2020	YES
Case 2	1700 1800 1900	NO
Case 3	1600 2000 2400	YES
Case 4	2013 2014 2015	NO



# Example 1: Leap Year [4]

- **STEP3:** พิจารณา ใส่ข้อยกเว้นที่สอง (Case 3)
  - ข้อยกเว้น (ในกรณีปีหาร 4 ลงตัว)
    - ☒ ปีที่หารด้วย 100 ลงตัว ไม่ใช่ปีอธิกสุรทิน (1700, 1800, 1900)
    - ☑ แต่ปีที่หารด้วย 400 ลงตัว เป็นปีอธิกสุรทิน (1600, 2000)

```
if year % 4 == 0:
    if year % 100 == 0:
        print("NO")
```

```
    else:
        print("YES")
```

```
else:
    print("NO")
```

	Test Case	output
Case 1	2012 2016 2020	YES
Case 2	1700 1800 1900	NO
Case 3	1600 2000 2400	YES
Case 4	2013 2014 2015	NO

# Example 1: Leap Year [5]

- **STEP3:** พิจารณา ใส่ข้อยกเว้นที่สอง (Case 3)
  - ข้อยกเว้น (ในกรณีปีหาร 4 ลงตัว)
    - ☒ ปีที่หารด้วย 100 ลงตัว ไม่ใช่ปีอธิกสุรทิน (1700, 1800, 1900)
    - ☑ แต่ปีที่หารด้วย 400 ลงตัว เป็นปีอธิกสุรทิน (1600, 2000)

```

if year % 4 == 0:
    if year % 100 == 0:
        print("NO")
    elif year % 400 == 0:
        print("YES")
    else:
        print("YES")
else:
    print("NO")

```

	Test Case	output
Case 1	2012 2016 2020	YES
Case 2	1700 1800 1900	NO
Case 3	1600 2000 2400	YES
Case 4	2013 2014 2015	NO

# Example 1: Leap Year [6]

## • STEP5: Testing

```

if year % 4 == 0:
    if year % 100 == 0:
        print("NO")
    elif year % 400 == 0:
        print("YES")
    else:
        print("YES")
else:
    print("NO")

```

วิธีแก้: สลับตำแหน่งเงื่อนไข

- Year = 2400 ได้ output = ?
- Why?
- จะแก้ bug อย่างไร?

	Test Case	output
Case 1	2012 2016 2020	YES
Case 2	1700 1800 1900	NO
Case 3	<u>1600 2000 2400</u>	YES
Case 4	2013 2014 2015	NO

# Example 1: Leap Year [7]

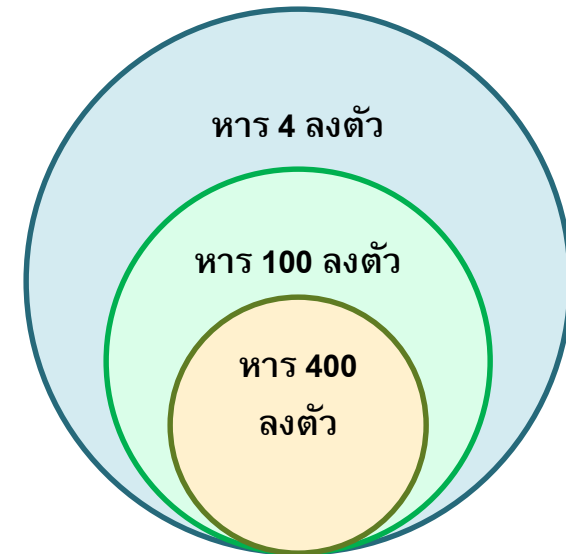
- STEP6: Reviewing**

```

if (year % 4 == 0):
    if (year % 400 == 0):
        print("YES")
    elif (year % 100 == 0):
        print("NO")
    else:
        print("YES")
else:
    print("NO")

```

กรณี condition เป็น subset ซึ่งกัน  
และกัน (ไม่แยกจากกันเด็ดขาด) ให้  
สร้าง condition จากกรณีที่ เจาะจง  
กว่าก่อน (เล็กไปหาใหญ่)



	Test Case	output
Case 1	2012 2016 2020	YES
Case 2	1700 1800 1900	NO
Case 3	<u>1600 2000 2400</u>	YES
Case 4	2013 2014 2015	NO

**Practice:** หากต้องการเปลี่ยนรูปแบบเป็น  
chain ล้วน ๆ แทน nested if จะทำอย่างไร

# References

- <http://openbookproject.net/thinkcs/python/english3e/conditionals.html>