Mahidol University

Wisdom of the Land

Mahidol University

# Lecture 03: Selections


Lecturers:

Aj. Jidapa Kraisangka

Aj. Akara Supratak

Aj. Tipajin Thaipisutikul

# Topic

- RECAP: Python Basics
- Flow Controls
- Boolean, Relational Expression and Logical Expression
- **`if`** statement
- **`else`** and **`elif`**

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

Mahidol University
Wisdom of the Land

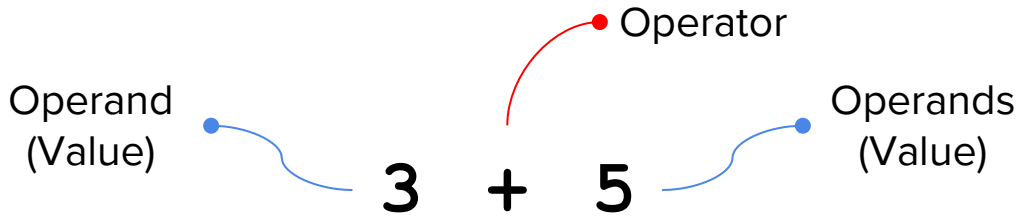
Mahidol University

# RECAP: Python Basics

# Expression



- Expression เป็นกลุ่มของค่าต่าง ๆ (values, e.g 3, 5) มากระทำต่อกันโดยใช้ตัวดำเนินการคำนวณ (operators, e.g. +)




- ผลจากการคำนวณจาก Expression จะเป็นค่าผลลัพธ์เพียงค่าเดียว

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

# Arithmetic Op. Precedence



**Table 1-1:** Math Operators from Highest to Lowest Precedence

| Operator | Operation | Example | Evaluates to . . . |
|---|---|---|---|
| ** | Exponent | 2 ** 3 | 8 |
| % | Modulus/remainder | 22 % 8 | 6 |
| // | Integer division/floored quotient | 22 // 8 | 2 |
| / | Division | 22 / 8 | 2.75 |
| * | Multiplication | 3 * 5 | 15 |
| - | Subtraction | 5 - 2 | 3 |
| + | Addition | 2 + 2 | 4 |

Table from https://automatetheboringstuff.com/2e/chapter1/

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

# Data Type เป็นการจำแนกประเภทของค่าข้อมูล

**Table 1-2:** Common Data Types

| Data type | Examples |
|---|---|
| Integers | -2, -1, 0, 1, 2, 3, 4, 5 |
| Floating-point numbers | -1.25, -1.0, -0.5, 0.0, 0.5, 1.0, 1.25 |
| Strings | 'a', 'aa', 'aaa', 'Hello!', '11 cats' |

ตัวเลข

ข้อความ

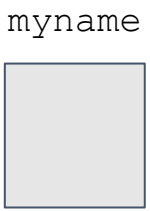
Table from https://automatetheboringstuff.com/2e/chapter1/

# Input from Users



- สามารถใช้คำสั่ง `input()` เพื่อรับค่า input จากผู้ใช้เพื่อเก็บค่านั้น ๆ ในตัวแปร

```
>>> myname = input()
_
```



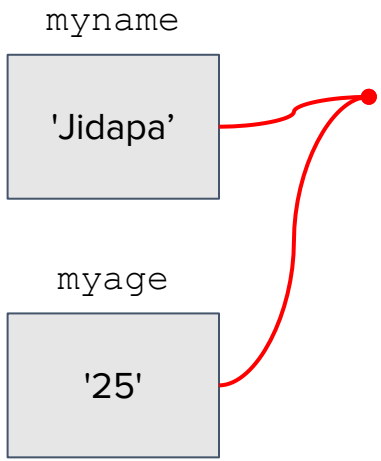

- `input()` **จะรอ** ข้อความจาก Keyboard จนกว่าผู้ใช้จะกด ENTER
- `myname` จะ **เก็บค่า** จาก Keyboard ในรูปแบบ String

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

# Example: `input()` and `print()`



```
>>> myname = input()
Jidapa
>>> print('Hello ' + myname)
Hello Jidapa


>>> myage = input()
25
>>> print(myage + 1)
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
TypeError: can only concatenate str
(not "int") to str
```




Note: Input จะถูก**เก็บค่า**ในรูปแบบ String ถึงแม้ว่ารูปข้อความจะเป็นตัวเลข

หากต้องการนำตัวเลขจาก input ไปใช้ จะ**ต้องเปลี่ยน**ประเภทของข้อมูลหรือ **Data type conversion (จาก `str` เป็น `int` หรือ `float`)** ก่อนการดำเนินการอื่น ๆ

# Data Type Conversion



- `int()` เป็นคำสั่งการเปลี่ยนประเภทข้อมูลจากค่าเดิมเป็น `int`
- `float()` เป็นคำสั่งการเปลี่ยนประเภทข้อมูลจากค่าเดิมเป็น `float`

```
>>> myage = input()
25
>>> print(int(myage) + 1)
26
>>> print(float(myage) + 1)
26.0

>>> '1234'+'5'
'12345'
>>> int('1234') + float('5')
1239.0
```

`int()` และ `float()` จะสามารถเปลี่ยนประเภทค่าข้อมูลได้ ก็ต่อเมื่อ String นั้น ๆ อยู่ในรูปแบบตัวเลข

** สามารถลองพิมพ์ `int('H')` เพื่อดูผลลัพธ์ได้

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

# Data Type Conversion (2)



- `str()` เป็นคำสั่งการเปลี่ยนประเภทข้อมูลจากค่าเดิมเป็น `String`

```
>>> myage = input()
25
>>> print(int(myage) + 1)
26
>>> print('Your age next year is ' + str(int(myage)+1))
Your age next year is 26

>>> str(100.5)
'100.5'
>>> print(str(100.5) + '124')
100.5124
```

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu



# Flow Controls

# Flow Controls



- **Flow controls** (in programming) เป็นลักษณะการเขียนโปรแกรม ที่มีลำดับการประมวลผลแตกต่างกันไป

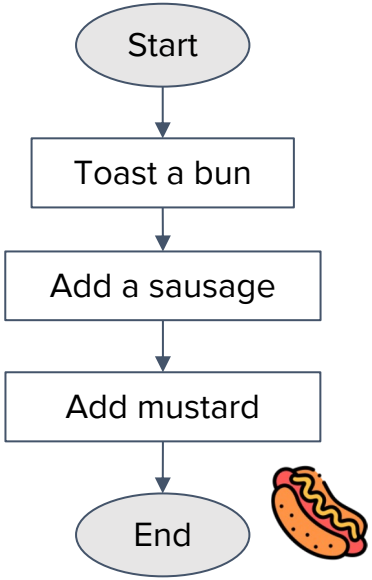
- รูปแบบของ Flow controls ได้แก่
  - Sequential: รูปแบบปกติที่เป็นลำดับขั้นตอนเรียงกัน
  - Selection: รูปแบบที่การเลือกทำชุดคำสั่งบางชุดตามเงื่อนไข
  - Repetition: รูปแบบปกติที่ทำซ้ำในบางชุดคำสั่ง
  - Invocation: รูปแบบการเรียกใช้ชุดคำสั่งอื่น ๆ หรือฟังก์ชัน

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

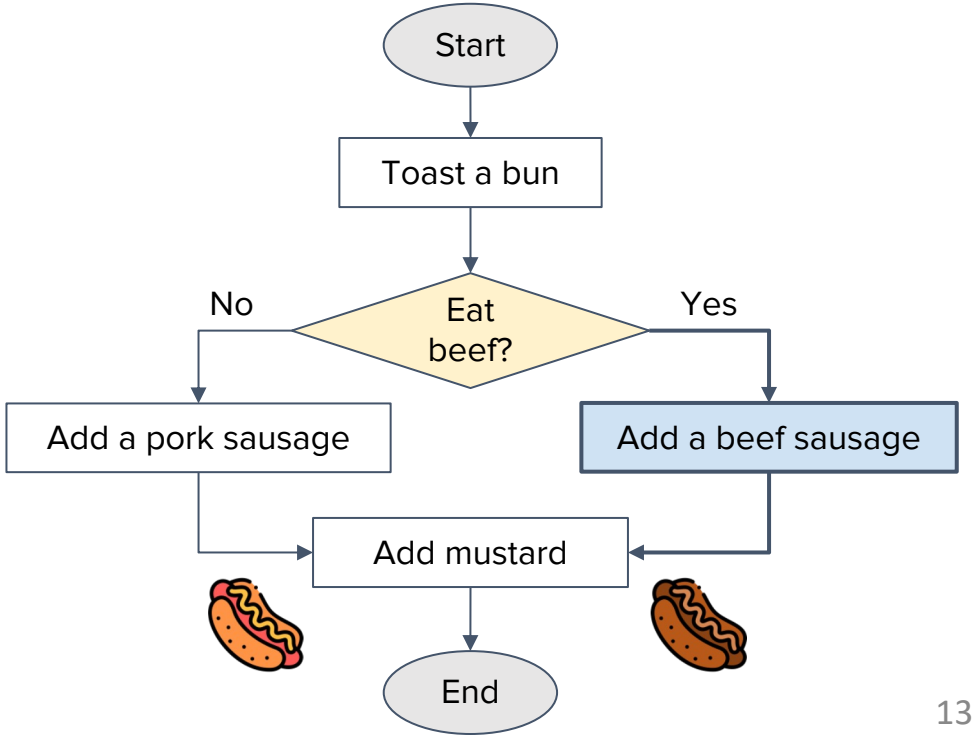
# Example: Sequential vs. Selection



**"Sequential"**

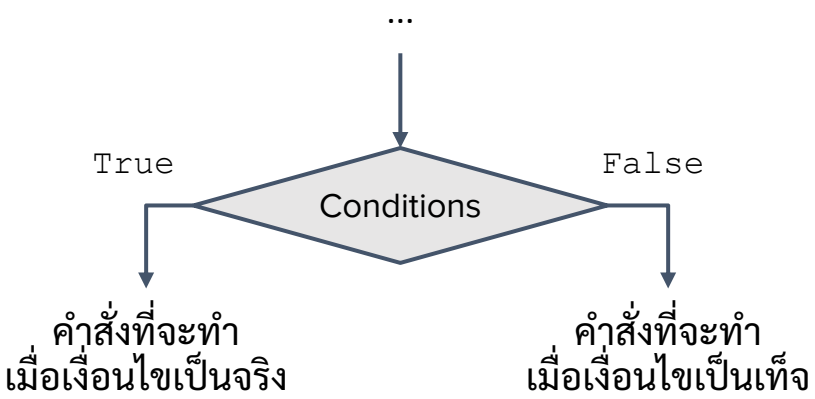


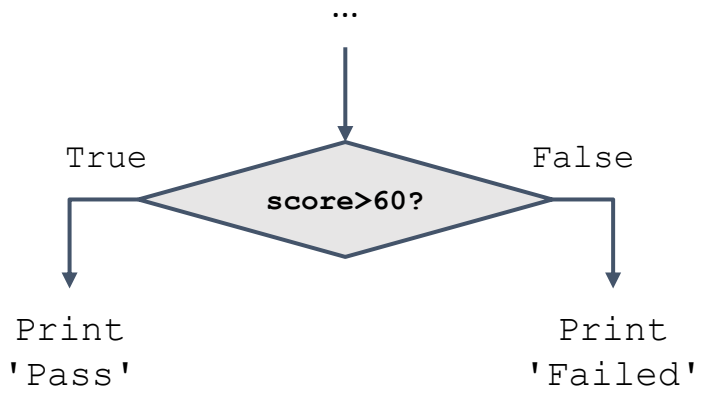

**"Selections"**




Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

- **Selection**: รูปแบบที่การเลือกทำชุดคำสั่งบางชุด โดยพิจารณาจากค่าที่เงื่อนไข (Conditions) เหล่านั้นเป็นจริง (True) หรือเท็จ (False)




# Boolean, Relational Expression and Logical Expression

# Data Type: Boolean



- ค่าความจริงใน Python จะเป็นประเภท **Boolean** ประกอบด้วยค่าที่เป็นจริง (**True**) และค่าความจริงเป็นเท็จ (**False**)

```
>>> my_boolean = True
>>> print(my_boolean)
True

>>> another_boolean = False
>>> print(another_boolean)
False
```

**Note:** `True` กับ `False` เป็นเหมือนคำเฉพาะ (Reserved words) ในภาษา Python ซึ่งต้องสะกดในรูปนี้เท่านั้น และต้องไม่มี Single quote (') ครอบ

- โดยปกติแล้ว Boolean จะได้จากการประมวลผลของนิพจน์เชิงสัมพันธ์หรือ **Relational expression**

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

# Relational Expression



**Relational expression** คือ expressions ที่ใช้ตัวดำเนินการเชิงเปรียบเทียบ (Comparison operators) โดยจะเปรียบเทียบค่าสองค่าและประมวลผลลัพธ์เป็นค่า Boolean

**Table 2-1:** Comparison Operators

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

เครื่องหมาย == ใช้เปรียบเทียบว่าค่าสองค่าเท่ากันหรือไม่ ซึ่งไม่ใช่ Assignment operator (=) ที่นำค่ามาใส่ตัวแปร

ใช้เปรียบเทียบเฉพาะ `int` และ `float` เท่านั้น

Table from https://automatetheboringstuff.com/2e/chapter2/

# **Exercise**: Relational Expression (1)



- == (เท่ากับ) / != (ไม่เท่ากับ)

```
5 == -5
5 =='5'
'5'=='5'
5 == 2+3
5 != 10//2
5 != int('5’)
'dst'!='DST'
str(5) != '5'
```

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

# **Exercise**: Relational Expression (2)



- `<` , `<=`, `>` และ `>=` (ใช้เปรียบเทียบได้เฉพาะ `int` และ `float`)

```
my_var =  10          # value of my_var is
my_var <  10
my_var <= 10
my_var = my_var-10    # value of my_var is
0   >  my_var
0   >= my_var
```

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

# Logical Expression



**Logical expression** คือ expressions ที่ใช้ตัวดำเนินการด้านตรรกศาสตร์ (Boolean operators ได้แก่ **`and`** **`or`** และ **`not`**) โดยจะเปรียบเทียบค่า Boolean และประมวลผลลัพธ์เป็นค่า Boolean

**Table 2-2:** The and Operator's Truth Table

| Expression | Evaluates to . . . |
|---|---|
| True and True | True |
| True and False | False |
| False and True | False |
| False and False | False |

**Table 2-3:** The or Operator's Truth Table

| Expression | Evaluates to . . . |
|---|---|
| True or True | True |
| True or False | True |
| False or True | True |
| False or False | False |

**Table 2-4:** The not Operator's Truth Table

| Expression | Evaluates to . . . |
|---|---|
| not True | False |
| not False | True |

Tables from https://automatetheboringstuff.com/2e/chapter2/

# **Exercise**: Logical Expression



```
(1!=-1) and (5!='5')

(1>=-1) and (5=='5')

(1==-1) or  (5!='5')

(1<=-1) or  (5=='5')

 not (10+10 > 1+1)

 not False
```

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

# Operator Precedence



- หากเรามีตัวดำเนินการทางตรรกศาสตร์หลายตัว
  เช่น หากเรามีตัวแปร `num` และ Logical expression เป็น

```
(num > 5) and (num<=10) or not(num==6)
```

- ลำดับการประมวลผลจะเรียง Operator Precedence ของ Logical expression ใน Python จะไล่ลำดับการประมวลจากซ้ายไปขวา และ `not → and → or`

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

# Example: Expression Evaluation



**Exercise:** `(num>5)and(num<=10)or not(num==6)`

กำหนดให้ `num=8`

**Step 1:**

Expression: `(num>5)and(num<=10)or not(num==6)`

เมื่อทำจากซ้ายไปขวา จะเห็นได้ว่า **`not`** มีลำดับความสำคัญสูงสุด;

Result: `(num>5)and(num<=10)or True`

**Step 2:**

Expression: `(num>5)and(num<=10)or True`

เมื่อทำจากซ้ายไปขวา จะเห็นได้ว่า **`and`** มีลำดับความสำคัญสูงสุด;

Result: `True or True`

**Final Result:** `True`

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

# Writing an Expression



- การเขียนโปรแกรมจะอาศัยต้องแปลงโจทย์ให้อยู่ในรูปแบบของ Expression
- บางครั้งต้องมีการตรวจสอบเงื่อนไขของค่าตัวแปร เช่น
  - x ต้องมีค่าอย่างน้อย 10 → `x >= 10`
  - x ต้องเป็นเลขจำนวนบวก → `x > 0`
  - x ต้องเป็นเลขที่อยู่ระหว่าง 2 กับ 5 → `x>2 and x<5`
  - x ต้องเป็นเลขตั้งแต่ 2 ถึง 5 → `x>=2 and x<=5`
  - x ต้องไม่เป็นเลขคู่ → `x%2 != 0`

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

# **Exercise**: Writing an Expression



จงเขียน Expression เพื่อตรวจสอบค่า x ให้ตรงตามคุณสมบัติที่กำหนด

- x ต้องมีค่าไม่เกิน 100
- x ต้องมีค่าอย่างน้อย 0 แต่ไม่เกิน 10
- x ต้องเป็นเลขคู่ที่ไม่เกิน 10
- x ต้องเป็นจำนวนลบที่ไม่น้อยกว่า-50
- x ต้องเป็นตัวเลขที่หาร 3 ลงตัว
- x ต้องเป็นตัวเลขที่ลงท้ายด้วย 8

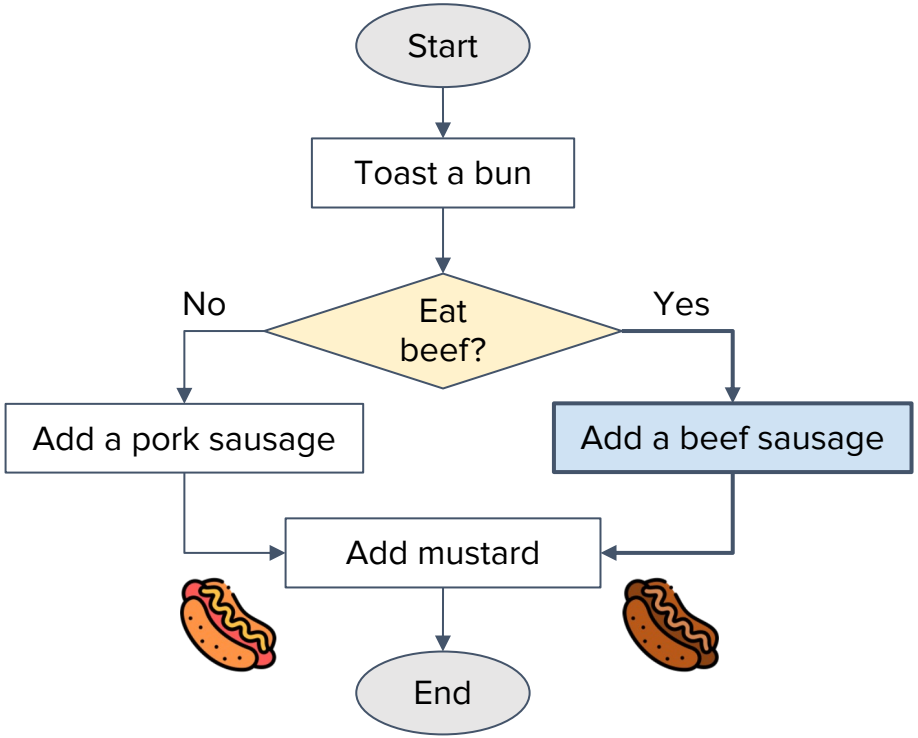
Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

# `if` statement

# Conditions



- Relational Expression และ Logical Expression เป็นพื้นฐานสำหรับการเขียนเงื่อนไข (Conditions)
- Conditions จะถูกประมวลผลเป็นค่า Boolean เสมอ
- ผลของ Conditions (ได้แก่ `True`, `False`) จะเป็นตัวกำหนดชุดคำสั่งที่จะกระทำ

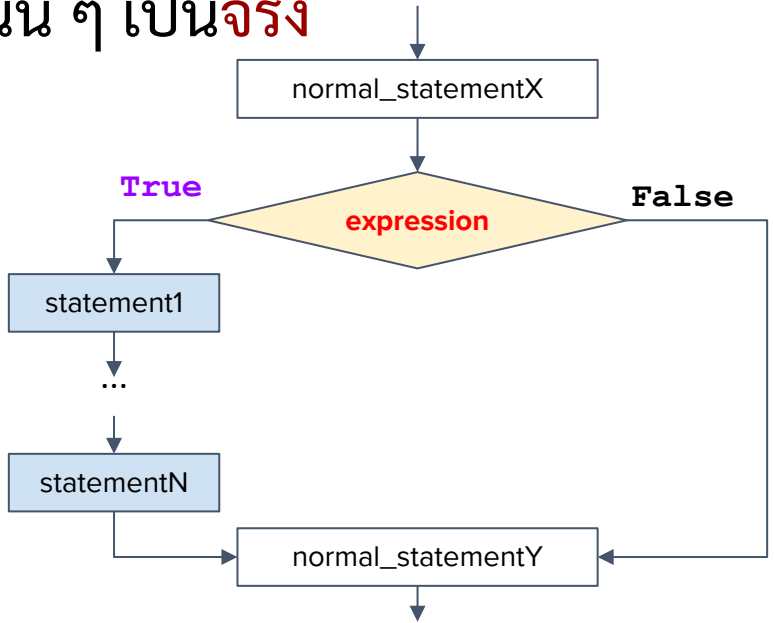


# if statement



- **if** เป็นคำสั่งที่ใช้ควบคุมการทำงานของโปรแกรมให้ทำงานตามเงื่อนไขที่กำหนดเมื่อเงื่อนไขนั้น ๆ เป็นจริง

```
normal_statementX

if expression:
        statement1
        ...
        statementN

normal_statementY
```



# Example: `if` statement



เมื่อถึงคำสั่ง `if` ตัว **`condition`** จะถูกประมวลผล และแสดงผลลัพธ์ตามค่า Boolean

greeting-by-sec.py

```
section=int(input());
if section==1:
    print('Hello, Aj.Pa')
    print('Nice to meet you')
print('Have a good day!!')
```

เมื่อ `section` มีค่าเป็น 1 ทำให้ `section==1` เป็น `True`

```
Hello, Aj.Pa
Nice to meet you
Have a good day!!
```

เมื่อ `section` มีค่าเป็นอื่นที่ไม่ใช่ 1 ทำให้ `section==1` เป็น `False`

```
Have a good day!!
```

# Block of Code



- จากการเขียนคำสั่ง if จะเห็นได้ว่า Statements ใน Python สามารถรวมกลุ่มเป็น blocks จากการย่อหน้าหรือ Indentation
- Block แต่ละ block จะเริ่มเมื่อกด Tab เพื่อย่อหน้าและสิ้นสุดขอบเขตของ block เมื่อยกเลิกการทำย่อหน้า
- Block สามารถซ้อนในอีก block ได้

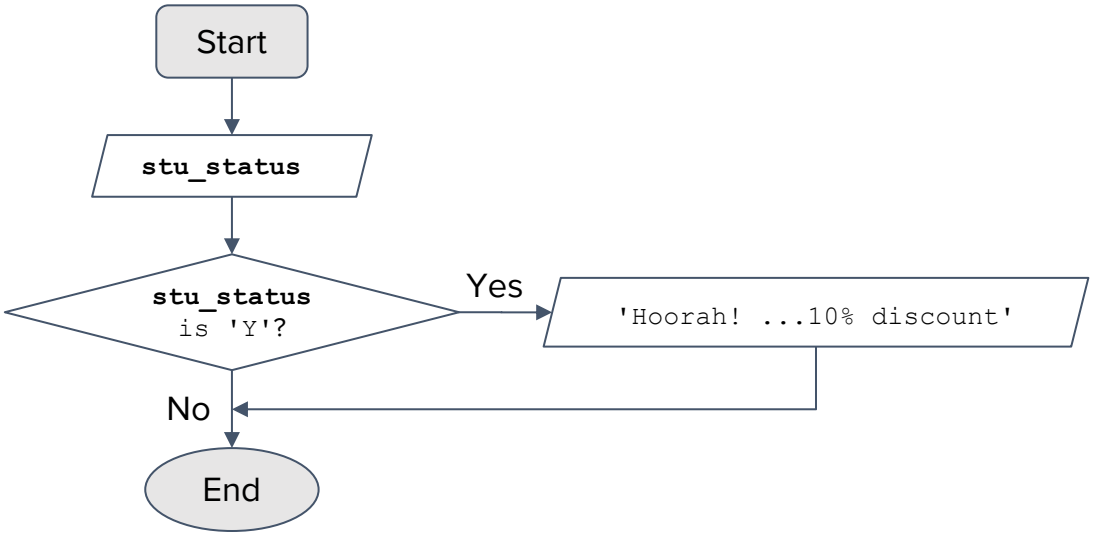
```
statement_outside_if
if expression1:
    statement_blue-1
    statement_blue-2
    ...
        if expression2:
        statement_pink-1
        statement_pink-2
        ...
    statement_blue-N
statement_outside_if
```

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

# Exercise: Student Discount



จงเขียนโปรแกรมเพื่อบอกส่วนลดสินค้าแก่นักศึกษา โดยรับ input สถานะของนักศึกษา `(stu_status: 'Y', 'N')` หากเป็นนักศึกษา โปรแกรมแสดงข้อความบอกนักศึกษาว่า `'Hoorah! You've got 10% discount'`




Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

# Exercise: Student Discount



จงเขียนโปรแกรมเพื่อบอกส่วนลดสินค้าแก่นักศึกษา โดยรับ input สถานะของนักศึกษา (stu_status: 'Y', 'N') หากเป็นนักศึกษา โปรแกรมแสดงข้อความบอกนักศึกษาว่า 'Hoorah! You've got 10% discount'

Source code: studentDisc.py

**input()** สามารถใส่ข้อความเพื่อให้การรับค่า input อ่านได้ง่ายขึ้น

```
stu_status = input('Are you a student? ')
if stu_status=='Y':
    print('Hoorah! You've got 10% discount')
print('Bye bye')
```

Run a program: **python studentDisc.py**

```
Are you a student? _
```

เมื่อ **stu_status** มีค่าเป็น **'Y'**

```
Are you a student? Y
Hoorah! You've got 10% discount
Bye bye
```

เมื่อ **stu_status** มีค่าเป็นอื่น ๆ

```
Are you a student? N
Bye bye
```

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

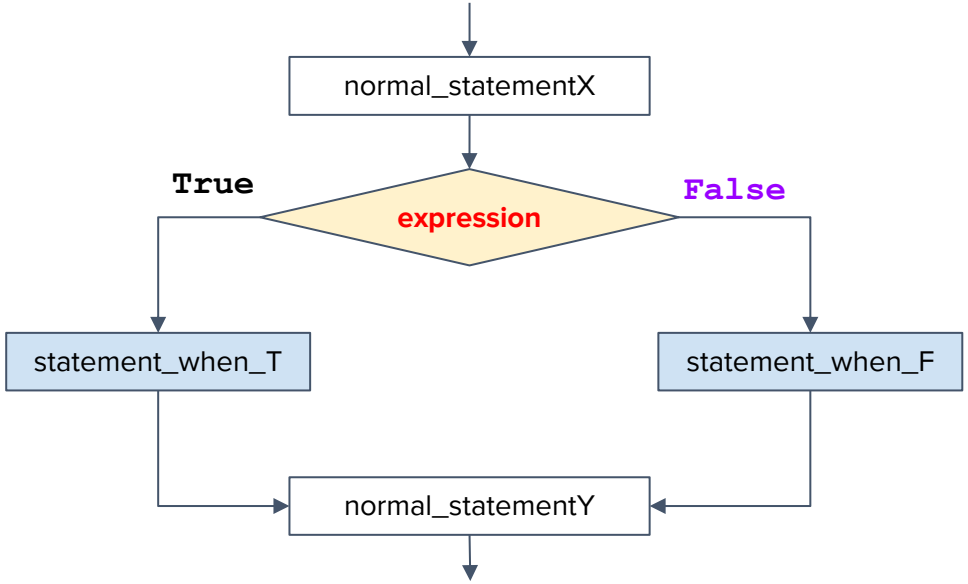
# `else` and `elif`

# `else` statement



- **`else`** เป็นคำสั่งที่ใช้ต่อเนื่องจากคำสั่ง **`if`** เพื่อควบคุมการทำงานของโปรแกรมให้ทำงานตามเงื่อนไขที่กำหนดเมื่อเงื่อนไขนั้น ๆ เป็นเท็จ

```
normal_statementX

if expression:
        statement_when_T
else :
    statement_when_F

normal_statementY
```



# if-else statement



เมื่อถึงคำสั่ง **if** ตัว **condition** จะถูกประมวลผล และแสดงผลลัพธ์ตามค่า Boolean หาก **condition** เป็นจริงจะทำใน Block **if** และ **condition** เป็นเท็จจะทำใน Block **else**

greeting-by-sec2.py

```
section=int(input());
if section==1: #True
    print('Hello, Aj.Pa')
    print('Nice to meet you')
else: #False
    print('Hello, Aj.Tip')
print('Have a good day!!')
```

เมื่อ `section` มีค่าเป็น 1 ทำให้ `section==1` เป็น **`True`**

```
Hello, Aj.Pa
Nice to meet you
Have a good day!!
```

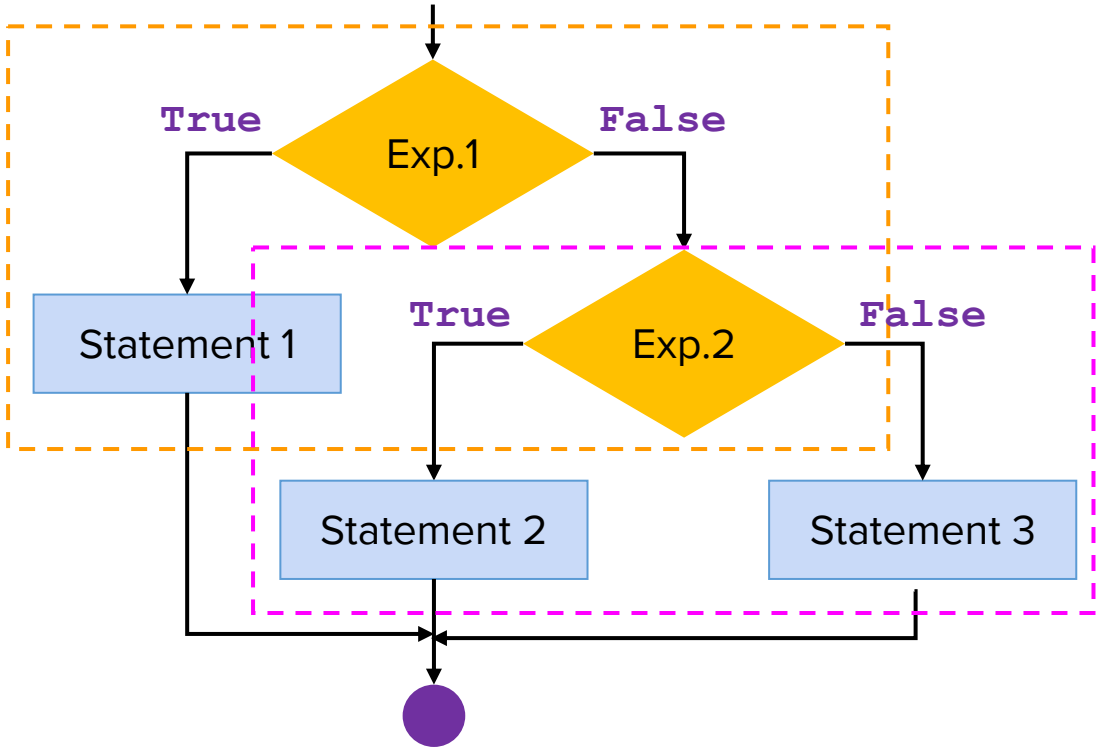
เมื่อ `section` มีค่าเป็นอื่นที่ไม่ใช่ 1 ทำให้ `section==1` เป็น **`False`**

```
Hello, Aj.Tip
Have a good day!!
```

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

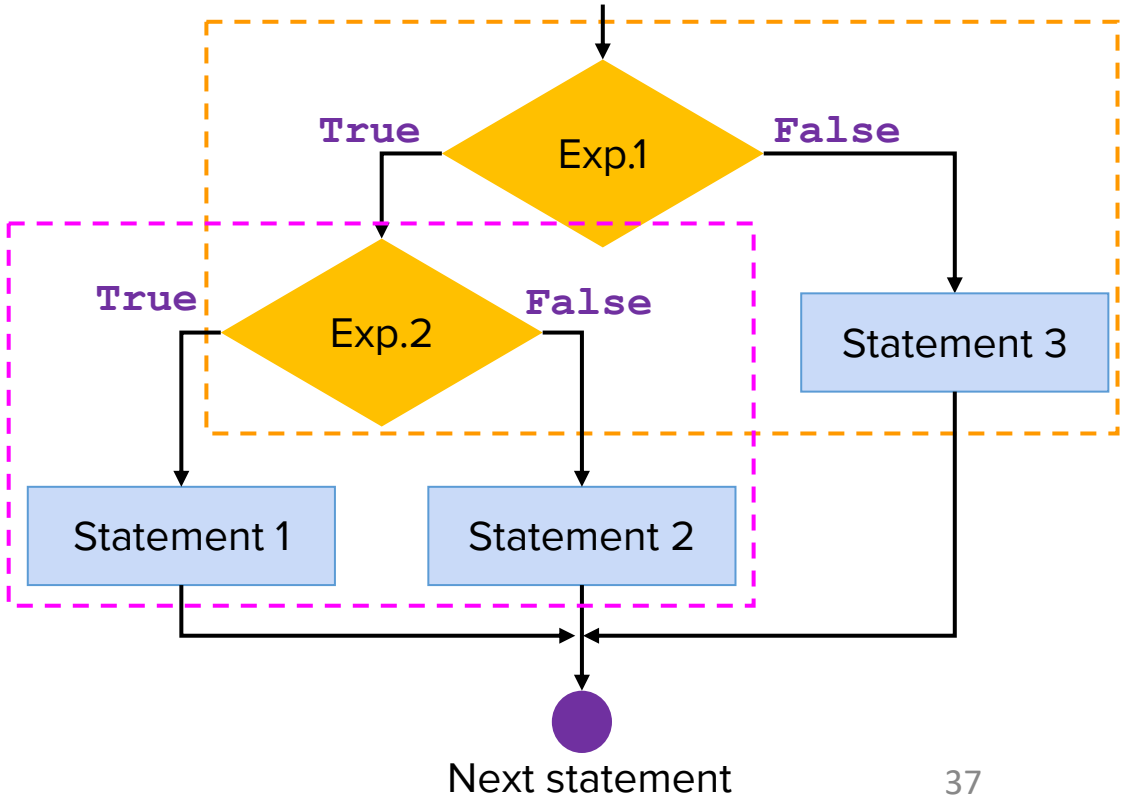
# if-else (nested)

```
if (expression1):
    statement1
else:
    if (expression2):
        statement2
    else:
        statement3

next_statement
```



# if-else (nested)

```
if (expression1):
    if (expression2):
        statement1
    else:
        statement2
else:
    statement3

next_statement
```



# Exercise: Female Student Discount



จงเขียนโปรแกรมเพื่อบอกส่วนลดสินค้า (discount) แก่นักศึกษา โดยรับ input สถานะของนักศึกษา (stu_status: 'Y', 'N') หากเป็นบุคคลธรรมดาจะไม่มีส่วนลด หากเป็นนักศึกษาให้รับ input เป็นเพศเพิ่มเติม (gender: 'M', 'F') หากเป็นนักศึกษาหญิงจะได้ส่วนลดเพิ่มเป็น 15% หากเป็นนักศึกษาชายจะได้ส่วนลด 10%

# Exercise: Female Student Discount



## Example Solutions

```
stu_status = input('Are you a student? ')
if (stu_status == 'Y'):
    gender = input('What is your gender? ')
    if(gender == 'F'):
        discount = 15
    else:
        discount = 10
else:
    discount = 0
print('Discount:',discount)
```

```
Are you a student? N
Discount:  0
```

```
Are you a student? Y
What is your gender? F
Discount: 15
```

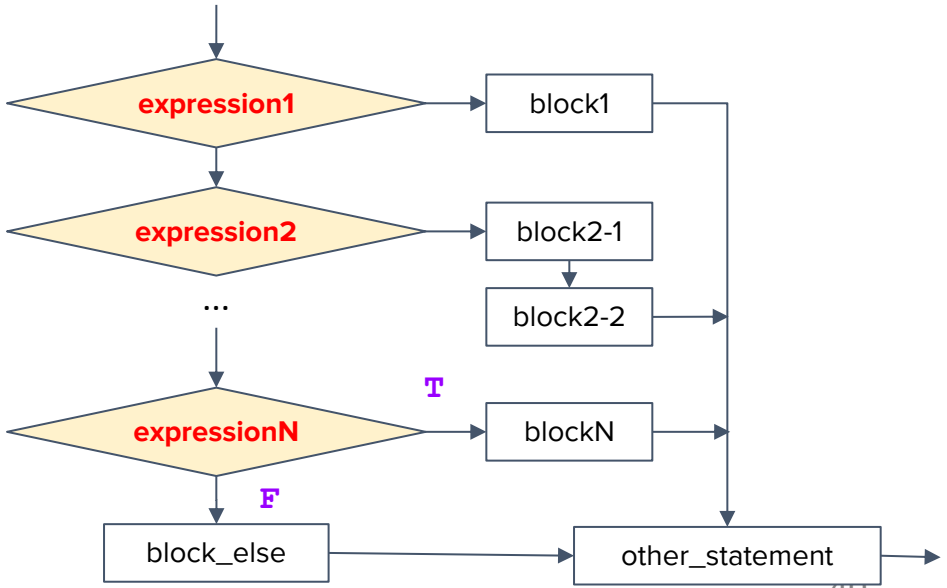
```
Are you a student? Y
What is your gender? M
Discount: 10
```

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

# elif statement

- **elif** (ย่อจาก else if ) เป็นคำสั่งที่ใช้ต่อเนื่องจากคำสั่ง **if** เพื่อควบคุมการทำงานของโปรแกรมให้ทำงานตามหลาย ๆ เงื่อนไขอย่างต่อเนื่อง
- ใช้ร่วมกับ **if** และ **else**

```
if expression1:
        statement_block1
elif expression2:
        statement_block2-1
    statement_block2-2
...
elif expressionN:
        statement_blockN
else:
    statement_block_else
other_statement
```



# elif statement



เมื่อเงื่อนไขใน `if` เป็นเท็จ โปรแกรมจะตรวจสอบเงื่อนไขใน `elif` ไปเรื่อย ๆ

- หากเจอเงื่อนไขที่เป็นจริง จึง run block นั้น ๆ
- หากไม่มีเงื่อนไขที่เป็นจริง จึง run block `else`

greeting-by-sec3.py

```
section=int(input());
if section==1:
    print('Hello, Aj.Pa')
elif section==2:
    print('Hello, Aj.Tip')
else: #False
    print('Hello World!')
print('Have a good day!!')
```

เมื่อ `section` มีค่าเป็น 1 ทำให้ `section==1` เป็น **`True`**

```
Hello, Aj.Pa
Have a good day!!
```

เมื่อ `section` มีค่าเป็น 2 ทำให้ `section==1` เป็น **`False`** และ `section==2` เป็น **`True`**

```
Hello, Aj.Tip
Have a good day!!
```

เมื่อ `section` มีค่าอื่น ๆ `section==1` และ `section==2` เป็น **`False`**

```
Hello World!
Have a good day!!
```

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

# Example: Grading



แสดงผลลัพธ์ของเกรดตามช่วงคะแนน

| Score | Grade |
|---|---|
| >= 80 | A |
| >= 70 | B |
| >=60 | C |
| >=50 | D |
| <50 | F |

```
score=int(input('Input a score: '))
if (score >= 80):
    print('Grade: A')
elif (score>=70):
    print('Grade: B')
elif (score>=60):
    print('Grade: C')
elif (score>=50):
    print('Grade: D')
else:
    print('Grade: F')
print('Bye bye')
```

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

# Exercise: What if `score=60`?



## แสดงผลลัพธ์ของเกรดตามช่วงคะแนน (1) `if` vs (2) `if-elif-else`

```
score=int(input('Input a score: '))
if (score >= 80):
    print('Grade: A')
if (score>=70):
    print('Grade: B')
if (score>=60):
    print('Grade: C')
if (score>=50):
    print('Grade: D')
else:
    print('Grade: F')
print('Bye bye')
```

1

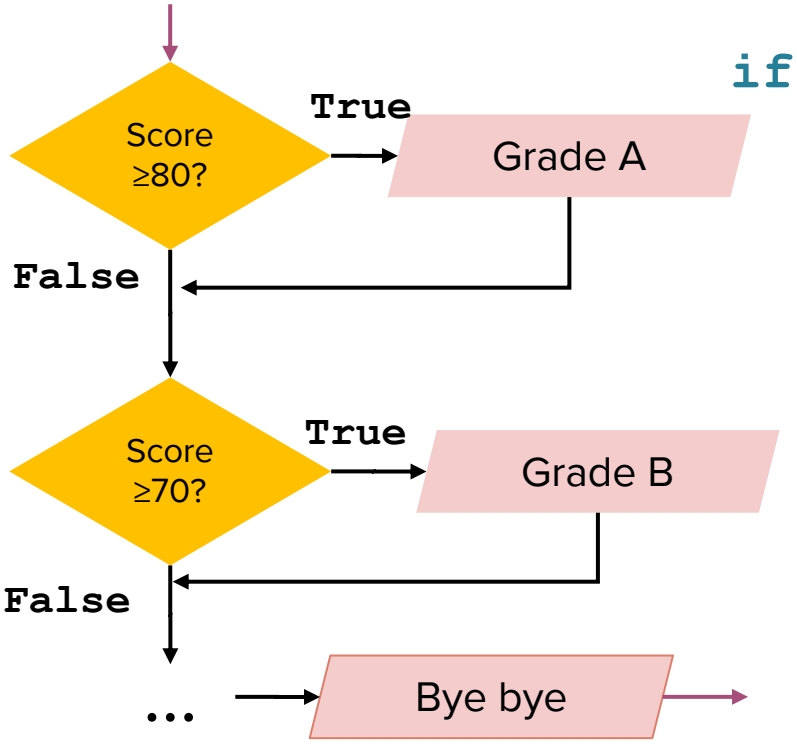
```
score=int(input('Input a score: '))
if (score >= 80):
    print('Grade: A')
elif (score>=70):
    print('Grade: B')
elif (score>=60):
    print('Grade: C')
elif (score>=50):
    print('Grade: D')
else:
    print('Grade: F')
print('Bye bye')
```

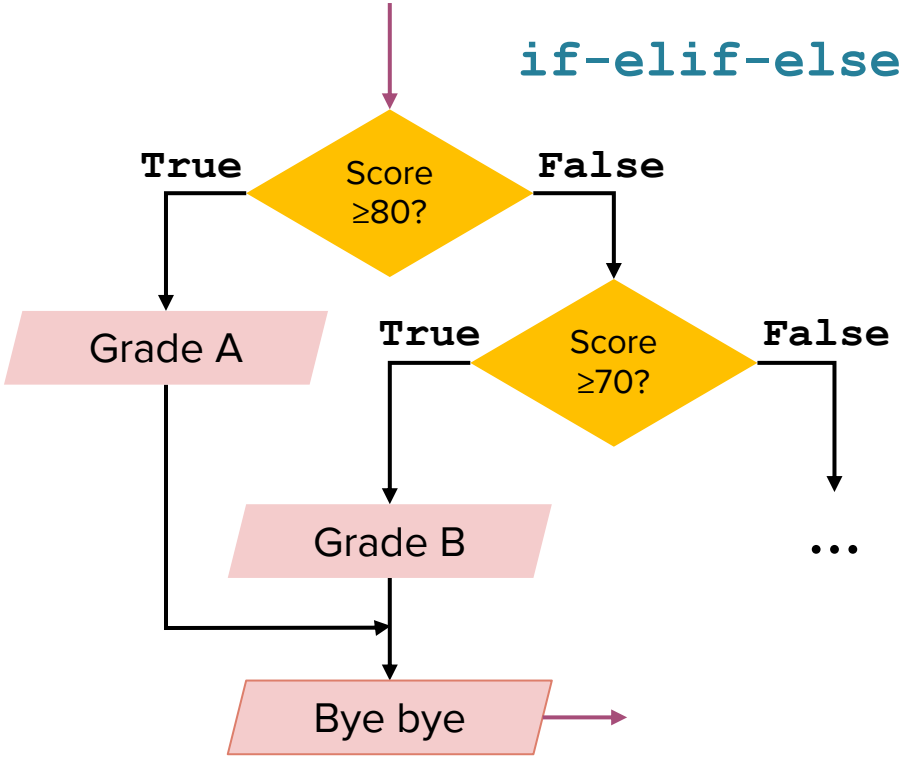
2

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

# Exercise: What if `score=60`?






Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

# Exercise: Age



หากต้องการจะต้องแสดงผลลัพธ์ตามช่วงอายุ age: (0-19] คือ 'young', age: [20-65] คือ 'adult' และ age: (65,200] คือ 'senior' ต้องทำอย่างไร

Contact: [jidapa.kra, akara.sup, tipajin.tha]@mahidol.edu

# Exercise: Age



หากต้องการจะต้องแสดงผลลัพธ์ตามช่วงอายุ age: (0-19] คือ 'young', age: [20-65] คือ 'adult' และ age: (65,200] คือ 'senior' ต้องทำอย่างไร

1

```
age = int(input())
if age>65:
    print('senior')
elif age>=20:
    print('adult')
else:
    print('young')
```

2

```
age = int(input())
if age<20:
    print('young')
elif (age>=20 and age<=65):
    print('adult')
else:
    print('senior')
```

3

```
age = int(input())
if (age>0 and age<20):
    print('young')
if (age>=20 and age<=65):
    print('adult')
if (age>65 and age<200):
    print('senior')
```

1 , 2, 3 ให้ผลลัพธ์เหมือนกัน แต่ในกรณีแบบที่ 3 ไม่แนะนำให้เขียน เพราะอาจเกิด error ได้

`if-elif-else` เหมาะสำหรับการเช็คเงื่อนไขที่เป็นจริงได้ในกรณีเดียวเพราะเป็นการเช็คทุกเงื่อนไขต่อเนื่อง แต่การใช้ `if` หลาย ๆ อันนั้นจะเป็นการเช็คแยกเงื่อนไขออกจากกัน อาจจะทำให้เกิด error ได้


Thanks to: https://keep-calm.net/keep-calm-and-code-on.html

# Happy Coding :)