การแก้ปัญหาด้วยภาษาไพทอน

จัดทำโดย

นายร่วมชาติ ชัยนา

ความรู้เบื้องต้นเกี่ยวกับไพทอน

ภาษาคอมพิวเตอร์ หมายถึง ภาษาที่ผู้ใช้งานใช้สื่อสารกับคอมพิวเตอร์หรือคอมพิวเตอร์สื่อสารกัน แล้วคอมพิวเตอร์สามารถทำงานตามคำสั่งนั้น ๆ ได้

โปรแกรมภาษาที่นิยมใช้ในการเขียน เช่น ภาษาซี (C) ภาษาจาวา (Java) ภาษา C++ ภาษา PHP และ ภาษา Python เป็นต้น ซึ่งแต่ละภาษาจะมีวิธีการเขียนที่คล้ายกัน แต่มีความซับซ้อนที่ต่างกัน

Python คือชื่อภาษาที่ใช้ในการเขียนโปรแกรมภาษาหนึ่งที่มีความสามารถสูง ถ้าเปรียบเทียบกับภาษาอื่น ๆ

ภาษา Python นั้นกำเนิดขึ้นในปลายปี 1980 และการพัฒนาได้เริ่มต้นใน ธันวาคม 1989 โดย Guido van Rossum ที่ Centrum Wiskunde & Information (CWI) ในประเทศเนเธอร์แลนด์

สิ่งที่ควรรู้ก่อนเขียนโปรแกรมภาษาไพทอน

ควรทำความเข้าใจกับความรู้พื้นฐานต่อไปนี้ เพื่อให้การเขียนโปรแกรมทำได้รวดเร็วขึ้น

1. ไอดีอีภาษาไพทอน (Python IDE) ภาษาไพทอนเป็นภาษาระดับสูง โปรแกรมที่เขียนจึงต้องถูกแปลให้เป็นภาษาเครื่อง โดยไอดีอีภาษาไพทอนจะทำงานได้ทั้งในโหมดอิมมีเดียท (Immediate mode) และโหมดสคริปต์ (Script mode)

    1.1 โหมดอิมมีเดียท เป็นการพิมพ์คำสั่งทีละคำสั่ง

    1.2 โหมดสคริปต์ เป็นการพิมพ์คำสั่งหลายคำสั่งเก็บไว้เป็นไฟล์ก่อน เมื่อผู้เขียนโปรแกรมสั่งให้ทำงาน

2. ข้อมูลเข้า เป็นข้อมูลที่ผู้ใช้นำเข้าสู่โปรแกรมขณะที่โปรแกรมกำลังทำงานอยู่
3. ข้อมูลออก เป็นผลลัพธ์ที่ได้จากการที่คอมพิวเตอร์ทำงานตามโปรแกรม

สิ่งที่ควรรู้ก่อนเขียนโปรแกรมภาษาไพทอน (ต่อ)

4. ข้อผิดพลาด (Error) คือ ความผิดพลาดที่เกิดขึ้นจากการเขียนโปรแกรม

    4.1 ข้อผิดพลากทางไวยากรณ์ (Syntax error)

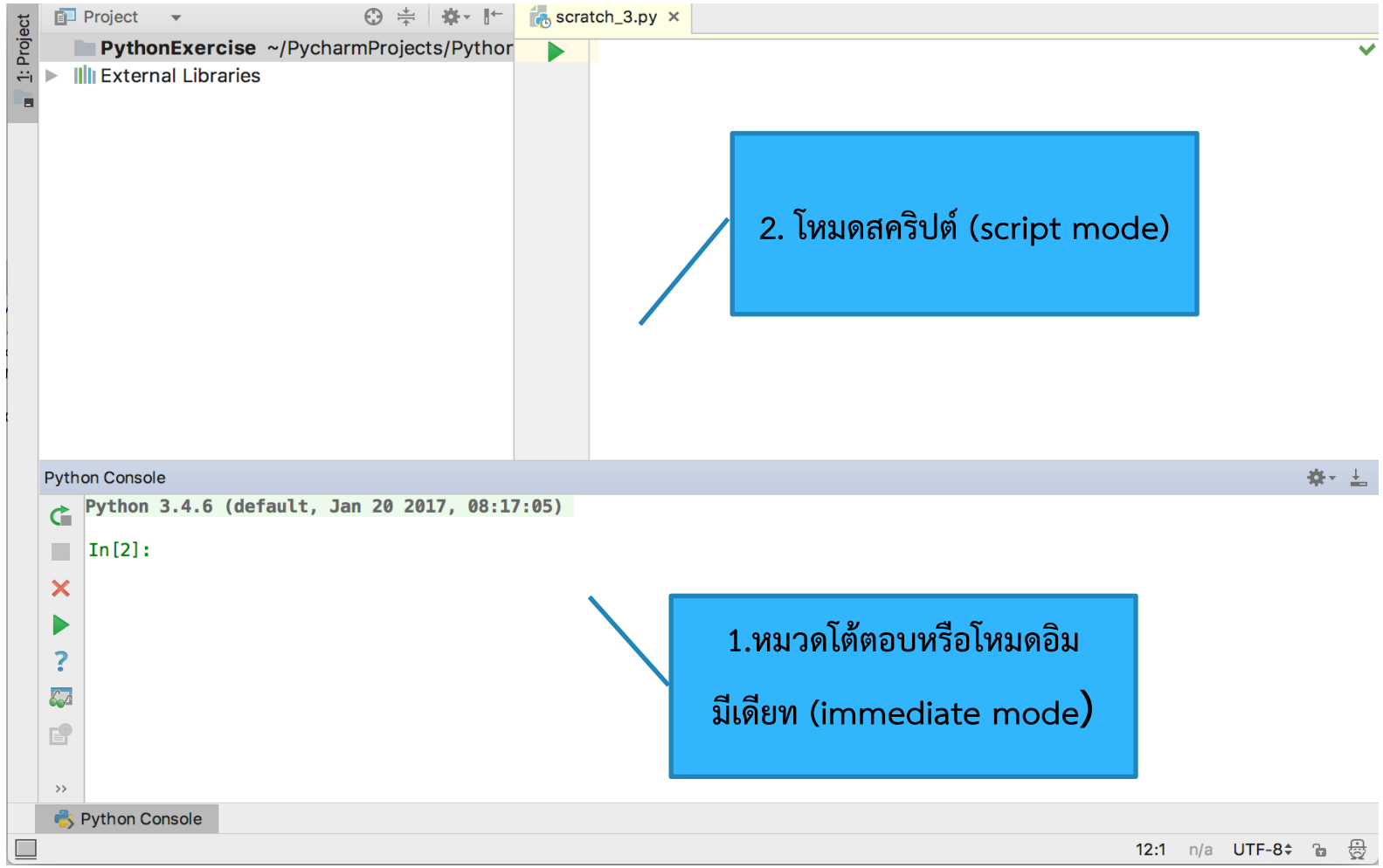
    4.2 ข้อผิดพลาดขณะโปรแกรมทำงาน (Runtime error)

    4.3 ขอผิดพลาดทางความหมาย (Semantic error)

5. การแก้ไขจุดบกพร่อง (Debugger) เป็นกระบวนการในการตรวจหาข้อผิดพลาดในโปรแกรมที่เขียนขึ้น

6. คอมเมนต์ (Comment) เป็นคำอธิบายที่ใส่ไว้เพื่อเตือนความจำ ในภาษาไพทอนจะใช้สัญลักษณ์ (#) แสดงจุดเริ่มต้นของคอมเมนต์ในแต่ละบรรทัด

เครื่องมือในการพัฒนาโปรแกรม (Thonny)

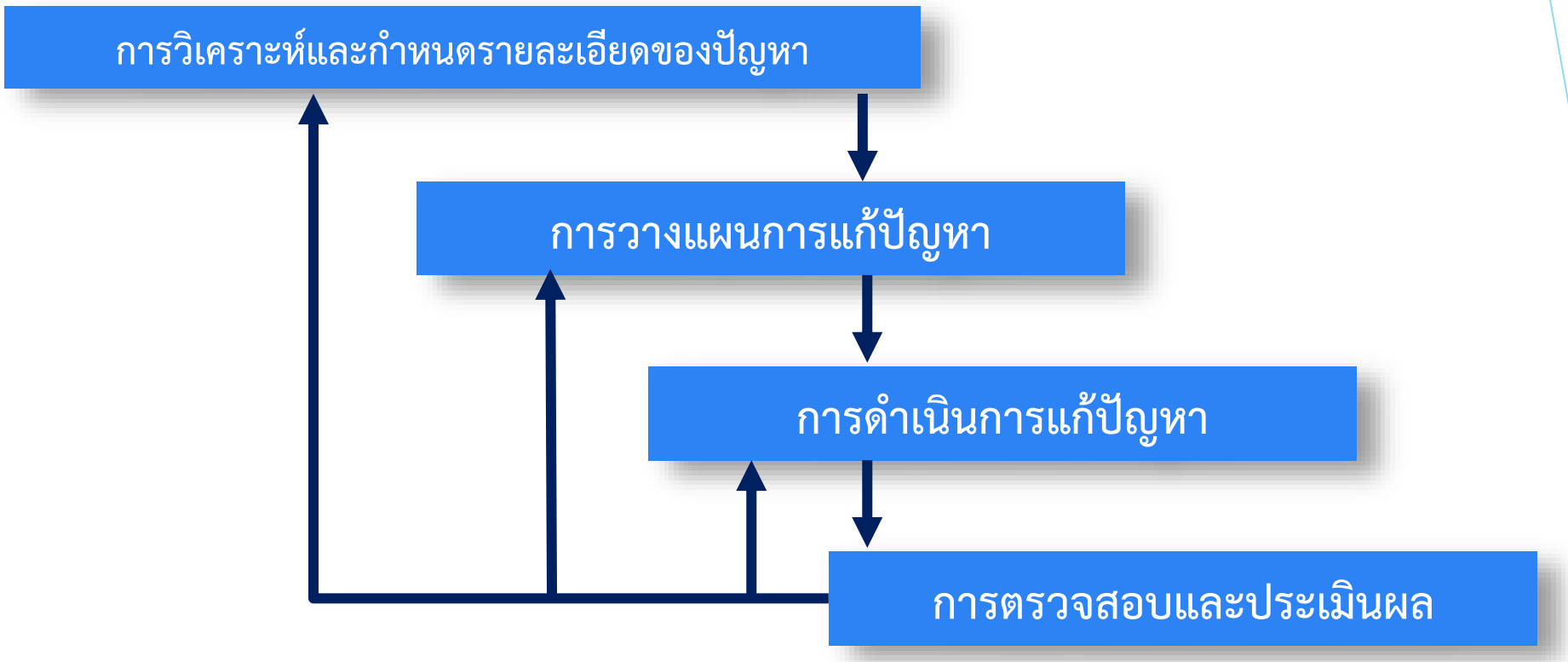


จุดเด่นของภาษาไพทอน

ภาษาไพทอน ได้ออกแบบมาเพื่อให้ทำงานได้กับ Web Application ที่ลักษณะคล้ายกับภาษา Perl, PHP, JAVA และ ASP ภาษาไพทอน มีจุดเด่น ดังนี้

1. สามารถใช้ได้ทุกแพลตฟอร์ม
2. ไม่ต้องเสียค่าใช้จ่ายในการจัดซื้อโปรแกรมต้นฉบับ
3. เขียนง่าย เมื่อเทียบกับภาษาคอมพิวเตอร์ส่วนใหญ่แล้ว
4. มีความปลอดภัยสูง เนื่องจากภาษาไพทอนทำงานอยู่ด้าน Server เป็นหลัก
5. ต่อยอดได้ง่าย ภาษาไพทอนมีไลบารีให้ใช้มากมาย

ขั้นตอนการแก้ปัญหา



เริ่มต้นเขียนโปรแกรมภาษาไพทอน

1. **คำสั่งแสดงผล** เป็นคำสั่งที่ให้แสดงผลออกทางจอภาพ

รูปแบบ: print() เป็นคำสั่งชนิดฟังก์ชัน เช่น

print(“Hello World”) หรือ print(‘Hello World’)

print(5+5)

2. **คำสั่งตัวแปร** เป็นคำสั่งที่ให้คอมพิวเตอร์เก็บค่าตัวแปร

รูปแบบ: ตัวแปร (variable) ตัวแปรต้องกำหนดให้เป็นตัวอักษรภาษาอังกฤษ เช่น

name=“บัวขาว” และ x=10

3. **คำสั่งรับข้อมูล** เป็นคำสั่งที่ให้คอมพิวเตอร์รับข้อมูลจากผู้ใช้ทางคีย์บอร์ด

รูปแบบ: input ( ) ทำหน้าที่รับข้อมูลจากผู้ใช้ทางคีย์บอร์ด เช่น

name=input(“โปรดใส่ชื่อ:”)

num = int(input(“please in input number: ”)

4. คำสั่งตรวจสอบชนิดข้อมูล เป็นคำสั่งที่ให้ตรวจสอบชนิดของข้อมูล

รูปแบบ: type() ตามด้วยข้อความที่จะตรวจสอบ เช่น

type (name) หรือ type (5)

5. คำสั่งแปลงชนิดข้อมูล เป็นคำสั่งที่ใช้แปลงข้อมูลก่อนนำไปใช้

ชื่อฟังก์ชัน	คำอธิบาย	ตัวอย่าง
str ()	แปลงค่าใน ( ) ให้เป็นข้อความ	str (10.789)
int ()	แปลงค่าใน ( ) ให้เป็นตัวเลขจำนวนเต็ม	x = “10” int (x)
long ()	แปลงค่าใน ( ) ให้เป็นตัวเลขจำนวนเต็มความจุไม่จำกัด	long (“456533464643”)
float ()	แปลงค่าใน ( ) ให้เป็นตัวเลขทศนิยม	x = input ( ) float (x)

ชนิดของข้อมูลในภาษาไพทอน

ชนิดของข้อมูล ที่ใช้บ่อย ๆ มีดังนี้

1. ข้อมูลชนิดตัวเลข หมายถึง ข้อมูลที่เป็นชุดของตัวเลขซึ่งประกอบด้วย ตัวเลขจำนวนเต็ม ตัวเลขที่มีจุดทศนิยม
2. ชนิดข้อมูลแบบอักขระ คือ ข้อมูลที่เป็นอักขระเพียงหนึ่งตัวเท่านั้น โดยที่ตัวอักขระนี้จะอยู่ในเครื่องหมาย Apostrophes หรือ single quote (‘ ’) หรือ Double quote (“ ”)
3. ชนิดข้อมูลแบบข้อความ คือ ข้อมูลที่เป็นอักขระตั้งแต่หนึ่งตัวอักขระขึ้นไป ซึ่งจะเรียงต่อกันเป็นกลุ่มโดยจะอยู่ภายในเครื่องหมาย Apostrophes หรือ single quote (‘ ’) หรือเครื่องหมาย Double quote (“ “)
4. ชนิดข้อมูลแบบตรรกศาสตร์ คือ ข้อมูลที่ให้ผลลัพธ์ที่ได้จากการตัดสินใจจากเงื่อนไขหรือนิพจน์ ข้อมูลชนิดนี้จะมีเพียงค่าจริงและเท็จเท่านั้น

ตัวดำเนินการในภาษาไพทอน

ตัวดำเนินการ (Operators) หมายถึง เครื่องหมายทางคณิตศาสตร์หรือคำสั่งที่ใช้ในการคำนวณผล หรือหาค่าต่าง ๆ

ชนิดของตัวดำเนินการ

1. ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic operators)
2. ตัวดำเนินการเปรียบเทียบ (Comparison operators)
3. ตัวดำเนินการทางตรรกะ (Logical operators)
4. ตัวดำเนินการกำหนดค่า (Assignment operators)
5. ตัวดำเนินการทางตรรกะ (List/String operators)

1. ตัวดำเนินการทางคณิตศาสตร์ (Arithmetic operators)

ตัวดำเนินการ	คำอธิบาย	ตัวอย่าง
+	บวก (addition)	x = 20 + 5
-	ลบ (subtraction)	x = 20 -5
*	คูณ (multiplication)	y = x * 4
/	หาร (division)	z = y / 3
//	การหารและตัดส่วนทิ้ง (Floor division)	x = 10 // 3 ตอบ 3
%	มอดูลัส (Modulus): เศษที่เหลือจากการหาร	x = 7 %5 ตอบ 2
**	เลขยกกำลัง (Exponent)	x = 10 ** 2 ตอบ 100

2. ตัวดำเนินการเปรียบเทียบ (Comparison (Relational) operators)

ตัวดำเนินการ	คำอธิบาย	ตัวอย่าง
>	มากกว่า	x > y (x มากกว่า y)
>=	มากกว่า หรือเท่ากับ	x >= y (x มากกว่า หรือเท่ากับ y)
<	น้อยกว่า	x < y (x น้อยกว่า y)
<=	น้อยกว่า หรือเท่ากับ	x <= y (x น้อยกว่า หรือเท่ากับ y)
==	ท่ากัน	x == y (x เท่ากันกับ y)
!= หรือ <>	ไม่เท่ากัน (ใช้ได้ทั้งสองแบบ != หรือ <>)	x != y (x ไม่เท่ากันกับ y)
is	เป็นตัวแปรที่ชี้ไปเก็บข้อมูลที่เดียวกัน	x = [1, 2, 3] x = y x is y (ให้คำตอบเป็น (True)

3. ตัวดำเนินการทางตรรกะ (Logical (Boolean) operators)

3.1 and (และ): จะให้ค่า True เมื่อเงื่อนไขทุกตัวเป็นจริง (True)

3.2 or (หรือ): จะให้ค่า True เมื่อเงื่อนไขตัวใดตัวหนึ่ง เป็นจริง (True)

ข้อมูลที่ 1	ข้อมูลที่ 2	เชื่อมด้วย And	ผลลัพธ์	เชื่อมด้วย OR	ผลลัพธ์
True	True	True and True	True	True and True	True
True	False	True and False	False	True and False	True
False	True	False and True	False	False and True	True
False	False	False and False	Fase	False and False	False

3.3 not (ตรงกันข้าม/ไม่): จะให้ค่าเป็น True เมื่อเงื่อนไข เป็น False และจะให้ค่าเป็น False เมื่อเงื่อนไขเป็น True

ข้อมูล	เชื่อมด้วย Not	ผลลัพธ์
True	not True	False
False	not False	True

4. ตัวดำเนินการกำหนดค่า (Assignment operators)

ตัวดำเนินการ	คำอธิบาย	ตัวอย่าง
=	กำหนดค่าฝั่งซ้าย เท่ากับ ฝั่งขวา	y = 2
+=	x += 10 มีค่าเท่ากับ x = x+10	x +=x (มีค่าเท่ากับ x + x)
-=	x -= 5 มีค่าเท่ากับ x = x-5	y -= y (มีค่าเท่ากับ y - y = 0)
*=	x *= 2 มีค่าเท่ากับ x = x*2	x *= x (มีค่าเท่ากับ x * x)
/=	x /= 2 มีค่าเท่ากับ x = x/2	x /= x (มีค่าเท่ากับ x / x = 1)

5. ตัวดำเนินการในลิสต์ หรือข้อความ (List/String operators)

ตัวดำเนินการ	คำอธิบาย	ตัวอย่าง
+	การนำ List หรือ String มาต่อเข้าด้วยกัน ตัวอย่างเช่น name = “โรงเรียน” Surname = “บัวขาว” fullname = name + “ “ + surname จะให้ค่าเท่ากับ “โรงเรียน บัวขาว”	x = [1, 2, 3] y = [4, 5, 6] z = x + y z (มีค่าเท่ากับ [1, 2, 3, 4, 5, 6])
*	การสร้าง List หรือ String ซ้ำ	x = [1, 2, 3] y = x * 2 y (มีค่าเท่ากับ [1, 2, 3, 1, 2, 3])
in	เช็คว่ามีสมาชิก/ตัวอักษร ใน List หรือ String (Element/character in) ให้ผลลัพธ์เป็นบูลีน (Boolean)	x = [1, 2, 3, 4] 1 in x (ให้ค่าเท่ากับ True) 10 in x (ให้ค่าเท่ากับ False)
not in	เช็คว่าไม่มีสมาชิก/ตัวอักษร ใน List หรือ String (Element/character not in) ให้ผลลัพธ์เป็นบูลีน (Boolean)	x = “วิทยาลัย” “ลัย” not in x (จะให้ค่าเป็น False) “การ” not in x (จะให้ค่าเป็น True)

ฟังก์ชันทางคณิตศาสตร์ที่ใช้บ่อย ๆ

1. math.factorial(x) คือ หาค่า แฟคทอเรียล
2. math.fabs(x) คือ หาค่าสัมบูรณ์
3. math.gcd(a,b) คือ หา ห.ร.ม.
4. math.exp(x) คือ ฟังก์ชันเลขชี้ยกกำลัง
5. math.log(x,[base]) คือ หาค่า logarithm
6. math.pow(x,y) คือ คืนค่าผลลัพธ์ที่ได้จากยกกำลัง
7. math.sqrt(x) คือ หารากที่สอง

การใช้เรียกใช้ฟังก์ชันทางคณิตศาสตร์ จะใช้ import math เช่น การหารากที่ 2

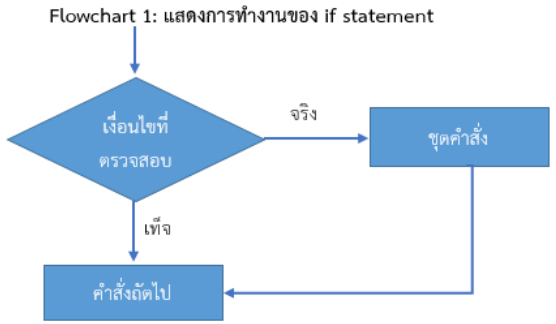
```
import math
x = math.sqrt(9)
print (x)
```

การทำงานตามเงื่อนไข

การทำงานแบบมีทางเลือกเป็นการเขียนคำสั่งที่ไม่จำเป็นต้องทำงานทุกคำสั่ง แต่จะทำงานเมื่อมีเงื่อนไขเป็นจริงตามที่ระบุเท่านั้น มีรูปแบบดังนี้

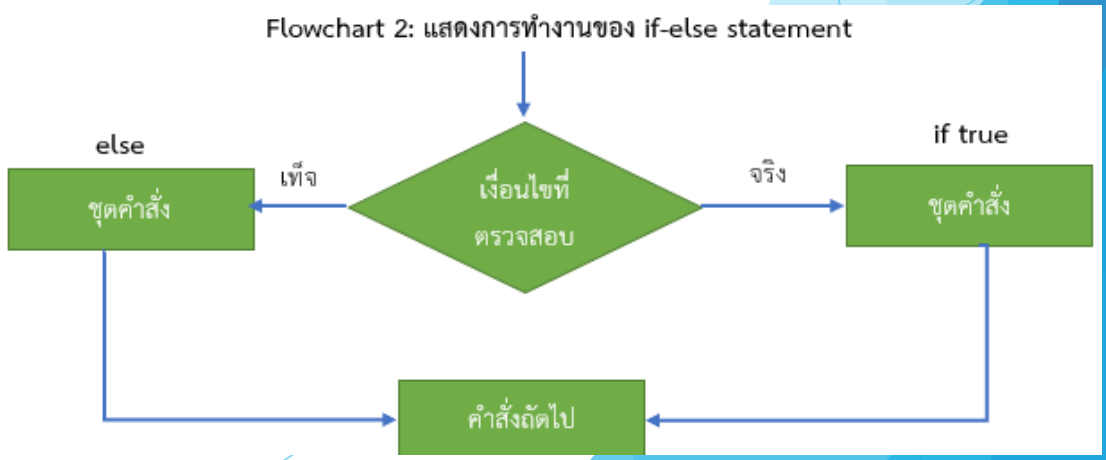
1. เงื่อนไขทางเดียว (if)

```
if เงื่อนไขทางเลือก :
        ชุดคำสั่ง
```



2. เงื่อนไขสองทาง (if – else)

```
if เงื่อนไข:
        คำสั่งที่จะให้ทำเมื่อเงื่อนไขเป็นจริง
else :
        คำสั่งที่จะให้ทำเมื่อเงื่อนไขเป็นเท็จ
```



การทำงานตามเงื่อนไข (ต่อ)

3. เงื่อนไขหลายทางเลือก (if – elif - else)

if เงื่อนไข:

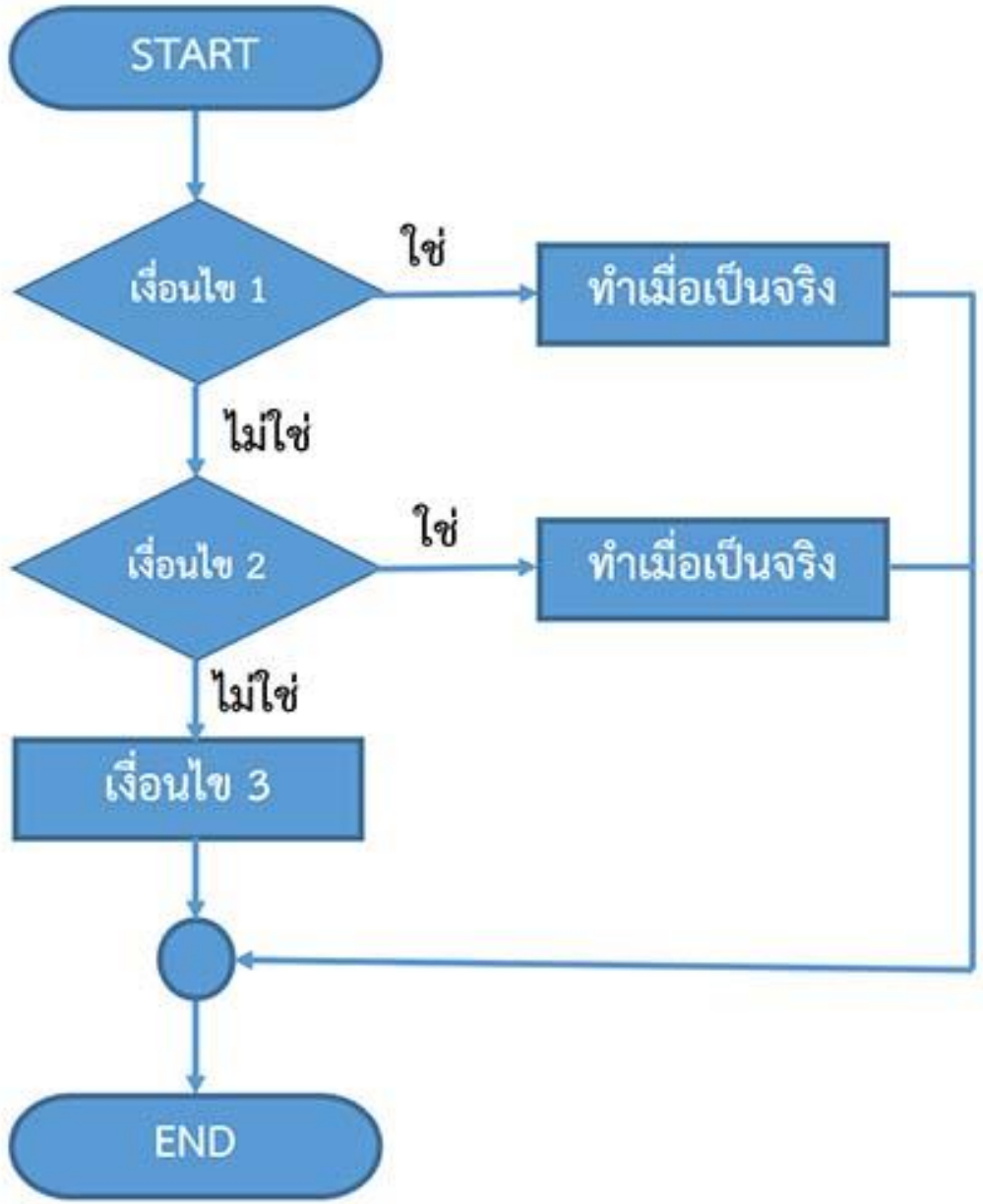
คำสั่งที่จะให้ทำเมื่อเงื่อนไขเป็นจริ

elif เงื่อนไข:

คำสั่งที่จะให้ทำเมื่อเงื่อนไขเป็นจริ

else :

คำสั่งที่จะให้ทำเมื่อเงื่อนไขเป็นเท็



การวนซ้ำ

การวนซ้ำ (For Loop)

จะทำซ้ำเรื่อยๆ ตามจำนวนรอบ เมื่อครบตามจำนวนรอบแล้วจะหลุดจากการทำซ้ำ

```
for ตัวแปร in ลิสต์ :
    ชุดคำสั่ง # ค่าตัวแปรที่ต้องการให้ทำซ้ำ
              # ค่าใน ตัวแปร จะเปลี่ยนไปเรื่อยๆ ตามลำดับใน ลิสต์
```

ตัวอย่าง เช่น

```
for c in 'word' :
    print (c)
```

และ

```
for a in range (5) :
    print (a)
```

การวนซ้ำ

การวนซ้ำ (While Loop)

จะทำซ้ำจนจะครบตามจำนวนรอบ ที่กำหนดไว้ คล้ายๆ กับ for

```
while นิพจน์ :
    ข้อความที่สั่ง (s)
    ข้อความที่จะแสดง
```

ตัวอย่าง เช่น

```
a = 0
while (a <=5) :
    a = a +1
    print (“Count”, count)
```

การวนซ้ำ

คำสั่ง Break

เป็นคำสั่งที่ใช้ร่วมกับคำสั่งวนซ้ำ เมื่อมีการประมวลผลคำสั่ง break จะมีผลให้โปรแกรมออกจากการวนซ้ำทันที

```
while นิพจน์ :
    ข้อความที่สั่ง (s)
    break
    ข้อความที่จะแสดง
```

ตัวอย่าง เช่น

```
target =65
print ('*** Guessing a number ***')
while (True):
    num = int(input('Input your guess number : '))
    if (num == target):
        break
    else:
        print ('Your guess is in incorrect, try again')
print ('คุณทำสำเร็จแล้ว ขอแสดงความยินดีด้วย')
```

การวนซ้ำ

คำสั่ง Continute

เป็นคำสั่งที่ใช้ร่วมกับคำสั่งวนซ้ำ เมื่อมีการประมวลผลคำสั่ง continute จะมีผลให้โปรแกรมทำคำสั่งถัดไปตามเงื่อนไขที่กำหนดไว้

```
while นิพจน์ :
    ข้อความที่สั่ง (s)
    continute
    ข้อความที่จะแสดง
```

ตัวอย่าง เช่น

```
num = int(input('Input a number : '))
count = 0
while (count < num):
    count = count + 1
    if (count % 2 == 0):
        continue
    print(count)
print('End Program')
```

การเขียนแบบฟังก์ชัน

ฟังก์ชัน

ฟังก์ชัน (function) หรือโปรแกรมย่อย (subroutine) เป็นกลุ่มคำสั่งที่ประกอบกันขึ้นเป็น ฟังก์ชันจะทำหน้าที่อย่างใดอย่างหนึ่งโดยเฉพาะ

การสร้างฟังก์ชันทำได้ดังนี้

```
def ชื่อฟังก์ชัน ():
    กลุ่มของคำสั่งที่จะให้ทำงานในฟังก์ชันนี้
```

ตัวอย่างการสร้างฟังก์ชั่น

```
def show_info():
    print("My name is Ruamchart  Chaina")
    print("My classroom is 631")
    print("My number is 99")

def triangle_area():
    base = int(input("Input base : "))
    hight = int(input("Input hight : "))
    area = 0.5*base*hight
    print("Area of Triangle is ", area)

print("First Function")
show_info()
print("Second Function")
triangle_area()
```