Chapter 2| Part1

การเขียนแอพพลิเคชันเบื้องต้นด้วยภาษา

Dart

Edit By Aj.Lattagarn Kuikaew

Outline

ประวัติ

โปรแกรมแรก

โครงสร้าง

คลาส

คลาส MaterialApp

ภาษา Dart

- เป็นภาษาโปรแกรมที่พัฒนาโดย Google
- โครงสร้างภาษาคล้าย Java , C และ C++
- มีความสามารถด้าน OOP คือ มี class และ inheritance ให้ใช้งาน

Dart เป็นภาษากลุ่ม Compiler จำเป็นต้อง Compile ก่อนเอาโปรแกรมไปรัน ไม่เหมือนภาษากลุ่ม Script ที่ใช้ interpreter ในการรันตัว source code ตรงๆ

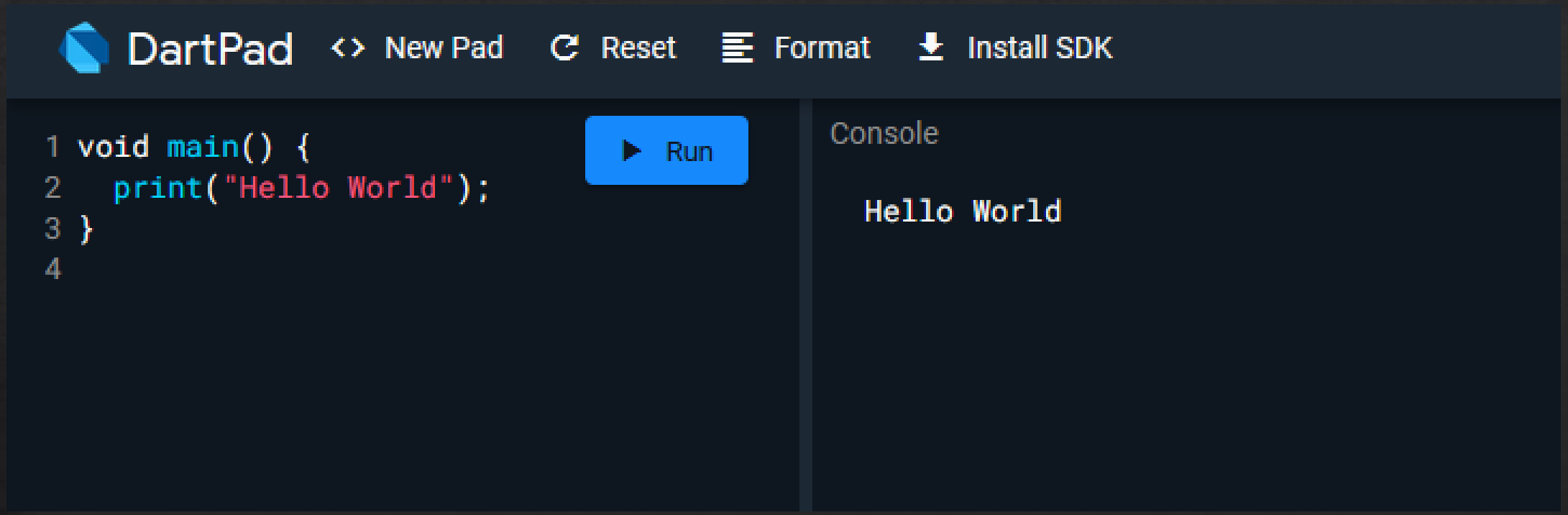
ถ้าพัฒนา Mobile App ด้วย Flutter

รันผ่าน Visual Studio Code โดยลง Extension Code Runner
ต่อจาก Dart & Flutter

ทดลองเขียนโปรแกรมภาษา Dart

http://dartpad.dev

โปรแกรมแรกในภาษา Dart



โครงสร้างภาษา Dart

ภาษา Dart มีฟังก์ชัน **main()** เป็นจุดเริ่มต้นที่จะนำทางเข้าสู่แอพพลิเคชั่น คำสั่งของภาษา Dart จำเป็นต้องใช้ฟังก์ชัน main() เป็นเมธอดสำหรับการดำเนินการ

เมธอดที่เป็นจุดเริ่มต้นการทำงานของโค้ดเราอันดับแรก ๆ

```
void main (){
    // คำสั่งต่าง ๆ
}
```

การแสดงผลข้อมูล

ฟังก์ชัน **print()** จะทำหน้าที่ในการพิมพ์ค่าสตริงหรือค่าของข้อมูลที่ระบุไว้ และส่งผลลัพธ์ไปแสดงผล

```
void main (){
    print(“ข้อความที่ต้องการแสดงผล”);
}
```

คำสั่งเต็มรูปแบบ

```
void main (){
    print('Hello World');
}
```

คำสั่งแบบย่อ

```
void main () => print('Hello World');
```

ภาษา Dart อนุญาตให้เขียนคำสั่งในรูปแบบย่อได้ หากมีคำสั่งภายในฟังก์ชันเพียงคำสั่งเดียว โดยใช้เครื่องหมาย **"=>"** แล้วเขียนต่อด้วยคำสั่งที่ต้องใช้งาน 1 คำสั่ง

การเขียนคำอธิบาย

วิธีที่ 1 โดยใช้เครื่องหมาย **Slash (/ /)** ใช้ในการอธิบาย**คำสั่งสั้น ๆ ในรูปแบบบรรทัดเดียว**

วิธีที่ 2 เขียนคำอธิบายไว้ในเครื่องหมาย **/* ... */** ใช้ในการอธิบาย**คำสั่งยาว ๆ หรือแบบหลายบรรทัด**

ตัวแปร

ตัวแปร คือ พื้นที่หน่วยความจำที่มีการตั้งชื่อเพื่อใช้อ้างอิงในการเก็บข้อมูล ซึ่งอาจจะถูกกำหนดฆ่าหรือถูกแก้ไขข้อมูลได้ มีตัวแปรบางประเภทที่ไม่สามารถเปลี่ยนแปลงค่าที่ถูกจัดเก็บได้ ค่าของข้อมูลที่จัดเก็บจะต้องมีความสัมพันธ์กันกับประเภทของตัวแปรที่ใช้ โดยการประกาศตัวแปรเป็นกระบวนการที่สร้างและกำหนดชื่อของตัวแปร ซึ่งการประกาศตัวแปรไว้ในกรรมในส่วนต่างๆ กัน ก็จะส่งผลในขอบเขตการใช้งานของตัวแปรแตกต่างกันไปด้วย

ชื่อที่ถูกนิยามขึ้นมาเพื่อใช้เก็บค่าข้อมูลสำหรับนำไปใช้งานในโปรแกรม โดยข้อมูลอาจจะประกอบด้วย ข้อความตัวเลข ตัวอักษรหรือผลลัพธ์จากการประมวลผลข้อมูล

ชนิดข้อมูล

ใช้**ระบุลักษณะ**การจัดเก็บข้อมูล การกำหนดชนิดข้อมูลที่เหมาะสม จะช่วยให้การประมวลผลมีความถูกต้อง แม่นยำ ข้อมูลไม่ผิดเพี้ยน

ชนิดข้อมูล	คำอธิบาย	รูปแบบข้อมูล
bool	ค่าทางตรรกศาสตร์	True /False
num	ตัวเลขที่ไม่มีจุดทศนิยม (int)	20
	ตัวเลขที่มีจุดทศนิยม (double)	30.15
string	ข้อความ	“kongruksiam”
List<type>	โครงสร้างข้อมูล	[”มะม่วง”, ”มะละกอ”, ”ส้ม”]
Map <index,value>	โครงสร้างข้อมูล	{firstName:”kong”, lastName:”ruksiam”, age:20};
Dynamic	ตัวแปรเปลี่ยนค่าได้	20,30,15,True,”kong”

รูปแบบตัวแปรในภาษา Dart

- **ตัวแปรแบบ Dynamic Typing** คือชนิดตัวแปรจะเป็นอะไรก็ได้ตามค่าที่ตัวมันเก็บโดยไม่ต้องประกาศชนิดข้อมูล

- **ตัวแปรแบบ Static Typing** ต้องประกาศชนิดข้อมูลในตอนเริ่มต้น เช่น int, double เพื่อบอกว่าตัวแปรนี้จะเก็บข้อมูลชนิดไหน

การประกาศตัวแปร (Static Type)

ชนิดข้อมูล ชื่อตัวแปร ;
ชนิดข้อมูล ชื่อตัวแปร = ค่าเริ่มต้น ;

```
int num;
double money = 20.25;
String fname, lname;
```

แบบหลายตัวแปรในบรรทัดเดียว

ชนิดข้อมูล ชื่อตัวแปร = ค่าเริ่มต้น,ชื่อตัวแปร = ค่าเริ่มต้น

***ตัวแปรที่ประกาศไว้แต่ยังไม่ได้กำหนดค่า จะมีค่าเป็น null โดยอัตโนมัติ

การประกาศตัวแปร (Dynamic Typing)

var (นิยามตัวแปรตามค่าที่กำหนดเริ่มต้น)
var ชื่อตัวแปร;
var ชื่อตัวแปร = ค่าเริ่มต้น ;

```
var money;
var money=100;
money= “200” ; // error
```

แบบหลายตัวแปรในบรรทัดเดียว
var ชื่อตัวแปร = ค่าเริ่มต้น,ชื่อตัวแปร = ค่าเริ่มต้น

***ตัวแปรที่ประกาศไว้แต่ยังไม่ได้กำหนดค่า จะมีค่าเป็น null โดยอัตโนมัติ

การประกาศตัวแปร (Dynamic Typing)

dynamic (นิยามตัวแปรตามค่าที่กำหนดเริ่มต้น)
dynamic ชื่อตัวแปร;
dynamic ชื่อตัวแปร = ค่าเริ่มต้น ;

```
dynamic money;
dynamic money=100;
money= “200” ; // ทำงานผ่าน
```

แบบหลายตัวแปรในบรรทัดเดียว
dynamic ชื่อตัวแปร = ค่าเริ่มต้น,ชื่อตัวแปร = ค่าเริ่มต้น

***ตัวแปรที่ประกาศไว้แต่ยังไม่ได้กำหนดค่า จะมีค่าเป็น null โดยอัตโนมัติ

สรุป Dynamic กับ Var ต่างกันยังไง

- **dynamic** ค่าที่อยู่ในตัวแปรสามารถ**เปลี่ยนค่าหรือชนิดข้อมูลได้เรื่อยๆ** ไม่สามารถตรวจสอบชนิดข้อมูลในตัวแปรนั้นได้

- **var** ค่าที่อยู่ในตัวแปรสามารถ**เปลี่ยนค่าได้แต่เปลี่ยนชนิดข้อมูลไม่ได้**ในภายหลังที่ประกาศตัวแปรออกไป

การประกาศตัวแปรค่าคงที่ Constant

const ชนิดข้อมูล ชื่อตัวแปร;
const ชนิดข้อมูล ชื่อตัวแปร = ค่าเริ่มต้น ;

```
int x = 10;
const int y= x+20; // ไม่สามารถนำค่าจากตัวแปรอื่นมาคำนวณได้
```

***ให้นำค่าทางขวามือของเครื่องหมาย = ไปเก็บไว้ในตัวแปรที่อยู่ด้านซ้ายมือ

การประกาศตัวแปรค่าคงที่ Final

final ชนิดข้อมูล ชื่อตัวแปร;
final ชนิดข้อมูล ชื่อตัวแปร = ค่าเริ่มต้น ;

```
int x = 10;
final int y= x+20; // สามารถนำค่าจากตัวแปรอื่นมาคำนวณได้
```

***ให้นำค่าทางขวามือของเครื่องหมาย = ไปเก็บไว้ในตัวแปรที่อยู่ด้านซ้ายมือ

กฎการตั้งชื่อตัวแปร

- ประกอบด้วยตัวเลข ตัวอักษร เครื่องหมาย
- อักษรตัวแรก**ห้ามขึ้นต้นด้วยตัวเลข**และสัญลักษณ์พิเศษ ยกเว้น **เครื่องหมาย _(Underscore) และ $**
- **ห้ามซ้ำ**กับคำสงวน (Keyword)
- ตัวพิมพ์ใหญ่พิมพ์เล็กถือว่าเป็น**คนละตัวกัน** (Case Sensitive)

ทดสอบ

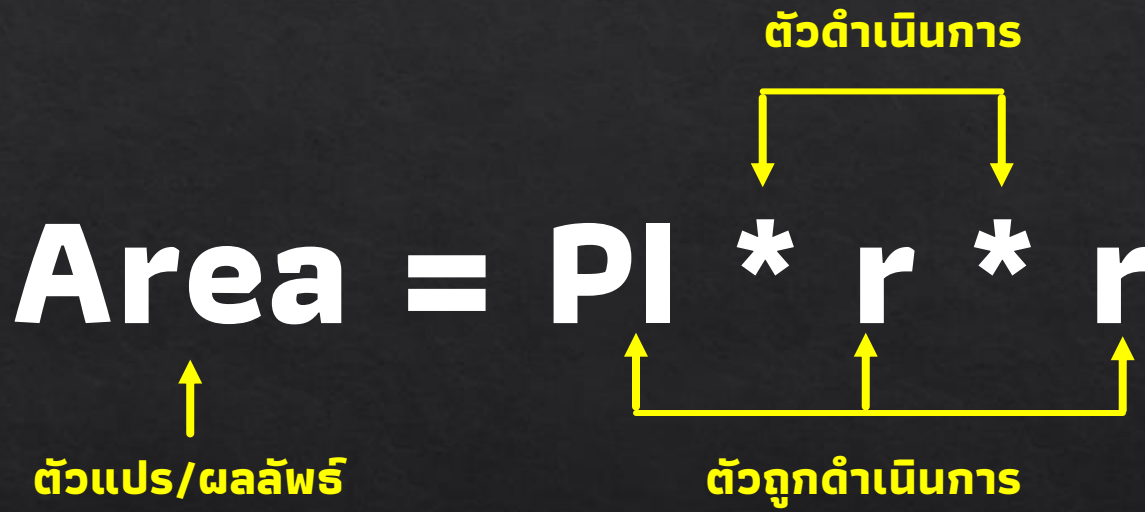
```
var num = 12;
final int id = 101;
final name = “Lattagarn”;
final age = 40 * num;

const int code = 1002;
const student = “ann”;
const years = 4* num;

void main()
{
      print(“ID : $id name: $name age: $age months”);
      print(“ID : $code name: $student age: $years monts”);
}
```

ตัวดำเนินการ (Operator)

กลุ่มของเครื่องหมายหรือสัญลักษณ์ที่ใช้ในการเขียนโปรแกรม



- ตัวดำเนินการ (Operator)
- ตัวถูกดำเนินการ (Operand)

ตัวดำเนินการทางคณิตศาสตร์

Operator	คำอธิบาย
+	บวก
-	ลบ
*	คูณ
/	หาร
%	หารเอาเศษ

ตัวดำเนินการเปรียบเทียบ

Operator	คำอธิบาย
==	เท่ากับ
!=	ไม่เท่ากับ
>	มากกว่า
<	น้อยกว่า
>=	มากกว่าเท่ากับ
<=	น้อยกว่าเท่ากับ

ชนิดข้อมูล bool

ตัวดำเนินการเพิ่มค่า - ลดค่า

Operator	รูปแบบการเขียน	ความหมาย
++ (Prefix)	++a	เพิ่มค่าให้ a ก่อน 1 ค่าแล้วนำไปใช้
++ (Postfix)	a++	นำค่าปัจจุบันใน a ไปใช้ก่อนแล้วค่อยเพิ่มค่า
-- (Prefix)	--b	ลดค่าให้ b ก่อน 1 ค่าแล้วนำไปใช้
-- (Postfix)	b--	นำค่าปัจจุบันใน b ไปใช้ก่อนแล้วค่อยลดค่า

ตัวดำเนินการทดสอบประเภท

Operator	รูปแบบการเขียน	ความหมาย
is	print (num is int);	จะให้ค่าเป็นจริงเมื่อวัตถุนั้นมีค่าตรงตามชนิดของข้อมูล
is!	var res = num is! int;	จะให้ค่าเป็นเท็จเมื่อวัตถุนั้นมีค่าตรงตามชนิดของข้อมูล

ตัวดำเนินการย่อ

Assignment	รูปแบบการเขียน	ความหมาย
+=	x+=y	x=x+y
-=	x-=y	x=x-y
*=	x*=y	x=x*y
/=	x~/=y	x=x~/y
%=	x%=y	x=x%y

ตัวดำเนินการตรรกะ

Operator	คำอธิบาย
&&	AND
||	OR
!	NOT

a	!a	a	b	a && b	a || b
true	false	false	false	false	false
false	true	false	true	false	true
		true	false	false	true
		true	true	true	true

ตัวดำเนินการย่อ

การประกาศ string ขึ้นมาใช้ ต้องกำหนดเนื้อหาหรือค่าอยู่ในเครื่องหมาย
‘ [single quote] หรือ “ [double quote]

```
int x = 10 , y =20;
print(“ค่า x = ”+ x.toString());
print(“ค่า x = $x”);
print(“ผลบวก = ${x+y} ”);
```

```
String name = “Lattagarn”;
String surname = “Kuikaew”;
String myName = name + “ ” + surname;
print(myName);
```

ตัวดำเนินการ string จะใช้เครื่องหมาย + เพื่อเชื่อมสตริง

โครงสร้างควบคุม [Control Structure]

กลุ่มคำส่งที่ใช้ควบคุมการทำงานของโปรแกรม

- **แบบลำดับ [Sequence]**
- **แบบมีเงื่อนไข [Condition]**
- **แบบทำซ้ำ [Loop]**

แบบมีเงื่อนไข [Condition]

กลุ่มคำส่งที่ใช้ตัดสินใจในการเลือกเงื่อนไขต่างๆ ภายในโปรแกรมมาทำงาน

- **if**
- **Switch..Case**

รูปแบบคำสั่งแบบเงื่อนไขเดียว

- **if statement**

เป็นคำส่งที่ใช้กำหนดเงื่อนไขในการตัดสินใจทำงานของโปรแกรม
ถ้าเงื่อนไขเป็นจริงจะทำตามคำสั่งต่างๆ ที่กำหนดภายใต้เงื่อนไขนั้นๆ

```
if(เงื่อนไข){
        คำสั่งเมื่อเงื่อนไขเป็นจริง ;
}
```

ตัวอย่าง

```
if(a > 10){
      print(“มากกว่า”);
}
```

ผลลัพธ์

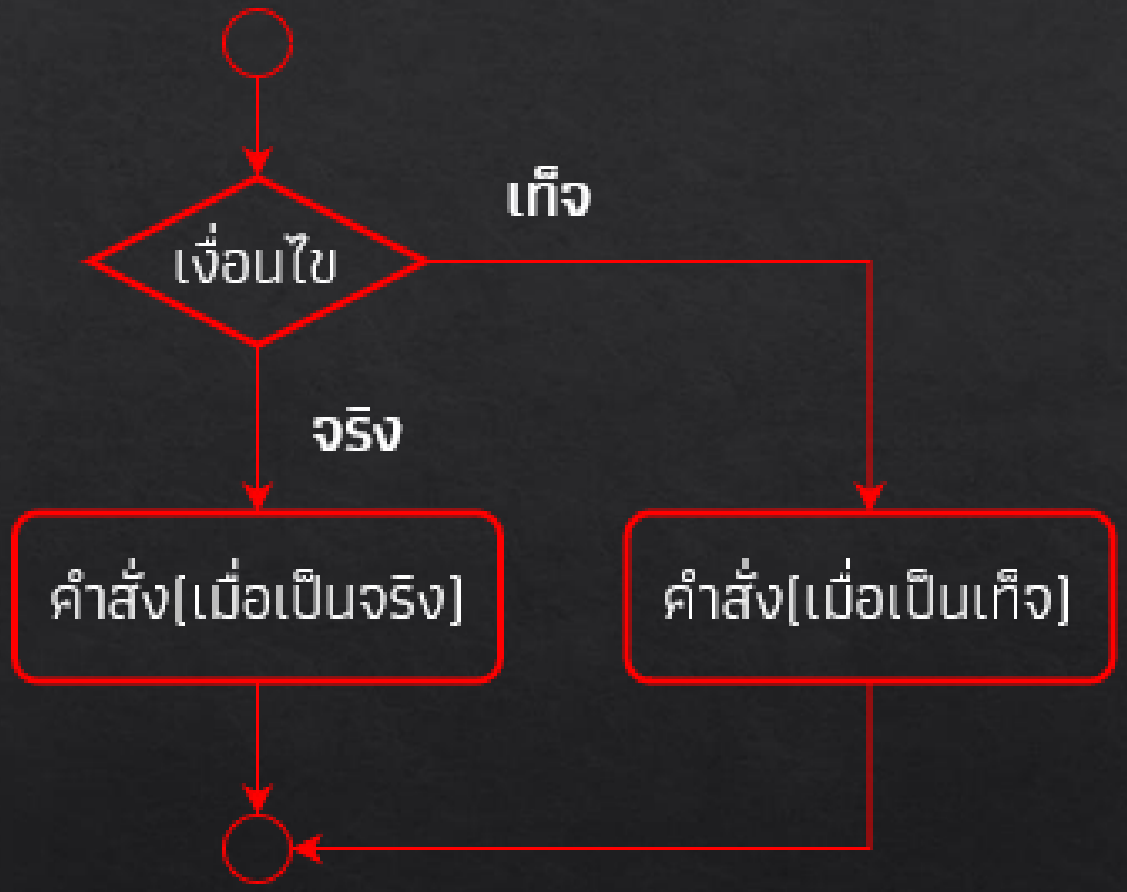
```
มากกว่า
```

รูปแบบคำสั่งแบบ 2 เงื่อนไข

```
if(เงื่อนไข){
        คำสั่งเมื่อเงื่อนไขเป็นจริง ;
}else{
        คำสั่งเมื่อเงื่อนไขเป็นเท็จ ;
}
```

ตัวอย่าง

```
if(a > 10){
      print(“มากกว่า”);
}else{
      print(“น้อยกว่าหรือเท่ากัน”) ;
}
```



ผลลัพธ์

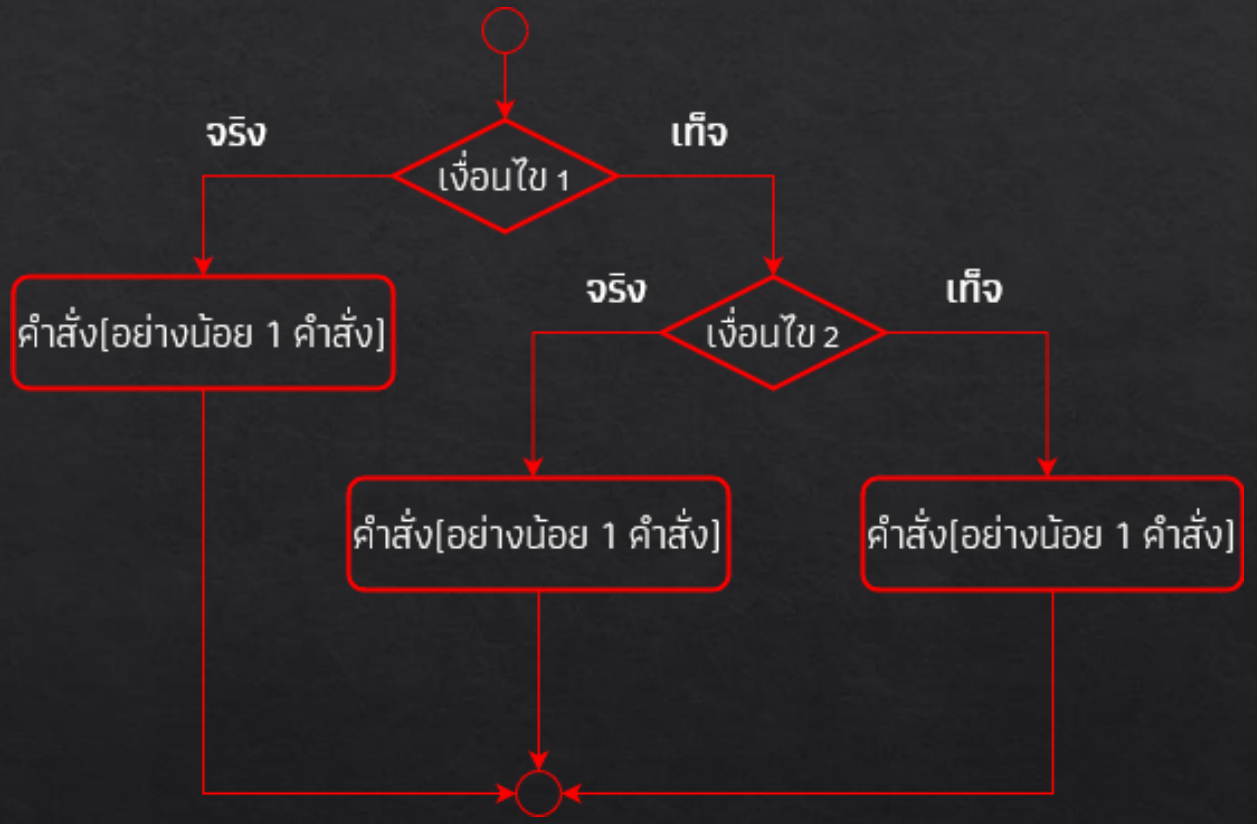
น้อยกว่าหรือเท่ากัน

รูปแบบคำสั่งแบบหลายเงื่อนไข

```
if(เงื่อนไขที่ 1){
    คำสั่งเมื่อเงื่อนไขที่ 1 เป็นจริง ;
}else if(เงื่อนไขที่ 2){
    คำสั่งเมื่อเงื่อนไขที่ 2 เป็นจริง ;
}else if(เงื่อนไขที่ 3){
    คำสั่งเมื่อเงื่อนไขที่ 3 เป็นจริง ;
}else{
    คำสั่งเมื่อทุกเงื่อนไขเป็นเท็จ ;
}
```

ตัวอย่าง

```
if(a > 10){
      print(“มากกว่า”);
}else if(a = 10){
      print(“เท่ากัน”) ;
}else{
      print(“น้อยกว่า”) ;
}
```



ผลลัพธ์

เท่ากัน

รูปแบบคำสั่งแบบมีเงื่อนไข

- **Switch..Case**

Switch เป็นคำสั่งที่ใช้กำหนดเงื่อนไขคล้ายๆกับ if แต่จะเลือกเพียงหนึ่งทางเลือกออกมาทำงาน โดยนำค่าในตัวแปรมากำหนดเป็นทางเลือกผ่านคำส่ง case

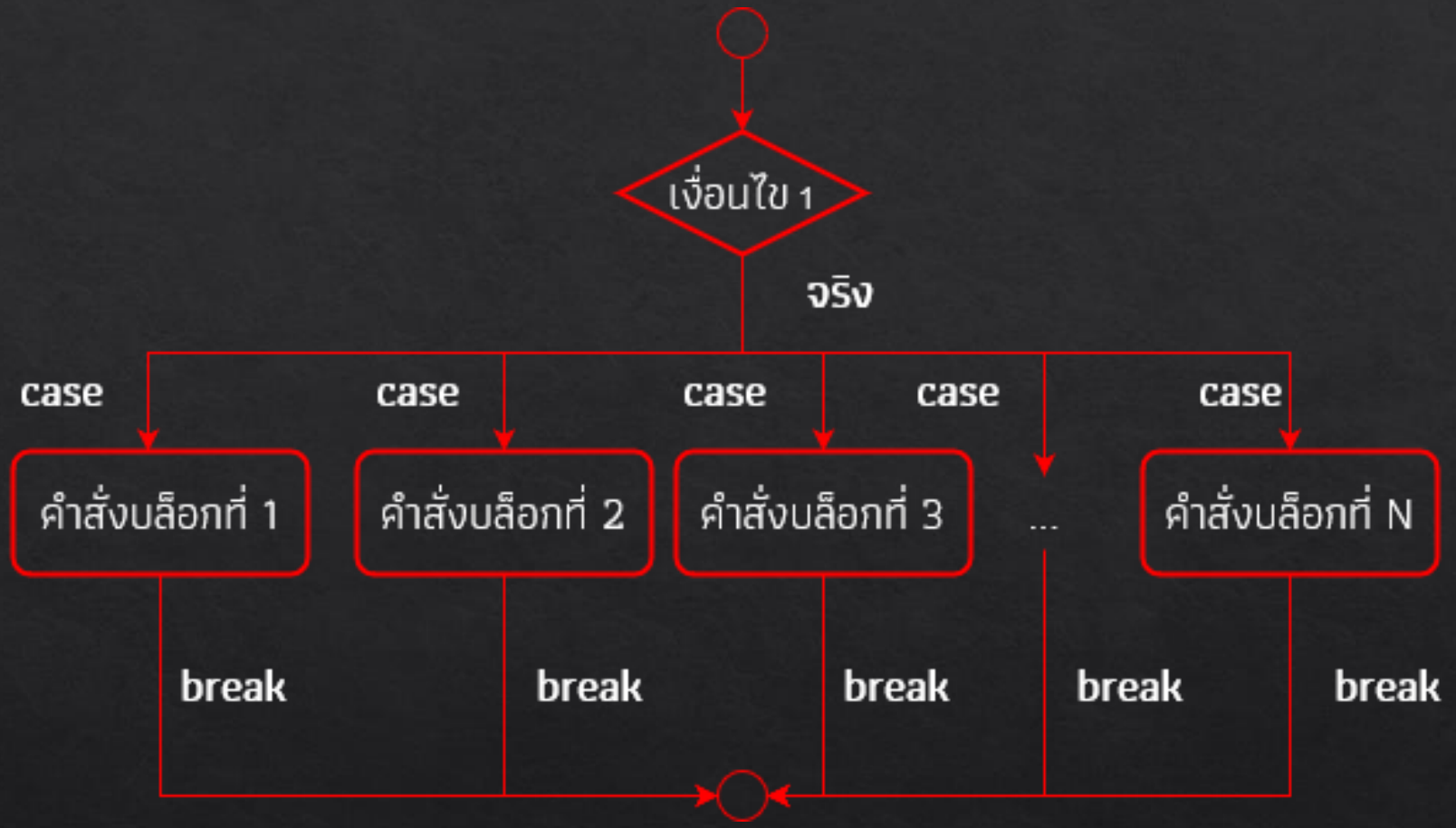
```
switch(งที่ต้องการตรวจสอบ){
    case ค่าที่ 1 :
        คำสั่งที่ 1;
    break;
    case ค่าที่ 2 :
        คำสั่งที่ 2;
    break;
    .....
    case ค่าที่ N :
        คำสั่งที่ N;
    break;
default : คำสั่งเมื่อไม่มีค่าที่ตรงกับที่ระบุใน case
```

*****คำสั่ง break**

จะทำให้โปรแกรมกระโดดออกไปทำงานนอกคำสั่ง switch ถ้าไม่มีคำสั่ง break โปรแกรมจะทำงานคำสั่งต่อไปเรื่อยๆ จนจบการทำงาน

ตัวอย่าง

```
var menu = 2;
switch(menu){
      case  1:
            print(“ข้าวผัด”);
      break;
      case  2:
            print(“ผัดหมี่”);
      break;
      default:{
            print(“กดผิด”);
      break;
}
```



แบบทำซ้ำ (Loop)

กลุ่มคำส่งที่ใช้ในการวนรอบ (loop) โปรแกรมจะทำงานไปเรื่อยๆ จนกว่าเงื่อนไขที่กำหนดไว้จะเป็นเท็จ จึงจะหยุดทำงาน

- **For**
- **While**
- **Do..While**

คำสั่ง For

- **For Loop**

เป็นรูปแบบที่ใช้ในการตรวจสอบเงื่อนไข มีการกำหนดค่าเริ่มต้น และเปลี่ยนค่าไปพร้อมๆกัน เมื่อเงื่อนไขในคำสั่ง for เป็นจริงก็จะทำงานตามคำสั่งที่แสดงไว้ภายในคำสั่ง for ไปเรื่อยๆ

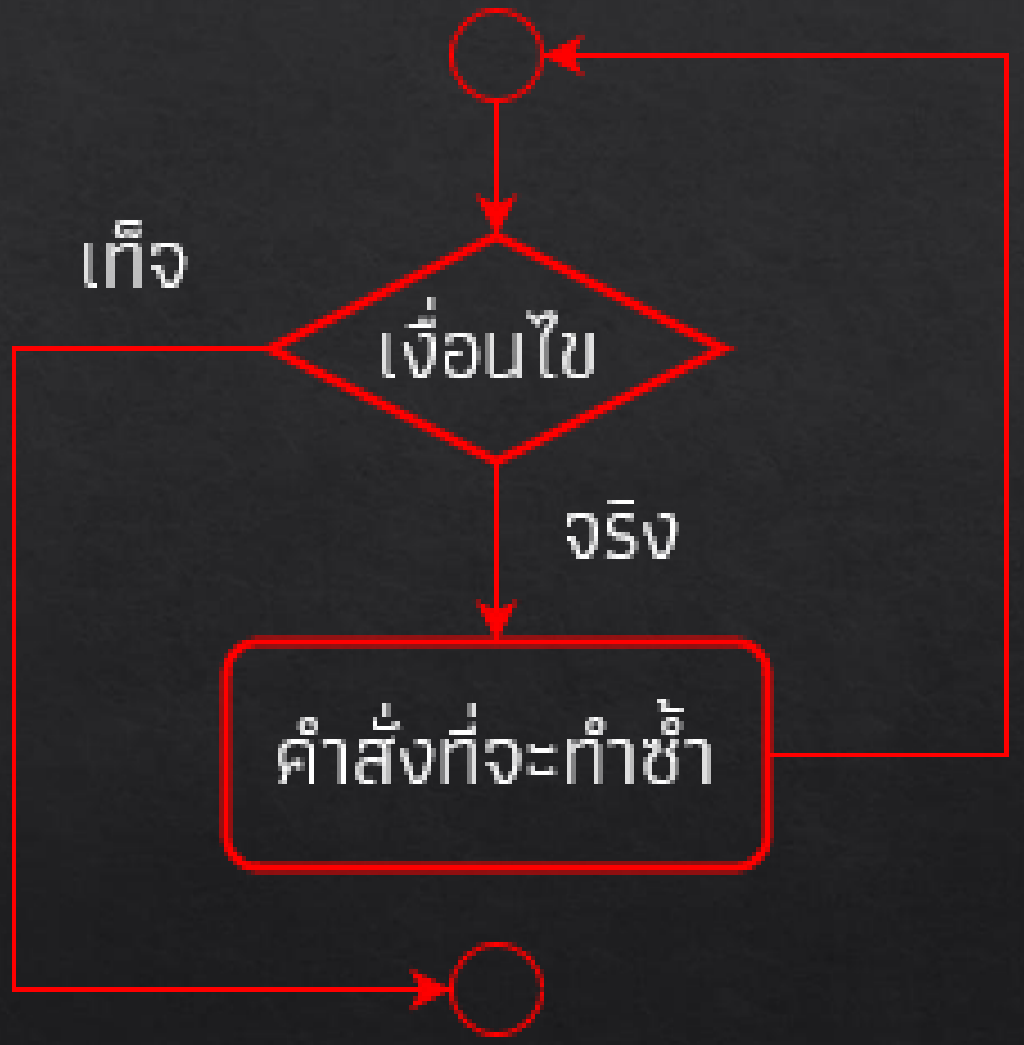
```
for(ค่าเริ่มต้นของตัวแปร; เงื่อนไข; เปลี่ยนแปลงค่าตัวแปร) {
    คำสั่งที่จะทำซ้ำเมื่อเงื่อนไขเป็นจริง ;
}
```

ตัวอย่าง

```
for(int i = 10; i < 100; i +=10){
        print(“i = $i”);
}
```

ผลลัพธ์

```
i = 10
i = 20
i = 30
i = 40
i = 50
i = 60
i = 70
i = 80
i = 90
i = 100
```



คำสั่ง While

- **While Loop**

จะทำงานตามคำสั่งภายใน while ไปเรื่อยๆ เมื่อเงื่อนไขที่กำหนดเป็นจริง

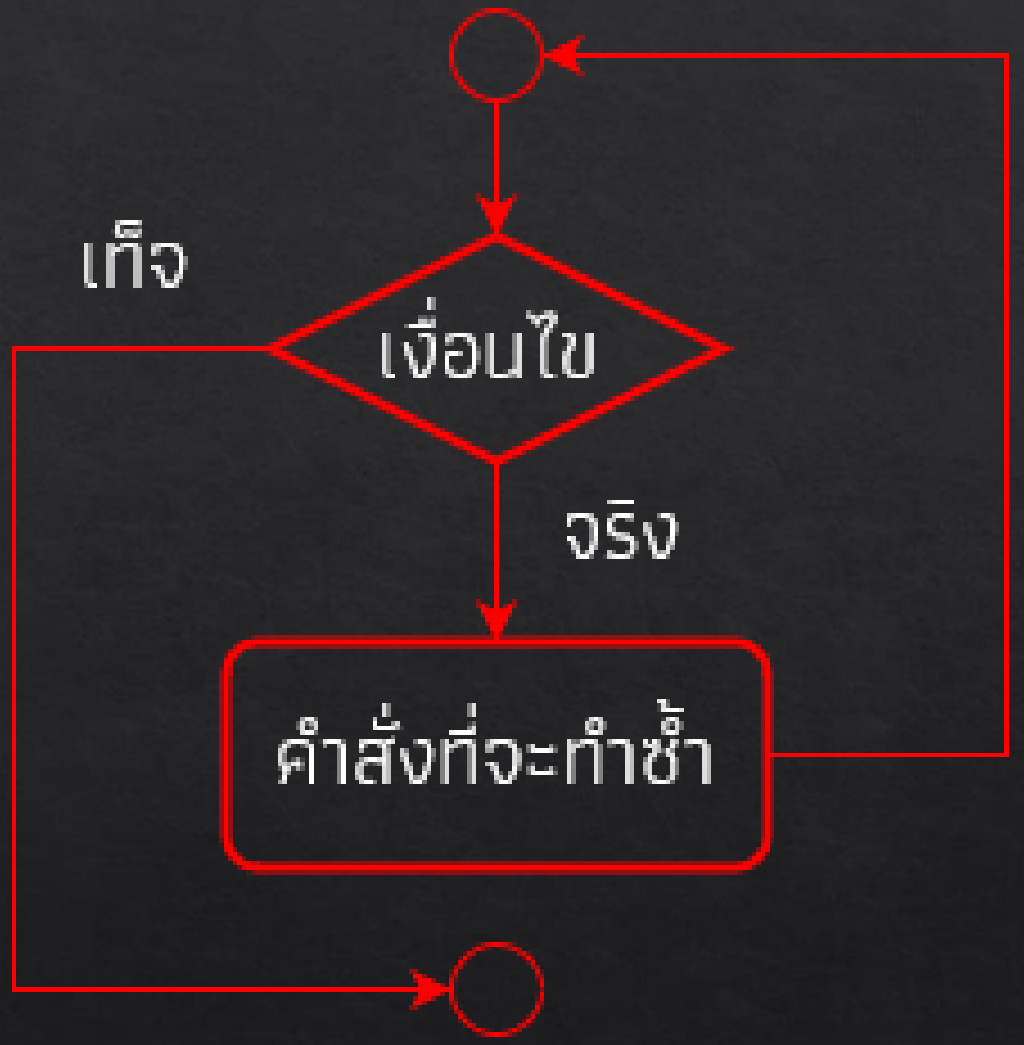
```
while(เงื่อนไข){
        คำสั่งที่จะทำซ้ำเมื่อเงื่อนไขเป็นจริง ;
}
```

ตัวอย่าง

```
int num = 1;
while(num <= 5){
        print(“num = $num”);
        num++;
}
```

ผลลัพธ์

```
num = 1
num = 2
num = 3
num = 4
num = 5
```



คำสั่ง Do..While

- **Do..while Loop**

โปรแกรมจะทำงานตามคำสั่งอย่างน้อย 1 รอบ เมื่อทำงานเสร็จจะมาตรวจสอบเงื่อนไขที่คำสั่ง while ถ้าเงื่อนไขเป็นจริงจะวนกลับขึ้นไปทำงานที่คำสั่งใหม่อีกรอบ แต่ถ้าเป็นเท็จจะหลุดออกจากลูป

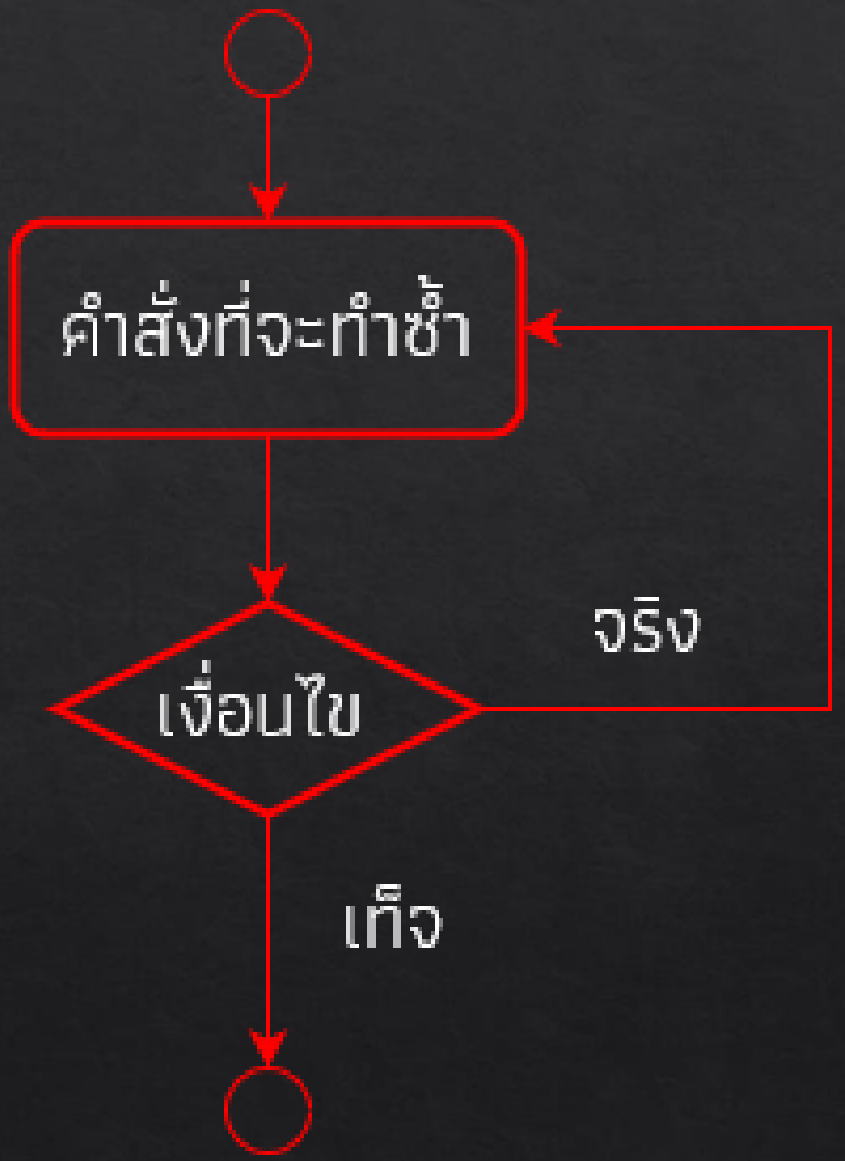
```
do{
        คำสั่งที่จะทำซ้ำเมื่อเงื่อนไขเป็นจริง ;
} while(เงื่อนไข);
```

ตัวอย่าง

```
int num = 1;
do{
        print(“num = $num”);
        num++;
} while(num <= 5);
```

ผลลัพธ์

```
num = 1
num = 2
num = 3
num = 4
num = 5
```



คำสั่งที่เกี่ยวข้องกับ Loop

- **break** ถ้าโปรแกรมพบคำสั่งนี้ จะหลุดจากการทำงานในลูปทันที เพื่อไปทำคำสั่งอื่นที่อยู่นอกลูป
- **continue** คำสั่งนี้จะทำให้หยุดการทำงานแล้วย้อนกลับไปเริ่มต้นการทำงานที่ต้นลูปใหม่

ข้อแตกต่างและการใช้งาน Loop

- **For** ใช้ในกรณีรู้จำนวนรอบที่ชัดเจน
- **While** ใช้ในกรณีที่ไม่รู้จำนวนรอบ
- **Do..while** ใช้ในกรณีที่อยากให้ลองทำก่อน 1 รอบแล้วทำซาไปเรื่อยๆ ทราบเท่าที่เงื่อนไขเป็นจริง

Reference

- KongRuksiam Official. https://www.youtube.com/c/KongRuksiamOfficial
- เอกรินทร์ วทัญญูเลิศสกุล. (2563). พัฒนา Mobile App ด้วย Flutter & Dart. โปรวิชั่น, บจก.
- จีราวุธ วารินทร์. (2564). ต่อยอดพัฒนาโมไบล์แอพด้วย Flutter + Firebase. ซิมพลิฟาย, สนพ.