Dart Programming Language

Mr. Thinnakorn Sathorn

Dart คืออะไร

Dart เป็นภาษาที่ออกแบบมาให้พัฒนาแอปบนแพลทฟอร์มต่าง ๆ ให้รวดเร็ว ช่วยใช้เขียนโปรแกรมในหลายๆ แพลทฟอร์มได้ง่ายและสะดวกยิ่งขึ้น มีความยืดหยุ่นทำให้สามารถ compile เป็นแอปพลิเคชันสำหรับ การทำงานได้ในหลาย ๆ แพลทฟอร์ม

เหมาะสำหรับการพัฒนาแอปในฝั่ง Client โดยมี คุณสมบัติอย่างเช่น Hot reload เพื่อช่วยให้นักพัฒนาเห็นความเปลี่ยนแปลงทันทีเมื่อแก้ไขโค้ดและยังรองรับการ compile ไปยังแพลทฟอร์มต่างๆ ทั้งmobile, desktop was web

Dart Type

ตัวแปร	คำอธิบาย
Int	เลขจำนวนเต็ม
Double	เลขทศนิยม
Number	เลขจำนวนเต็ม และเลขทศนิยม
Bool	ค่าทางตรรกศาสตร์
String	ข้อความ หรือตัวอักษร

Dart Type (ต่อ)

ตัวแปร	คำอธิบาย
Map<index,value>	ชนิดข้อมูลที่เป็น array โดยมี key เป็นตัวเลขเรียงเป็นลำดับ เป็น base zero หรือเริ่มต้นที่ 0
List<type>	ชนิดข้อมูลที่เป็น object โดยมาเป็นชุดข้อมูลที่มี key และ value จับคู่กัน
Dynamic	ตัวแปรที่สามารถเปลี่ยนแปลงค่าได้

ตัวแปรชนิดพิเศษในภาษา Dart

ตัวแปร	คำอธิบาย
Var	เป็นการละเว้น Type เอาไว้ให้โปรแกรมกำหนดให้ (ตาม Value)
Final	เหมือน Var แต่ไม่สามารถเปลี่ยนแปลงค่าได้
Const	ค่าคงที่

ข้อแตกต่างระหว่าง Dynamic vs Var คือ

Dynamic เป็นตัวแปรเก็บค่าชนิดไหนก็ได้ เปลี่ยนแปลงได้เรื่อย ๆ การใช้ Dynamic มีความเสี่ยงทำให้เกิด Runtime error ได้เพราะ Compiler ไม่สามารถช่วยเช็คชนิดของตัวแปรได้

var จะเป็นการกำหนดชนิดตัวแปร โดยดูชนิดตัวแปรจาก value หลังจากนั้นตัวแปรจะถูกกำหนดเป็น type นั้นไปตลอด ไม่สามารถเปลี่ยนแปลงได้แล้ว ๆ

ข้อแตกต่างระหว่าง final vs const คือ

final เป็นการกำหนดว่าตัวแปรที่ไม่สามารถเปลี่ยนแปลงค่าได้ ซึ่งเป็นตัวแปรประเภท runtime ดังนั้นสามารถกำหนดค่า final จากตัวแปรหรือฟังก์ชันอื่นได้

const เป็นการประกาศตัวแปรที่ไม่ต้องการเปลี่ยนแปลงค่า หากประกาศไปแล้วจะไม่สามารถแก้ไขค่าตัวแปร นั้นซ้ำได้อีก

ตัวอย่างการประกาศตัวแปร

ไม่ระบุ Type

```
void main() {
        var a = "Hello world"; //string
        var b = 9876; //int
        var c = 1.5; //double
        var d = true; //boolean
        dynamic e = true;
        print(a.runtimeType); //check type
}
```

ระบุ Type

```
void main() {
        String a = "Hello world";
        int b = 9876;
        double d = 1.5;
        boole = true;
}
```

คำสั่งในการรัน dart บน Terminal vs code

หากต้องการรันคำสั่งเพื่อทำงานของภาษา dart บน Terminal ใน vs code ให้ทำพิมพ์คำสั่ง dart ตามด้วยชื่อไฟล์ที่เราตั้งไว้ดังตัวอย่างด้านล่าง

```
Warning: PowerShell detected that you might be using a screen
you want to re-enable it, run 'Import-Module PSReadLine'.

PS C:\Users\thinn\Downloads\dart> dart test.dart
String
PS C:\Users\thinn\Downloads\dart>
```

dart test.dart

คำสั่ง ตามด้วยชื่อไฟล์

การใช้งาน String Methods

- length นับจำนวน String
- subString(start, [end]) เลือกส่วนของ String โดยระบุตำแหน่งเริ่มต้นที่ต้องการตัด ส่วนตำแหน่งสิ้นสุด จะระบุหรือไม่ก็ได้
- split(object) แยก String ออกเป็นส่วน ๆ แล้วคืนค่าเป็น List โดยกำหนดว่าจะให้แยก String ด้วยอะไร
- toLowerCase() ปรับ String เป็นตัวพิมพ์เล็กทั้งหมด
- toUpperCase() ปรับ String เป็นตัวพิมพ์ใหญ่ทั้งหมด

ตัวอย่างการใช้งาน String Methods

```
void main() {
        String message = 'hello dart';
        print('lenght is $message.length');
        print('substring is ${message.substring(2)}');
        print('split is ${message.split(' ')}');
        print('toLowerCase is ${message.toLowerCase()}');
        print('toUpperCase is ${message.toUpperCase()}');
}
```

Operator in Dart

Operator

Operator คือ ตัวดำเนินการทางคณิตศาสตร์ที่ใช้ประเมินผล (Evaluate) หรือกำหนดค่า (Assign) แบ่งออกเป็นกลุ่มตามการใช้งานได้ดังนี้

- Arithmetic Operators
- Equality and Relational Operators
- Type test Operators
- Bitwise Operators
- Assignment Operators
- Logical Operators

Arithmetic Operators Operator

Arithmetic Operators ใน Dart เป็นการเพิ่ม และลดค่าที่ละ 1 ให้กับตัวแปรที่ได้ประกาศไว้ ตัวอย่างการใช้งานดังโค้ด

```
void main() {
        int number = 10;
        print(number++);
        print(number--);
        print(++number);
        print(--number);
}
```

Equality and Relational Operators

Equality and Relational Operators เอาไว้กำหนดความสัมพันธ์ระหว่างตัวแปรสองตัว เช็คค่าตัวแปร ทำให้ได้คำตอบออกมาเป็น Boolean (True, False) ได้แก่ > (มากกว่า), < (น้อยกว่า), => (เท่ากับหรือมากกว่า), =< (เท่ากับหรือน้อยกว่า), == (เท่ากับ), != (ไม่เท่ากับ)

```
void main() {
        int number = 10;
        int number2 = 8;
        print(number > number2);
        print(number < number2);
        print(number >= number2);
        print(number <= number2);
        print(number == number2);
        print(number != number2);
}
```

Type test Operators

Arithmetic Operators Operator ใน Dart เป็นการเพิ่ม และลดค่าที่ละ 1 ให้กับตัวแปรที่ได้ประกาศไว้ ตัวอย่างการใช้งานดังโค้ด

```
void main() {
        int number = 10;
        print(number is int);
        print(number is! int);
}
```

Bitwise Operators

Bitwise Operators คือ ตัวดำเนินการแบบบิต เป็นตัวดำเนินการที่ใช้กับชนิดข้อมูลเลขจำนวนเต็ม (integer) เท่านั้น โดยจะแปลงเป็นเลขฐาน 2 (8-bit) ก่อนดำเนินการ หลังจากนั้น จะแปลงค่ากลับเป็นฐาน 10 กลับมา ซึ่งมีตัวเนินการได้แก่ & (AND) เป็นการหาเลข 1 ที่ตำแหน่ง(บิด)เดียวกัน มีเงื่อนไขดังนี้ ถ้า 1&1 เป็น 1, ถ้า 1&0 เป็น 0, ถ้า 0&1 เป็น 0, ถ้า 0&0 เป็น 0

Operator	Meaning
&	AND
|	OR
^	XOR
~expr	Unary bitwise complement (0s become 1s; 1s become 0s)
<<	Shift left
>>	Shift right
>>>	Unsigned shift right

Assignment Operators

Assignment Operators เป็นตัวดำเนินการที่เป็นการกำหนดค่าให้ตัวแปร มีดังนี้

= (Simple Assignment) คือ การกำหนดค่าตัวแปรให้เป็นค่าต่าง ๆ คืนค่าเป็นค่าที่กำหนดให้

?? = กำหนดค่าให้ตัวแปร ถ้าตัวแปรนั้นเป็น null ถ้าตัวแปรนั้นไม่ใช่ null ค่าของตัวแปรนั้นจะคงเดิม

+ = (Add and Assignment) คือ การบวกค่าตัวแปรนั้นด้วยค่าที่กำหนด คืนค่าเป็นตัวแปรที่ผ่านการบวกแล้ว

- = (Subtract and Assignment) คือ การบวกค่าตัวแปรนั้นด้วยค่าที่กำหนด คืนค่าเป็นตัวแปรที่ผ่านการลบแล้ว

* = (Multiply and Assignment) คือ การบวกค่าตัวแปรนั้นด้วยค่าที่กำหนด คืนค่าเป็นตัวแปรที่ผ่านการคูณแล้ว

/ = (Divide and Assignment) คือ การบวกค่าตัวแปรนั้นด้วยค่าที่กำหนด คืนค่าเป็นตัวแปรที่ผ่านการหารแล้ว

~/ (Floored integer division) คือ การหารค่าที่มีทศนิยมแล้วต้องแปลง cast เพื่อให้เป็น integer

% (Division Remainder) คือ การหารเพื่อเอาเศษที่เหลือจากการหารไปใช้งาน โดยที่ทั้งตัวตั้งและตัวหารต้องเป็นเลขจำนวนเต็ม

ตัวอย่างการใช้ Assignment Operators

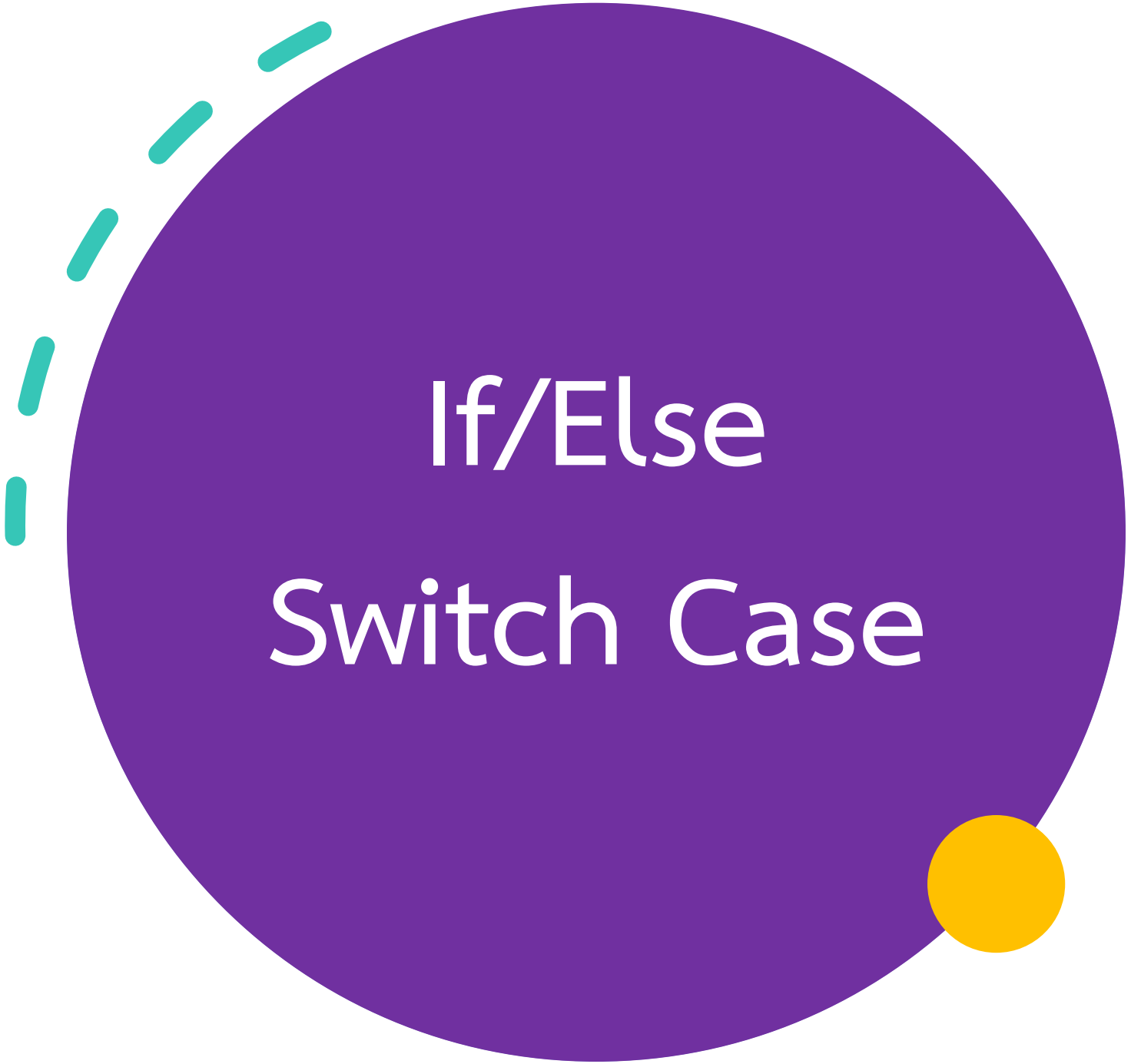
```
void main() {
        int number = 10;
        int number2 = 9;
        print(number + number2);
        print(number - number2);
        print(number * number2);
        print(number / number2);
        print(number ~/ number2);
        print(number % number2);
        print(number ?? number2);
        print(number = number2);
}
```

Logical Operators

Logical Operators คือตัวดำเนินการทางตรรกศาสตร์ เป็นองค์ประกอบที่ใช้ในการประมวลผลและจัดการกับค่าที่เป็นข้อมูลประเภทบูลีน (Boolean) ซึ่งมีค่าเป็นเพียงสองค่าเท่านั้นคือ True (จริง) และ False (เท็จ) โดยมักใช้ในการทำเงื่อนไขและการควบคุมกระบวนการทำงานของโปรแกรม

Operator	ความหมาย
!	กลับค่าระหว่าง True หรือ False
||	เหมือน OR ในตรรกศาสตร์
&&	เหมือน AND ในตรรกศาสตร์

```
void main() {
        int number = 10;
        int number2 = 9;
        print(number > 5 && number2 >5);
        print(number > 10 || number2 >10);
        print(!(number > 15 || number2 >15));
}
```


If/Else
Switch Case

If/Else, Switch Case

กลุ่มคำสั่งที่ใช้ในการควบคุมการทำงานของโปรแกรม โดยในภาษา Dart จะประกอบไปด้วย 3 รูปแบบ

- แบบลำดับ (Sequence) คือ การทำงานจากบนลงล่าง ซ้ายไปขวา
- แบบมีเงื่อนไข (Condition) คือ กลุ่มคำสั่งที่ใช้ตัดสินใจในการเลือกเงื่อนไขต่าง ๆ ภายในโปรแกรมมาทำงาน ซึ่งจะประกอบไปด้วย 2 คำสั่ง คือ if else และ Switch.Case

If เป็นคำสั่งที่ใช้กำหนดเงื่อนไขในการตัดสินใจทำงานของโปรแกรม ถ้าเงื่อนไขเป็นจริงก็จะทำตามคำสั่งต่าง ๆ ที่ได้กำหนดในเงื่อนไขนั้น ๆ

ตัวอย่างการใช้งาน If แบบเงื่อนไขเดียว

```
void main() {
        int number = 10;
        int number2 = 9;
        if (number > number2) {
        print('$number มากกว่า $number2');
        }
        print('จบการทำงาน');
}
```

ตัวอย่างการใช้งาน if แบบ 2 เงื่อนไข

```
void main() {
        int number = 10;
        int number2 = 9;
        if (number > number2) {
                print('$number มากกว่า $number2');
        } else {
                print('$number น้อยกว่า $number2');
        }
}
```

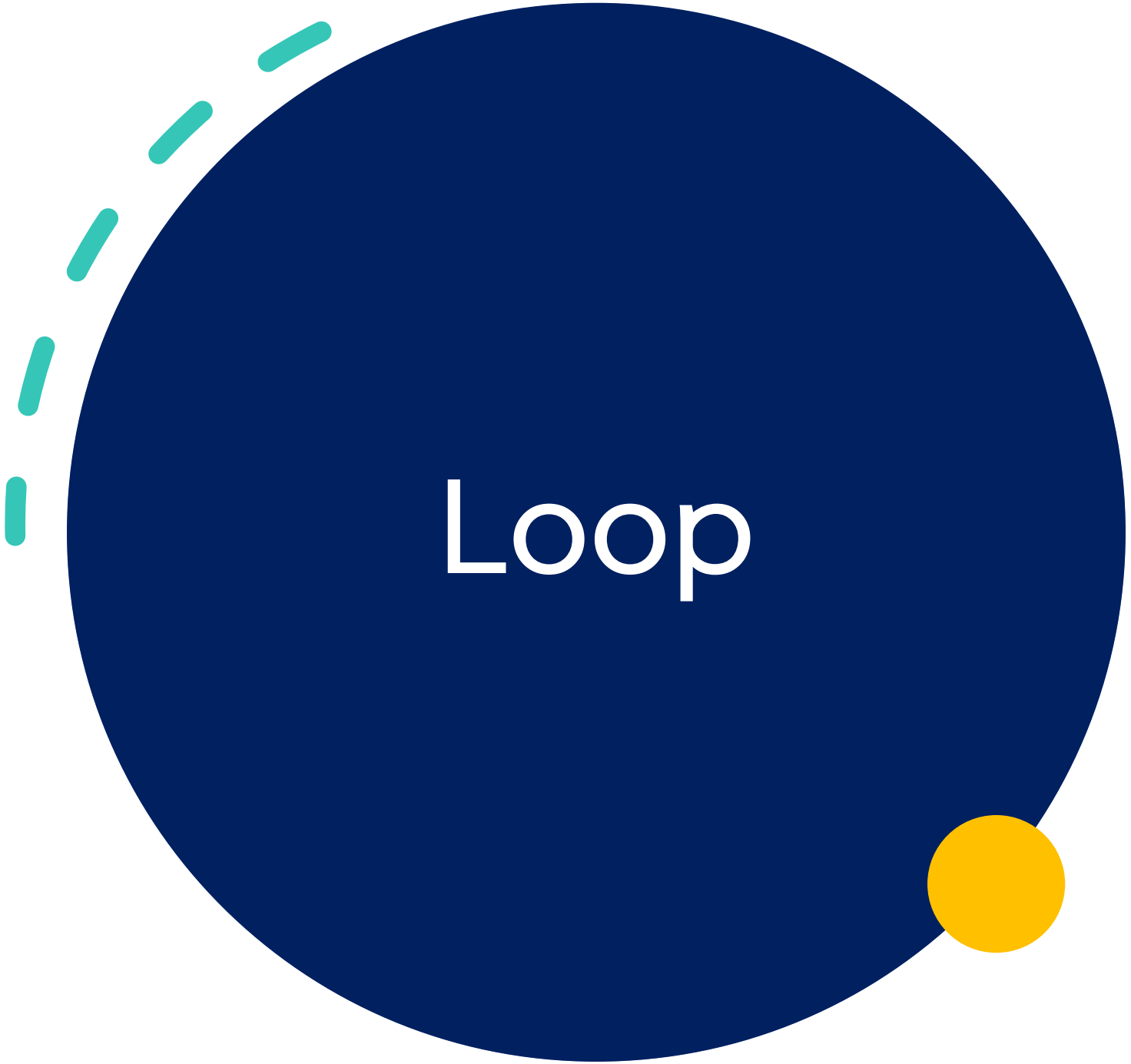
ตัวอย่างการใช้งาน If แบบหลายเงื่อนไข

```
void main() {
        int number = 10;
        int number2 = 9;
        if (number > number2) {
                print('$number มากกว่า $number2');
        } else if (number < number2) {
                print('$number น้อยกว่า $number2');
        } else {
                print('$number เท่ากับ $number2');
        }
}
```

Switch..Case

เป็นคำสั่งที่ใช้กำหนดเงื่อนไขคล้าย ๆ กับ If แต่จะเลือกเพียงหนึ่งทางเลือกออกมาทำงานโดยนำค่าในตัวแปรมากำหนดทางเลือกผ่านคำสั่ง Case ตัวอย่างเช่น เจอ Case ที่ 1 จะให้ทำอะไร Case ที่ 2 จะทำอะไร

```
void main() {
  var score = 2;
  switch (score = 4) {
    case 4:
      print('ดีมาก');
      break;
    case 3:
      print('ดี');
      break;
    case 2:
      print('พอใช้');
      break;
    case 1:
      print('ปรับปรุง');
      break;
    default:
  }
}
```


Loop

Loop

กลุ่มคำสั่งที่ใช้ในการวนวนลูป โดยโปรแกรมจะทำงานไปเรื่อย ๆ จนกว่าเงื่อนไขที่กำหนดนั้นจะเป็นเท็จ โปรแกรมจึงจะหยุดทำงาน โดยในที่นี้จะมีอยู่ 3 คำสั่งคือ While, For และ Do While ซึ่งแต่ละตัวก็จะมีรูปแบบการใช้งานที่แตกต่างกันไป

While Loop จะทำงานภายในคำสั่ง While ไปเรื่อย ๆ เมื่อเงื่อนไขที่กำหนดเป็นจริง เมื่อเงื่อนไขเป็นจริงก็จะทำซ้ำไปเรื่อย ๆ

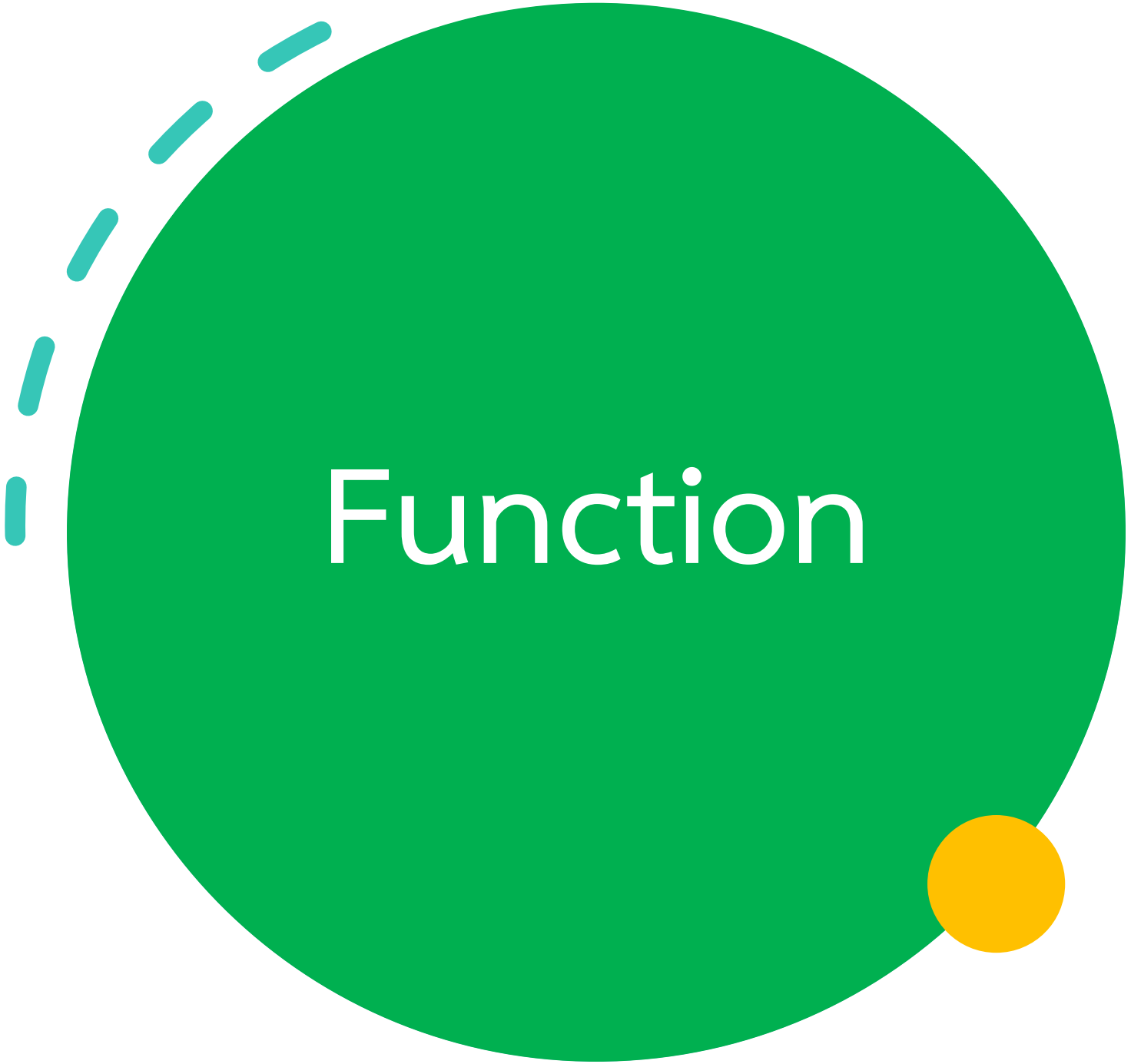
```
void main() {
        var score = 1;
        while (score <= 3) {
        print('Hello dart');
        score++;
 } }
```

Loop (ต่อ)

For Loop เป็นรูปแบบที่ใช้ตรวจสอบเงื่อนไข มีการกำหนดค่าเริ่มต้น และเปลี่ยนค่าไปพร้อม ๆ กัน เมื่อเงื่อนไขที่กำหนดเป็นจริง เมื่อเงื่อนไขเป็นจริงก็จะทำซ้ำไปเรื่อย ๆ

```
void main() {
        for (var i = 1; i <= 5; i++) {
        print('number is $i');
        }
}
```

Do While โปรแกรมจะทำงานตามคำสั่งก่อนอย่างน้อย 1 รอบแล้วมาตรวจสอบเงื่อนไขที่คำสั่ง While ถ้าเป็นจริงก็จะวนกลับขึ้นไปทำงานอีกรอบแล้วถ้าเป็นเท็จก็จะหลุดออกจากลูป


Function

Function

Function เป็นชุดของการทำงาน Function เป็นส่วนหนึ่งของ Code ซึ่งถูกเรียกโดยชื่อของฟังก์ชัน ฟังก์ชันสามารถส่งผ่านข้อมูล (Parameter) เพื่อทำอะไรบางอย่าง และส่งคืนค่า

รูปแบบการใช้งานแบบทั่วไป

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

รูปแบบการแบบย่อ

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

ประเภทของ Function

- Function: No return & No Args
- Function: No return & Args
- Function: return & No Args
- Function: return & Args
- Optional Positional Argument
- Optional Named Argument

Function: No return & No Args

เป็น Function ที่ไม่ต้องรับค่าเข้ามาเพื่อประมวลผล ไม่คืนค่ากลับหลังจากทำงานเสร็จ (void + No return)

```
void main() {
        test1();
}
void test1() {
        print("No Return & No Args Function");
}
```

Function: No return & Args

เป็น Function ที่ไม่ต้องรับค่าเข้ามาเพื่อประมวลผล ไม่คืนค่ากลับหลังจากทำงานเสร็จ (void + No return)

```
void main() {
        test2("Hello", "Dart");
}
void test2(String name, String surname) {
        print("name is $name $surname");
}
```

Function: return & No Args

เป็น Function ที่ไม่ต้องรับค่าเข้ามาเพื่อประมวลผล แต่มีการคืนค่ากลับหลังจากทำงานเสร็จ (type + return)

```
void main() {
        print(test3());
}
String test3() {
        String a = 'Hello Dart';
        return a;
}
```

Function: return & Args

เป็น Function ที่มีการรับค่าเข้ามาเพื่อประมวลผลและมีการคืนค่ากลับหลังจากทำงานเสร็จ

```
void main() {
        print(test4(9));
}
int test4(int a ) {
        return a+50;
}
```

Optional Positional Argument Optional

เป็น optional parameter ที่มีลักษณะดังนี้

- Optional Positional Argument ใช้ [ ] ครอบ Argument ที่ต้องการให้ เป็น Optional

- Optional Positional Argument การส่งค่าจะอิงกับตำแหน่งของ Argument ที่กำหนดไว้ ถ้าส่งค่าไม่ครบหรือไม่ตรงตำแหน่ง ค่าที่ออกมาจะมีโอกาสเพี้ยนได้

```
void main() {
        test5('Hello Dart', 'M');
        test5('Hello Dart', 23, 'M');
}
void test5(name, [age, gender]) {
        print("name is $name age is $age gender is $gender");
}
```

Optional Named Argument

เป็น optional parameter ที่มีลักษณะดังนี้

- Optional Named Argument ใช้ { } ครอบ Argument ที่ต้องการให้เป็น Optional Named Argument

- Optional Named Argument ให้เรียกชื่อ Argument พร้อมทั้งกำหนดค่าด้วย สามารถสลับตำแหน่งกันได้ จะต่างกับ Optional Positional Argument ที่ต้องส่งค่าในพารามิเตอร์ อิงตามตำแหน่งนั้น ๆ

```
void main() {
        test6('Hello Dart', 'M');
        test6('Hello Dart', 23, 'M');
}
void test6(name, [age, gender]) {
        print("name is $name age is $age gender is $gender");
}
```


List

List

ลิสต์ (List) คือ รายการข้อมูลที่สามารถจัดเรียงต่อเนื่องกันอยู่ในรูปแบบของอาร์เรย์ สามารถกำหนดชนิดของข้อมูลที่จัดเก็บได้ทั้งแบบข้อความ ตัวเลข และอื่น ๆ ในการใช้งานพื้นฐานทั่วไปแบ่งออกได้ 2 แบบ

1. ลิสต์แบบขนาดคงที่ Fixed-length (Array) คือ ลิสต์ที่มีจำนวนของสมาชิกที่มีจำนวนคงที่แน่นอน มีรูปแบบที่ใช้ในการ ประกาศค่าดังนี้

var <ชื่อของลิสต์> = new List (ขนาด)

2. ลิสต์แบบปรับเปลี่ยนขนาดได้ Growable (Link List) เป็นลิสต์ที่สามารถปรับเปลี่ยนขนาดหรือจำนวนของสมาชิกในลิสต์ได้ตลอดเวลา จะมีรูปแบบในการประกาศค่าดังนี้

var <ชื่อของลิสต์> = [ ]

คุณสมบัติสำคัญที่ใช้งานเกี่ยวกับลิสต์

คุณสมบัติ	คำอธิบาย	ชนิดข้อมูล
First	คืนค่าข้อมูลสมาชิกในลิสต์รายการลำดับแรก	ตามชนิดที่จัดเก็บ
isEmpty	คืนค่า จริง เมื่อไม่มีสมาชิกในลิสต์	ตรรกะ
isNotEmpty	คืนค่า จริง เมื่อมีสมาชิกในลิสต์ อย่างน้อย 1 ข้อมูล	ตรรกะ
length	คืนค่าขนาดหรือจำนวนสมาชิก	เลขจำนวนเต็ม
reversed	ชนิดข้อมูล ตามชนิดที่จัดเก็บ	ลิสต์

Fixed-length (Array)

List<type> name = List<type>(size);

List<int> list1 = List<int>(5);

List: ประกาศตัวแปร List

type: กำหนดประเภทของตัวแปร

name: ตั้งชื่อตัวแปร

size: กำหนดขนาดที่ต้องการ

การกำหนดค่าเริ่มต้นให้ List

```
void main() {
        var filling = new List.filled(5, 0);
        print("filling is: $filling");
}
```

List.filled(Length, Fill);

Length:ขนาดที่ต้องการ

Fill:ค่าเริ่มต้นที่ต้องการ

Fixed-length (Array) (ต่อ)

```
void main() {
        // ตัวเลข
        List<int> list1 = [1, 2, 3, 4, 5, 6, 7, 8, 9];
        // ตัวอักษร
        List<String> list2 = ['q', 'e', 't', 'u', 'o', 'a', 'd', 'g', 'j', 'l'];
        print(list1);
        print(list2);
}
```

Growable (Link List)

การเพิ่มจำนวนสมาชิกภายใน List

```
void main() {
        List<String> growable = [];
        growable.add(‘a');
        growable.add(‘c');
        print("growable: $growable");
}
```

ชื่อ list.add(ข้อมูลที่ต้องการ)

การแทรกข้อมูลภายใน List

```
void main() {
        List<String> growable = [];
        growable.add('a');
        growable.add('c');
        growable.insert(1, 'b');
        print("growable: $growable");
}
```

ชื่อ list.insert(index, ข้อมูลที่ต้องการ)

Growable (Link List) (ต่อ)

การลบสมาชิกภายใน List

```
void main() {
        List<String> growable = [];
        growable.add('a');
        growable.add('c');
        growable.insert(1, 'b');
        growable.remove('a');
  print("growable: $growable");
}
```

ชื่อ list.remove(ข้อมูลที่อยู่ภายใน List)

Growable (Link List) (ต่อ)

การลบสมาชิกในลิสต์ มีรูปแบบการลบโดยใช้ฟังก์ชัน 4 แบบ คือ

3.1 remove() คือ การลบสมาชิกในลิสต์ที่ละรายการโดยระบุข้อมูลที่ต้องการลบไว้ภายในวงเล็บ หากสมาชิกมีข้อมูลซ้ำหรือตรงกัน จะลบรายการแรกสุดที่มีข้อมูลตรงกันเท่านั้น

3.2 removeAt() คือ การลบสมาชิกในลิสต์ที่ละรายการโดยระบุหมายเลขอินเด็กซ์ที่ต้องการลบไว้ ภายในวงเล็บ

3.3 removeLast() คือ การลบสมาชิกรายการสุดท้ายของลิสต์ รูปแบบการใช้งาน คือ ไม่มีการกำหนด พารามิเตอร์ในขณะใช้งานฟังก์ชันนี้

3.4 removeRange() คือ การลบสมาชิกด้วยการระบุช่วง โดยมีพารามิเตอร์ 2 ค่า ได้แก่ อินเด็กซ์ เริ่มต้น และอินเด็กซ์สุดท้าย

List Sort

การเรียงลำดับข้อมูล

```
void main() {
  var sort = ['B', 'C', 'A', 'D'];
  sort.sort();
  print("sort is: $sort");
}
```

การ Loop ข้อมูลออกมาใช้ร่วมกับ Inline Function มี 2 รูปแบบ

List for Each

```
void main() {
  var sort = ['B', 'C', 'A', 'D'];
  sort.sort();
  print("sort is: $sort");
  sort.forEach((String value) {
    print("value is $value");
  });
}
```

List for Loop

```
void main() {
  var sort = ['B', 'C', 'A', 'D'];
  sort.sort();
  print("sort is: $sort");
  for (String v in sort) {
    print("v is $v");
  }
}
```


Null
Safety

Null Safety

Null Safety ฟีเจอร์ใหม่ในภาษา Dart ตั้งแต่เวอร์ชั่น 2.12 ขึ้นไป Null Safety คือ การตรวจสอบตัวแปรที่เป็นค่า null ถ้าเอา ตัวแปร null ไปใช้งาน จะแสดง Error ตั้งแต่ตอนเขียนโค้ดทันที โดยที่ยังไม่ได้รันโปรแกรม เพราะฉะนั้นต้องกำหนดค่าให้ตัวแปรด้วย

ถ้าจะอนุญาตให้ตัวแปรนั้น เป็นค่า null ได้ ให้ใส่เครื่องหมาย ? หลังชนิดของตัวแปร (Data Type) ตัวอย่างการใช้งานการประกาศตัวแปรให้สามารมีค่าเป็น Null ได้

```
void main() {
        int? age;
        print(age);
}
```

Null Safety เมื่อใช้งานกับตัวแปร List, Set, Map มี 4 รูปแบบ

- ตัวแปรห้ามเป็น null และสมาชิกภายในตัวแปรห้ามมี null
- ตัวแปรห้ามเป็น null แต่สมาชิกภายในตัวแปรมี null ได้
- ตัวแปรเป็น null ได้ แต่สมาชิกภายในตัวแปรห้ามมี null
- ตัวแปรเป็น null ได้ และสมาชิกภายในตัวแปรมี null ได้

ตัวอย่างการประกาศตัวแปร List

```
List<String> list1; // ห้ามเป็น null และสมาชิกใน List ห้ามมีค่า null
List<String?> list2; // ห้ามเป็น null แต่สมาชิกใน List มีค่า null ได้
List<String>? list3; // null ได้ แต่สมาชิกใน List ห้ามมีค่า null
List<String?>? list4; // null ได้ และสมาชิกใน List มีค่า null ได้
```

ตัวอย่างการประกาศตัวแปร List

```
void main(){
  List<String> list1;   // ห้ามเป็น null และสมาชิกใน List ห้ามมีค่า null
  List<String?> list2;   // ห้ามเป็น null แต่สมาชิกใน List มีค่า null ได้
  List<String>? list3;   // null ได้ แต่สมาชิกใน List ห้ามมีค่า null
  List<String?>? list4;  // null ได้ และสมาชิกใน List มีค่า null ได้
  list1 = [];
  list2 = [null, 'Thinnakorn'];
  list3 = ['A'];
  list4 = ['Thinnakorn', null];
  print(list1);
  print(list2);
  print(list3);
  print(list4);
}
```

Thank you