ใบงานการทดลองที่ 2

เรื่อง ภาษา Dart

1. จุดประสงค์

1. เขียนประกาศตัวแปรประเภท List และใช้ Method ของ List ได้
2. เขียน function ที่สร้างขึ้นเอง และ เรียกใช้ function library ได้

2. ทฤษฎี

บทที่ 1.5 ชุดข้อมูล

ชุดข้อมูลในภาษา Dart มี 2 ประเภทหลักๆ คือ

- List เก็บข้อมูลแบบเรียงลำดับ มีค่าเท่ากับ array ในบางภาษา
- Map เก็บข้อมูลแบบ key-value มีค่าเท่ากับ dictionary หรือ hash-table ในบางภาษา

1.5.1 ชุดข้อมูลในแบบลิสต์

List คือชุดข้อมูลที่เก็บข้อมูลแบบเรียงลำดับ ข้อมูลแต่ละตัวใน List เรียกว่าสมาชิก (element) สมาชิกใน List จะถูกเรียงลำดับตามเลขลำดับเริ่มต้นที่ 0

โครงสร้างชุดข้อมูลในแบบลิสต์

```
List<type_list> variable=[value1,value2]; หรือ
List variable = <type_list>[value1,value2];
```

type_list คือ ชนิดข้อมูลของลิสต์
variable คือ ชื่อตัวแปรของลิสต์ที่สร้าง
value คือ ค่าในลิสต์

1.5.1.1 การสร้าง List

การสร้าง List มี 2 วิธีหลักๆ ดังนี้

- การสร้าง List แบบไม่ระบุชนิดข้อมูล สามารถทำได้โดยการใช้ตัวแปรชนิด var หรือ dynamic แล้วกำหนดสมาชิกของ List ในวงเล็บ [ ]

```
var numbers = [-10, 01, 3.141, 7, 10.5, 11, 66.99];
dynamic data= [ true, 0.39, false, “Drink”];
```

- การสร้าง List แบบระบุชนิดข้อมูล สามารถทำได้โดยการใช้คำสั่ง List() แล้ว กำหนดชนิดข้อมูลของสมาชิกของ List ในวงเล็บ [ ]

```
List<int> numbers = [-1, 1, 3, 7];
List numbers =<int> [-1, 1, 3, 7]; // หรือจะประกาศแบบนี้ก็ได้เช่นกัน

List<String> names=[‘Drink’, ‘Ninja’, ‘AJ’];
List names=<String>[‘Drink’, ‘Ninja’, ‘AJ’]; // หรือจะประกาศแบบนี้ก็ได้เช่นกัน

List <dynamic> data= [ true, 0.39, false, “Drink”];
```

เราไม่จำเป็นจะต้องกำหนดค่าในลิสต์ตั้งแต่แรก เราสามารถกำหนดไว้บางส่วนก่อนหรือไม่มีเลย (ลิสต์ว่าง) แล้วค่อยเพิ่มข้อมูลในลิสต์ภายหลัง

```
List<String> name = [];
  name = name + ['Drink'];
  name += ['AJ'];
  var name2 = 'Ninja';
  name += [name2];
  print(name);
```

ผลลัพธ์

[Drink, AJ, Ninja]

1.5.1.2 การเข้าถึงสมาชิกของ List

เราสามารถเข้าถึงสมาชิกของ List ได้โดยใช้เลขลำดับของสมาชิกเริ่มต้นที่ 0 ดังนี้

```
void main() {
List<dynamic> numbers = [-10, 01, 3.141, 7, 10.5, 11, 66.99];
print(numbers[2]); // 3.141
numbers[2] = “Ninja”;
print(numbers[2]); // Ninja
}
```

1.5.1.3 การใช้ลูป for-in ร่วมกับลิสต์

อีกทางนึงที่จะเข้าถึงข้อมูลในลิสต์ ก็สามารถใช้ลูป for โดยใช้ตัวนับจำนวนสมาชิกในลิสต์เพื่อใช้ในการวนลูป แต่มีอีกวิธีที่สะดวกกว่าการใช้ for เฉยๆคือ for-in สามารถดูได้ตามตัวอย่างด้านล่าง

```
void main() {
List<String>
names=['Drink','Ninja','Athan'];

  for(int n=0;n<names.length;n++){
    print(names[n]);
  }
  for(String i in names){
    print(i);
  }
}
```

```
ผลลัพธ์
Drink
Ninja
Athan
Drink
Ninja
Athan
```

1.5.1.4 พร็อปเพอร์ตี้และเมธอดที่สำคัญของลิสต์

ลิสต์จัดอยู่ใน คลาสในภาษา Dart ซึ่งจะประกอบด้วยพร็อปเพอร์ตี้และเมธอดอยู่จำนวนนึงที่เราจะได้นำไปใช้งานได้ในบางกรณี สามารถดูคำสั่งและความหมายได้ดังตาราง 1.5.1.4.1 และ 1.5.1.4.2

- พร็อปเพอร์ตี้ (property) คือ ตัวแปรที่ผูกติดกับวัตถุ (object) โดยพร็อปเพอร์ตี้สามารถเข้าถึงได้โดยตรงผ่านตัวดำเนินการจุด (.)

คำสั่ง	ความหมาย
length	ตรวจสอบจำนวนสมาชิกในลิสต์
first	สมาชิกตัวแรกของลิสต์
last	สมาชิกตัวสุดท้ายในลิสต์
reversed	เรียงลำดับย้อนกลับ
isEmpty	ตรวจสอบว่าเป็นลิสต์ว่างเปล่าหรือไม่ ถ้าใช่จะได้ค่า true

ตารางที่ 1.5.1.4.1 คำสั่งและความหมายของพร็อปเพอร์ตี้ของลิสต์

```
void main() {
  // สร้างชุดข้อมูล List
  List<int> numbers = [1, 2, 3, 4, 5];

  // ตรวจสอบจำนวนสมาชิกในลิสต์
  print(numbers.length); // 5

  // ตรวจสอบว่าชุดข้อมูลว่างหรือไม่
  print(numbers.isEmpty); // false

  // เข้าถึงสมาชิกลำดับแรกในชุดข้อมูล
  print(numbers.first); // 1

  // เข้าถึงสมาชิกลำดับสุดท้ายในชุดข้อมูล
  print(numbers.last); // 5

  // สร้างชุดข้อมูลแบบกลับด้าน
  print(numbers.reversed); // [5, 4, 3, 2, 1]
}
```

- เมธอด (method) คือ ฟังก์ชันที่ผูกติดกับวัตถุ โดยเมธอดสามารถเรียกใช้ผ่านตัวดำเนินการจุด (.)

คำสั่ง	ความหมาย
add()	เพิ่มสมาชิกเข้าไปในชุดข้อมูล
addAll()	เพิ่มสมาชิกหลายตัวเข้าไปในชุดข้อมูลพร้อมกัน
clear()	ลบสมาชิกทั้งหมดออกจากชุดข้อมูล
contains()	ตรวจสอบว่าสมาชิกที่กำหนดอยู่ในชุดข้อมูลหรือไม่
forEach()	เรียกใช้การทำงานที่กำหนดสำหรับแต่ละสมาชิกในชุดข้อมูล
indexOf()	ค้นหาสมาชิกในชุดข้อมูลตามค่าที่กำหนด
insert()	เพิ่มสมาชิกเข้าไปในชุดข้อมูลที่ตำแหน่งที่ต้องการ
join()	ต่อสมาชิกในชุดข้อมูลเข้าด้วยกันด้วยค่าที่กำหนด
remove()	ลบสมาชิกออกจากชุดข้อมูล
removeAt()	ลบสมาชิกออกจากชุดข้อมูลตามตำแหน่งที่ต้องการ
sort()	เรียงลำดับสมาชิกในชุดข้อมูล

ตารางที่ 1.5.1.4.2 คำสั่งและความหมายของเมธอดของลิสต์

```
void main() {
  List<int> numbers = [1, 2, 3];
  // เพิ่มสมาชิกลงในชุดข้อมูล
  numbers.add(4); // [1,2,3,4]
  numbers.addAll([5, 6]); // [1,2,3,4,5,6,]

  // ตรวจสอบว่าสมาชิกที่กำหนดอยู่ในชุดข้อมูลหรือไม่
  print(numbers.contains(2)); // true

  // เรียกใช้การทำงานที่กำหนดสำหรับแต่ละสมาชิกในชุดข้อมูล
  numbers.forEach((number) {
    print(number); }); // 1...6

  // ค้นหาสมาชิกในชุดข้อมูลตามค่าที่กำหนด
  int index = numbers.indexOf(3); // 2

  // เพิ่มสมาชิกลงในชุดข้อมูลในตำแหน่งที่ต้องการ
  numbers.insert(0, 0); //[0, 1, 2, 3, 4, 5, 6]
```

```
  // ต่อสมาชิกในชุดข้อมูลเข้าด้วยกันด้วยค่าที่กำหนด
  String joinedNames = numbers.join(" ");
  print(joinedNames); // 0 1 2 3 4 5 6

  // ลบสมาชิกออกจากชุดข้อมูล
  numbers.remove(3); // [0, 1, 2, 4, 5, 6]

  // ลบสมาชิกออกจากชุดข้อมูลตามตำแหน่งที่ต้องการ
  numbers.removeAt(1); // [0, 2, 4, 5, 6]

  // เรียงลำดับสมาชิกในชุดข้อมูล
  numbers.sort(); // [0, 2, 4, 5, 6]

  numbers.clear();
  print(numbers); //[ ]ลิสต์ว่าง
}
```

1.5.2 ชุดข้อมูลในแบบแมป

ชุดข้อมูลในแบบแมป (Map) เป็นชุดข้อมูลชนิดหนึ่ง ที่ใช้เก็บข้อมูลคู่กันระหว่างคีย์ (Key) และค่า (Value) โดยคีย์จะต้องเป็นค่าที่ไม่ซ้ำกัน ชุดข้อมูลในแบบแมป มีลักษณะคล้ายกับพจนานุกรม ที่สามารถค้นหาค่าได้จากคีย์

โครงสร้างชุดข้อมูลในแบบแมป

```
Map<key_type, value_type> variable = { key1:value1, key2:value2};
                                หรือ
Map variable = <key_type, value_type>{ key1:value1, key2:value2};
```

key_type คือ ชนิดข้อมูลของ key ซึ่งส่วนใหญ่เรานิยมให้เป็นชนิด String แต่ที่สำคัญคือไม่ควรให้ค่า key ซ้ำกัน

value_type คือ ชนิดข้อมูลของ value

1.5.2.1 การสร้าง Map

ชุดข้อมูลในแบบแมป (Map) สามารถสร้างขึ้นได้ 2 วิธี ดังนี้

- การสร้างชุดข้อมูลเปล่า การสร้างชุดข้อมูลเปล่าสามารถทำได้โดยใช้ตัวสร้าง (constructor) ของคลาส Map โดยไม่มีการระบุค่าใดๆ ให้กับชุดข้อมูล

```
Map<String, String> countries = {};
```

ตัวอย่างการใช้งานการสร้างชุดข้อมูลเปล่ามีดังนี้

```
// สร้างชุดข้อมูลเปล่า
Map<String, String> countries = {};
// แสดงผลชุดข้อมูล
print(countries); // {}
```

- การสร้างชุดข้อมูลจากค่าคงที่ การสร้างชุดข้อมูลจากค่าคงที่สามารถทำได้โดยใช้ตัวสร้าง (constructor) ของคลาส Map โดยระบุค่าคงที่ให้กับชุดข้อมูล

```
Map<String, String> countries = {
  "TH": "Thailand",
  "US": "United States",
  "JP": "Japan"
};
```

ตัวอย่างการใช้งานการสร้างชุดข้อมูลจากค่าคงที่มีดังนี้

```
// สร้างชุดข้อมูลจากค่าคงที่
Map<String, String> countries = {
  "TH": "Thailand",
  "US": "United States",
  "JP": "Japan"
};
// แสดงผลชุดข้อมูล
print(countries); // {TH: Thailand, US: United States, JP: Japan}
```

1.5.2.2 การเข้าถึงสมาชิกของ Map

การเข้าถึงสมาชิกของชุดข้อมูลในแบบแมป (Map) สามารถดำเนินการได้โดยใช้ตัวดำเนินการ [ ] โดยระบุคีย์ของสมาชิกที่ต้องการเข้าถึง

ตัวอย่างการเข้าถึงสมาชิกของชุดข้อมูลในแบบแมปมีดังนี้

```
void main() {
  // สร้างชุดข้อมูล
Map<String, String> countries = {
  "TH": "Thailand",
  "US": "United States",
  "JP": "Japan"
};
// เข้าถึงสมาชิกของชุดข้อมูล
 String? countryName = countries["TH"];
print(countryName); // Thailand
print(countries["US"]); // United States
countries["JP"] = "Faker";
print(countries["JP"]); // Faker
}
```

1.5.2.3 การใช้ลูป for-in ร่วมกับ Map

การเข้าถึงสมาชิกของชุดข้อมูลในแบบแมป (Map) โดยใช้ for in สามารถดำเนินการได้โดยใช้ตัวดำเนินการ for...in โดยชุดข้อมูลในแบบแมปจะวนซ้ำผ่านคีย์และค่าของสมาชิกแต่ละตัว

ตัวอย่างการเข้าถึงสมาชิกของชุดข้อมูลในแบบแมปโดยใช้ for in มีดังนี้

```
// สร้างชุดข้อมูล
Map<String, String> countries = {
  "TH": "Thailand",
  "US": "United States",
  "JP": "Japan"};
// เข้าถึงสมาชิกของชุดข้อมูลโดยใช้ for in
for (var entry in countries) {
  print(entry.key); // TH
  print(entry.value); // Thailand
}
```

1.5.2.4 พร็อปเพอร์ตี้และเมธอดที่สำคัญของ Map

แมปก็จัดอยู่ใน คลาสในภาษา Dart ซึ่งจะประกอบด้วยพร็อปเพอร์ตี้และเมธอดอยู่จำนวนนึงที่เราจะได้นำไปใช้งานได้ในบางกรณี สามารถดูคำสั่งและความหมายได้ดังตาราง 1.5.2.4.1 และ 1.5.2.4.2

- พร็อปเพอร์ตี้ (property) คือ ตัวแปรที่ผูกติดกับวัตถุ (object) โดยพร็อปเพอร์ตี้สามารถเข้าถึงได้โดยตรงผ่านตัวดำเนินการจุด (.)

คำสั่ง	ความหมาย
length	ตรวจสอบจำนวนสมาชิกในลิสต์
isEmpty	ตรวจสอบว่าเป็นลิสต์ว่างเปล่าหรือไม่ ถ้าใช่จะได้ค่า true
keys	key ทั้งหมดในแมป
values	value ทั้งหมดในแมป

ตารางที่ 1.5.2.4.1 คำสั่งและความหมายของพร็อปเพอร์ตี้ของแมป

```
// สร้างชุดข้อมูล
Map<String, String> countries = {
  "TH": "Thailand",
  "US": "United States",
  "JP": "Japan"};
// ตรวจสอบจำนวนสมาชิกในชุดข้อมูล
int numberOfCountries =
countries.length;
print(numberOfCountries); // 3
```

```
// ตรวจสอบว่าชุดข้อมูลว่างหรือไม่
bool isEmpty = countries.isEmpty;
print(isEmpty); // false
// เข้าถึงชุดข้อมูลของคีย์ทั้งหมดในชุดข้อมูล
Set<String> countriesKeys = countries.keys;
print(countriesKeys); // {TH, US, JP}
// เข้าถึงชุดข้อมูลของค่าทั้งหมดในชุดข้อมูล
Set<String> countriesValues = countries.values;
print(countriesValues); // {Thailand, United States, Japan}
```

- เมธอด (method) คือ ฟังก์ชันที่ผูกติดกับวัตถุ โดยเมธอดสามารถเรียกใช้ผ่านตัวดำเนินการจุด (.)

คำสั่ง	ความหมาย
addAll()	เพิ่มสมาชิกหลายตัวเข้าไปในชุดข้อมูลพร้อมกัน
clear()	ลบสมาชิกทั้งหมดออกจากชุดข้อมูล
containsKey()	ตรวจสอบว่า key ที่กำหนดอยู่ในแมปหรือไม่ ถ้ามีคืนค่า true
containsValue()	ตรวจสอบว่า value ที่กำหนดอยู่ในแมปหรือไม่ ถ้ามีคืนค่า true
forEach()	เรียกใช้การทำงานที่กำหนดสำหรับแต่ละสมาชิกในชุดข้อมูล
remove()	ลบสมาชิกออกจากชุดข้อมูล

ตารางที่ 1.5.1.4.2 คำสั่งและความหมายของเมธอดของแมป

```
// สร้างชุดข้อมูล
Map<String, String> countries = {
  "TH": "Thailand",
  "US": "United States",
  "JP": "Japan"
};

// เพิ่มสมาชิกใหม่ลงในชุดข้อมูล
countries.addAll({
  "CN": "China",
  "FR": "France",
  "DE": "Germany"
});

// ตรวจสอบว่าชุดข้อมูลมีคีย์ที่กำหนดหรือไม่
bool isThailandInCountries =
countries.containsKey("TH");
print(isThailandInCountries); // true
```

```
// ตรวจสอบว่าชุดข้อมูลมีค่าที่กำหนดหรือไม่
bool isThailandInCountries =
countries.containsValue("Thailand");
print(isThailandInCountries); // true

// วนซ้ำผ่านชุดข้อมูลโดยใช้ for in
for (MapEntry<String, String> entry in
countries.entries) {
  print(entry.key); // TH, US, JP, CN, FR,
DE
  print(entry.value); // Thailand, United
States, Japan, China, France, Germany
}

// ลบสมาชิกออกจากชุดข้อมูล
countries.remove("TH");

// ตรวจสอบว่าชุดข้อมูลมีคีย์ที่กำหนดหรือไม่
bool isThailandInCountries =
countries.containsKey("TH");
print(isThailandInCountries); // false
```

บทที่ 1.6 ฟังก์ชัน (Function)

1.6.1 ความหมายของฟังก์ชัน

เป็นการแยกส่วนที่ต้องทำซ้ำๆ ของการกระทำอย่างใดอย่างหนึ่งไปสร้างเป็นบล็อกไว้ต่างหาก ซึ่งสามารถเรียกเฉพาะส่วนนี้ขึ้นมาใช้งานได้ทันทีเมื่อต้องการ จึงไม่ต้องเขียนโค้ดซ้ำซ้อนกัน

1.6.2 ประเภทของฟังก์ชัน

1.6.2.1 ฟังก์ชันมาตรฐาน (Standard Function)

เป็นฟังก์ชันที่สามารถเรียกใช้งานได้ โดยไม่ต้องเขียนขึ้นเอง เช่น ฟังก์ชันคำนวณทางคณิตศาสตร์ ในภาษา Dart สามารถเขียนได้ว่า import 'dart:math'; เป็นต้น โดยฟังก์ชันเหล่านี้สามารถเรียกได้อีกอย่างว่าไลบรารีฟังก์ชัน (Library Function)

1.6.2.2 ฟังก์ชันที่ผู้ใช้สร้างขึ้นเอง (User-Define Function)

เป็นฟังก์ชันที่ผู้ใช้งานสร้างขึ้นมาเอง ซึ่งส่วนใหญ่เขียนขึ้นมาเพื่อแก้ไขปัญหาการคำนวณตามสูตรต่างๆ หรือตอบสนองกับความต้องการในการแก้ปัญหาของผู้ใช้

1.6.3 ฟังก์ชันที่ผู้ใช้สร้างขึ้นเอง (User-Define Function)

1.6.3.1 การสร้างฟังก์ชัน

การสร้างฟังก์ชันในภาษา Dart ให้เริ่มต้นด้วยการระบุชนิดข้อมูลที่จะส่งกลับ แล้วตามด้วยชื่อฟังก์ชัน และกำหนดคำสั่งต่างๆ ไว้ในบล็อก { } ดังรูปแบบต่อไปนี้

```
ชนิดข้อมูลส่งกลับ ชื่อฟังก์ชัน ( ) {
        คำสั่งต่างๆ
}
```

สำหรับชนิดข้อมูลกลับ ถ้าไม่มีการส่งค่ากลับ ให้ระบุคำว่า void ส่วนชื่อฟังก์ชันใช้หลักการเดียวกับการตั้งชื่อตัวแปร โดยในภาษา Dart นิยมให้อักขระตัวแรกของชื่อฟังก์ชันเป็นตัวพิมพ์เล็ก เช่น ฟังก์ชันที่ชื่อ printOneToHundred พิมพ์ตัวเลข 1 - 100 โดยอาจเขียนได้ดังนี้

```
void printOneToHundred ( ) {
        for (int i = 1; i <= 100; i++) {
                print( i );
        }
}
```

1.6.3.2 การกำหนดพารามิเตอร์ในเบื้องต้น

หากใช้เฉพาะข้อมูลที่กำหนดขึ้นภายในฟังก์ชัน จะขาดความยืดหยุ่นต่อการทำงาน ดังนั้น เราอาจสร้างฟังก์ชันที่สามารถรับข้อมูลจากภายนอกเข้ามาประมวลผลภายในฟังก์ชันได้ ซึ่งสามารถเรียกข้อมูลดังกล่าวได้ว่า **พารามิเตอร์ (Parameter)** โดยใช้รูปแบบดังนี้

```
ชนิดข้อมูลส่งกลับ ชื่อฟังก์ชัน (ชนิดข้อมูล ชื่อพารามิเตอร์, ...) {
        คำสั่งต่างๆ
}
```

พารามิเตอร์ คือตัวแปรที่ใช้รับข้อมูลจากภายนอกเข้ามาใช้ในฟังก์ชัน โดยมีข้อกำหนดที่ควรรู้จักในเบื้องต้นดังนี้

- ต้องระบุชนิดข้อมูลและชื่อของพารามิเตอร์เช่นเดียวกับการประกาศตัวแปร
- แต่ละฟังก์ชันจะมีพารามิเตอร์กี่ตัวก็ได้ ขึ้นอยู่กับความจำเป็นในการใช้งาน
- ถ้ามีพารามิเตอร์หลายตัว ให้คั่นแต่ละตัวด้วยเครื่องหมาย ,
- เราสามารถนำพารามิเตอร์ไปใช้งานต่าง ๆ ในฟังก์ชันคล้ายกับตัวแปร

แนวทางตัวอย่างการสร้างฟังก์ชันแบบมีพารามิเตอร์ มีดังนี้

```
//ฟังก์ชันตรวจสอบเลขว่าเป็นเลขคู่หรือเลขคี่
void checkOddEven (int num) {
        if (num % 2 == 0) {
                print('Even');
        } else {
                print('Odd');
        }
}
//ฟังก์ชันแสดงชื่อ
void showName (int number, String firstname, String lastname) {
        print('$number : $firstname $lastname');
}
```

ทั้งนี้ยังมีแนวทางการกำหนดพารามิเตอร์แบบต่างๆ ดังต่อไปนี้

1. พารามิเตอร์แบบชุดข้อมูล

พารามิเตอร์ที่รับเข้ามาในฟังก์ชันอาจเป็นชุดข้อมูลแบบ ลิสต์ หรือแมป เช่นเดียวกับการส่งค่ากลับออกจากฟังก์ชันที่กล่าวไป ซึ่งหลักการต่างๆ เหมือนเดิมทั้งหมด ซึ่งสามารถดูแนวทางจากโค้ดต่อไปนี้

```
void main ( ) {
        List<int> nums = randIntNums(1, 100, 5);
        Map<String, int> stat = minMaxAvg(nums);
        print(nums);
        print(stat);

        //ลักษณะของผลลัพธ์
        /*
        [75, 25, 56, 42, 96]
        [min: 25, max: 96, sum: 294]
        */
}

List<int> randIntNums(int min, int max, int count) {
        List<int> result = [ ];
        var rand = Random();
        int n;
        for (int i = 0; i < count; i++) {
                n= min + rand.nextInt(max min + 1);
                result += [n];
        }
```

```
        return result;
}

Map<String, int> minMaxAvg(List<int> data)  {
        int min = data[0], max = data[0], sum = 0;

        data.forEach((element)  {
                min = (element < min) ? element : min;
                max = (element > max) ? element : max;
                sum += element;
        });

        Map<String, int> result = {
                'min': min, 'max': max, 'sum': sum
        };

        return result;
}
```

2. พารามิเตอร์แบบกำหนดค่าดีฟอลต์

ถ้าพารามิเตอร์ตัวใดของฟังก์ชัน เรามักกำหนดค่าใดค่าหนึ่งให้กับมันเป็นส่วนใหญ่ แทนที่เราจะต้องระบุค่าเดิมๆ ทุกครั้งเมื่อเรียกฟังก์ชัน ก็สามารถกำหนดเป็นค่าล่วงหน้าให้กับมันได้ ซึ่งเราเรียกลักษณะดังกล่าวนี้ว่า Default Parameter โดยหลักการที่สำคัญคือ

- พารามิเตอร์ตัวใดที่มีค่าดีฟอลต์ ให้ใช้วงเล็บ [ ] ครอบพารามิเตอร์เหล่านั้นทั้งหมด
- พารามิเตอร์ตัวใดที่มีค่าดีฟอลต์ ให้กำหนดค่าในรูปแบบ

```
ชนิดข้อมูล ชื่อพารามิเตอร์ = ค่าดีฟอลต์
```

- พารามิเตอร์ที่ไม่มีค่าดีฟอลต์ให้เขียนตามปกติ แต่ต้องอยู่ก่อนตัวที่มีค่าดีฟอลต์เสมอและไม่ต้องเขียนในวงเล็บ [ ]

```
void func1 ([int a=10])  {
        …
}
```

```
void func2 ([int min=1, int max=100])  {
        …
}

void func3(String s, bool b, [double d=1.23, bool b2=false])  {
        …
}
```

- ในการเรียกฟังก์ชัน หากเราต้องการให้พารามิเตอร์ตัวที่มีค่าดีฟอลต์ตรงกับค่าดังกล่าว ไม่จำเป็นต้องกำหนดค่าสำหรับพารามิเตอร์ (อาร์กิวเมนต์) ตัวนั้น และไม่ต้องเขียนในวงเล็บ [ ] ส่วนพารามิเตอร์ที่ไม่มีค่าดีฟอลต์ ต้องกำหนดต่ำตามปกติ เช่น

```
func1( );                  //ใช้ค่าดีฟอลต์
func1(20);                 //ใช้ค่าที่กำหนดเอง

func3('hello', true);      //ใช้ค่าดีฟอลต์ d=1.23, b2=false
func3('hi', false, 3.14)   //b2 = false
```

3. การกำหนดค่าพารามิเตอร์แบบระบุชื่อ

เมื่อเรียกฟังก์ชัน หากเราระบุเฉพาะค่าที่เป็นอาร์กิวเมนต์ เช่น test(10, 1.5, true) ซึ่งถ้าเราดูเฉพาะโค้ดส่วนนี้จะเข้าใจได้ยากว่าอาร์กิวเมนต์แต่ละค่านั้นหมายถึงสิ่งใด ดังนั้น เพื่อให้ดูชัดเจนยิ่งขึ้น เราสามารถเขียนชื่อพารามิเตอร์กำกับไว้ที่ค่าของอาร์กิวเมนต์แต่ละตัวหรือเพียงบางตัว แต่อย่างไรก็ตาม การจะใช้วิธีนี้ได้ ต้องปรับเปลี่ยนรูปแบบการเขียนพารามิเตอร์ที่ตัวฟังก์ชันให้รองรับการระบุชื่อพารามิเตอร์เมื่อเรียกใช้ฟังก์ชันได้ ดังแนวทางต่อไปนี้

- ที่ตัวฟังก์ชัน พารามิเตอร์ตัวใดที่สามารถระบุชื่อได้เมื่อเรียกฟังก์ชัน ให้ใช้วงเล็บ ( ) ครอบพารามิเตอร์เหล่านั้นทั้งหมด และต้องระบุค่าดีฟอลต์เอาไว้ล่วงหน้า
- หากพารามิเตอร์ที่ไม่ต้องระบุชื่อ ให้เขียนตามปกติ แต่ต้องอยู่ก่อนตัวที่สามารถระบุชื่อทั้งหมด และไม่ต้องเขียนในวงเล็บ ( )

```
void func1 ({int a=10})  {
        ...
}

void func2 ({int min=1, int max=100})  {
        ...
}
```

```
void func3(String s, bool b, {double d=1.23, bool b2=false})  {
        ...
}
```

- ในการเรียกฟังก์ชัน พารามิเตอร์ตัวใดที่เขียนไว้ใน { } ต้องระมุชื่อของมันเสมอ แต่ไม่ต้องเขียนในวงเล็บ { } และเนื่องจากในกรณีนี้ เรากำหนดค่าดีฟอลต์ให้กับมันด้วย ดังนั้นหากเราต้องการให้พารามิเตอร์ตัวที่มีค่าดีพอลใช้ค่าดังกล่าว ก็ไม่จำเป็นต้องกำหนดค่าสำหรับพารามิเตอร์ (อาร์กิวเมนต์) ตัวนั้น ส่วนพารามิเตอร์ที่ไม่มีค่าต้องกำหนดค่าตามปกติ

```
func1( );                                   //ใช้ค่าดีฟอลต์
func1(a:5);                                 //Ok
func1(5);                                   //Error เพราะไม่ระบุชื่อ

func3('hello', true, d:5.55, b2:false);
func3('hello', false, 0.355, false);        //Error (ไม่ระบุชื่อ d, b2)
```

1.6.4 การส่งผลลัพธ์กลับจากฟังก์ชัน

เราสามารถส่งผลลัพธ์บางอย่างที่เกิดขึ้นภายในฟังก์ชัน กลับออกไปยังส่วนที่เรียกใช้ฟังก์ชัน เพื่อนำผลลัพธ์ดังกล่าวไปใช้งานอื่นๆ ต่อไป ซึ่งการส่งค่ากลับจากฟังก์ชันจะแบ่งตามลักษณะข้อมูล ดังนี้

1.6.4.1 การส่งค่ากลับแบบพื้นฐาน

ค่าที่จะส่งกลับจากฟังก์ชันในแบบพื้นฐาน จะเป็นการกำหนดชนิดข้อมูลไว้อย่างแน่นอน ซึ่งอาจเป็นชนิดพื้นฐาน เช่น int, double, bool หรืออาจเป็นคลาสต่างๆ ก็ได้ รวมถึงชนิด dynamic **แต่จะเป็นชนิด var ไม่ได้** และในกรณีนี้จะต้องส่งค่ากลับไปเสมอ แต่จะส่งค่า null กลับไปไม่ได้โดยมีหลักการดังต่อไปนี้

- ที่ตัวฟังก์ชัน ให้ระบุชนิดข้อมูลผลลัพธ์ที่จะส่งกลับคืนไป ไว้ก่อนชื่อฟังก์ชัน หรือเขียนแทนคำว่า void นั่นเอง

- ภายในฟังก์ชัน หลังจากได้ผลลัพธ์มาแล้ว ให้ส่งกลับออกไปด้วยคำสั่ง return ตามด้วยข้อมูล ซึ่งข้อมูลดังกล่าวต้องเป็นชนิดเดียวกับที่ได้ระบุไว้หน้าชื่อฟังก์ชัน

```
ชนิดข้อมูลผลลัพธ์ ชื่อฟังก์ชัน (ชนิดข้อมูล พารามิเตอร์, ...) {
        คำสั่งต่างๆ
        return ผลลัพธ์
}
```

- ในส่วนที่เรียกใช้ฟังก์ชัน หากต้องการนำค่าที่ถูกส่งกลับจากฟังก์ชันไปใช้งานต่อ ให้กำหนดตัวแปรมารับค่าดังกล่าว และตัวแปรนั้นต้องเป็นชนิดเดียวกับที่ฟังก์ชันจะส่งคืนมา หรือถ้าเราไม่สร้างตัวแปรรับคำ สามารถเรียกฟังก์ชันแทนตัวแปรในตำแหน่งที่ต้องการใช้งานได้เลย หรือถ้าต้องการแทรกผลลัพธ์ในสตริง ก็ใช้รูปแบบ ${ฟังก์ชัน}
- กรณีที่เราต้องเปรียบเทียบเงื่อนไขด้วย if สามารถนำคำสั่ง return ไปใส่ไว้ในแต่ละเงื่อนไขได้เลย

```
int add (int n1, int n2)  {          //ส่งข้อมูลกลับเป็นชนิด int
        int x = n1 + n2;
        return x;                    //x ต้องเป็นชนิด int
}

int randomInt (int min, int max)  {
        var rand = Random();
        return min + rand.nextInt(max – min + 1);
}

String oddEven (int num)  {
        if (num % 2 == 0) {
                return 'Even';
        } else {
                return 'Odd';
        }
}
void main ( )  {
        int n  = add(10, 20);        //n ต้องเป็นชนิด int
        var m = add(40, 50);         //m กลายเป็นชนิด int
        int a  = 5, b = 15;
        dynamic p = '$a + $b = ${add(a, b)}';
        print('random num is ${randInt(1, 10)}');

        if (randInt(1, 10) < 50)  {
                print('Error');
        }

        print('99 is ${oddEven(99)}');
}
```

- คำสั่ง return ที่ใช้ส่งผลลัพธ์กลับ ต้องวางไว้เป็นคำสั่งสุดท้ายของฟังก์ชัน ทั้งนี้หากเรากำหนดคำสั่งอื่นๆ ถัดจาก return คำสั่งนั้นจะไม่ถูกประมวลผล เช่น

```
Int increment (int n)  {
        var result = n + 1;
        return result;
        print(result);                  //คำสั่งนี้ไม่มีผล
}
```

1.6.4.2 การส่งค่ากลับแบบ Nullable Type

สำหรับชนิดข้อมูลแบบ Nullable ได้กล่าวมา ซึ่งหากเป็นกรณีที่ค่าจะส่งกลับอาจเป็น null ได้ ในส่วนของฟังก์ชัน เราก็แค่วางเครื่องหมาย ? ตามหลังชนิดข้อมูลที่จะส่งกลับ แล้วภายในฟังก์ชันอาจส่งค่ากลับแบบใดแบบหนึ่งคือ

- อาจส่งค่ากลับด้วย return ตามปกติ แต่ต้องสอดคล้องกับชนิดข้อมูลที่เขียนกำกับไว้หน้าชื่อฟังก์ชัน
- อาจส่งค่า null กลับไปด้วย return
- ไม่ส่งค่าใดๆ กลับไปเลย (ไม่มีคำสั่ง return) ซึ่งผลลัพธ์จะเป็น null โดยอัตโนมัติ

ในการเรียกใช้ฟังก์ชันที่อาจส่งค่ากลับแบบ Nullable Type เราต้องกำหนดชนิดข้อมูลของผลลัพธ์เป็นแบบ Nullable Type แล้วจัดการร่วมกับเครื่องหมายในกลุ่มนี้ตามที่ได้กล่าวไว้ เช่น if-null หรือ nullable-aware เป็นต้น โดย ส่วนใหญ่แล้ว การส่งค่ากลับแบบ Nullable Type มักเป็นกรณีที่ผลลัพธ์ขึ้นกับเงื่อนไข เช่น หากคำบางอย่างหรือพารามิเตอร์ที่รับเข้ามาไม่อยู่ในหลักเกณฑ์ที่จะใช้ประมวลผลได้ ก็อาจส่งค่า null กลับไป เป็นต้

```
String? digitToText (int digit) {
        var text = [
                'zero', 'one', 'two', 'three', 'four',
                'five', 'six', 'seven', 'eight', 'nine'
        ];
        If (digit >= 0 && digit <= 9) {
                return text;
        } else {
                return null;
        }
}
void main ( ) {
        String? minus5 = digitToText(-5);
        print(minus5);     //null
        String? minus1 = digitToText(-1) ?? 'error!';
        print(minus1);     //error!
        String? num = digitToText(5);
        print(num);        //five
}
```

1.6.4.3 การส่งค่ากลับแบบชุดข้อมูล

นอกจากการส่งคำกลับเป็นชนิดข้อมูลพื้นฐานหรือคลาสดังที่กล่าวมาแล้ว หากผลลัพธ์มีข้อมูลมากกว่า 1 คำ เราอาจส่งกลับในแบบชุดข้อมูล เช่น ลิสต์ หรือแมป ซึ่งในกรณีนี้ ก็ใช้หลักการเดิมทั้งหมด โดยระบุชนิดข้อมูลส่งกลับเป็น List หรือ Map ไว้หน้าชื่อฟังก์ชัน แล้วภายในฟังก์ชันก็สร้างรายการข้อมูลชนิดดังกล่าว จากนั้นก็ส่งกลับด้วยคำสั่ง return ตามปกติ เช่น

```
void main ( ) {
        List<int> nums = randIntNums(1, 100, 5);
        print(nums);
}

List<int> randIntNums (int min, int max, int count) {
        var range = Random();
        List<int> result = [ ];
        Int n;
        for(int i = 0; i < count; i++) {
                n = min + range.nextInt(max - min +1);
                result += [n];
        }
        return result;
}
```

1.6.5 ฟังก์ชันแบบ Arrow

ฟังก์ชันที่เราสร้างขึ้น มีเพียงคำสั่งเดียว สามารถเขียนแบบลดรูปให้สั้นลงซึ่งเรียกว่า Arrow Function โดยมีหลักการพื้นฐานดังนี้

- ใช้เครื่องหมาย > หรือ => เชื่อมโยงระหว่างส่วนชื่อฟังก์ชัน กับส่วนที่เป็นตัวฟังก์ชัน
- ส่วนที่เป็นตัวฟังก์ชัน ไม่ต้องเขียนในบล็อกของวงเล็บ ( )
- ถ้ามีการคืนค่ากลับไป ไม่ต้องใช้คำสั่ง return โดยผลลัพธ์ที่เกิดขึ้น จะถูกส่งออกไปโดยอัตโนมัติ

ตัวอย่างโค้ดเปรียบเทียบระหว่างฟังก์ชันแบบปกติ (Regular) กับฟังก์ชันแบบ Arrow มีดังต่อไปนี้

ตัวอย่างโค้ดที่ 1

Regular Function

```
int add (int a, int b) {
        return a + b;
}
```

Arrow Function

```
Int add (int a, int b) => a + b;
```

ตัวอย่างโค้ดที่ 2

Regular Function

```
int randInt (int min, int max)  {
        var rand = Random( );
        return min + rand.nextInt(max-min+1);
}
```

Arrow Function

```
Int randInt(int min, int max)  =>  min + Random().nextInt(max-min+1);
```

การเรียกใช้ฟังก์ชันแบบ Arrow ก็ใช้หลักการเดิมดังนี้

```
int a = add(5, 15);
int r = randInt(1, 10);
```

คำสั่ง ให้นักศึกษาเขียนคำตอบตามที่โจทย์กำหนดให้ถูกต้อง (สามารถแนบรูปโค้ดและผลลัพธ์คำตอบของโปรแกรมได้)

1. กำหนดให้ List<String> fruits = [“apple”, “orange”, “mango”] และใช้ Method ของ List ทำดังข้อต่อไปนี้
    1.1 แทรก “dragonfruit” ไว้หน้า “apple”
    1.2 เพิ่ม “pineapple” ไว้หลังสุด
    1.3 ลบ “orange” ออก
    1.4 print ผลลัพธ์สุดท้ายให้เป็น [dragonfruit, apple, mango, pineapple]

```dart
void main() {
  List<String> fruits = ['apple', 'orange', 'mango'];
  fruits.insert(0, 'dragonfruit');
  fruits.add('pineapple');
  fruits.removeAt(2);
  print(fruits);
}
```

Console

```
[dragonfruit, apple, mango, pineapple]
```

2. เขียนฟังก์ชัน max(List<int> dat) หาค่า **max** ของ dat เช่น List<int> dat=[1,2,3,-4] ให้ คืนค่า 3

```dart
int max(List<int> dat) {
  int max = dat[0];
  for (int i = 1; i < dat.length; i++) {
    if (dat[i] > max) {
      max = dat[i];
    }
  }
  return max;
}

void main() {
  List<int> dat = [1, 2, 3, -4];
  print(max(dat));
}

```

Console

```
3
```

3. จงเติมคำในช่องที่เว้นว่างให้ถูกต้อง เพื่อให้ได้ผลลัพธ์ตาม Output แสดง

Output

{id: 1, name: ninja, salary: 20000}

```
void toDict(int? id, String? name, double? salary) {
  Map<String, dynamic> dict = {

    " id ": id,
    " name ": name,
    " salary ": salary

  };
  print(dict);
}
void main () {
  toDict(1, "ninja", 20000);
}
```

4. เขียน function sum(List<double> dat) หาค่า **sum** ของ dat เช่น List<double> dat=[2,-3.5,5.5] ให้ คืนค่า 4

```dart
double sum(List<double> dat) {
  double sum = dat[0];
  for (int i = 0; i < dat.length; i++) {
    sum += dat[i];
  }
  return sum;
}

void main() {
  List<double> dat = [2, -3.5, 5.5];
  print(sum(dat));
}
```

Console

```
6
```

5. หาค่า **min และ max** จาก Method / function ของ List โดยให้สมมติตัวเลขใน List 10 จำนวน

```dart
import 'dart:math' show min, max;

void main() {
  List<int> number = [10, 5, 2, 6, 4, 9, 1, 100, 50, 22];
  print("The minimum number is ${number.reduce(max)}");
  print("The maximum number is ${number.reduce(min)}");
}

```

Console

```
The minimum number is 100
The maximum number is 1
```

6. หาค่า **sum** จาก Method / function ของ List โดยให้สมมติตัวเลขใน List 10 จำนวน

```dart
import 'package:collection/collection.dart';

void main() {
  final list = [10, 22, 33, 44, 55, 1, 2, 3, 4, 8];
  final sum = list.sum;
  print("ALL sum is $sum");
}

```

Console

```
ALL sum is 182
```

5. สรุปผลการทดลอง

จากการทดลองสรุปว่า การใช้funtiion กับงานต่างๆ เพื่อให้การทำงานมีประสิทธิภาพและให้งานมีความสมบูณมากขึ้น โดยเราจะต้องรู้วิธีการใช้งานให้ถูกต้อง เพื่อจะได้นำไปใช้งานได้อย่างดียิ่งขึ้น โดยในแบบฝึกหัดได้ให้ทำการทดลองใช้ funtion และ Properties methods และการใช้ Map โดยจะมุ่งเน้นไปทางการใช้ funtion และ List

6. คำถามหลังการทดลอง

6.1 List และ Map แตกต่างกันอย่างไร

ชุดข้อมูลในแบบแมป (Map) เป็นชุดข้อมูลชนิดหนึ่ง ที่ใช้เก็บข้อมูลคู่กันระหว่างคีย์ (Key) และค่า (Value) โดยคีย์จะต้องเป็นค่าที่ไม่ซ้ำกัน ชุดข้อมูลในแบบแมป มีลักษณะคล้ายกับพจนานุกรม ที่สามารถค้นหาค่าได้จากคีย์

List คือชุดข้อมูลที่เก็บข้อมูลแบบเรียงลำดับ ข้อมูลแต่ละตัวใน List เรียกว่าสมาชิก (element) สมาชิกใน List จะถูกเรียงลำดับตามเลขลำดับเริ่มต้นที่ 0

6.2 การเขียนพารามิเตอร์แบบกำหนดค่าดีฟอลต์ไว้ก่อน มีข้อดีอย่างไร

ลดการเรียกใช้ข้อมูลซ้ำซ้นกันหลายรอบ ถ้าค่านั้นเราจำเป็นต้องใช้หายครั้ง

6.3 การเขียนฟังก์ชันแบบปกติ (regular function) แตกต่างกับ การเขียนฟังก์ชันลูกศร (arrow function) อย่างไร

ฟังก์ชันปกติจะมีการเขียนคำสั่งการทำงานไว้ข้างในเป็นจำนวนมาก แต่ฟังชั้น ลูกศรสวนมากจะใช้ในการทำคำสั่งเดียวโดยจะมีการใช้ => เป็นการเรียกใช้การทำงาน