CADT IDT

W1 PRACTICE

PART 1 – EXPLORATION

Learning objectives

- Apply type **inference** for variable declarations.
- Handle **nullable** and **non-nullable** variables.
- Differentiate between **final** and **const**.
- Manipulate **strings, lists, and maps.**
- Use **loops** and **conditions** to control flow.
- Define and call functions with positional and **named arguments**, understand **arrow syntax**

L0 NO AI
Solve problems entirely on your own.

How to run Dart code?

You can write you code on VS Code, or using this online editor

- Install Dart and Flutter SDK
- Online Dart Compiler

Resources for this research

To help you complete this handout, use the following resources:

- ✓ Variables
- ✓ Null Safety
- ✓ Built-in types
- ✓ Lists
- ✓ Loops
- ✓ Conditions
- ✓ Functions

EX 1 - Type Inference

EXPLAIN : Explain how Dart infers the type of a variable.

Dart is can identify type of variable by itself if you just declare Object or var but you if declare specific type of variable you need to follow then variable rule. Example:
Object variable1 = “any_type”; // can be any type of varible
int variable2 = 1; // must be a number

CODE : Complete the bellow code to illustrate the concepts:

```
// Declare a int variable and let Dart infer its type
int variable2 = 1; // must be a number
// Define a variable with an explicit String type
Object variable1 = “any_type”; // can be any type of varible
```

EX 2 - Nullable and Non-Nullable Variables

EXPLAIN : Explain nullable vs non-nullable variables.

Nullable is the variable can be null or not null (the program is not error) while non-nullable is the variable that must not null (need to initialize it).

EXPLAIN : When is it useful to have nullable variables?

I think that it useful when attribute is unnecessary example the person should variable id, first_name, last_name and age so age is not unnecessary so can declare as the nullable.

CODE : Complete the bellow code to illustrate the concepts:

```
// Declare a nullable integer variable and assign it a null value
int? age;
// Declare a non-nullable integer variable and assign it a value
int age = 100;
// Assign a new value to the nullable variable
age = 101;
```

EX 3 - Final and const

EXPLAIN : Describe the difference between final and const.

Final is define afterward and const is the specific variable that when you initialized it then it can't change or modify it any to another value;

CODE : Complete the bellow code to illustrate the concepts:

```
// Declare a final variable and assign it the current date and time
final string dateTime = DateTime.now();
// Can you declare this variable as const? Why?
No, I can't declare this const because final is initialize once time.
// Declare a const variable with a integer value
const pi = 3.14;
// Can you reassign the value of this final variable? Why?
No, I can't because it is the const can't change to another value.
```

EX 4 - Strings, Lists and Maps

CODE : Complete the bellow code to illustrate the concepts:

Strings:

```
// Declare two strings: firstName and lastName and an integer:age
String firstName, lastName;
int age;
// Concatenate the 2 strings and the age
String combined = firstName + lastName + "$age";
// Print result
print("This is the the result:  ${combined} ");
```

Lists:

```
// Create a list of integers
List<int> number = [1, 2, 3];
// Add a number to the list
number.add(4);
// Remove a number from the list
number.remove(4); // remove by specific element(value)
number.removeAt(4); // remove by indexing
// Insert a number at a specific index in the list
number.insert(1, 99); // insert number 99 at index number 1
```

```
// Iterate over the list and print each number
for( int num in list){
        printInt("${number[num]} \n");
}
// using forEach
number.forEach((num) => print(num));
```

Maps:

```
// Create a map with String keys and integer values
Map<String, int> scores = {
        'Rayu' : 100,
        'Mengheng' : 99,,
        'Somnang' : 20,
}
// Add a new key-value pair to the map
scores[ 'Elit' ] = 100;
// Remove a key-value pair from the map
scores.remove ('Somnang' );
// Iterate over the map and print each key-value pair
scores.forEach((key, value){
        print("$key : $value");
});
```

EXPLAIN : When should I use a Map instead of a List?

I use map instead of a list when the data collection is link together and for better search (reduce time complexity of searching) don't care about the order or indexing. Example we want to search the any location.

EXPLAIN : When should I use a Set instead of a List?

I use Set instead of a List when the I need the collection that require no duplicate value. Example: the collection to store students id so Set is not insert the duplicate value into the collaction.

EX 5 - Loops and Conditions

CODE : Complete the bellow code to illustrate the concepts:

```
// Use a for-loop to print numbers from 1 to 5

// Use a while-loop to print numbers while a condition is true

// Use an if-else statement to check if a number is even or odd
```

EX 6 - Functions

EXPLAIN : Compare positional and named function arguments

positional function is identify by its position in the function call or order in which you pass them matters while named function is identify by its name when you call a function and no need order.
Example: You declare a function like void functions(String name, int age) and then you call that function positional like this functions("rayu", 20); // (need order following funciton that you declared but for named function you use like this functions(age: 20, name: "rayu"); // (no need order) just pass the correct parameter.

EXPLAIN : Explain when and how to use arrow syntax for functions?

Using arrow syntax functions when the function is using only the function's body consists of a single expression, can say one statement of code and no loop or complex login with if-else blocks. To use the arrow syntax is just using sample function that you declared but instead of using { function's body} and use => with return value instead.

CODE : Complete the bellow code to illustrate the concepts:

Defining and Invoking a Function:

```
// Define a function that takes two integers and returns their sum
int sum(int a, int b) => a + b; // using arrow function to reduce function's body just only line of code
// Call the function and print the result
print("result: ${sum(2,2)}");
```

Positional vs Named Arguments:

```
// Define a function that uses positional arguments
void positionalFunction(String mae_ah_nang, int mi_knang) {
  print("here is: $mae_ah_nang and $mi_knang");
}

// Define another function that uses named arguments with the required keyword (ex:
getArea with rectangle arguments)

void getName({required String mae_ah_nang, required int mi_knang}) {
  print("Hello $mae_ah_nang, you are $mi_knang years old.");
}

// Call both functions with appropriate arguments

void main() {
  // Call positional function
  positionalFunction("Rayu", 20); // no error
  positionalFunction(20, “Rayu”); // it will error
  // Call named function (order doesn’t matter)
  getArea(mae_ah_nang: "Rayu", mi_knang: 20);
  getArea(mi_knang: 25, mae_ah_nang: "Choeng"); // still valid no error
}
```

EXPLAIN : Can positional argument be omitted? Show an example

```
Positional argument can be omitted. Example:
// “name” is required positional argument
// “gender” is an optional positional argument
void getInfo(String name, [String? gender]) {
        if (gender != null) {
                print(“Name: $name, Gender: $gender “);
        }else {
        print(“Name: $name”);
        }
}
```

EXPLAIN : Can named argument be omitted? Show an example

CODE : Complete the bellow code to illustrate the concepts:

Arrow Syntax:

```
// Define a function using arrow syntax that squares a number
int squareNumber(int number) => number * number;
// Call the arrow function and print the result
print(“ Square of 4: ${ squareNumber(4) } “);
```