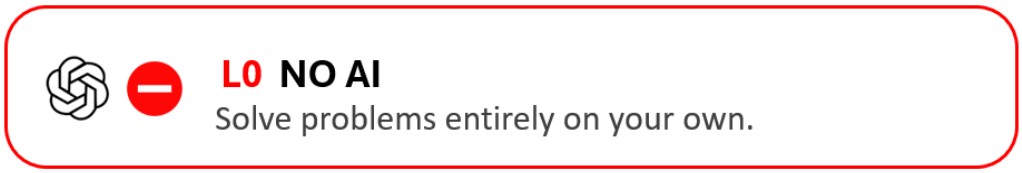
W1 PRACTICE

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**



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 infers the type of a variable by using keyword var and it will decide the type of the variable based on its value.

CODE : Complete the bellow code to illustrate the concepts:

```
// Declare a int variable and let Dart infer its type
var num = 10;
// Define a variable with an explicit String type
String name = “hab”;
```

EX 2 - Nullable and Non-Nullable Variables

EXPLAIN : Explain nullable vs non-nullable variables.

Nullable is when variables can be null if it has no value it will be null and the syntax is int? age.
Non-nullable is variable that does not allow to be null it has to have value and the syntax is normal as we declare.

EXPLAIN : When is it useful to have nullable variables?

Nullable variables are useful when you don’t want to initialize the value of the variable because as default you have to initialize value of the variable in dart and when you accept user input.

CODE : Complete the bellow code to illustrate the concepts:

```
// Declare a nullable integer variable and assign it a null value
int? i = null;
// Declare a non-nullable integer variable and assign it a value
int j = 10;
// Assign a new value to the nullable variable
i = 10;
```

EX 3 - Final and const

EXPLAIN : Describe the difference between final and const.

Final and const are both use when we don’t want to change the value of the variable but for const is a compile time constant

CODE : Complete the bellow code to illustrate the concepts:

```
// Declare a final variable and assign it the current date and time
final date = DateTime.now();
// Can you declare this variable as const? Why?
We can’t declare this variable as const because date time has to be change
and const can’t be changed;
```

```
// Declare a const variable with a integer value
const int i = 10;
// Can you reassign the value of this final variable? Why?
We can not reassign the value of this final variable because when you declare
it as final it can not be change.
```

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 = “te”;
String lastName = “Chhenghab”;
int age = 19;
// Concatenate the 2 strings and the age
String person = firstName + lastName + age.toString();
// Print result
print(person);
```

Lists:

```
// Create a list of integers
var list = [1, 2, 3, 4];
// Add a number to the list
list.add(5);
// Remove a number from the list
list.removeAt(2);
// Insert a number at a specific index in the list
list.insert(4, 6);
// Iterate over the list and print each number
var interator = list.iterator;
while(interator.moveNext()){
  print(interator.current);
}
```

Maps:

```
// Create a map with String keys and integer values
var map = {
  'one': 1,
  'two': 2,
  'three': 3,
};
// Add a new key-value pair to the map
map['four'] = 4;
// Remove a key-value pair from the map
```

```
map.remove('three');
// Iterate over the map and print each key-value pair
print(map);
```

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

We should use Map instead of list when we have a key-value pairs not just list of arrays.

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

We should use Set instead of list when we have a list of unique value unlike list that can have duplicate value.

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
for(int i = 1; i <= 5; i++){
  print(i);
}
// Use a while-loop to print numbers while a condition is true
var list = [1, 2, 3, 4];
var interator = list.iterator;
while(interator.moveNext()){
  print(interator.current);
}
// Use an if-else statement to check if a number is even or odd
int number = 2;
if(number%2 == 0){
  print(“Number is even”);
}else{
  print(“Number is odd”);
}
```

EX 6 - Functions

EXPLAIN : Compare positional and named function arguments

Positional function arguments are pass down base on their order in the parameter list in the function, while named function arguments are pass down base on their name and its must be nullable unless it mark as required.

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

We use arrow function when we only have 1 line of code in the function and the syntax is <type> <name> (arguments) => <logic>;

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){
 return a +b;
}
// Call the function and print the result
int result = sum(4, 5);
print(result);
```

Positional vs Named Arguments:

```
// Define a function that uses positional arguments
String name(String firstName, String lastName, [String? nickname]){
  String fullName = firstName + lastName;
  if(nickname != null){
    print(nickname);
  }
  return fullName;
}
// Define another function that uses named arguments with the required
keyword (ex: getArea with rectangle arguments)
double getArea({double? width, double? height}){
  double result;
  if(width != null && height != null){
    result = width * height;
    return result;
  }
  return 0.00;
}

// Call both functions with appropriate arguments
String fullName = name("te", "chheng", "hab");
print(fullName);

double areaResult = getArea(width: 10, height: 20);
print(areaResult);
```

EXPLAIN : Can positional argument be omitted? Show an example

It can not be omitted otherwise the value will be out of order.
Example:

```
String name(String firstName, String lastName){
  String fullName = firstName + lastName;
  return fullName;
}
String fullName = name( “te”); // it cause error
```

EXPLAIN : Can named argument be omitted? Show an example

It can be omitted if it set to optional by putting ?
Example

```
void hello({String? name, int? age}) {
  if (name != null && age != null) {
    print("Hello $name and you are $age years old.");
  } else if (name != null) {
    print("Hello $name!");
  } else {
    print("Hello...!");
  }
}
hello(name: “hab”,age: 10);
hello(name: “hab”);
```

CODE : Complete the bellow code to illustrate the concepts:

Arrow Syntax:

```
// Define a function using arrow syntax that squares a number
int square(int num) => num * num;

// Call the arrow function and print the result
int result = square(5);
print(result);
```