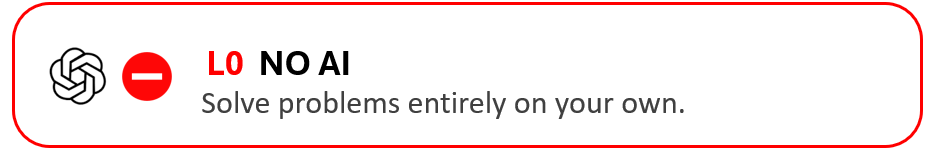
**W1**  PRACTICE

*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](https://docs.flutter.dev/get-started/install)
* [Online Dart Compiler](https://dartpad.dev/)

## *Resources for this research*

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

* [Variables](https://dart.dev/language/variables)
* [Null Safety](https://dart.dev/null-safety)
* [Built-in types](https://dart.dev/language/built-in-types)
* [Lists](https://dart.dev/language/collections)
* [Loops](https://dart.dev/language/loops)
* [Conditions](https://dart.dev/language/branches)
* [Functions](https://dart.dev/guides/language/language-tour#functions)

*EX 1 - Type Inference*

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

Dart infers the type of a variable by referring to the object of the variable’s value.

**CODE** : Complete the bellow code to illustrate the concepts:

// Declare a int variable and let Dart infer its type

var num = 0;

// Define a variable with an explicit String type

String name = “Hello!”;

*EX 2 - Nullable and Non-Nullable Variables*

**EXPLAIN** : Explain nullable vs non-nullable variables.

Nullable variables allow declaration without initializing.

Non-nullable variables throw runtime errors if their declaration doesn’t get initialized or initialized to a null value.

**EXPLAIN** : When is it useful to have nullable variables?

It is useful when it has unknown type or value.

**CODE** : Complete the bellow code to illustrate the concepts:

// Declare a nullable integer variable and assign it a null value

int? nullNum = null;

// Declare a non-nullable integer variable and assign it a value

int num = 0;

// Assign a new value to the nullable variable

nullNum = num;

*EX 3 - Final and const*

**EXPLAIN** : Describe the difference between final and const.

Final is not read when during compiling whereas const is read during compiling.

**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?

No, you can’t because DateTime.now() doesn’t allow constant cast, and const also gets initial during the compile phase.

// Declare a const variable with a integer value

const int num = 0;

// Can you reassign the value of this final variable? Why?

No, you can’t because final value can only be set once.

*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

var firstName = “I”, lastName = “ am handsome”, age = 20;

// Concatenate the 2 strings and the age

String result = firstName + lastName + age.toString();

// Print result

print(result);

Lists:

// Create a list of integers

List<int> nums = [0, 1, 2];

// Add a number to the list

nums.add(3);

// Remove a number from the list

nums.removeAt(1);

// Insert a number at a specific index in the list

const i = 1;

nums.insert(i, 10);

// Iterate over the list and print each number

nums.forEach(print);

**Maps**:

// Create a map with String keys and integer values

var price = Map<String, int>();

// Add a new key-value pair to the map

price[‘apple’] = 2;

// Remove a key-value pair from the map

price.remove(‘apple’);

// Iterate over the map and print each key-value pair

  price.forEach((key, value) {

    print("$key: $value");

  });

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

I should use map when there is specific key for each value.

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

I should use a Set when the values have different data type.

*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 (var i = 1; i <= 5; i++) print(i);

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

var num = 0;

while (num < 10) {

print(num);

num++;

}

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

if (num % 2 == 0) {

print(“Even”);

} else {

Print(“Odd”);

}

*EX 6 - Functions*

**EXPLAIN** :  Compare positional and named function arguments

Positional function arguments are the traditional style that are namely suggested positional which means the position of the arguments cannot be changed when called, and arguments don’t have to be specified, but they must follow the initial order. All arguments must be filled in by default.

Named function arguments must have the name of argument followed by its value when called. They aren’t in order and require all arguments to be filled in by default.

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

Arrow syntax for functions is used when the length isn’t too long, and it returns value.

To use arrow syntax for functions, first, name the variable or function, and its type is optional in Dart. Then, specify the argument(s) in the parenthesis after the equal symbol. After the parenthesis, use the equal symbol follow by greater than symbol to form an arrow (=>). Lastly, put operation inside curly brace.

**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 num1, int num2) { return num1 + num2; }

// Call the function and print the result

print(sum(2, 3));

Output: 5

**Positional vs Named Arguments:**

// Define a function that uses positional arguments

void getInfo(String id, String password) {}

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

double getArea({required double? length, required double? width}) {

return length! \* width!;

}

// Call both functions with appropriate arguments

getInfo(‘user1’, ‘legitPassword’);

getArea(width: 10.2, length: 6.9);

**EXPLAIN** :  Can positional argument be omitted? Show an example

Yes, it can be omitted by using square brackets.

For example, void order(String item, [String? Note]) {}

When called, order(‘Milk Tea’); == order(‘Milk Tea’, ‘No sugar’);

**EXPLAIN** :  Can named argument be omitted? Show an example

By default, named argument can be omitted.

For example, void drive({double? steeringAngle, double? throttleInput}) {}

When called, drive();

**CODE** : Complete the bellow code to illustrate the concepts:

**Arrow Syntax:**

// Define a function using arrow syntax that squares a number

double square(double num) => num \* num;

// Call the arrow function and print the result

print(square(2.5));

Output: 6.25