**module-2 : Theory Assignments :**

1. Fundamental Data Types in Dart:

int: Represents integer values (e.g., int age = 25;).

double: Represents floating-point numbers (e.g., double price = 19.99;).

String: Represents a sequence of characters (e.g., String name = 'John';).

bool: Represents a boolean value – true or false.

List: An ordered collection of items (e.g., List<int> numbers = [1, 2, 3];).

Map: A collection of key-value pairs (e.g., Map<String, int> scores = {'Math': 90};).

Set: An unordered collection of unique items (e.g., Set<int> ids = {1, 2, 3};).

2. Control Structures in Dart:

if / else: Executes code based on condition.

if (age > 18) { print("Adult"); } else { print("Minor"); }

for loop: Repeats a block of code a set number of times.

for (int i = 0; i < 5; i++) { print(i); }

while loop: Repeats code while a condition is true.

while (count < 5) { print(count); count++; }

do-while loop: Executes code at least once before checking the condition.

do { print(count); count++; } while (count < 5);

switch: Executes code based on matching a value.

switch (day) {

case 'Mon': print('Work'); break;

default: print('Rest');

}

3. Object-Oriented Programming Concepts in Dart:

Class: A blueprint for creating objects.

class Car { String color; void drive() {} }

Object: An instance of a class.

Car myCar = Car();

Inheritance: Allows a class to inherit properties from another class.

class Dog extends Animal {}

Polymorphism: Allows objects to be treated as instances of their parent class.

Animal a = Dog(); a.makeSound();

Interface: A class that defines a contract; implemented using implements.

class Printer { void printData(); }

class PDFPrinter implements Printer { void printData() {} }

4. Asynchronous Programming in Dart:

Future: Represents a value that will be available in the future.

Future<String> fetchData() async { return 'Data'; }

async: Marks a function that performs asynchronous operations.

Future<void> getData() async { await fetchData(); }

await: Waits for a future to complete before continuing.

var result = await fetchData();

Stream: Handles multiple asynchronous events over time.

Stream<int> numberStream() async\* { yield 1; yield 2; }