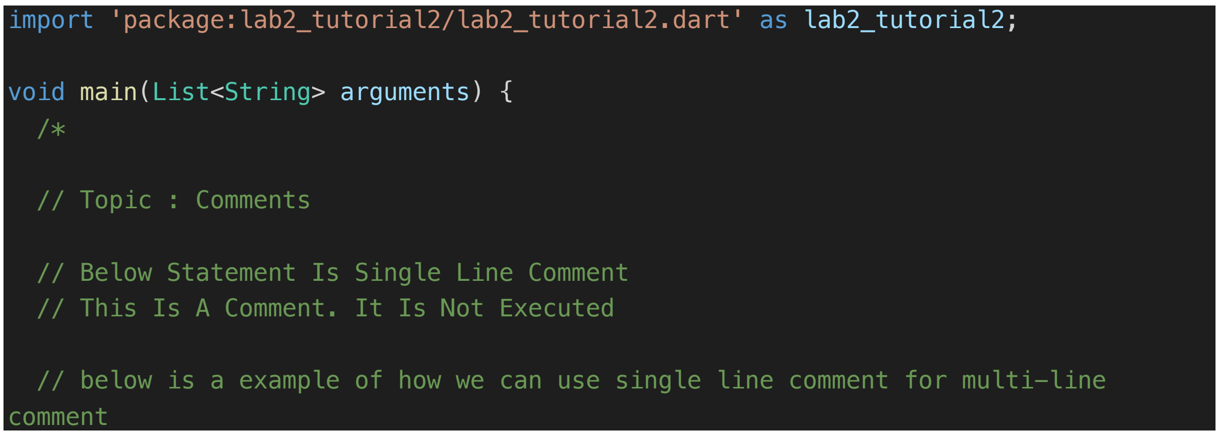
NAME : MAKVANA NEEL
ROLL NO : CE064
ID : 20CEUOS086
BATCH : A4

Lab 02 code and comment :

Github link:

https://github.com/NK16082002/SDP-LABS/tree/main/LAB2

tutorial 2:



```
import 'package:lab2_tutorial2/lab2_tutorial2.dart' as lab2_tutorial2;

void main(List<String> arguments) {
  /*

  // Topic : Comments

  // Below Statement Is Single Line Comment
  // This Is A Comment. It Is Not Executed

  // below is a example of how we can use single line comment for multi-line
comment
```

```
  // This is also a comment
  // over multiple lines

  // Below is an example of comment block
  /* This is mul
  tiline com
  ment */

  // Below is  A Example of nest comment
  /* this is a comment
  /* and inside ther is another
  comment */
  back */

  // below is a example of documentation comment
  /// I am a documentation comment
  /// hi
  /**
   * hi
   */

  // Below statement print Hello,Dart Apprentice reader!"
  // print("Hello,Dart Apprentice reader!");
 */

/*

  // Topic : Statement and expressions


  // Below statement print Hello,Dart Apprentice reader!"
  // print("Hello,Dart Apprentice reader!");

  // Example Of If Statement
  int a=5;
  if(a>0){
    print("yes!!");
  }

  // Example of Expression:
  // print 42
  // print(42);
  // print 5
  // print(3 + 2);
  // print the statement
  // print("Hello,Dart Apprentice reader!");
  // below statement give us error of undefined name
  // print(x);

  */
  // Topic : Arithmatic Expressions:

  // simple operations:

  // print 2+6=8
  // print(2 + 6);
  // print 10-2=8
  // print(10 - 2);
  // print 2*4=8
  // print(2 * 4);
  // print 24/3=8.0
  // print(24 / 3);

  // Decimal Numbers:

  // print 3.142857142857143
  // print(22 / 7);

  // if we want integer division
  // print 3
  // print(22 ~/ 7);

  // Euclidean modulo operation
  // it print 8 because remainder is 8
  // print(28 % 10);

  // order of operations:

  // it print 32
  // beacause it will first calculate the expression which are in () (parenthesis)
the it evalute the value from left to right so firt 8000/50 is evaluate and its
substract 32 then he compute 29%5 and then perform ~/ between them And Give Result
32
  // print(((8000 / (5 * 10)) - 32) ~/ (29 % 5));

  // first it will calculate 350/5 because / has higher precedence then + so it
will give us 72
  // print(350 / 5 + 2);

  // parenthesis has higher precedence compare to / so it will first perform 5+2
and then divide from 350
  // print(350 / (5 + 2));

  // Topic : Math Functions:

  // We Have To Import dart:math library
  // These convert an angle from degrees to radians, and then compute the sine and
cosine respectively.
  // it will take pi value as 3.14 and perform the expression and give it's sin
value
  // print(sin(45 * pi / 180));
  // it will take pi value as 3.14 and perform the expression and give it's cos
value
  // print(cos(135 * pi / 180));
  // it will give us a value of square root of 2
  // print(sqrt(2));

  // These compute the maximum of two numbers respectively.
  // print(max(5, 10));
  // These compute the minimum of two numbers respectively.
  // print(max(-5, -10));
  // it will first compute sqrt(2) and pi/2 and then give us max of this two
numbers
  // print(max(sqrt(2), pi / 2));

  // print the value of 1 over the square root of 2 in Dart.
  // It Will Give Us 1 As Answer.
  // print(sin(pi / 2));

  // Topic : Naming data

// Identifiers can include both, characters and digits. However, the identifier
cannot begin with a digit.
// Identifiers cannot include special symbols except for underscore (_) or a dollar
sign ($).
// They must be unique.
// Identifiers cannot be keywords.
// Identifiers are case-sensitive.
// Identifiers cannot contain spaces.

// Variables :

// This Below statement declares a variable called number of type int and set the
value is 10.
// The int part of the statement is known as a type annotation.
// int number = 10;

// If you want to change the value of a variable, then you can just give it a
different value of the same type:
// int number = 10;
// number = 15;

// This Below statement declares a variable called apple of type double and set the
value is 3.14159.
// double apple = 3.14159;

// This Beloe statement print true
// print(10.isEven);

// This Below statement print 3.
// print(3.14159.round());
// Topic : Type safety

// int myInteger = 10;
// myInteger = 3.14159; // No, no, no. That's not allowed.

// The num type can be either an int or a double, However, the string value 'ten'
is of a different type, so the compiler complains.

// num myNumber;
// myNumber = 10;      //ok
// myNumber = 3.14159; //ok
// myNumber = "t";     //No, no , no

// This lets you assign any data type you like to your variable, and the compiler
won’t warn you about anything.

// dynamic myNumber = 10;      //ok
// myNumber = 3.14159;         //ok
// myNumber = "t";             //ok

// Type inference :

// There’s no need to tell Dart that 10 is an integer. Dart the type  and makes
someNumber an int.
// var someNumber = 10;
// someNumber = 15; // OK
// someNumber = 3.14159; // No, no, no.

// Constants :

// constant variable is immutable , we can't change it's value after assigning it.

// const myConstant = 10;
// myConstant = 0; // Not allowed.

// final constants :

// Here is another example of a runtime value:
// final hoursSinceMidnight = DateTime.now().hour;

// If you try to change the final constant afterit’s already been set, This will
produce error.
// final hoursSinceMidnight = DateTime.now().hour;
// hoursSinceMidnight = 0;

// Mini-exercises :

// 1. Declare a constant of type int called myAge and set it to your age.
// Ans1 :
// const myAge = 19;
// 2. Declare a variable of type double called averageAge.
// Initially, set the variable to your own age. Then, set it to
// the average of your age and your best friend’s age.
// Ans2 :
// double averageAge = 19;
// averageAge = 20;

// 3. Create a constant called testNumber and initialize it
// with whatever integer you’d like. Next, create another
// constant called evenOdd and set it equal to testNumber
// modulo 2. Now change testNumber to various numbers.
// What do you notice about evenOdd?
// Ans3 :
// const testNumber = 1025;
// const evenOdd = testNumber % 2;

// Increment and decrement :

// var counter = 0;
// counter += 1;
// counter = 1;
// counter -= 1;
// counter = 0;

// In other words, the code above is shorthand for the following:

// var counter = 0;
// counter = counter + 1;
// counter = counter - 1;

// If you only need to increment or decrement by 1, then you can use the ++ or --
operators:

// var counter = 0;
// counter++; // 1
// counter--; // 0

// This given Below given operation is perform similar operations for
multiplication and division, respectively:
// double myValue = 10;

// same as myValue = myValue * 3;// myValue = 30.0;
// myValue *= 3;
// same as myValue = myValue / 2;// myValue = 15.0;
// myValue /= 2;

// Chanllanges :

// Challenge 1: Variables
// Declare a constant int called myAge and set it equal to your
// age. Also declare an int variable called dogs and set that
// equal to the number of dogs you own. Then imagine you
// bought a new puppy and increment the dogs variable by one.
// Ans1 :
// const myAge = 19;
// int dogs = 0;
// dogs++;
// print(dogs);

// Challenge 2: Make it compile
// Modify the first line so that the code compiles. Did you use var?
// Ans2 :
// var age = 16;
// print(age);  //print : 16
// age = 30;
// print(age); //print : 30

// Challenge 3: Compute the answer

// Consider the following code:
// const x = 46;
// const y = 10;

// Work out what each answer equals when you add the
// following lines of code to the code above:

// const answer1 = (x * 100) + y;
// const answer2 = (x * 100) + (y * 100);
// const answer3 = (x * 100) + (y / 10);

// Ans3 :
// print(answer1); // 4610
// print(answer2); // 5600
// print(answer3); // 4601.0

// Challenge 4: Average rating
// Ans4 :
// const rating1 = 3.4;
// const rating2 = 4.2;
// const rating3 = 4.5;
// const averageRating = (rating1 + rating2 + rating3) / 3;
// print(averageRating); // 4.033333333333333
}
```

Tutorial 3:

```
import 'package:lab2_tutorial3/lab2_tutorial3.dart' as lab2_tutorial3;

void main(List<String> arguments) {
  // Annotating variables explicitly :

// It’s fine to always explicitly add the when you declare a variable.
// int myInteger = 10;
// double myDouble = 3.14;

// Creating constant variables :

// These forms of declaration are fine with const:
// const int myInteger = 10;
// const double myDouble = 3.14;

// They’re also fine with final:
// final int myInteger = 10;
// final double myDouble = 3.14;

// Checking the type at runtime

// num myNumber = 3.14;
// print true
// print(myNumber is double);
// print false
// print(myNumber is int);
// print double
// print(myNumber.runtimeType);

// Type conversion
// var integer = 100;
// var decimal = 12.5;
// A value of type 'double' can't be assigned to a variable of type 'int'.
// integer = decimal;

// You can convert this double to an int like so:
// integer = decimal.toInt();

// Operators with mixed types
// const hourlyRate = 19.5;
// const hoursWorked = 10;
// const totalCost = hourlyRate * hoursWorked;

// If you actually do want an int as the result, then you need to perform the
conversion explicitly:
```

```
// The parentheses tell Dart to do the multiplication first, and after that, to
take the result and convert it to an integer value.
// Const variables must be initialized with a constant value.
// const totalCost = (hourlyRate * hoursWorked).toInt();

// Just change const to final:
// final totalCost = (hourlyRate * hoursWorked).toInt();
// print int
// print(totalCost.runtimeType);

// Ensuring a certain type

// const wantADouble = 3;
// Dart infers the type of wantADouble as int. But what if you wanted the constant
to store a double instead?
// final actuallyDouble = 3.toDouble();

// Another option would be to not use type inference at all, and to add the double
annotation:
// const double actuallyDouble = 3;

// Casting down

// num someNumber = 3;
// Below statement gives an error The getter 'isEven'isn't defined for the type
'num'.
// print(someNumber.isEven);

// int my=3;
// Print false
// print(my.isEven);

// final someInt = someNumber as int;
// Print false
// print(someInt.isEven);

// num someNumber = 3;
// final someDouble = someNumber as double;
// _CastError (type 'int' is not a subtype of type 'double' in type cast)
// print(someNumber.isEven);

// If you do need to convert an int to a double atruntime, use the toDouble method
that you saw earlier:

// final someDouble = someNumber.toDouble();

// Mini-exercises :

// 1. Create a constant called age1 and set it equal to 42.
// Create another constant called age2 and set it equal to
// 21. Check that the type for both constants has been
// inferred correctly as int by hovering your mouse pointer
// over the variable names in VS Code.
// Ans1 :
// const age1 = 42;    // int
// const age2 = 21;    // int

// 2. Create a constant called averageAge and set it equal to
// the average of age1 and age2 using the operation (age1
// + age2) / 2. Hover your mouse pointer over
// averageAge to check the type. Then check the result of
// averageAge. Why is it a double if the components are
// all int?
// Ans2 :
// const averageAge = ( age1 + age2 )/2;   // double

// Strings :

// Print Hello, Dart!
// print('Hello, Dart!');

// You can extract that same string as a named variable:
// var greeting = 'Hello, Dart!';
// print(greeting);

// var greeting = 'Hello, Dart!';
// you completely discarded the string 'Hello, Dart!'and replaced it with a whole
new string whose value was 'Hello, Flutter!'.
// greeting = 'Hello, Flutter!';
// print(greeting);

// const letter = 'a';
// print a
// print(letter);

// Single-quotes vs. double-quotes :

// var a = 'I like cats';
// print I like cats
// print(a);
// var b = "I like cats";
// print I like cats
// print(b);

// You might want to use double-quotes, though, if your string includes any
apostrophes.
// var c = "my cat's food";
// print(c);

// Otherwise you would need to use the backslash \
// var d ='my cat\'s food';
// print(d);
// Concatenation :

// var message = 'Hello' + ' my name is ';
// const name = 'Ray';
// message += name;
// print(message);
// 'Hello my name is Ray'

// If you find yourself doing a lot of concatenation, you should use a string
buffer, which is more efficient.
// final message = StringBuffer();
// message.write('Hello');
// message.write(' my name is ');
// message.write('Ray');
// message.toString();
// print(message);
// this above statement print : Hello my name is Ray

// Interpolation :

// const name = 'Ray';
// const introduction = 'Hello my name is $name';
// print(name);
// this above statement print : Hello my name is Ray

// The syntax works in the same way to build a string from other data types such as
numbers:

// const oneThird = 1 / 3;
// const sentence = 'One third is $oneThird.';
// print(sentence);
// This above statement print : One third is 0.3333333333333333.

// final sentence = 'One third is ${oneThird.toStringAsFixed(3)}';
// print(sentence);
// This above statement print :One third is 0.333


// Multi-line strings :

// You can support multi-line text like so:
// const bigString = '''
// You can have a string
// that contains multiple
// lines
// by
// doing this.''';
// print(bigString);

// This above things print :
```

```
// You can have a string
// that contains multiple
// lines
// by
// doing this.

// const oneLine = 'This is only '
// 'a single '
// 'line '
// 'at runtime.';
// print(oneLine);

// This above things print :
// This is only a single line at runtime.

// const oneLine = 'This is only ' +
// 'a single ' +
// 'line ' +
// 'at runtime.';
// print(oneLine);

// This above things print :
// This is only a single line at runtime.


// Print :
// This is
// two lines.
// const twoLines = 'This is\ntwo lines.';
// print(twoLines);

// print : My name \n is $name.
// const rawString = r'My name \n is $name.';
// print(rawString);


// Mini-exercises :

// 1. Create a string constant called firstName and initialize it to your first
name. Also create a string constant called lastName and initialize it to your last
name.
// Ans1 :
// const firstName = "Mit";
// const lastName = "Virani";

// 2. Create a string constant called fullName by adding the firstName and lastName
constants together, separated by a space.
// Ans2 :
// const firstName = "Mit";
// const lastName = "Virani";
// const fullName = firstName + " " + lastName;
// 3. Using interpolation, create a string constant called myDetails that uses the fullName constant to create a string introducing yourself. For example, Ray Wenderlich’s string would read: Hello, my name is Ray Wenderlich.
// Ans3 :
// const firstName = "Mit";
// const lastName = "Virani";
// const myDetails = "Hello, my name is $firstName $lastName."; //Hello, my name is Mit Virani.


// Object and dynamic types :

// Here is an example in JavaScript
// var myVariable = 42;
// myVariable = 123.23;
// print(myVariable);

// var answer = myVariable * 3; // runtime error

// If you try to do the following in Dart:

// var myVariable = 42;
// myVariable = 'hello'; // compile-time error

// dynamic myVariable = 42;
// myVariable = 'hello'; // OK

// var myVariable; // defaults to dynamic
// myVariable = 42; // OK
// myVariable = 'hello'; // OK
// print : hello
// print(myVariable);

// If you need to explicitly say that any type is allowed, you should consider using the Object? type.

// Object? myVariable = 42;
// myVariable = 'hello'; // OK
// print(myVariable);
// This above statement print :hello

// Challenges :

// Challenge 1: Teacher’s grading
// You’re a teacher, and in your class, attendance is worth 20% of the grade, the homework is worth 30% and the exam is worth 50%. Your student got 90 points for her attendance, 80 points for her homework and 94 points on her exam. Calculate her grade as an integer percentage rounded down.
// Ans1 :
// final grade = ((90 * 0.20) + (80 * 0.30) + (94 * 0.50)).toInt();
// print(grade);   // 89

// Challenge 2: Find the error

// What is wrong with the following code?
// const name = 'Ray';
// name += ' Wenderlich';
// Ans2 : We can't change the value of const variable after declaring it.

// Challenge 3: What type?
// What’s the type of value?
// const value = 10 / 2;
// Ans3 : Double

// Challenge 6: In summary
// What is the value of the constant named summary?
// const number = 10;
// const multiplier = 5;
// final summary = '$number∗ $multiplier = ${number ∗ multiplier}';

// Ans6 :
// print(summary); // 10 ∗ 5 = 50

}
```

Tutorial 4:

```dart
import 'package:lab2_tutorial4/lab2_tutorial4.dart' as lab2_tutorial4;

void main(List<String> arguments) {

// Topic : Boolean values

// create some Boolean variables like so:
// const bool yes = true;
// const bool no = false;

// Because of Dart’s type inference, you can leave off the type annotation:

// const yes = true;
// const no = false;

// Boolean operators :

// Booleans are commonly used to compare values.

// Testing equality :

// const doesOneEqualTwo = (1 == 2);
// print : false
// print(doesOneEqualTwo);

// const doesOneEqualTwo = 1 == 2;
// print : false
// print(doesOneEqualTwo);

// Testing inequality :

// const doesOneNotEqualTwo = (1 != 2);
// print : true
// print(doesOneNotEqualTwo);

// print : true
// const alsoTrue = !(1 == 2);
// print(alsoTrue);

// Testing greater and less than :

// const isOneGreaterThanTwo = (1 > 2);
// print : false
// print(isOneGreaterThanTwo);

// const isOneLessThanTwo = (1 < 2);
// print : true
// print(isOneLessThanTwo);

// print(1 <= 2); // true
// print(2 <= 2); // true

// print(2 >= 1); // true
// print(2 >= 2); // true

// Boolean logic :

// AND operator :

// const isSunny = true;
// const isFinished = true;
// const willGoCycling = isSunny && isFinished;
// print(willGoCycling);
// This above given statement print : true
// Print willGoCycling and you’ll see that it’s true. If either isSunny or
isFinished were false, then willGoCycling

// OR operator :

// const willTravelToAustralia = true;
// const canFindPhoto = false;
// const canDrawPlatypus = willTravelToAustralia || canFindPhoto;
// print(canDrawPlatypus);
// This above given statement print : true
// Print canDrawPlatypus to see that its value is true. If both values on the right
were false, then canDrawPlatypus would be false. If both were true, then
canDrawPlatypus would still be true.

// Operator precedence :

// const andTrue = 1 < 2 && 4 > 3;
// print : andTrue
// print(andTrue);
// const andFalse = 1 < 2 && 3 > 4;
// print : andFalse
// print(andFalse);
// const orTrue = 1 < 2 || 3 > 4;
// print : orTrue
// print(orTrue);
// const orFalse = 1 == 2 || 3 == 4;
// print : orFalse
// print(orFalse);

// In This given below expressions precedence is comes in picture ,here &&
precedence is higher than || so first (3 > 4 && 1 < 2) is evaluated and than after
(false || true) is evaluated and final answer is true.
// print(false && true || true)
// print(3 > 4 && 1 < 2 || 1 < 4);

// Overriding precedence with parentheses :

// 3 > 4 && (1 < 2 || 1 < 4) // false
// (3 > 4 && 1 < 2) || 1 < 4 // true

// String equality :

// const guess = 'dog';
// const dogEqualsCat = guess == 'cat';
// print(dogEqualsCat);
// This above given statement print : false

// Mini-exercises :

// 1. Create a constant called myAge and set it to your age.Then, create a constant
named isTeenager that uses Boolean logic to determine if the age denotes someone in
the age range of 13 to 19.
// Ans1 :
// const myAge = 19;
// const isTeenager = (myAge>=13 && myAge<=19);

// 2. Create another constant named maryAge and set it to 30.Then, create a
constant named bothTeenagers that uses Boolean logic to determine if both you and
Mary are teenagers.
// Ans2 :
// const maryAge=30;
// const myAge=19;
// const bothTeenagers = (myAge>=18 && maryAge>=18);

// 3. Create a String constant named reader and set it to your name. Create another
String constant named ray and set it to 'Ray Wenderlich'. Create a Boolean constant
named rayIsReader that uses string equality to determine if reader and ray are
equal.
// Ans3 :
// const String reader = "Mit";
// const String ray = "Ray Wenderlich";
// const bool rayIsReader = (ray==reader) ;

// The if statement :

  // if (2 > 1) {
  //   print('Yes, 2 is greater than 1.');
  // }

// This given above statment print : Yes, 2 is greater than 1.

// The else clause :

  // const animal = 'Fox';
  // if (animal == 'Cat' || animal == 'Dog') {
  //   print('Animal is a house pet.');
  // } else {
  //   print('Animal is not a house pet.');
  // }
// This given above statment print : Animal is not a house pet.

// Else-if chains :

  // const trafficLight = 'yellow';
  // var command = '';
  // if (trafficLight == 'red') {
  //   command = 'Stop';
  // } else if (trafficLight == 'yellow') {
  //   command = 'Slow down';
  // } else if (trafficLight == 'green') {
  //   command = 'Go';
  // } else {
  //   command = 'INVALID COLOR!';
  // }
  // print(command);
// This given above statment print : Slow down.

// Variable scope :

  // const global = 'Hello, world';
  // void main() {
  //   const local = 'Hello, main';
  //   if (2 > 1) {
  //     const insideIf = 'Hello, anybody?';
  //     print(global);
  //     print(local);
  //     print(insideIf);
  //   }
  //   print(global);
  //   print(local);
  //   // print(insideIf); // Not allowed!
  // }

// The ternary conditional operator :
  // const score = 83;
  // String message;
  // if (score >= 60) {
  //   message = 'You passed';
  // } else {
  //   message = 'You failed';
  // }
  // print : You passed
  // print(message);

// (condition) ? valueIfTrue : valueIfFalse;
// const score = 83;
// const message = (score >= 60) ? 'You passed':'You failed';
// print : You passed
// print(message);

// Mini-exercises :

// 1. Create a constant named myAge and initialize it with
// your age. Write an if statement to print out “Teenager”
// if your age is between 13 and 19, and “Not a teenager” if
// your age is not between 13 and 19.
// Ans1 :
// const myAge =19;
// if(myAge>=13 && myAge<=19)
//   print("Teenager");
// else
//   print("Not a teenager");

// 2. Use a ternary conditional operator to replace the else-
// if statement that you used above. Set the result to a variable named answer.
// Ans2 :
// const myAge =19;
// const message = (myAge>=13 && myAge<=19) ? 'Teenager':'Not a teenager';

/*
// Switch Statement :

// An alternate way to handle control flow, especially for multiple conditions, is
with a switch statement.

int c = 1;
  // where c is variable and which can be int,string or compile-time
constant,switch will redirect the program control to one of the case value that
follow.
  switch (c) {
   // each case keywords takes value and compare with the value using == to the
variable and we can add many cases statements and last we add default statement
    case 1:
     // the break keyword used to break the switch statement and exit from the
switch statement
      break;
    case 2:
      break;
    default:
*/

//replacing else-if chains :

  // const number = 3;
  // if (number == 0) {
```

```
  //   print("Zero");
  // } else if (number == 1) {
  //   print("One");
  // } else if (number == 2) {
  //   print("Two");
  // } else if (number == 3) {
  //   print("Three");
  // } else if (number == 4) {
  //   print("Four");
  // } else {
  //   print("Something else");
  // }
// Above given print : Three

  // rewrire upper code in switch statement
  // const number1 = 3;
  // switch (number1) {
  //   case 1:
  //     print("One");
  //     break;
  //   case 2:
  //     print("Two");
  //     break;
  //   case 3:
  //     print("Three");
  //     break;
  //   case 4:
  //     print("Four");
  //     break;
  //   default:
  //     print("Something else");
  // }

// in dart, switch statements don't support ranges like number > 5. only ==
equality checking is allowed. if your condition involve ranges,then you should use
if statements.
// switching on string
//   const weather = 'cloudy';
//   switch (weather) {
//     case 'sunny':
//       print('Put on sunscreen.');
//       break;
//     case 'snowy':
//       print('Get your skis.');
//       break;
//     case 'cloudy':
//     case 'rainy':
//       print('Bring an umbrella.');
//       break;
//     default:
//       print("I'm not familiar with that weather.");
//   }

// here cloudly case was empty with no break statements .Therefore,the code "falls
through" to the "rainy" case. this means that is they

// Enumerated types :

// Enumerated types are also known as enums, play especially well with switch
statement

// const weather1 = "i like turtles";
/* That’s what the default case was there for — to catch all the
   weird stuff that gets through. Wouldn’t it be nice to make
   weird stuff impossible, though? That’s where enums come
   in. */
// enum define our different kinds of weather.

// Create the enum as follows, placing it outside of the main function:

// enum Weather {
//   sunny,
//   snowy,
//   cloudy,
//   rainy,
// }

// Naming enum
// when we write enum name with an intial capital letter,the value of an enum
should use LowerCamelCase unless you have a special raeson to do otherwise

// switching on enum
  // const WeatherToday = Weather.cloudy;
  // switch (WeatherToday) {
  //   case Weather.sunny:
  //     print("Put on sunscreen");
  //     break;
  //   case Weather.snowy:
  //     print("get your skills");
  //     break;
  //   case Weather.cloudy:
  //   case Weather.rainy:
  //     print("bring an umbrella");
  //     break;
  // }

// Enum values and indexes  :

// print(weatherToday);
// print : Weather.cloudy
// Unlike some languages, a Dart enum isn’t an integer. However, you can get the
index, or ordinal placement, of a
// value in the enum like so :

// final index = weatherToday.index;

// Since cloudy is the third value in the enum, the zero-based index is 2.

// Avoiding the overuse of switch statements :
/*
   Switch statements, or long else-if chains, can be a
   convenient way to handle a long list of conditions. If you’re
   a beginning programmer, go ahead and use them; they’re
   easy to use and understand.
   However, if you’re an intermediate programmer and still
   find yourself using switch statements a lot, there’s a good
   chance you could replace some of them with more
   advanced programming techniques that will make your
   code easier to maintain. If you’re interested, do a webs and
   read a few articles about it.

*/

}
```