Practical No. 01

Aim: Program to demonstrate the features of Dart language.

Description:

Simple Dart program to print Hello World−

```
void main() {
  print("Hello World");
}
```

Variables in DART

Variables are containers used to store value in the program. There are different types of variables where you can keep different kinds of values. Here is an example of creating a variable and initializing it.

```
// here variable name contains value Meena.
var name = "Meena";
```

Variable Types
They are called data types.

String: For storing text value. E.g. “John” [Must be in quotes]
int: For storing integer value. E.g. 10, -10, 8555 [Decimal is not included]
double: For storing floating point values. E.g. 10.0, -10.2, 85.698 [Decimal is included]
num: For storing any type of number. E.g. 10, 20.2, -20 [both int and double]
bool: For storing true or false. E.g. true, false [Only stores true or false values]
var: For storing any value. E.g. ‘Bimal’, 12, ‘z’, true

Rules For Creating Variables In Dart

- Variable names are case sensitive, i.e., a and A are different.
- A variable name can consist of letters and alphabets.
- A variable name cannot start with a number.
- Keywords are not allowed to be used as a variable name.
- Blank spaces are not allowed in a variable name.
- Special characters are not allowed except for the underscore (_) and the dollar ($) sign.

Dart Constant

Constant is the type of variable whose value never changes. In programming, changeable values are mutable and unchangeable values are immutable. Sometimes, you don't need to change the value once declared. Like the value of PI=3.14, it never changes. To create a constant in Dart, you can use the const keyword.

```
void main(){
const pi = 3.14;
pi = 4.23; // not possible
print("Value of PI is $pi");
}
```

DATA TYPES IN DART

Data types help you to categorize all the different types of data you use in your code. For e.g. numbers, texts, symbols, etc. The data type specifies what type of value will be stored by the variable. Each variable has its data type. Dart supports the following built-in data types :

1. Numbers
2. Strings
3. Booleans
4. Lists
5. Maps

Data Type	**Keyword**	**Description**
Numbers	int, double, num	It represents numeric values
Strings	String	It represents a sequence of characters
Booleans	bool	It represents Boolean values true and false
Lists	List	It is an ordered group of items
Maps	Map	It represents a set of values as key-value pairs

Lists and Maps − It is used to represent a collection of objects. A simple List can be defined as below −.

```
void main() {
  var list = [1,2,3,4,5];
  print(list);
}
```

The list shown above produces [1,2,3,4,5] list. **(Also add the functions you had used for the List during practical session)**

Map can be defined as shown here −

```
void main() {
  var student = {'Name':'Heena','Age':19,'Marks':90.88};
  print(student);
}
```

Dynamic − If the variable type is not defined, then its default type is dynamic. The following example illustrates the dynamic type variable −

```
void main() {
  dynamic name = "Dart";
  print(name);
}
```

Type Conversion In Dart

In dart, type conversion allows you to convert one data type to another type. For e.g. to convert String to int, int to String or String to bool, etc.

Convert String To Int In dart

You can convert String to int using int.parse() method. The method takes String as an argument and converts it into an integer.

```
void main() {
String strvalue = "1";
print("Type of strvalue is ${strvalue.runtimeType}");
int intvalue = int.parse(strvalue);
print("Value of intvalue is $intvalue");
// this will print data type
print("Type of intvalue is ${intvalue.runtimeType}");
}
```

Convert String To Double In Dart

You can convert String to double using double.parse() method. The method takes String as an argument and converts it into a double.

```
void main() {
String strvalue = "1.1";
print("Type of strvalue is ${strvalue.runtimeType}");
double doublevalue = double.parse(strvalue);
print("Value of doublevalue is $doublevalue");
// this will print data type
print("Type of doublevalue is ${doublevalue.runtimeType}");
}
```

Convert Int To String In Dart

You can convert int to String using the toString() method. Here is example:

```
void main() {
int one = 1;
print("Type of one is ${one.runtimeType}");
String oneInString = one.toString();
print("Value of oneInString is $oneInString");
// this will print data type
print("Type of oneInString is ${oneInString.runtimeType}");
}
```

Convert Double To Int In Dart

You can convert double to int using the toInt() method.

```
void main() {
   double num1 = 10.01;
   int num2 = num1.toInt(); // converting double to int

  print("The value of num1 is $num1. Its type is ${num1.runtimeType}");
  print("The value of num2 is $num2. Its type is ${num2.runtimeType}");
}
```

Optionally Typed Language

You may have heard of the statically-typed language. It means the data type of variables is known at compile time. Similarly, dynamically-typed language means data types of variables are known at run time. Dart supports dynamic and static types, so it is called optionally-typed language.

Statically Typed

A language is statically typed if the data type of variables is known at compile time. Its main advantage is that the compiler can quickly check the issues and detect bugs.

```
void main() {
  var myVariable = 50; // You can also use int instead of var
  myVariable = "Hello"; // this will give error
  print(myVariable);
}
```

Dynamically Typed Example

A language is dynamically typed if the data type of variables is known at run time.

```
void main() {
  dynamic myVariable = 50;
  myVariable = "Hello";
  print(myVariable);
}
```

Difference Between var and dynamic type in Dart

Use var if you expect a variable assignment to change during its lifetime:

```
var msg = "Hello world.";
msg = "Hello world again.";
```

You can change the type of x but not a

```
void main() {
 dynamic x = 'hal';
 x = 123;
 print(x);
 var a = 'hal';
 a = 123;
 print(a);
}
```

Note:
dynamic: can change TYPE of the variable, & can change VALUE of the variable later in code.
var: can't change TYPE of the variable, but can change the VALUE of the variable later in code.

Decision Making and Loops

A decision making block evaluates a condition before the instructions are executed. Dart supports If, If..else and switch statements.

Loops are used to repeat a block of code until a specific condition is met. Dart supports for, for..in , while and do..while loops.

This is the most common type of loop. You can use for loop to run a code block multiple times according to the condition. The syntax of for loop is:

```
for(initialization; condition; increment/decrement){
        statements;
}
```

Initialization is executed (one time) before the execution of the code block.
Condition defines the condition for executing the code block.
Increment/Decrement is executed (every time) after the code block has been executed.

To Print 1 To 10 Using For Loop
This example prints 1 to 10 using for loop. Here int i = 1; is initialization, i<=10 is condition and i++ is increment/decrement.

```
void main() {
  for (int i = 1; i <= 10; i++) {
    print(i);
  }
}
```

Example for For in loop

```
  var names = ["John", "Mary", "Reena", 1, 20, 30];
  for (var fname in names)
  {
    print(fname);
  }
```

Let us understand another simple example about the usage of control statements and loops −

```
void main() {
  for( var i = 1 ; i <= 10; i++ ) {
    if(i%2==0) {
      print(i);
    }
  }
}
```

The above code prints the even numbers from 1 to 10.

Functions

A function is a group of statements that together performs a specific task. Let us look into a simple function in Dart as shown here −

```
void main() {
  add(3,4);
}
void add(int a,int b) {
  int c;
  c = a+b;
  print(c);
return c;
}
```

The above function adds two values and produces 7 as the output.

Object Oriented Programming

Dart is an object-oriented language. It supports object-oriented programming features like classes, interfaces, etc.

A class is a blueprint for creating objects. A class definition includes the following −

- Fields
- Getters and setters
- Constructors
- Functions

Now, let us create a simple class using the above definitions −

```
class Employee {
  String name;

  //getter method
  String get emp_name {
    return name;
  }
  //setter method
  void set emp_name(String name) {
    this.name = name;
  }
  //function definition
  void result() {
    print(name);
  }
}
void main() {
  //object creation
  Employee emp = new Employee();
  emp.name = "employee1";
  emp.result(); //function call
}
```