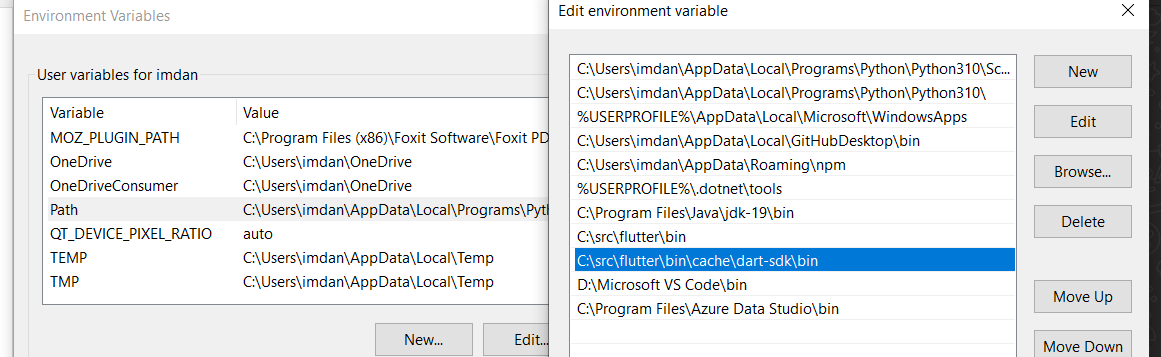
**Byte Wise Fellow Ship (Flutter)**

During my three months flutter fellowship at Byte wise I have done multiple task and projects. All of these task explained below

**Week 1 + 2**

**Environment Setup**

I installed flutter and dart and add its path to environment variable  
 

Then I downloaded android studio.

**Introduction to Flutter**

Flutter is a growing mobile framework developed by Google, designed to help in the development of mobile and web applications with only one code base and the ability to export them as native apps for Android and iOS.

Flutter is a tool that allows developers to build native cross-platform apps with just one programming language and one codebase. It will not create an app that runs in the browser or something that gets wrapped by native apps. Instead, it creates a native app for both iOS and Android that can be published to the stores later.  
Flutter is made if **2 major components; 1) SDK Software Development Kit** is a collection of tools that will assist you in developing your applications. Tools for compiling your code into native machine code are included one for ios and other for android. **2) Framework** is a widget-based UI library a set of reusable UI elements such as buttons, text inputs, and sliders and so on so that you can customize to meet your specific needs.

**Dart**

Dart is a programming language that will be used to create flutter applications. It is focused of building front-end user interfaces and front-end apps. By using it, you will be able to create either web apps or mobile apps.

It is developed by Google and is class based, object oriented and strongly typed programming language. Dart’s syntax is very similar to languages like Java or JavaScript.

Flutter and Dart are actually two independent things that work together. Flutter is a framework for Dart, and Dart uses the capabilities of Flutter to build the app. In the end, Flutter working as an SDK will help to build the app.

In this week, I learned about basics of Dart language.

I use [DartPad](https://dartpad.dev/?) for practicing dart language

**Dart- Data Types**

The Dart Language supports the following types

* Integer
* Double
* String
* Bool
* Lists
* Maps
* Dynamic

**List-** A list is an ordered group of objects. The list data type in Dart is synonymous to concept of an array in other programming languages.

**Map-** The Map data type represents a set of values as key-values pairs. The **dart: core** library enables creation and manipulation of these collections through the predefined List and Maps classes respectively.

**Dynamic Type**Dart is an optionally typed language. If the type of a variable is not explicitly specified, the variable’s type is **dynamic**. The **dynamic** keyword can also be used as a type annotation explicitly.

**Type Syntax**

A variable must be declared before it is used. Dart uses the var keyword to achieve the same. The syntax for declaring a variable is as given below.

**Note:** Dart supports both type inference and statically type defined

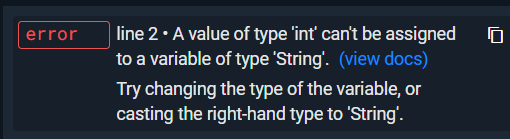
var name='Daniyal';

It means value store in **name** variable is a string because Dart compiler automatically guess it because of **var** keyword.

var num=10;  
It means value store in **num** is a integer value.  
  
The above mentioned syntax are called type inference

String name='Daniyal';  
int num=10;  
This is called statically type defined as I define the data type of name and number   
If I use String Data Type and assign a integer value in it then it shows an error

**For Example:**

String num=10; ****

**Dart Programming- Operators**

Following operators are available in dart

* Arithmetic Operators
* Equality and Relational Operators
* Assignment Operators
* Logical Operators
* Type Test Operators
* Bitwise Operators

**Type test Operators**

These operators are handy for checking types at runtime

|  |  |
| --- | --- |
| **Operator** | **Meaning** |
| is | True if the object has the specified type |
| is! | False if the object has the specified type |

**Dart Programming- Loops**

At times, certain instructions require repeated execution. Loops are an ideal way to do the same. A loop represents a set of instructions that must be repeated.

**Dart Programming- Enumeration**

An enumeration is used for defining named constant values. An enumerated type is declared using the enum keyword.

**Syntax**

enum enum\_name{

enumeration list

}

enum Status {

none,

running,

stopped,

paused

}

void main() {

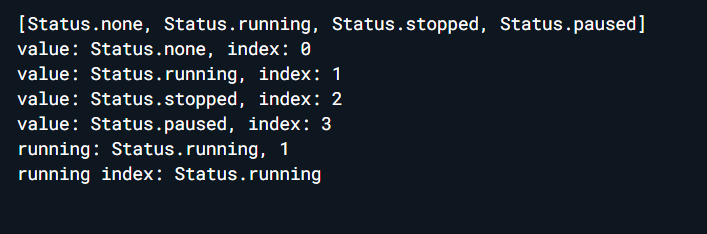
print(Status.values);

Status.values.forEach((v) => print('value: $v, index: ${v.index}'));

print('running: ${Status.running}, ${Status.running.index}');

print('running index: ${Status.values[1]}');

}



**Dart Programming - Classes**

Dart is an object-oriented language. It supports object-oriented programming features like classes, interfaces, etc. A class in terms of OOP is a blueprint for creating objects. A class encapsulates data for the object. Dart gives built-in support for this concept called class.

## Declaring a Class

Use the class keyword to declare a class in Dart. A class definition starts with the keyword class followed by the class name; and the class body enclosed by a pair of curly braces. The syntax for the same is given below −

**Syntax**

class class\_name {

<fields>

<getters/setters>

<constructors>

<functions>

}

The class keyword is followed by the class name. The rules for identifiers must be considered while naming a class.

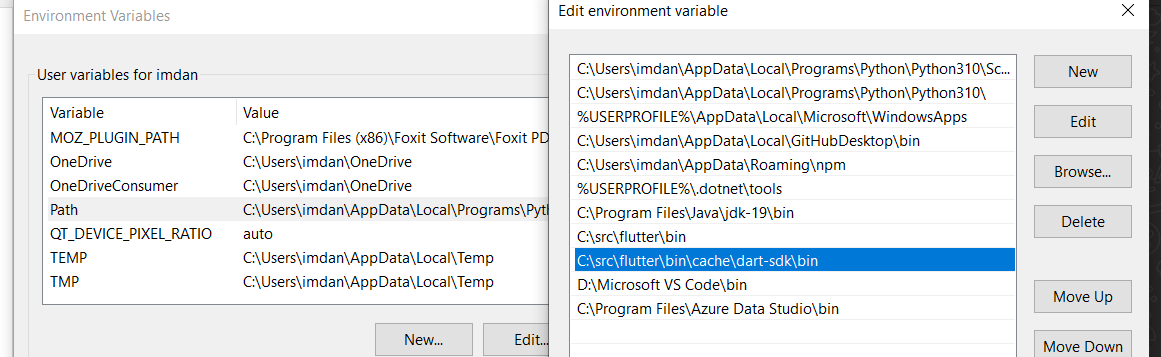
A class definition can include the following −

* **Fields** − A field is any variable declared in a class. Fields represent data pertaining to objects.
* **Setters and Getters** − Allows the program to initialize and retrieve the values of the fields of a class. A default getter/ setter is associated with every class. However, the default ones can be overridden by explicitly defining a setter/ getter.
* **Constructors**− responsible for allocating memory for the objects of the class.

**Functions** − Functions represent actions an object can take. They are also at times referred to as methods.

**Week 2**

**Environment Setup**

I installed flutter and dart and add its path to environment variable  
 

Then I downloaded android studio.

**Introduction to Flutter**

Flutter is a growing mobile framework developed by Google, designed to help in the development of mobile and web applications with only one code base and the ability to export them as native apps for Android and iOS.

Flutter is a tool that allows developers to build native cross-platform apps with just one programming language and one codebase. It will not create an app that runs in the browser or something that gets wrapped by native apps. Instead, it creates a native app for both iOS and Android that can be published to the stores later.  
Flutter is made if **2 major components; 1) SDK Software Development Kit** is a collection of tools that will assist you in developing your applications. Tools for compiling your code into native machine code are included one for ios and other for android. **2) Framework** is a widget-based UI library a set of reusable UI elements such as buttons, text inputs, and sliders and so on so that you can customize to meet your specific needs.

**Dart**

Dart is a programming language that will be used to create flutter applications. It is focused of building front-end user interfaces and front-end apps. By using it, you will be able to create either web apps or mobile apps.

It is developed by Google and is class based, object oriented and strongly typed programming language. Dart’s syntax is very similar to languages like Java or JavaScript.

Flutter and Dart are actually two independent things that work together. Flutter is a framework for Dart, and Dart uses the capabilities of Flutter to build the app. In the end, Flutter working as an SDK will help to build the app.

In this week, I learned about basics of Dart language.

I use [DartPad](https://dartpad.dev/?) for practicing dart language

**Dart- Data Types**

The Dart Language supports the following types

* Integer
* Double
* String
* Bool
* Lists
* Maps
* Dynamic

**List-** A list is an ordered group of objects. The list data type in Dart is synonymous to concept of an array in other programming languages.

**Map-** The Map data type represents a set of values as key-values pairs. The **dart: core** library enables creation and manipulation of these collections through the predefined List and Maps classes respectively.

**Dynamic Type**Dart is an optionally typed language. If the type of a variable is not explicitly specified, the variable’s type is **dynamic**. The **dynamic** keyword can also be used as a type annotation explicitly.

**Type Syntax**

A variable must be declared before it is used. Dart uses the var keyword to achieve the same. The syntax for declaring a variable is as given below.

**Note:** Dart supports both type inference and statically type defined

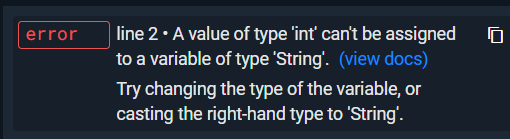
var name='Daniyal';

It means value store in **name** variable is a string because Dart compiler automatically guess it because of **var** keyword.

var num=10;  
It means value store in **num** is a integer value.  
  
The above mentioned syntax are called type inference

String name='Daniyal';  
int num=10;  
This is called statically type defined as I define the data type of name and number   
If I use String Data Type and assign a integer value in it then it shows an error

**For Example:**

String num=10; ****

**Dart Programming- Operators**

Following operators are available in dart

* Arithmetic Operators
* Equality and Relational Operators
* Assignment Operators
* Logical Operators
* Type Test Operators
* Bitwise Operators

**Type test Operators**

These operators are handy for checking types at runtime

|  |  |
| --- | --- |
| **Operator** | **Meaning** |
| is | True if the object has the specified type |
| is! | False if the object has the specified type |

**Dart Programming- Loops**

At times, certain instructions require repeated execution. Loops are an ideal way to do the same. A loop represents a set of instructions that must be repeated.

**Dart Programming- Enumeration**

An enumeration is used for defining named constant values. An enumerated type is declared using the enum keyword.

**Syntax**

enum enum\_name{

enumeration list

}

enum Status {

none,

running,

stopped,

paused

}

void main() {

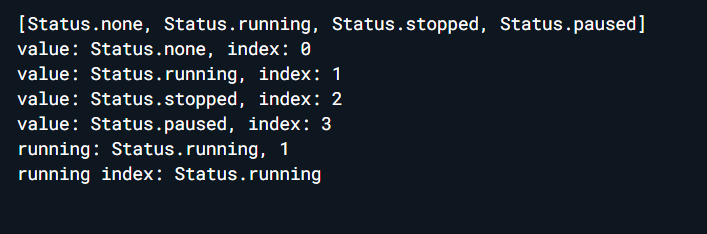
print(Status.values);

Status.values.forEach((v) => print('value: $v, index: ${v.index}'));

print('running: ${Status.running}, ${Status.running.index}');

print('running index: ${Status.values[1]}');

}



**Dart Programming - Classes**

Dart is an object-oriented language. It supports object-oriented programming features like classes, interfaces, etc. A class in terms of OOP is a blueprint for creating objects. A class encapsulates data for the object. Dart gives built-in support for this concept called class.

## Declaring a Class

Use the class keyword to declare a class in Dart. A class definition starts with the keyword class followed by the class name; and the class body enclosed by a pair of curly braces. The syntax for the same is given below −

**Syntax**

class class\_name {

<fields>

<getters/setters>

<constructors>

<functions>

}

The class keyword is followed by the class name. The rules for identifiers must be considered while naming a class.

A class definition can include the following −

* **Fields** − A field is any variable declared in a class. Fields represent data pertaining to objects.
* **Setters and Getters** − Allows the program to initialize and retrieve the values of the fields of a class. A default getter/ setter is associated with every class. However, the default ones can be overridden by explicitly defining a setter/ getter.
* **Constructors**− responsible for allocating memory for the objects of the class.
* **Functions** − Functions represent actions an object can take. They are also at times referred to as methods.