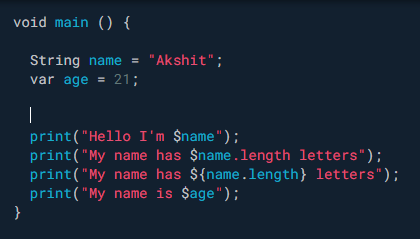
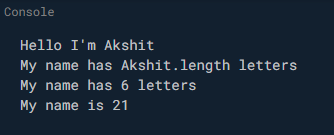
In flutter everything is a widget.

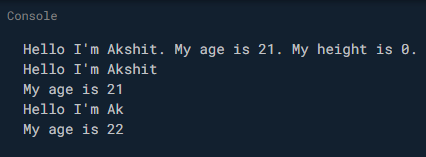
We need dart language.

**Introduction to Dart**

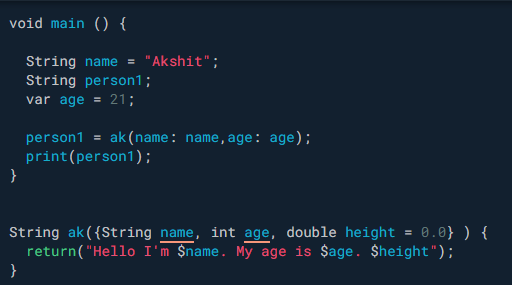
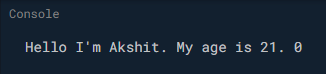
* The dart language
  + Dart is OOPS language.
  + C-style syntax
  + Statically typed
  + Multiple environments
  + Productive and fast
* Introduction to dartpad
  + <https://dartpad.dartlang.org>

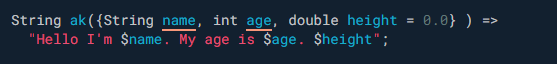
* + - **String interpolation**.
    - **Type inference**
      * Auto detect DT
    - **Statically** typed.
      * Cannot change DT once assigned.
      * Cannot store string in var age after int assigned.
    - We can use **var** for mutable variable.
    - We can use **final** for **immutable** variable.
      * Change value of variable once assigned.
    - We can use **dynamic** keyword.
      * After this we can change DT of string to int and so on.
  + **Function**

* + - We have to use final keyword for return something. .
    - We can make some arguments optional.
      * For this we have to use argument inside [ ].
      * But if we don’t have any value our code will print NULL, so instead of it we can assign a default value to the function.
  + If we have lots of arguments sometimes it is harder to manage, so here we have both ways, we can pass positional argument as above or named arguments.
    - For this we have to use curly braces.
    - They will by default work as a default.

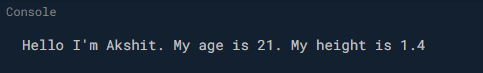
 

* + - We also have arrow operator in dart, it will work when you have single line in function.

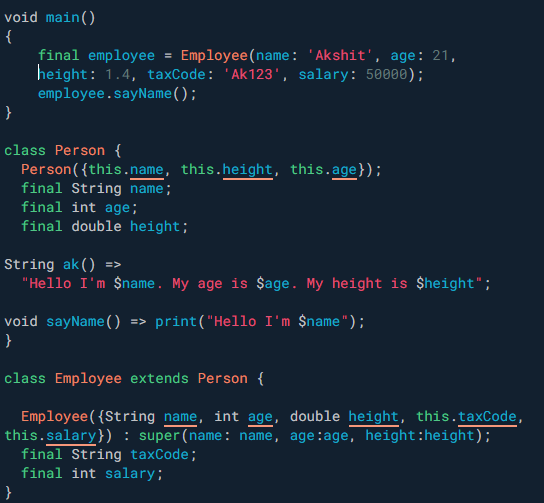


* Classes
  + If we make variables final then we cannot change it after calling a class.
    - Eg, we cannot write person.name = “pal”
    - Here we have used Person constructor to initialize value of string int double.

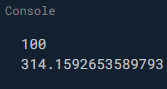
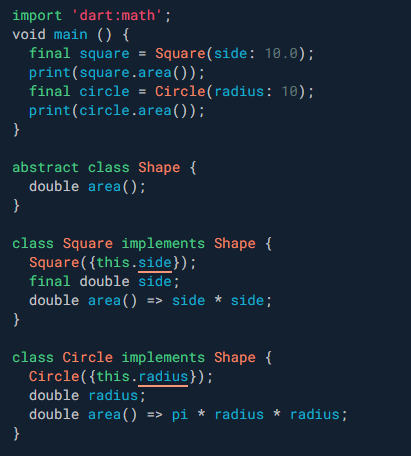




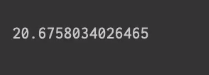
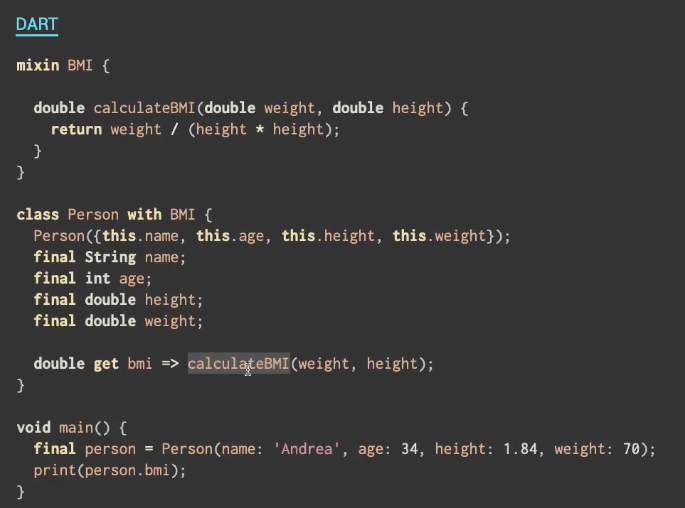
* + As we know sub class **extends** super class and has all the functions of superclass.
  + We can also pass parameter from sub class to super class.

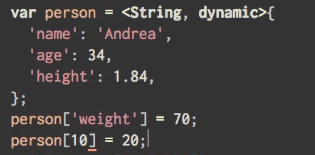

* + **Object class** is a root of Dart class hierarchy.
    - Every other class are subclass of Object class.
    - By deault we have one function toString(), written in object class it will return method belongs to which class.
    - We can also override this.
    - You have to write @override above toString because you are overriding it.
  + If you don’t write anything after class name, then it will by default use classname.toString function.
    - Eg, print(employee) == print(employee.toString())
  + **Abstract class** method cannot be initiated.
    - We use this class when we want to implement this in next class.
    - We must have to implement all method of abstract class.



* **Computed properties** of dart.
  + We have getters and setters as dart computed property.
  + In above code in abstract class, we can write double get area.
  + We don’t require brackets; we have to do same in below where we have implemented shape methods.
  + While printing we can write circle.area, we don’t require brackets here also.
  + They will improve **code readability**.
* Mixins
  + Classes and mixins cannot define same variables & methods name.
  + We can add mixins to the existing class.



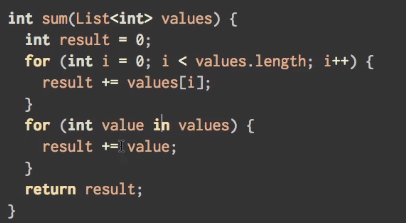
* Dart Collections
  + Collection is a group of objects.
  + We have 3 types of dart collections.
  + Lists sets & Maps
    - Lists is ordered group of objects.
    - Maps is a collection of key value pair.
* We can use generics for security.
  + Suppose in list we can add any number or string.
  + For specify only numbers or strings we can use generics.
  + Its also said type annotation.
  + For this we have to give DT inside angular brackets.



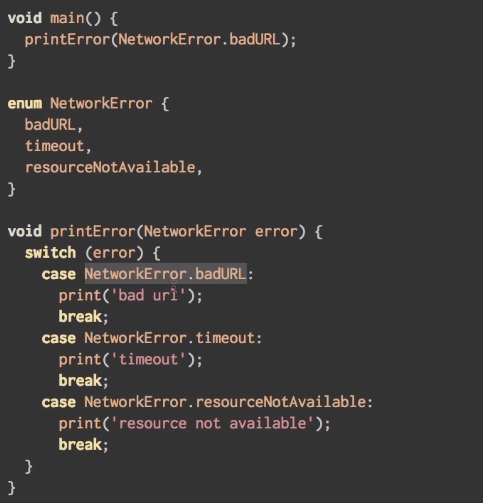
* + Here you can see we cannot have key = number.



* For loop



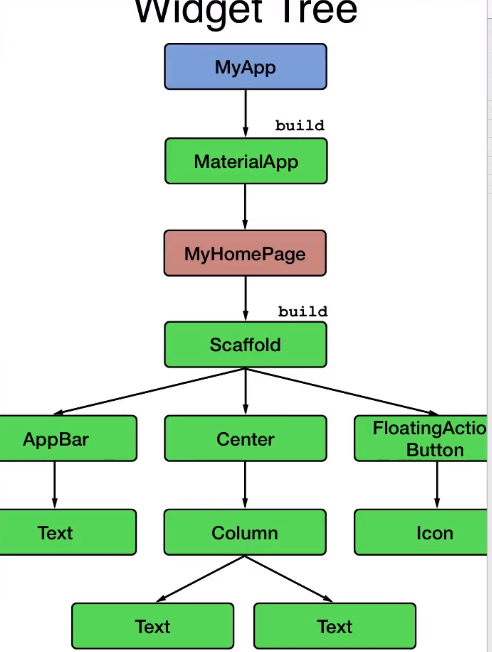
* Enumeration
  + An enum is a class used to represent a fixed number of constant values.



* Installation
  + Go to flutter.io and download for your specific OS.
  + After downloading extract in some location (Document’s folder).
  + Update path variable for it.
    - In windows search for edit env… click on it.
    - Click on Path in user variables.
    - Add new path.
    - C:\Users\admin\Documents\flutter\bin

Flutter

* We have created a folder inside our Documents folder namely flutter\_apps.
  + In this folder we are going to build our app.
  + We will create a new flutter project with flutter create first\_app.
  + Go inside app folder and write flutter run.
* Hot reload & hot restart
* Widget
  + Object that describes a part of user interface.
  + Something we can see in the screen are widget.
* Material App
* Scaffold
* Flutter UI is declarative.
* Widget tree



* Stateless & stateful widget
  + Stateless widgets are immutable.
  + UI always look the same state once build.
    - State is a data might vary during lifecycle of widget.
  + Stateful widget can update his state.

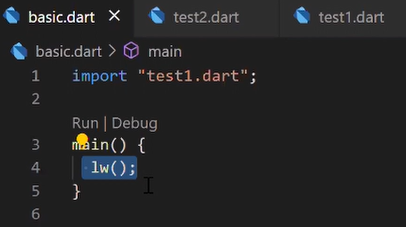
Session 01 – Intro to flutter

Session 02 – gradle//hot reload//host restart

* Flutter devices
* Flutter emulators
* Flutter emulators –launch <Nexus\_5x\_API\_28\_classroom\_flutter>
* For android we use build (compile) tool called **gradle**.
* If you create a OS on top of other OS.
  + It will reduce your performance.
  + To increase performance of sub-OS we have to install HAXM (intel x86 emulator accelerator).
* Flutter has concept called hot reload.
  + App is running you don’t have to compile everything again; it will compile only part which is changed.
    - They will automatically know what is changed.
* Here save is equal to hot reload.

Session 03 – physical mobile//dart start//runApp//materialApp

* For running code inside your physical phone, you have to connect to your laptop using USB.
  + Turn on developer mode & USB debug.
  + Flutter is a framework and understand only dart language.
    - They have given you some pre created methods.
    - We have to just use this methods//program files.
  + Installing dart, create a path for dart till bin.
    - You can try your code with dart file.dart.
    - Dart code always start with main () function.
      * It is an entry point.



* + - In main file we write only main ().
* **runApp()**
  + run app function will send your app to mobile//AVD and run it for you.
  + In flutter for running your app you have to use this method.
* **materialApp()**
  + For creating a app we need materialApp() function.
  + There are lots of widget created by flutter team.
  + Which type of widget you want to launch.
  + Here we are using material App type of widget.
    - We have to write home inside materialApp.
    - **Home** is a entry point for application.
  + We have to write something in home page.
    - Suppose we are writing something, but you have to specify it is image or Text.
    - For this we have to write text inside **Text()** function.

Session 04

* Data types
  + String
  + Int
  + Var
  + List<String>
    - If you want to fix data type inside list you can set this constraint using < >.
  + Map
    - Key-value pair.
* For auto format use comma at last.
* Multi line comment /\* \*/
* Dynamic function
* Flutter always look (by default) for main.dart in lib folder.
  + And start executing from main() function.
* For build & run a app we have to use runApp().
  + Inside app what you want to show, view of App, User Interface we have to mention.
  + Lot of company has created their own design language/platform (framework) how UI should like.
    - Use this kind of icons, use this font size.
  + Google has also created this type of platform called materialApp().
  + When you create a App you have to create a base (canvas).
    - On top of this we draw icon, login page button.
    - Without this base canvas you cannot create a button and other things.
  + If you want to write some Text in center you have to use Text function & put inside a parent center function.
    - Both are a different function.
    - **Text** is a child of **center** function.
      * For this you have to use keyword called **child**.
    - In flutter almost all function comes with key value pair.

