**Day2**

**Difference between value type vs reference type**

**Value type**: If variable is assigned to another variable a whole new memory is created and values are copied to that memory and it assigned to that variable hence if something is changed in new variable it will not be reflected in old memory e.g.: structure, enums, tuples, primitive types. etc.

**Reference type**: if a variable is assigned to another variable, its base address is assigned to the new variable so whenever something is changed using any of the variable it changes in the base address, hence it is reflected in both the variables e.g.: classes, closures etc.

Mainly used value and reference type are structure and classes, swift gives more advantages to structures than in Many languages, Hence the view of where to use value type are changed in swift

**Why we should prefer value type in collections**

In swift collection are in value type, mainly because values are thread safe and logic can be easily viewed in scope

**When/where should prefer call by value over call by reference**

**Use a value type when:**

* Comparing instance data with == makes sense. A double equal operator (aka ==) compares **values**.
* You want copies to have independent state.
* The data will be used in code across multiple threads. So that you don’t have to worry about the data being changed from another thread.

**Use a reference type (e.g. use a class) when:**

* Comparing instance identity with === makes sense. **===** checks if two objects are *exactly* identical, right down to the memory address that stores the data.
* You want to create shared, mutable state.

**Where/why we should prefer value type for return type:**

Mostly we should return of using value types hence everything is inside the scope .

Using reference type will be error prone since we don’t have control over where all the values are changed