Lecture 3 : Dart basics **Level : 3**

Department : cyber security

Operations on lists

List is a data type stores multiple values based on its index

Example

```
  List names = ['ahmed', 'mohammed', 'Majed', 'Abdullah'];
```

names.length : shows the length of list

names.isEmpty : returns true if the list is empty , false otherwise

names.contains(value) : returns true if value exists ,

names.add(value) : add to list

names.addAll(iterable) : add multiple values to list

names.insert (position,value) : insert at specific position

Maps

Maps is datatype contains key and value

Example

```
  Map info = {
    'name': 'mohammed ',
    'age': 25,
    'specificaiton': 'cyber security'
  };
```

Converts from Map to List

```
  List names = ['ahmed', 'mohammed', 'Majed', 'Abdullah'];

  Map info = {
    'name': 'mohammed ',
    'age': 25,
    'specificaiton': 'cyber security'
  };

  info.forEach((key, value) {
    names.add(key);
  });
  print(names);
}
```

Null safety concept

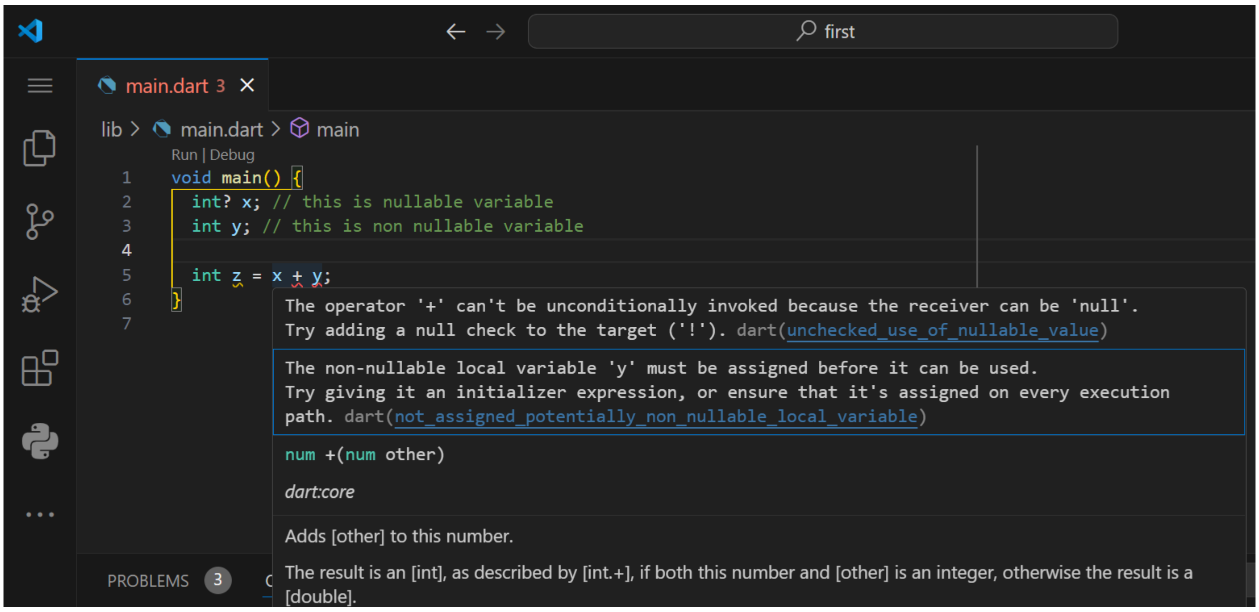
Null safety is a feature in programming helps to avoid null reference errors ,

Null safety categorize variables into 2 categories

Nullable

Non-Nullable

Example



```
void main() {
  int? x; // this is nullable variable
  int y; // this is non nullable variable

  int z = x + y;
}
```

```
The operator '+' can't be unconditionally invoked because the receiver can be 'null'.
Try adding a null check to the target ('!'). dart(unchecked_use_of_nullable_value)

The non-nullable local variable 'y' must be assigned before it can be used.
Try giving it an initializer expression, or ensure that it's assigned on every execution
path. dart(not_assigned_potentially_non_nullable_local_variable)

num +(num other)

dart:core

Adds [other] to this number.

The result is an [int], as described by [int.+], if both this number and [other] is an integer, otherwise the result is a
[double].
```

Here we have 2 errors in this code

+ operator can not apply operation on null value

How to solve it ?

Adding null check operator (!) to the variable x!

Now we telling the compiler this variable is not null

Second error is when directly using not nullable variable y without initialize a value to it

So

```
void main() {
  int? x; // this is nullable variable
  int y=8; // this is non nullable variable

  int z = x! + y;
}
```

Now the compiler will never give a compile error because we ensured that the variable x is not null anymore.

But running the code will throw an exception

Null check operator used in null value

This means the error here is your responsibility

To avoid this error we never add a null check operator if we not sure that the variable is not initialized

```
void main() {
  int? x; // this is nullable variable
  int y = 8; // this is non nullable variable

  if (x != null) {
    int z = x + y;
    print(z);
  }
}
```

Try catch

Try catch is a technique give us ability to handle the exceptions thrown by the compiler and prevents crash of program

Syntax

try {

// here we put the code that we assume it will cause an error

}

Catch (exception)

{

// here we handle the error

}

Finally {

// this brackets always executed

}

try : always executed

catch : executed only when error arise

finally : always executed (optional)

```
void main() {
  int? x = 44; // this is nullable variable
  int y = 8; // this is non nullable variable

  try {
    int z = x + y;
    print(z);
    print('this print from try');
  } catch (e) {
    print(e.toString());
    print('this print from catch ');
  } finally {
    print('this print from finally ');
  }
```

Asynchrony support

Asynchrony means the statement will take some time to execute

Keywords :

Async : this keyword mark the function as an asynchronous function

this is a simple asynchronous function

```
add2Numbers(int a, int b) async {
 return a+b;
}
```

Await : this keyword pause the execution until value is returned

```
add2Numbers(int a, int b) async {
  print('loading...');
  return await a + b;
```

Future :

Is the datatype returned when using async functions

```
void main() async {
  print(await add2Numbers(2, 1));
}

add2Numbers(int a, int b) async {
  print('loading...');
 await Future.delayed(Duration(seconds: 3));
  return a + b;
}
```

When we invoke or call a sync function in main method we must identify main method as an async method .

Labwork

Write async function called myName to print your first name then delay by 2 seconds then second name and delay by 2 seconds then last name