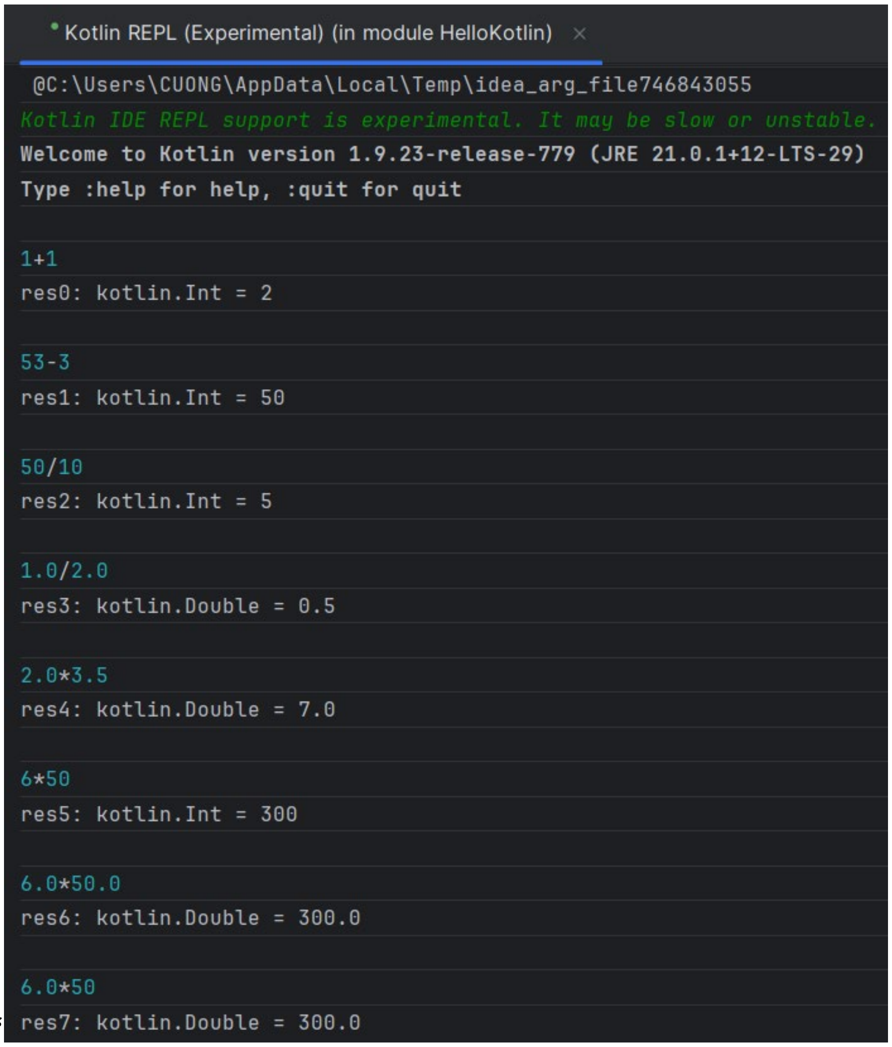
Name: Nguyễn Văn Cường – Student ID: 20215006

Report - Lesson 1: Kotlin basics

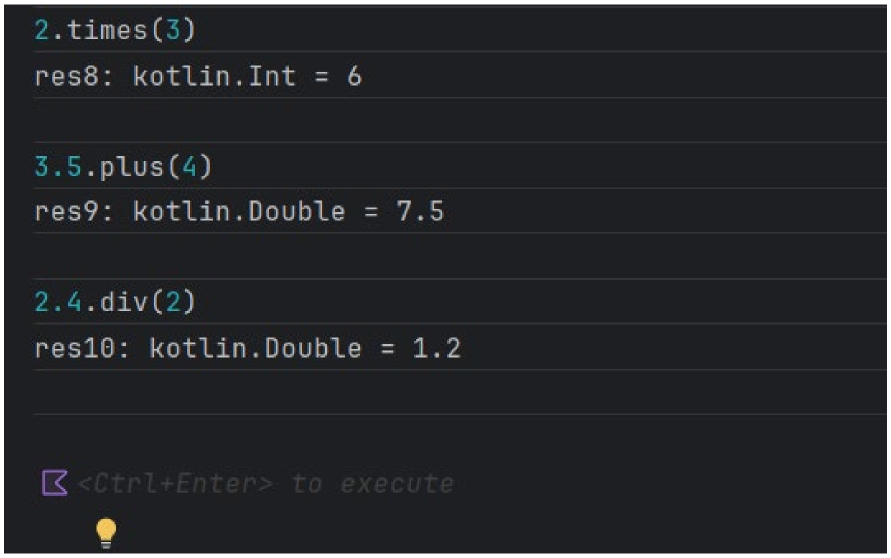
1. Learn about operators and types

a. Step 1: Explore numeric operators



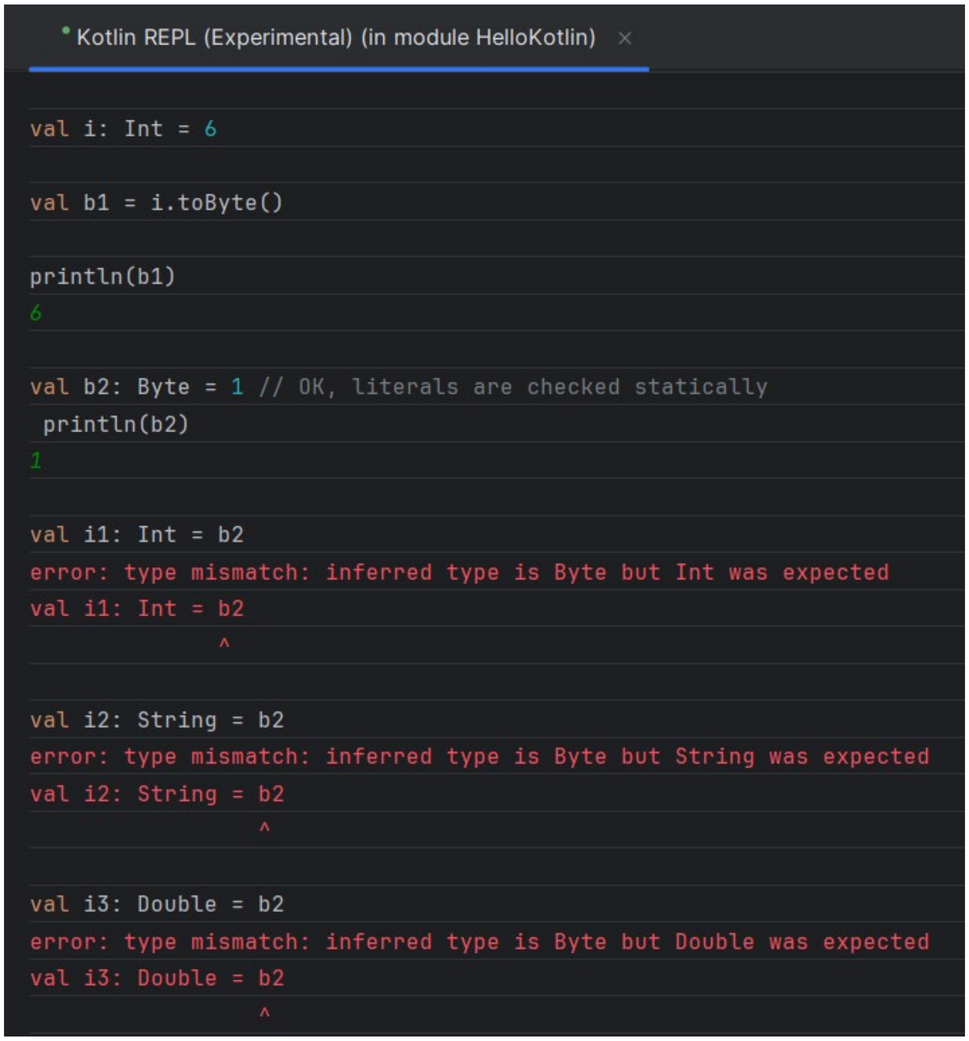
Kotlin REPL (Experimental) (in module HelloKotlin)

```
@C:\Users\CUONG\AppData\Local\Temp\idea_arg_file746843055
Kotlin IDE REPL support is experimental. It may be slow or unstable.
Welcome to Kotlin version 1.9.23-release-779 (JRE 21.0.1+12-LTS-29)
Type :help for help, :quit for quit

1+1
res0: kotlin.Int = 2

53-3
res1: kotlin.Int = 50

50/10
res2: kotlin.Int = 5

1.0/2.0
res3: kotlin.Double = 0.5

2.0*3.5
res4: kotlin.Double = 7.0

6*50
res5: kotlin.Int = 300

6.0*50.0
res6: kotlin.Double = 300.0

6.0*50
res7: kotlin.Double = 300.0
```



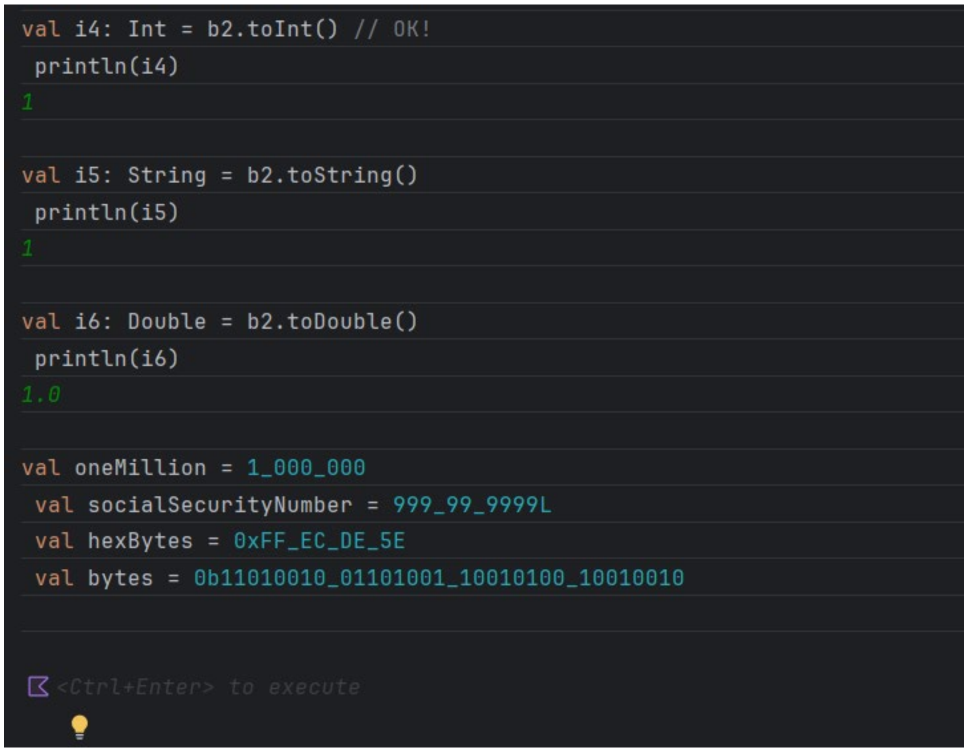
```
2.times(3)
res8: kotlin.Int = 6

3.5.plus(4)
res9: kotlin.Double = 7.5

2.4.div(2)
res10: kotlin.Double = 1.2

<Ctrl+Enter> to execute
```

b. *Step 2: Practice using types*



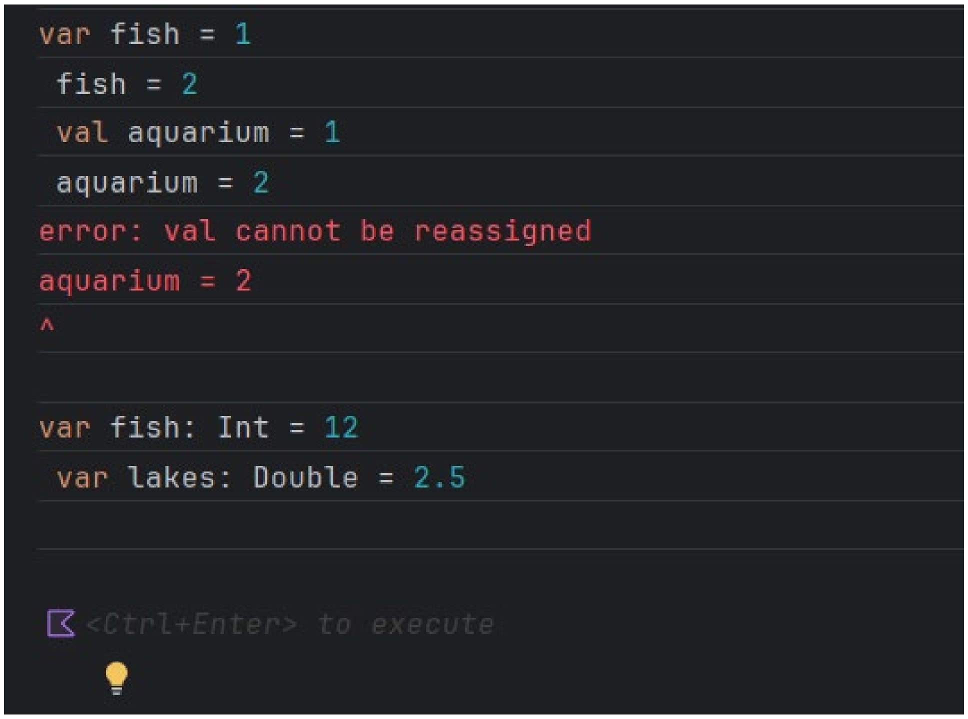
Kotlin REPL (Experimental) (in module HelloKotlin)

```
val i: Int = 6

val b1 = i.toByte()

println(b1)
6

val b2: Byte = 1 // OK, literals are checked statically
 println(b2)
1

val i1: Int = b2
error: type mismatch: inferred type is Byte but Int was expected
val i1: Int = b2
              ^

val i2: String = b2
error: type mismatch: inferred type is Byte but String was expected
val i2: String = b2
                 ^

val i3: Double = b2
error: type mismatch: inferred type is Byte but Double was expected
val i3: Double = b2
                 ^
```



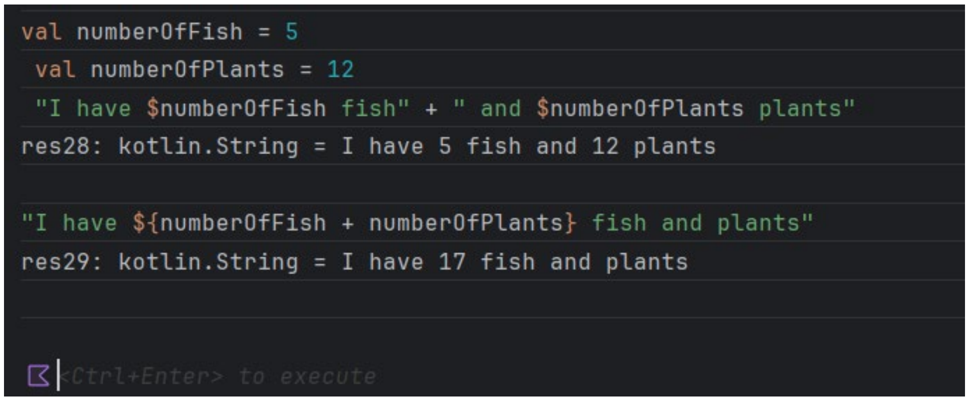
```
val i4: Int = b2.toInt() // OK!
 println(i4)
1

val i5: String = b2.toString()
 println(i5)
1

val i6: Double = b2.toDouble()
 println(i6)
1.0

val oneMillion = 1_000_000
 val socialSecurityNumber = 999_99_9999L
 val hexBytes = 0xFF_EC_DE_5E
 val bytes = 0b11010010_01101001_10010100_10010010

<Ctrl+Enter> to execute
```

c. *Step 3: Learn the value of variable types*



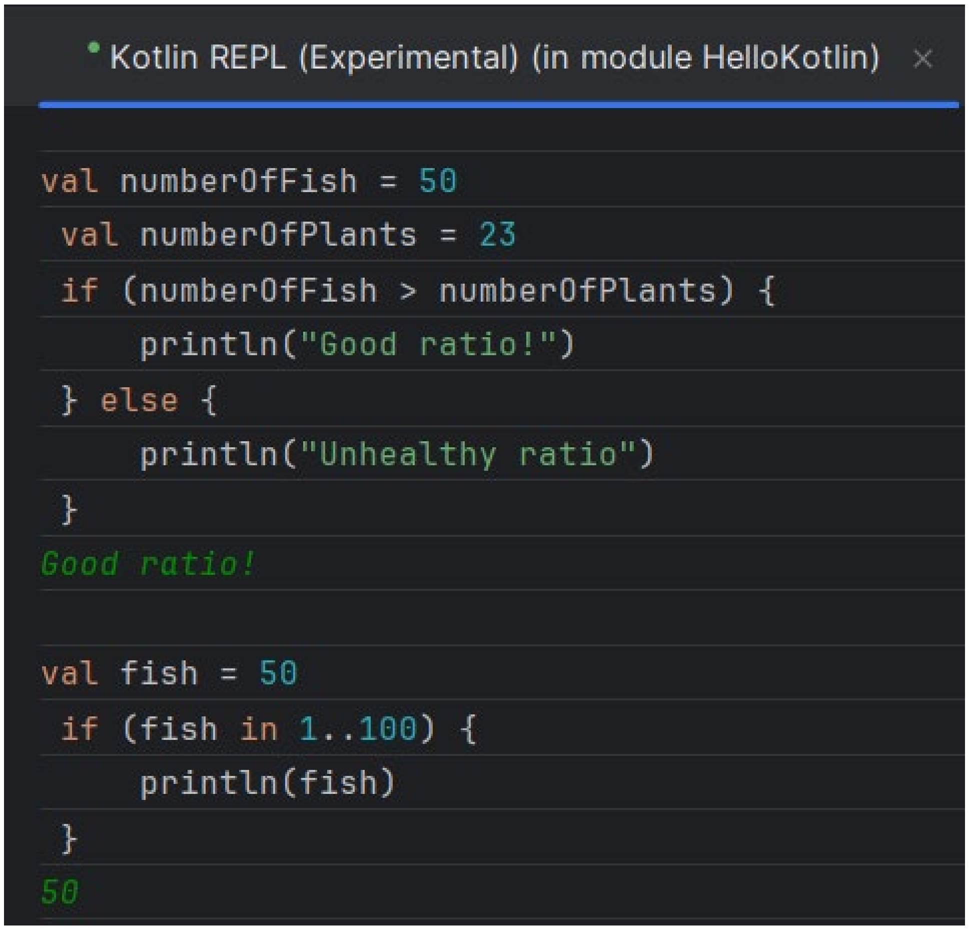
```
var fish = 1
 fish = 2
 val aquarium = 1
 aquarium = 2
error: val cannot be reassigned
aquarium = 2
^

var fish: Int = 12
 var lakes: Double = 2.5
```

d. *Step 4: Learn about strings and characters*

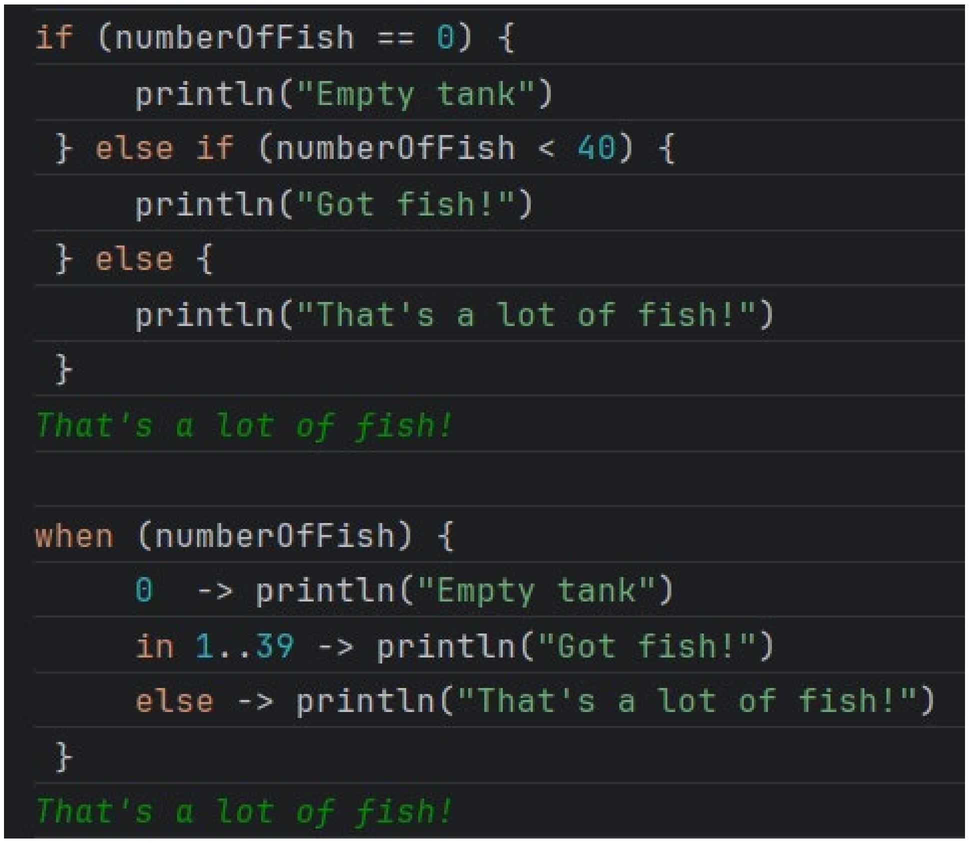


```
val numberOfFish = 5
 val numberOfPlants = 12
 "I have $numberOfFish fish" + " and $numberOfPlants plants"
res28: kotlin.String = I have 5 fish and 12 plants

"I have ${numberOfFish + numberOfPlants} fish and plants"
res29: kotlin.String = I have 17 fish and plants
```

2. **Compare conditions and booleans**

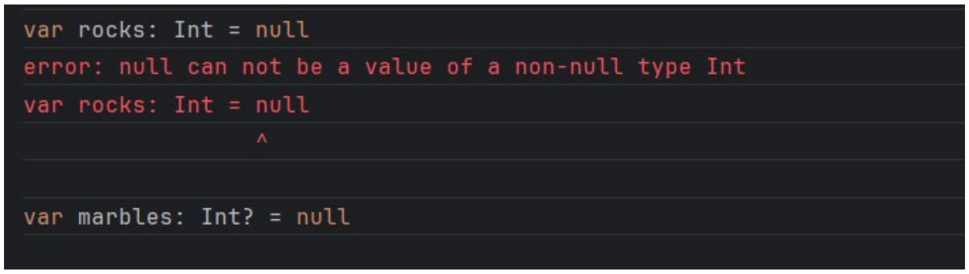


```
val numberOfFish = 50
 val numberOfPlants = 23
 if (numberOfFish > numberOfPlants) {
     println("Good ratio!")
 } else {
     println("Unhealthy ratio")
 }
Good ratio!

val fish = 50
 if (fish in 1..100) {
     println(fish)
 }
50
```



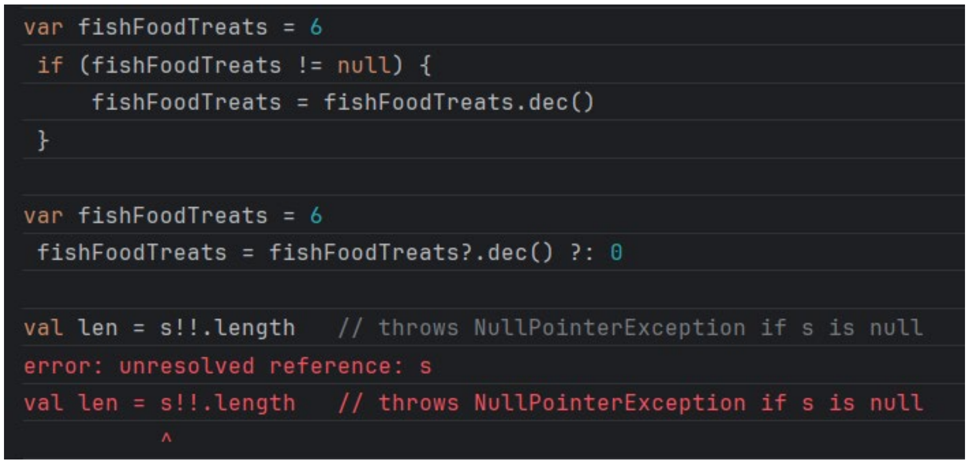
```
if (numberOfFish == 0) {
     println("Empty tank")
 } else if (numberOfFish < 40) {
     println("Got fish!")
 } else {
     println("That's a lot of fish!")
 }
That's a lot of fish!

when (numberOfFish) {
     0  -> println("Empty tank")
     in 1..39 -> println("Got fish!")
     else -> println("That's a lot of fish!")
 }
That's a lot of fish!
```

3. **Learn about nullability**
   a. *Step 1: Learn about nullability*



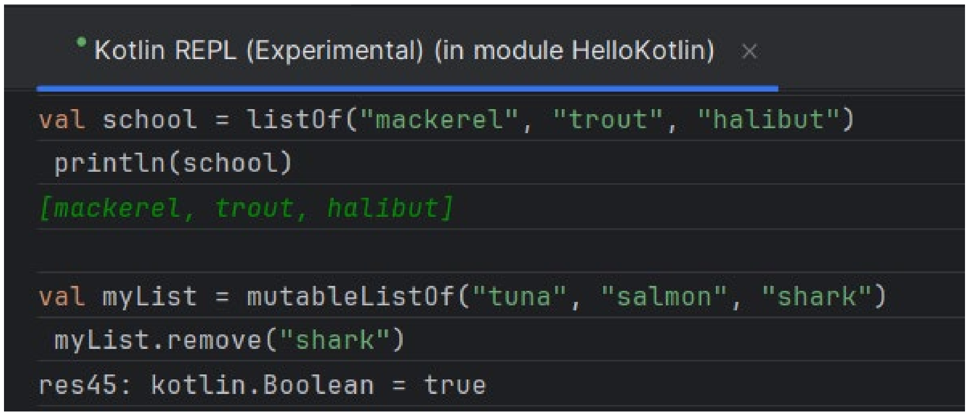
```
var rocks: Int = null
error: null can not be a value of a non-null type Int
var rocks: Int = null
                 ^

var marbles: Int? = null
```

   b. *Step 2: Learn about the ? and ?: operators*



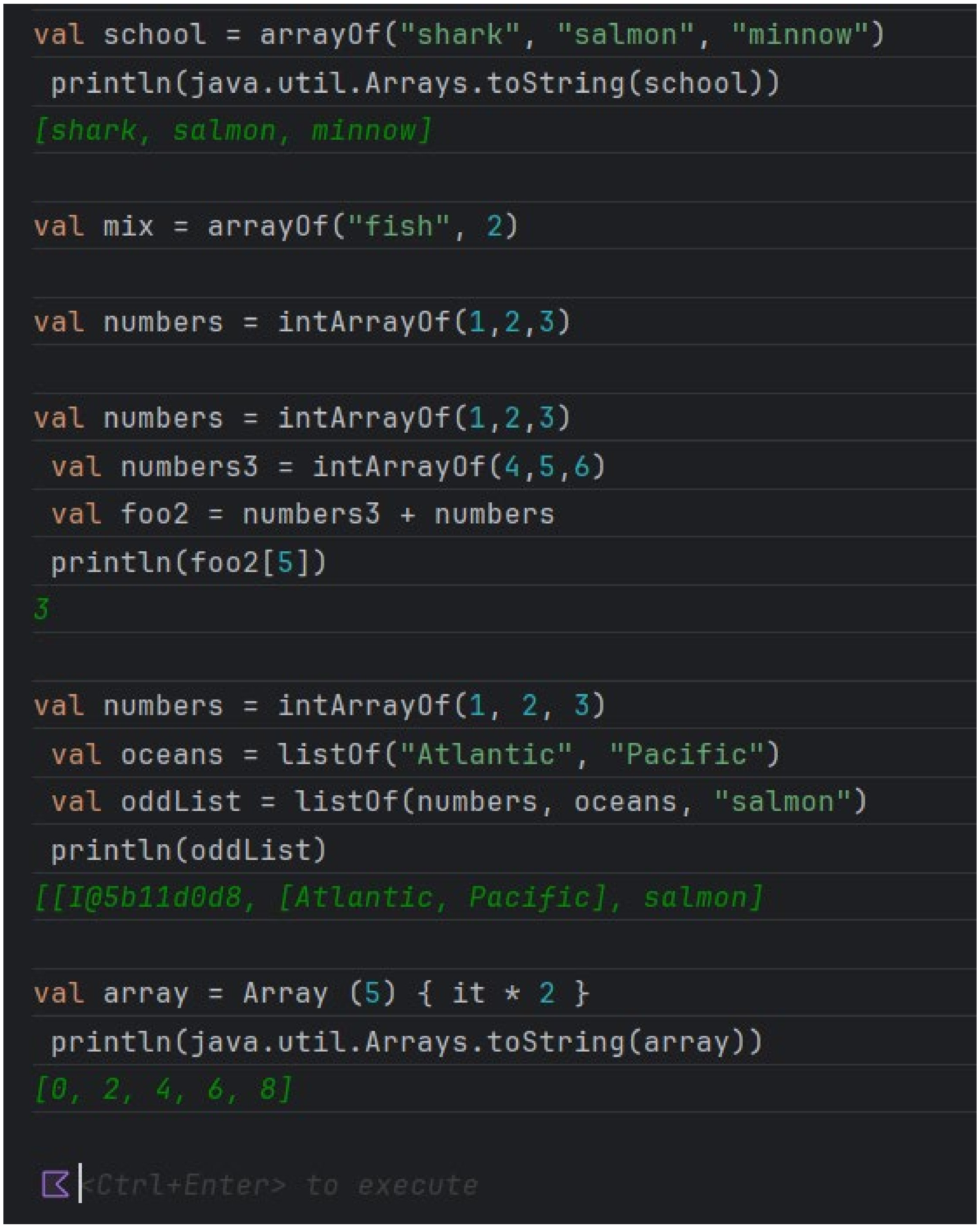
```
var fishFoodTreats = 6
 if (fishFoodTreats != null) {
     fishFoodTreats = fishFoodTreats.dec()
 }

var fishFoodTreats = 6
 fishFoodTreats = fishFoodTreats?.dec() ?: 0

val len = s!!.length   // throws NullPointerException if s is null
error: unresolved reference: s
val len = s!!.length   // throws NullPointerException if s is null
          ^
```

4. **Explore arrays, lists, and loops**
   a. *Step 1: Make lists*



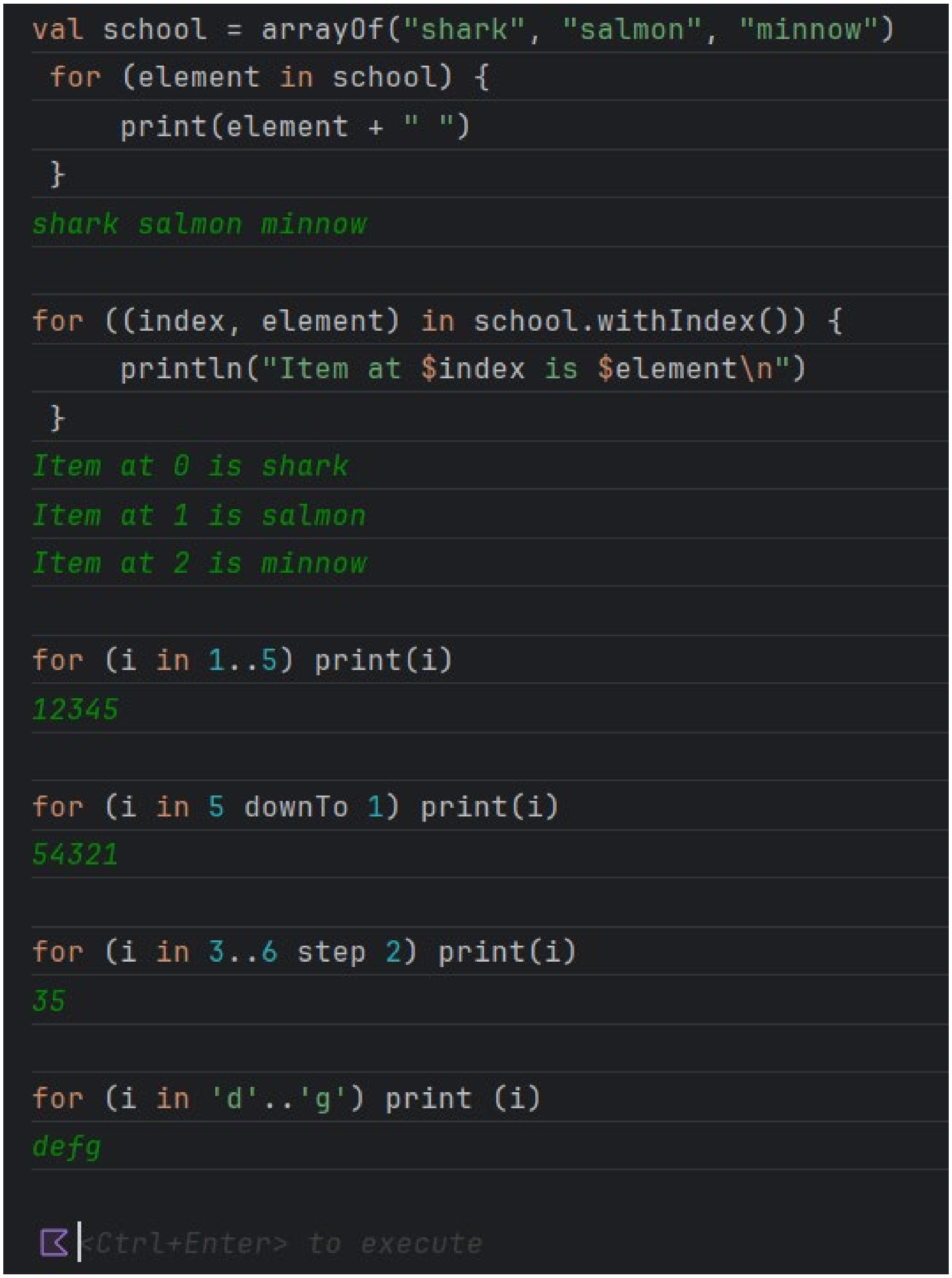
```
val school = listOf("mackerel", "trout", "halibut")
 println(school)
[mackerel, trout, halibut]

val myList = mutableListOf("tuna", "salmon", "shark")
 myList.remove("shark")
res45: kotlin.Boolean = true
```

b. *Step 2: Create arrays*

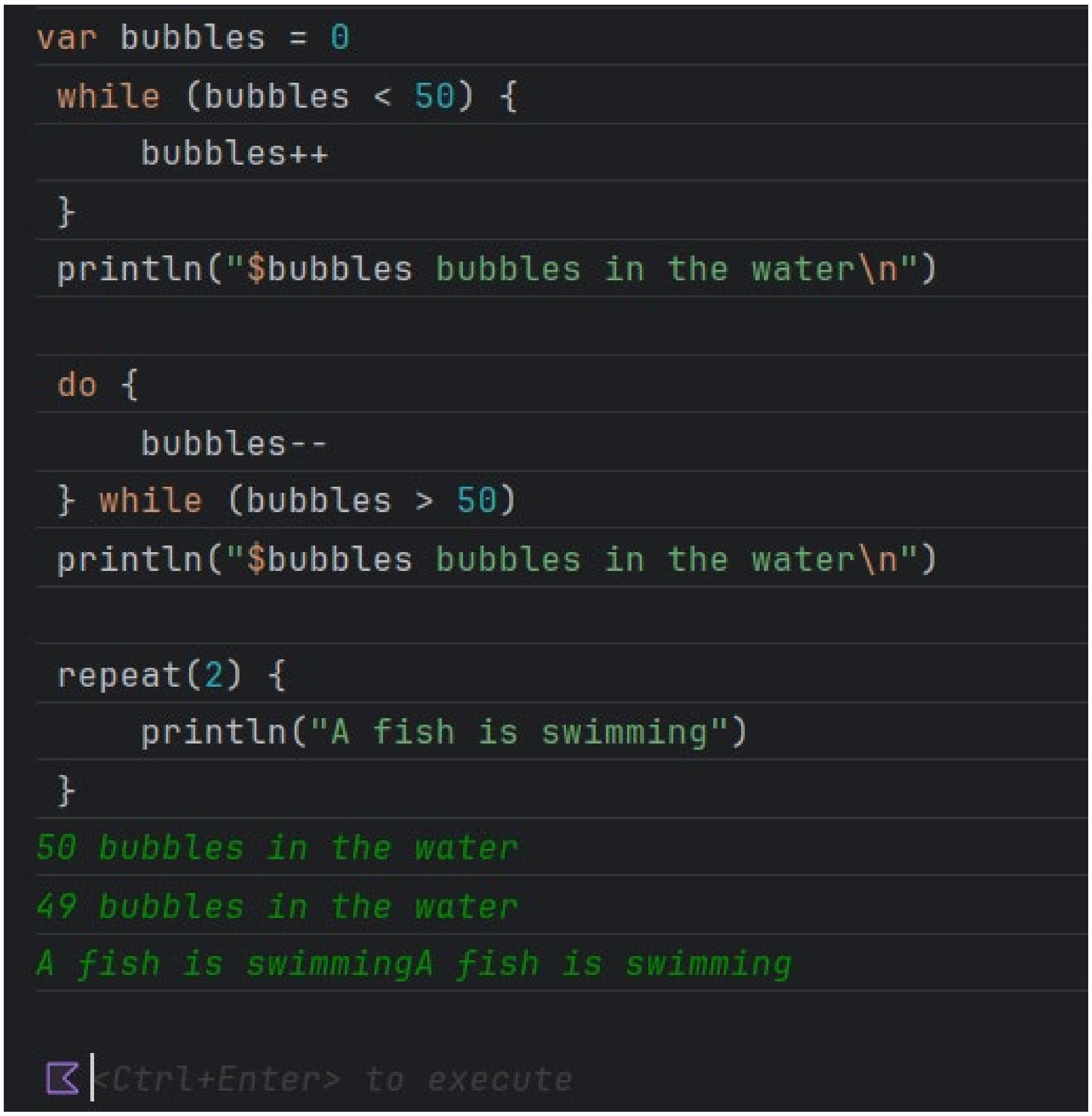


```
val school = arrayOf("shark", "salmon", "minnow")
 println(java.util.Arrays.toString(school))
[shark, salmon, minnow]

val mix = arrayOf("fish", 2)

val numbers = intArrayOf(1,2,3)

val numbers = intArrayOf(1,2,3)
 val numbers3 = intArrayOf(4,5,6)
 val foo2 = numbers3 + numbers
 println(foo2[5])
3

val numbers = intArrayOf(1, 2, 3)
 val oceans = listOf("Atlantic", "Pacific")
 val oddList = listOf(numbers, oceans, "salmon")
 println(oddList)
[[I@5b11d0d8, [Atlantic, Pacific], salmon]

val array = Array (5) { it * 2 }
 println(java.util.Arrays.toString(array))
[0, 2, 4, 6, 8]

<Ctrl+Enter> to execute
```

c. *Step 3: Create loops*



```
val school = arrayOf("shark", "salmon", "minnow")
 for (element in school) {
     print(element + " ")
 }
shark salmon minnow

for ((index, element) in school.withIndex()) {
     println("Item at $index is $element\n")
 }
Item at 0 is shark
Item at 1 is salmon
Item at 2 is minnow

for (i in 1..5) print(i)
12345

for (i in 5 downTo 1) print(i)
54321

for (i in 3..6 step 2) print(i)
35

for (i in 'd'..'g') print (i)
defg

<Ctrl+Enter> to execute
```



```
var bubbles = 0
 while (bubbles < 50) {
     bubbles++
 }
 println("$bubbles bubbles in the water\n")

 do {
     bubbles--
 } while (bubbles > 50)
 println("$bubbles bubbles in the water\n")

 repeat(2) {
     println("A fish is swimming")
 }
50 bubbles in the water
49 bubbles in the water
A fish is swimmingA fish is swimming
```