Welcome to Kotlin version 1.9.23-release-779 (JRE 21.0.2+13-LTS)

Type :help for help, :quit for quit

fun printHello() {

println("Hello World")

}

printHello()

Hello World

1+1

res1: kotlin.Int = 2

53-3

res2: kotlin.Int = 50

90/10

res3: kotlin.Int = 9

1.2/2.3

res4: kotlin.Double = 0.5217391304347826

3\*5.5

res5: kotlin.Double = 16.5

6.3\*5.6

res6: kotlin.Double = 35.279999999999994

2.times(3)

res8: kotlin.Int = 6

9.plus(5)

res9: kotlin.Int = 14

4.div(5)

res10: kotlin.Int = 0

val i: int = 6

val i = 7

val b1 = i.toByte()

println(b1)

7

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

println(b2)

⇒ 1

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

error: expecting an element

? 1

^

error: expecting an element

? 1

^

error: expecting an element

? 1.0

^

val b1 = i.toByte()

println(b1)

7

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

println(b2)

1

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

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

val numberOfFish = 5

val numberOfPlants = 12

"I have $numberOfFish fish" + " and $numberOfPlants plants"

res26: kotlin.String = I have 5 fish and 12 plants

"I have ${numberOfFish + numberOfPlants} fish and plants"

res27: kotlin.String = I have 17 fish and plants

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!

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

var fishFoodTreats = 6

if (fishFoodTreats != null) {

fishFoodTreats = fishFoodTreats.dec()

}

var fishFoodTreats = 6

fishFoodTreats = fishFoodTreats?.dec()

error: type mismatch: inferred type is Int? but Int was expected

fishFoodTreats = fishFoodTreats?.dec()

^

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

^

val school = listOf("mackerel", "trout", "halibut")

println(school)

[mackerel, trout, halibut]

val myList = mutableListOf("tuna", "salmon", "shark")

myList.remove("shark")

res39: kotlin.Boolean = true

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@758ac46, [Atlantic, Pacific], salmon]

val array = Array (5) { it \* 2 }

println(java.util.Arrays.toString(array))

[0, 2, 4, 6, 8]

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

incomplete code

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

defg

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