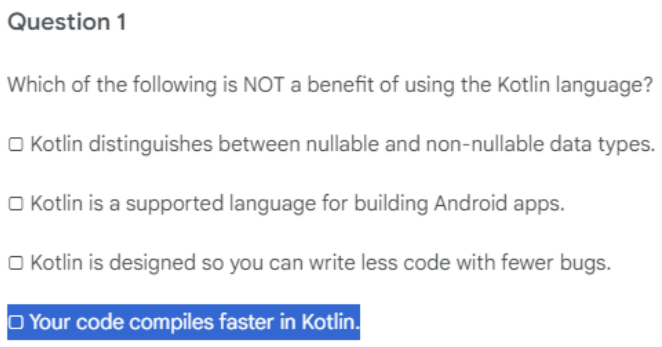
Kotlin bootcamp for programmers

1. Get started.



Question 1

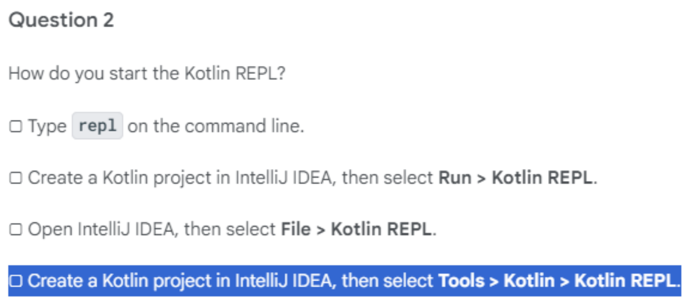
Which of the following is NOT a benefit of using the Kotlin language?

▢ Kotlin distinguishes between nullable and non-nullable data types.

▢ Kotlin is a supported language for building Android apps.

▢ Kotlin is designed so you can write less code with fewer bugs.

[✓ ▢ Your code compiles faster in Kotlin.]



Question 2

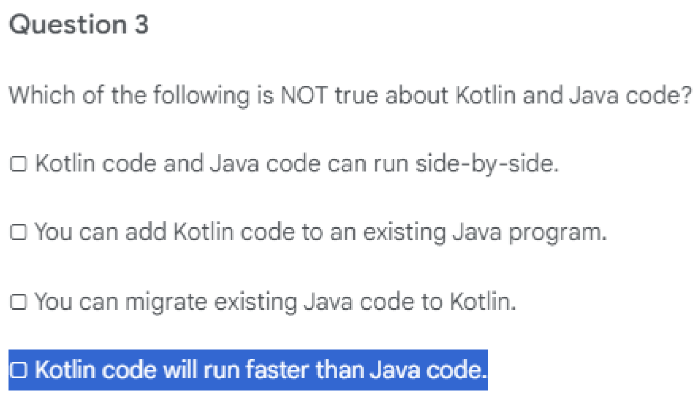
How do you start the Kotlin REPL?

▢ Type `repl` on the command line.

▢ Create a Kotlin project in IntelliJ IDEA, then select **Run > Kotlin REPL**.

▢ Open IntelliJ IDEA, then select **File > Kotlin REPL**.

[✓ ▢ Create a Kotlin project in IntelliJ IDEA, then select **Tools > Kotlin > Kotlin REPL**.]



Question 3

Which of the following is NOT true about Kotlin and Java code?

▢ Kotlin code and Java code can run side-by-side.

▢ You can add Kotlin code to an existing Java program.

▢ You can migrate existing Java code to Kotlin.

[✓ ▢ Kotlin code will run faster than Java code.]

2. Kotlin basics

Question 1

Which of the following declares an unchangeable list of strings?

▢ `val school = arrayOf("shark", "salmon", "minnow")`

▢ `var school = arrayOf("shark", "salmon", "minnow")`

[✓ ▢ `val school = listOf("shark", "salmon", "minnow")`]

▢ `val school = mutableListOf("shark", "salmon", "minnow")`

Question 2

What will be the output of the following code? `for (i in 3..8 step 2) print(i)`

▢ 345678

▢ 468

▢ 38

[✓ ▢ 357]

Question 3

What is the purpose of the question mark in this code? `var rocks: Int? = 3`

▢ The type of the variable `rocks` isn't fixed.

[✓ ▢ The variable `rocks` can be set to null.]

▢ The variable `rocks` cannot be set to null.

▢ The variable `rocks` shouldn't be initialized right away.

3. functions

Question 1

The `contains(element: String)` function returns `true` if the string `element` is contained in the string it's called on. What will be the output of the following code?

```
val decorations = listOf ("rock", "pagoda", "plastic plant", "alligator", "flowerpot")
```

```
println(decorations.filter {it.contains('p')})
```

☐ `[pagoda, plastic, plant]`

☐ `[pagoda, plastic plant]`

[✓ `[pagoda, plastic plant, flowerpot]`]

☐ `[rock, alligator]`

Question 2

In the following function definition, which one of the parameters is required? `fun shouldChangeWater (day: String, temperature: Int = 22, dirty: Int = 20, numDecorations: Int = 0): Boolean {...}`

☐ `numDecorations`

☐ `dirty`

[✓ `day`]

☐ `temperature`

Question 3

You can pass a regular named function (not the result of calling it) to another function. How would you pass `increaseDirty( start: Int ) = start + 1` to `updateDirty(dirty: Int, operation: (Int) -> Int)`?

☐ `updateDirty(15, &increaseDirty())`

☐ `updateDirty(15, increaseDirty())`

☐ `updateDirty(15, ("increaseDirty()"))`

[✓ `updateDirty(15, ::increaseDirty)`]

4. Object-oriented programming

Question 1

Classes have a special method that serves as a blueprint for creating objects of that class. What is the method called?

☐ A builder

☐ An instantiator

[✓ ☐ A constructor]

☐ A blueprint

Question 2

Which of the following statements about interfaces and abstract classes is NOT correct?

☐ Abstract classes can have constructors.

☐ Interfaces can't have constructors.

[✓ ☐ Interfaces and abstract classes can be instantiated directly.]

☐ Abstract properties must be implemented by subclasses of the abstract class.

Question 3

Which of the following is NOT a Kotlin visibility modifier for properties, methods, etc.?

☐ `internal`

[✓ ☐ `nosubclass`]

☐ `protected`

☐ `private`

Question 4

Consider this data class: `data class Fish(val name: String, val species:String, val colors:String)`
Which of the following is NOT valid code to create and destructure a `Fish` object?

☐ `val (name1, species1, colors1) = Fish("Pat", "Plecostomus", "gold")`

☐ `val (name2, _, colors2) = Fish("Bitey", "shark", "gray")`

☐ `val (name3, species3, _) = Fish("Amy", "angelfish", "blue and black stripes")`

[✓ ☐ `val (name4, species4, colors4) = Fish("Harry", "halibut")`]

Question 5

Let's say you own a zoo with lots of animals that all need to be taken care of. Which of the following would NOT be part of implementing caretaking?

☐ An `interface` for different types of foods animals eat.

☐ An `abstract Caretaker` class from which you can create different types of caretakers.

☐ An `interface` for giving clean water to an animal.

[✓ ☐ A `data` class for an entry in a feeding schedule.]

5.1. Extensions

Question 1

Which one of the following returns a copy of a list?

☐ `add()`

☐ `remove()`

[✓ ☐ `reversed()`]

☐ `contains()`

Question 2

Which one of these extension functions on `class AquariumPlant(val color: String, val size: Int, private val cost: Double, val leafy: Boolean)` will give a compiler error?

☐ `fun AquariumPlant.isRed() = color == "red"`

☐ `fun AquariumPlant.isBig() = size > 45`

[✓ ☐ `fun AquariumPlant.isExpensive() = cost > 10.00`]

☐ `fun AquariumPlant.isNotLeafy() = leafy == false`

Question 3

Which one of the following is not a place where you can define constants with `const val`?

☐ at the top level of a file

[✓ ☐ in regular classes]

☐ in singleton objects

☐ in companion objects

5.2. Generics.

Question 1

Which of the following is the convention for naming a generic type?

▢ `<Gen>`

▢ `<Generic>`

[✓ ▢ `<T>`]

▢ `<X>`

Question 2

A restriction on the types allowed for a generic type is called:

▢ a generic restriction

[✓ ▢ a generic constraint]

▢ disambiguation

▢ a generic type limit

Question 3

Reified means:

▢ The real execution impact of an object has been calculated.

▢ A restricted entry index has been set on the class.

[✓ ▢ The generic type parameter has been made into a real type.]

▢ A remote error indicator has been triggered.

6. Functional manipulation

Question 1

In Kotlin, SAM stands for:

☐ Safe Argument Matching

☐ Simple Access Method

[✓ ☐ Single Abstract Method]

☐ Strategic Access Methodology

Question 2

Which one of the following is not a Kotlin Standard Library extension function?

[✓ ☐ `elvis()`]

☐ `apply()`

☐ `run()`

☐ `with()`

Question 3

Which one of the following is not true of lambdas in Kotlin?

☐ Lambdas are anonymous functions.

☐ Lambdas are objects unless inlined.

[✓ ☐ Lambdas are resource intensive and shouldn't be used.]

☐ Lambdas can be passed to other functions.

Question 4

Labels in Kotlin are indicated with an identifier followed by:

☐ `:`

☐ `::`

[✓ ☐ `@:`]

☐ `@`