1. **Variables:** Kotlin uses `val` for immutable (read-only) variables and `var` for mutable variables. You can skip type declarations as Kotlin infers types automatically.

   ```
   val readOnly = 5
   var changeable = 10
   changeable = 15
   ```

2. **Classes and Objects:** In Kotlin, classes define the structure and behavior of objects. You instantiate objects using class constructors. Properties can be defined directly in the primary constructor.

   ```
   class Person(val name: String, var age: Int) {
   fun introduce() = println("Hi, I'm $name and I'm $age years old")
   }
   val michael = Person("Michael", 23)
   michael.introduce()
   ```

3. **Data Types:** Kotlin supports various data types such as Int, Long, Float, Double, Boolean, Char, and String. It also allows nullable types using ?.

   ```
   val number: Int = 42
   val optionalText: String? = null
   ```

4. **Functions:** Functions in Kotlin are declared with the `fun` keyword. They can have parameters and return values. Single-expression functions can be shortened using the = sign.

   ```
   fun sayHello(name: String): String { return "Hello, $name!" }
   fun sayHelloCompact(name: String) = "Hello, $name!"
   ```

5. **Control Flow:** Kotlin provides `if-else` statements and `when` expressions for control flow. It also supports `for`, `while`, and `do-while` loops for iteration.

   ```
   val comparison = if (10 > 5) "Greater" else "Smaller"
   when (comparison) {
   "Greater" -> println("10 is greater than 5")
   else -> println("Unexpected value")
   }
   ```

6. **Inheritance:** By default, Kotlin classes cannot be inherited. Use the `open` keyword to allow inheritance. Subclasses are created by extending the base class.

```
open class Animal(val name: String)
class Dog(name: String) : Animal(name) {
fun Bow() = println("$name says: Bow!")
}
```

7. **Ranges:** Ranges represent a sequence of values. They can be created with the `..` operator or `rangeTo()` function. Ranges can be used with numbers and characters.

   Numeric Ranges:
   ```
   for (i in 1..5) print(i)
   println((5 downTo 1).toList())
   ```

   Char Ranges:
   ```
   for (char in 'a'..'e') print(char)
   println(('D' downTo 'A').toList())
   ```

8. **Null Safety:** Kotlin's type system differentiates between nullable and non-nullable types to avoid null pointer exceptions. Use `?.` for safe calls and `?:` for default values.

```
val nullableString: String? = null
val length = nullableString?.length ?: 0
println("Length is: $length")
```