1.Hello World

```
fun main(){
    println(“Hello World”)
}
```

The main() function is the starting off point in the kotlin program and must be included in the code before execution.

The println() statement will print hello world on new line while print() statement print on same line.

2. Comments

```
//Single line comments

/*
This
Is
Multiple
Line
Comments
*/
```

3. Order of Execution

Code is read, compiled, and executed in top down order

4. Variables and Constants

Constant is immutable variable that cannot be reassigned in the program

Example:

```
val pi: Double = 3.14
```

Variable is mutable that can be reassigned in the program

Example:

```
var count: Int = 10
```

5. Data Types

Kotlin has a rich set of data types including ‘Int’, ‘Double’, ‘Float’, ‘Long’, ‘Char’, ‘String’, ‘Boolean’,etc

6. Type Inference Language

Kotlin can infer type based on the value assigned to the variable in short does not have to mention data type while assigning value.

Example:

```
val name = "Sritika" // type inferred as String
```

7. Control Flow
if-else statement Example:

```
val age=21
if(age>=18){
    println(“You are an adult.”)
else{
    println(“You are a mirror.”)
}
```

when statement (similar to switch statement in other languages) example:

```
val day=3
when(day) {
    1 -> println(“Monday”)
    2 -> println(“Tuesday”)
    3 -> println(“Wednesday”)
    4 -> println(“Thursday”)
    else -> println(“Other day”)
}
```

8. Loops
for loop example:

```
for (i in 1..5){
    println(i)
}
```

Output:
1
2
3
4
5
for loop using “step” example:

```
for(i in 1..10 step 2){
    println(i)
}
```

Output:
1
3

5
7
9

for loop using “downTo” example:

```
for (i in 5 downTo 1){
    println(i)
}
```

while loop example:

```
var i=0
while(i<5){
    println(i)
    i++
}
```

9. Functions

```
fun addNumbers(a:Int, b:Int): Int{
    return a+b
}

//can be also written as

fun addNumbers(a:Int, b:Int):Int=a+b
```

10. String Interpolation

```
val name= “Sritika”
val greeting= “Hello, $name!”
```

11. Lists

```
val numbers=listOf(1,2,3,4,5)
```

12. Ranges

```
val range= 1..5 // Represents the range from 1 to 5
```

13. Class and Objects

```
class Person(val name:String, var age:Int)
```

```
val person =Person(“Sritika”,20)
println(person.name)
person.age=21
println(person.age)
```

14. Lambda Expression

```
val sum={a:Int, b:Int -> a+b}
println(sum(2,3)) //Output 5
```

15. Companion Object

In Kotlin, a Companion Object is a special object within a class that allows you to define static members. These members belong to the class itself rather than to any specific instance of the class. It is similar to static members in other programming languages.

```
class MyClass {
    companion object {
        const val PI = 3.14
        fun staticFunction() {
            println("Static function called.")
        }
    }
}
```

16. Nullable Types

Kotlin encourages avoiding null references. For nullable types, you use the `?` symbol:

```
val nullableValue: String? = null
```

17. Null Safety

Kotlin provides safe calls and the Elvis operator to handle null values:

```
val length: Int? = nullableValue?.length // Safe call
val name: String = nullableValue ?: "Default" // Elvis operator
```