FUNCTIONS AND OBJECTS

Functions

Definition: Blocks of code designed to perform specific tasks, reusable throughout the code.

Syntax:

```
kotlinCopy code
fun functionName(parameter1: Type,
parameter2: Type): Retu
rnType {
// Code to execute
return value
}
```

Naming Conventions:

Camel Case: fun myFunction()

Descriptive: Use verbs & avoid abbreviations.

Examples:

No Parameters:

```
kotlinCopy code
fun displayMessage() {
println("Hello, Kotlin Learner!")
}
displayMessage() // Output: Hello,
Kotlin Learner!
```

With Parameters:

```
kotlinCopy code
fun greet(name: String) {
println("Hello, $name!")
}
greet("John") // Output: Hello, John!
```

With Return Value:

```
kotlinCopy code
fun addNumbers(a: Int, b: Int): Int {
return a + b
}
val sum = addNumbers(5, 3)
println(sum) // Output: 8
```

Print vs. Return:

Print: Displays output to console, does not affect function flow.

```
kotlinCopy code
fun showMessage() {
println("Hello, Kotlin Learner!")
}
showMessage() // Output: Hello, Kotlin
Learner!
```

Return: Passes a value back to the caller, exits the function.

```
kotlinCopy code
fun add(a: Int, b: Int): Int {
return a + b
}
val result = add(2, 3)
println(result) // Output: 5
```

tutorials.EU

Classes and Objects

Definition: Templates for creating objects with properties and methods.

Syntax:

```
kotlinCopy code
class ClassName {
// class body
}
```

Property vs. Parameter:

- **Parameter:** Values passed to the constructor.
- **Property:** Variables in the class storing passed or default values.

Key Concepts:

- **Constructor:** Initializes properties when an object is created.
- **Properties:** Variables that hold data within a class.
- **Initializers:** Code blocks that run when an object is instantiated.
- **Objects/Instances:** Individual instances created from a class.

Examples:

Basic Class:

```
kotlinCopy code
class Person(val name: String, val
age: Int)
val person = Person("Alice", 25)
```

Initializer Block:

```
kotlinCopy code
class Person(name: String, age: Int) {
val name = name
val age = age
init {
println("Person named $name is created.")
}
}
val person = Person("Alice", 25) // Output: Person name
d Alice is created.
```

Default Values:

```
kotlinCopy code
class Person(val name: String =
"John", val age: Int =
30)
val person = Person() // Uses default
values: John, 30
```

Data Classes

Definition: Classes primarily used for holding data.

Syntax:

```
kotlinCopy code
data class Person(val name: String, val age: Int)
```

- **Immutability:** Encourages use of immutable properties.
- **Standard Methods:** Automatically provides toString() , equals() , hashCode() .
- **Destructuring Declarations:** Decompose the data class into its properties.

```
kotlinCopy code
val person = Person("Alice", 25)
val (name, age) = person
println(name) // Output: Alice
println(age) // Output: 25
```