20CYS312 - Principles of Programming Languages
Exploring Programming Paradigms

Assignment-01

Presented by Penugonda V S Ganasekhar
CB.EN.U4CYS21055
TIFAC-CORE in Cyber Security
Amrita Vishwa Vidyapeetham, Coimbatore Campus

Feb 2024

AMRITA
VISHWA VIDYAPEETHAM

Outline

Scripting

- Scripting refers to the practice of writing scripts, which are typically interpreted or executed by a runtime environment rather than compiled into machine code.
- Scripting has become an integral part of modern software development, providing a flexible and dynamic approach to programming. It enables developers to write code that is interpreted at runtime rather than being compiled into machine code, allowing for rapid prototyping, customization, and automation.
- Scripting involves writing sequences of commands or instructions that are interpreted or executed by a runtime environment. Unlike compiled languages, scripting languages allow for quick development cycles and on-the-fly modifications.
- Scripting is widely used for automation, configuration, and extending the functionality of existing software.

Scripting (cont.)

- Some Programming Languages which support Scripting are,
  1. JavaScript
  2. Python
  3. Lua
  4. Ruby
  5. Perl

  and many more..

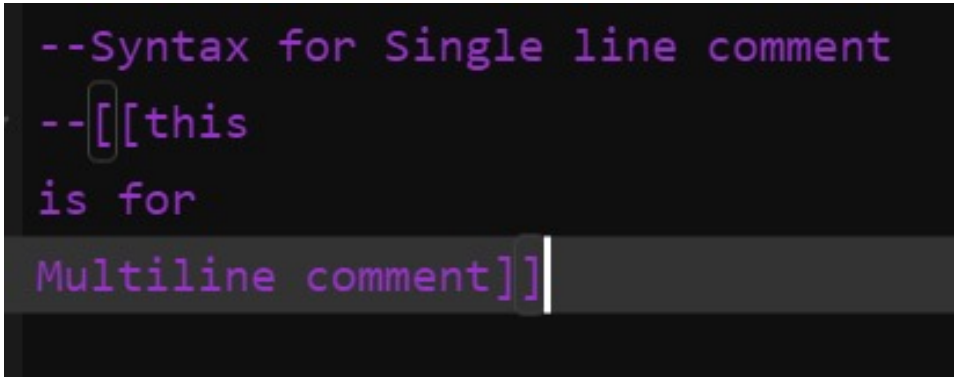
Lua

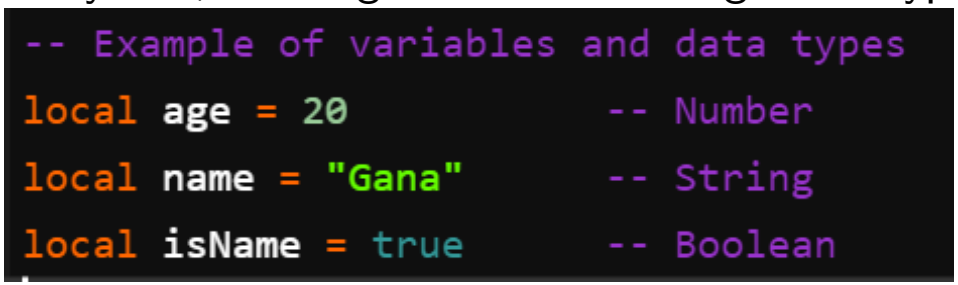
- Lua is often used as a scripting language in various applications and is known for its simplicity, efficiency, and ease of integration.
- **Lua**, initially developed in the early 1990s, has evolved into a powerful scripting language. Known for its lightweight design, Lua is often embedded in larger applications to provide users with the ability to customize and extend functionalities, and it is used for general-purpose programming.
- Lua is commonly used in game development, scripting, and automation tasks. It has a clean and simple syntax that makes it accessible for beginners while remaining powerful enough for more experienced developers.

Lua (cont.)

- Lua supports multiple programming paradigms, making it a versatile language that can be used in various contexts. Some of them are,
  1. Imperative (Procedural) Programming
  2. Functional Programming
  3. Object-Oriented Programming (OOP)
  4. Scripting
  5. Event-Driven Programming
  6. Data-Driven Programming

Basic Syntax and Structure

- **Comments:**
  - For a Single line comment, we have to use '–' as the syntax.
  - For a Multiple line comment, we have to use '–[[]]' as the syntax.
  - 

- **Data Types and Variables:**
  - Lua supports fundamental data types, including numbers, strings, booleans, tables, and functions.
  - Variables are used to store values, and Lua employs dynamic typing just like how it is in Python, allowing variables to change their type during runtime.
  - 

Basic Syntax and Structure (cont.)

- **Control Structures:**
  - Lua provides standard control structures for branching and looping. Conditionals like if, else, and elseif are used for decision-making, and loops such as for and while enable iteration.

```lua
-- Example of control structures
local x = 10

if x > 0 then
    print("Positive number")
elseif x < 0 then
    print("Negative number")
else
    print("Zero")
end

-- Loop example
for i = 1, 5 do
    print(i)
end
```

Comparison with Other Scripting Languages:

- **Advantages:**
  - Lua stands out for its minimalistic design, which allows for easy integration and efficient execution.
  - Unlike some scripting languages that might carry a larger runtime overhead, Lua's small and efficient interpreter makes it suitable for resource-constrained environments.
- **Disadvantages:**
  - Compared to languages like Python or JavaScript, Lua may lack some built-in libraries and frameworks, but its simplicity, speed, and ease of integration make it an attractive choice, particularly for scenarios where a lightweight scripting language is preferable.

Object-Oriented

- Object-Oriented Programming (OOP) is a programming paradigm that organizes code into objects, which are instances of classes.
- In the dynamic landscape of software development, Object-Oriented Programming (OOP) has emerged as a fundamental paradigm that facilitates the organization and structuring of code.
- OOP focuses on modeling software systems as a collection of interacting objects, each encapsulating data and behavior.

Object-Oriented (cont.)

- Some of the Famous Object-Oriented Programming Languages are,
  1. Java
  2. C++
  3. Javascript
  4. Swift
  5. PHP
- **Fundamental Principles of Object-Oriented**
  1. Classes and Objects
  2. Encapsulation
  3. Inheritance
  4. Polymorphism
  5. Abstraction

Kotlin

- Kotlin, initially announced by JetBrains in 2011, was developed as a statically-typed programming language for the Java Virtual Machine (JVM). JetBrains, known for their integrated development environments (IDEs), sought to create a language that addressed certain limitations of Java while maintaining interoperability with it.
- In 2012, JetBrains open-sourced Kotlin under the Apache 2 license, encouraging community collaboration and adoption. This move accelerated Kotlin's growth, as developers outside of JetBrains began contributing to its development.
- The Kotlin community actively contributes to the language's development through discussions, forums, and open-source projects. The collaborative nature of the community enhances the language's ecosystem and encourages the sharing of best practices.
- Its concise syntax, null safety, and extensive standard library make it an attractive choice for a wide range of applications, including Android development, server-side programming, and more.

Kotlin (cont.)

- **Key Features of Kotlin,**
  1. Conciseness and Readability
  2. Null Safety
  3. Interoperability with Java
  4. Extension Functions
- **Classes and Objects,**

```kotlin
class Car {
    // Properties
    var brand: String = ""
    var model: String = ""

    // Method
    fun startEngine() {
        println("Engine started!")
    }
}
```

```kotlin
val myCar = Car()
myCar.brand = "Toyota"
myCar.model = "Camry"
myCar.startEngine()
```

Kotlin (cont.)

- **Encapsulation,**

```
class BankAccount {
    private var balance: Double = 0.0

    fun deposit(amount: Double) {
        // Logic for deposit
        balance += amount
    }

    fun withdraw(amount: Double) {
        // Logic for withdrawal
        if (amount <= balance) {
            balance -= amount
        } else {
            println("Insufficient funds.")
        }
    }
}
```

Kotlin (cont.)

- **Inheritance,**

```
open class Animal(val name: String)

class Dog(name: String, val breed: String) : Animal(name)
```

- **Abstraction,**

```
abstract class Shape {
    abstract fun draw()

    fun resize() {
        println("Resizing the shape")
    }
}
```

- **Interfaces,**

```
class Car : Drivable {
    override fun start() {
        println("Car starting")
    }

    override fun accelerate() {
        println("Car accelerating")
    }

    override fun brake() {
        println("Car braking")
    }
}
```

Comparison

Similarities between Scripting and Object-oriented programming,

- Both paradigms are used to develop software systems.
- Both paradigms use variables to store data.
- Both paradigms use control structures such as loops and conditionals to control program flow.
- Both paradigms use functions to perform operations on data.

Differences between Scripting and Object-oriented programming,

- Scripting languages are interpreted, while object-oriented programming languages are compiled.
- Scripting languages are dynamically typed, while object-oriented programming languages are statically typed.
- Scripting languages are often used for small and simple tasks, while object-oriented programming languages are often used for large and complex software systems.
- Scripting languages are often used for web development, system administration, and scientific computing, while object-oriented programming languages are often used for developing desktop applications, mobile applications, and video games.

Comparison (cont.)

Similarities between Lua and Kotlin,

- Both languages are open source and free to use.
- Both languages are highly portable and can be run on a wide range of platforms and operating systems.
- Both languages support functional programming and object-oriented programming paradigms.
- Both languages have a growing community of developers, which means that there are many resources and libraries available for them.

Differences between Lua and Kotlin,

- Lua is a scripting language, while Kotlin is a compiled language.
- Lua is dynamically typed, while Kotlin is statically typed.
- Lua is often used for video games, game engines, and network and system programs, while Kotlin is often used for Android development, web development, and server-side development.
- Lua is often used as an embedded scripting language, while Kotlin is often used as a standalone programming language.

References

- https://www.lua.org/manual/5.4/manual.html
- https://www.geeksforgeeks.org/difference-between-python-and-lua-programming-language/
- https://www.geeksforgeeks.org/kotlin-extension-function/?ref=lbp
- https://kotlinlang.org/docs/js-overview.html
- https://www.w3schools.com/kotlin/index.php
- https://www.youtube.com/watch?v=1srFmjt1Ib0
- https://www.w3schools.com/kotlin/kotlin-oop.php
- https://betterprogramming.pub/object-oriented-programming-in-kotlin-1e8b9a95adbe
- https://www.geeksforgeeks.org/kotlin-class-and-objects/
- https://towardsdev.com/kotlin-oop-full-guide-762c18fd3b42