20CYS312 - Principles of Programming Languages
Exploring Programming Paradigms

Assignment-01

Presented by Gokulachselvan C D
CB.EN.U4CYS21019
TIFAC-CORE in Cyber Security
Amrita Vishwa Vidyapeetham, Coimbatore Campus

Feb 2024

AMRITA
VISHWA VIDYAPEETHAM

Outline

Programming paradigms

Programming paradigm:

- A programming paradigm is a fundamental style or approach to programming that provides a set of principles, concepts, and guidelines for designing and structuring code.
- They decide how tasks are expressed, organized, and executed in a programming language and it defines the overall flow and structure of a program.
- In simple words, programming paradigm is the classification of programming languages based on their features.
- There are various programming paradigms. They are broadly divided in two sub classes. They are:
  - **Imperative programming paradigm**
  - **Declarative programming paradigm**

Imperative programming paradigm

Imperative programming paradigm:

- It is one of the oldest programming paradigm. The main focus of this paradigm is based on how to achieve the goal. It works by changing the program state through assignment statements. It performs step by step task by changing state.
- In simple words, Imperative programming is an approach that focuses on describing a sequence of steps for the computer to perform to achieve a specific result. Here's a simple example in python, which is an imperative programming language:

```python
def factorial(n):
    result = 1
    for i in range(1, n + 1):
        result *= i
    return result

number = 5
result = factorial(number)
print(f"The factorial of {number} is {result}")
```

- This example shows the imperative nature of the code, it explicitly outlines the steps to achieve the desired outcome, by keep on changing the program's state through a series of instructions till it obtains a desired result.

Imperative programming paradigm (Contd.)

Types of Imperative programming paradigm:

- **Procedural programming:**
  - It involves organizing the code into procedures or routines that perform specific tasks. It focuses on step-by-step instructions for the computer to execute.
  - **Example:** C is a procedural programming language. In C, we write functions that contain a series of steps to execute, and the main program calls these functions as needed.
- **Object Oriented programming**
  - OOP organizes code into objects, which encapsulate data and behavior. It emphasizes concepts like encapsulation, inheritance, and polymorphism to model real-world entities.
  - **Example:** Java is an object-oriented programming language. In Java, you create classes representing objects, and objects interact through methods, allowing for a modular and reusable code structure.
- **Parallel processing**
  - It is the processing of program instructions by dividing them among multiple processors.
  - A parallel processing system possess many numbers of processor with the objective of running a program in less time by dividing them.
  - This approach seems to be like divide and conquer.
  - **Example:** NESL (Nested Data-Parallel Language).

Declarative programming paradigm

Declarative programming paradigm:

- Declarative programming is a programming paradigm in which the programmer defines what needs to be accomplished by the program without defining how it needs to be implemented.
- In simple words, the approach focuses on what needs to be achieved instead of instructing how to achieve it.
- **Example:** Consider a simple database with a table named employees that contains information about employees, such as employee_id, name, and salary.

```sql
-- Declarative Approach
SELECT employee_id, name, salary FROM employees;
```

- This query does not specify how the database system should perform the retrieval and it abstracts away the implementation details. The database system takes care of executing the query.

Declarative programming paradigm (Contd.)

Types of Declarative programming paradigm:

- **Functional:**
  - Functional programming focuses on expressing computation as the evaluation of mathematical functions.
  - **Example:** Haskell is a functional programming language. Functions in Haskell are pure, meaning they produce the same output for the same input without any side effects.
- **Logic:**
  - Logic programming represents computation as a series of logical relationships and constraints. It uses rules and facts to express knowledge and lets the system derive solutions.
  - **Example:** Prolog is a logic programming language. In Prolog, we define facts and rules, and the system uses logical inference to find solutions to queries based on these rules.
- **Data-Driven:**
  - It allows users to describe the desired result without specifying the step-by-step process. It's commonly used for querying and manipulating relational databases.
  - **Example:** SQL (Structured Query Language) is a data-driven language used for database operations.
  - A query like SELECT * FROM Employees WHERE Department = 'IT' does not specify how the database retrieves data but declares the desired result.

Declarative programming paradigm (Contd.)

- **Implementation of any basic Declarative paradigm:**
  - **Define the Problem:**
    - **Problem:** Retrieve a list of customers who made a purchase in the last month.
  - **Identify Data and Relationships:**
    - Relevant Tables: customers, purchases
    - Relationships: Each customer has a unique identifier (customer_id). Purchases are linked to customers through the customer_id foreign key.
  - **Choose Declarative Language:**
    - Here we choose SQL as it is a declarative language commonly used for database management.
  - **Write Declarative Statements:**

```sql
SELECT customer_name, email
FROM customers
WHERE customer_id IN (
    SELECT DISTINCT customer_id
    FROM purchases
    WHERE purchase_date >= DATE_SUB(CURDATE(), INTERVAL 1 MONTH)
);
```

  - **Utilize Built-in Functions:**
    - SQL provides built-in functions like DATE_SUB for date manipulation.
    - CURDATE for the current date, and IN for filtering based on a list of values use it for our ease.
  - **Test the results:**
    - Execute the query against the database and verify the results.

Declarative programming paradigm (Contd.)

- **Some of the key features of Declarative paradigm:**
  - **Specifies style** - It focuses on expressing the desired result, allowing us to declare what needs to be done rather than specifying the step-by-step instructions on how to achieve it.
  - **Improves code readability** - It's code tends to be more readable and expressive. By removing the low-level details of implementation, the code becomes more closer to natural language making it easier to understand.
  - **Provides more abstraction** - Declarative languages provides higher levels of abstraction, allowing developers to work with concepts and entities at a more abstract level.
  - **Uses immutable data** - It increases the use of immutable data structures, rather than modifying existing data, this results in creating new data structures, which can lead to better program stability and easier reasoning about code.

Declarative programming paradigm (Contd.)

- **Difference between Declarative and imperative paradigms:**

Imperative Programming	Declarative Programming
We specify **how** to get the desired result by providing detailed instructions.	We specify **what** result we expect from the program.
Imperative programming specifies and directs the control flow of the program.	Declarative programming specifies the expected result and core logic without directing the program's control flow.
The programmer makes the major decisions about how the program works.	The compiler makes the major decisions about how the program works.
It is easy to learn and understand.	The code is clean, readable, and effective.
It uses mutable variables, i.e., the values of variables can change during program execution.	It uses immutable variables, i.e., the values of variables cannot change.
Due to the use of mutable variables, imperative programming often changes the program's state by changing the internal data.	Declarative programming doesn't change the program's state.
Procedural Programming and Object-Oriented Programming are examples of imperative programming.	Functional Programming and Logic Programming are examples of declarative programming.
C, C++, Java, PHP, JavaScript, Python, etc., are programming languages that focus on the imperative paradigm.	SQL, LISP, Scala, Haskell, Prolog, Absys, Alice, etc., are programming languages that focus on the declarative paradigm.

- **HiveQL**
  - **Inspiration:** HiveQL draws inspiration from SQL (Structured Query Language), specifically the SQL syntax used in RDBMS. It was initially developed by Facebook in 2007. It is mostly written in Java.
  - **Paradigm used:** Declarative.
  - Apache Hive is a data warehouse implementation on top of Hadoop/HDFS.
  - HDFS (Hadoop Distributed File System) is an open source framework that quickly moves data between nodes. It's often used by companies that need to store and manage big data. HDFS can manage large amounts of data using low-cost hardware.
  - SQL query written in Hive is called as HiveQL or HQL.
  - HQL enables users to write SQL-like queries to interact with and analyze large datasets stored in Hadoop's distributed file system(HDFS).
  - It reuses common concepts from relational databases, such as tables, rows, columns, and schema, to make it as simple and more efficient.
  - HiveQL supports four file formats which are: TEXTFILE, SEQUENCEFILE, ORC and RCFILE (Record Columnar File).

HiveQL (Contd.)

- **Basic and simple example using HiveQL:**
  - Almost all the queries of HiveQL are similar to SQL queries only some are different.
  - Let us consider that we already have an emp table with these values,

Id	Name	Salary	Department
1	Gaurav	30000	Developer
2	Aryan	20000	Manager
3	Vishal	40000	Manager
4	John	10000	Trainer
5	Henry	25000	Developer
6	William	9000	Developer
7	Lisa	25000	Manager
8	Ronit	20000	Trainer

  - Now we are going to sort employee deatils based on their salary in descending order.
  - **select * from emp order by salary desc;** (This shows the characteristics of declarative paradigm as we are just telling what to do than specifying how to do)

```
Total MapReduce CPU Time Spent: 22 seconds 340 msec
OK
3       "Vishal"        40000.0 Manager
1       "Gaurav"        30000.0 Developer
7       "Lisa"  25000.0 Manager
5       "Henry" 25000.0 Developer
8       "Ronit" 20000.0 Trainer
2       "Aryan" 20000.0 Manager
4       "John"  10000.0 Trainer
6       "William"       9000.0  Developer
NULL    NULL    NULL    NULL
Time taken: 257.304 seconds, Fetched: 9 row(s)
hive>
```

HiveQL (Contd.)

- **Characteristics and features found in HiveQL associated with the declarative programming paradigm:**
  - **HQL Syntax are like SQL:**
    - HiveQL supports a SQL-like syntax, making it accessible and familiar to users with relational database query experience.
    - **Declarative aspect:** Users express what data they want to retrieve or manipulate without specifying the detailed procedural steps.
  - **Join and Aggregation Capabilities:**
    - HiveQL supports various types of joins and aggregation functions for data analysis and reporting.
    - **Declarative aspect:** Users declare the desired outcome of joins and aggregations without detailing how these operations should be executed.
  - **Integration with Hadoop Ecosystem:**
    - HiveQL integrates seamlessly with the broader Hadoop ecosystem, allowing users to leverage features like HBase, Pig, and Spark.
    - **Declarative aspect:** Users interact with different components of the Hadoop ecosystem declaratively, expressing their intentions without dealing with low-level details.

HiveQL (Contd.)

- **Features and advantages of HiveQL:**
  - HiveQL is designed for querying and managing only structured data stored in tables.
  - HiveQL is scalable, fast, and uses familiar concepts like SQL.
  - Schema gets stored in a database, while processed data goes into a Hadoop Distributed File System (HDFS).
  - Tables and databases get created first; then data gets loaded into the proper tables.
  - HiveQL supports four file formats: ORC, SEQUENCEFILE, RCFILE (Record Columnar File), and TEXTFILE.
  - Hive uses an SQL-inspired language, so it makes learning more accessible by utilizing familiar concepts found in relational databases, such as columns, tables, rows, and schema, etc.
- **Limitations of HiveQL:**
  - Hive is not designed for the OLTP (Online transaction processing).
  - It does not offer real-time queries.
  - It provides limited subquery support.
  - Latency of HQL is generally very high.

Scripting programming paradigm

- **Scripting paradigm:**
  - Scripting is a programming paradigm that emphasizes the automation of tasks by using scripts, which are sequences of commands or instructions.
  - Scripting is commonly used for tasks such as automation, system administration, and rapid prototyping.
- Scripting language is a language which is used to implement scripting approach.
- **Scripting Language:**
  - A Scripting language (aka scripting, or script) is a series of commands that can be executed without the need for compiling.
  - While all scripting languages are programming languages, but all programming languages are not scripting languages.
  - PHP, Perl, and Python, Lua are common examples of scripting languages.
  - Most of scripting languages uses an interpreter to translate commands directly from source code without compilation step.
- **Is Scripting equals to Coding?**
  - Scripting is not the same as coding, but these two are very similar.
  - Though both are used in the backend of websites and applications, there are key differences:
  - A programming language allows you to create a new program but a scripting langu allows you to provide instructions for a program that already exists.
  - Scripting provides functionality, while coding provides structure.

Scripting programming paradigm (Contd.)

- **Some of the key features of scripting:**
  - Scripting languages can be directly embedded into an application or in a variety of environments.
  - They are quite easy to learn and use. For example, JavaScript and PHP are the easiest scripting languages.
  - They provide all of the basic features common to modern programming languages, such as powerful variable types, basic operations (addition, subtraction, etc.), standard control statement, functions, etc.
  - Scripting languages require less memory from the system because they are not compiled, are interpreted.
- **Implementation of any basic scripting**
  - **Define the Task:**
    - Rename all text files in a directory by replacing "old_" with "new_" in their filenames.
  - **Write Script:**
    - Utilize any scripting language features to iterate through files and perform the renaming. Here we use python.

```python
import os

for filename in os.listdir('.'):
    if filename.endswith('.txt'):
        new_name = filename.replace('old_', 'new_')
        os.rename(filename, new_name)
```

  - **Execute and Test:**
    - Run the script and verify that the files are renamed as expected.

Scripting programming paradigm (Contd.)

- **Types of scripting:**
  - There are two main types of scripting languages: **server-side and client-side.**
  - **Server-side scripting languages:**
    - The term server-side scripting language refers to those that run in a web server.
    - Since it performs from the back-end side, the script is not visible to the visitor. so, it is a more secure approach.
    - They are often used to create dynamic websites and platforms, handle user queries, and generate and provide data and others.
    - A famous example of server-side scripting is the use of PHP in WordPress.
    - Examples of server-side scripting languages: PHP, Python, Node.js, Perl, and Ruby.
  - **Client-side scripting languages:**
    - Unlike the above, client-side scripting languages run in the user's browser.
    - It is usually performed at the front end, which makes it visible to visitors and makes it quite vulnerable to exploits and leaks.
    - Since it runs locally, they usually provide better performance and therefore, they do not overload our server.
    - Examples of client-side scripting languages: HTML, CSS, jQuery, and JavaScript.

Lua

- **Lua:**
  - **Inspiration:** Lua draws inspiration from Simple Object Language (SOL) and Data-Entry Language (DEL). It was created in 1993 by the members of the Computer Graphics Technology Group (Tecgraf). It is mostly written in C.
  - **Paradigm used:** Multi-paradigm and majorly used for scripting.
  - Lua is a scripting language that supports multiple programming methods, including procedural, object-oriented, functional, and data-driven programming.
  - It is used for many applications, including games, web applications, and image processing.
  - It is considered fast and easy to learn and use.
  - It is named after the moon and was designed by a team of computer scientists in Brazil in 1993.
  - It is most useful for people who want to add new functionality to an existing game, website, or application that allows Lua code to extend it.
  - For example, the mobile payment app Venmo and the game Angry Birds were both made using Lua.

Lua (Contd.)

- **Characteristics and features found in Lua associated with the scripting programming paradigm**
  - **Dynamic Typing:**
    - Lua is dynamically typed, meaning variable types are determined at runtime rather than compile-time.
    - This flexibility is a common trait in scripting paradigm, allowing for rapid development and ease of use.
  - **Interpreted:**
    - Lua is an interpreted language, which means it doesn't require a separate compilation step.
    - Scripts can be executed directly by an interpreter, enabling quick testing and modification of code.
  - **Garbage collection:**
    - Lua has automatic garbage collection, handling memory management without explicit developer intervention.
    - This helps simplify the scripting process and reduces the risk of memory-related errors.
  - **Embeddable:**
    - Lua is designed to be easily embedded into other applications, making it an excellent choice for scripting within larger systems. It provides a simple and lightweight API for integration.
  - **Portability:**
    - Lua is designed to be portable across different platforms, making it suitable for scripting in various environments.
    - This portability is essential for scripting languages to be adaptable to diverse systems.

Lua (Contd.)

- **Basic and simple example using Lua:**

```lua
-- Lua script

function calculateRectangleArea(length, width)
    return length * width
end

print("Enter the length of the rectangle:")
local length = io.read("*n") -- Reads a number from user

print("Enter the width of the rectangle:")
local width = io.read("*n") -- Read another number from user

local area = calculateRectangleArea(length, width)
print("The area of the rectangle is: " .. area)

```

```
Enter the length of the rectangle:
5
Enter the width of the rectangle:
2
The area of the rectangle is: 10
```

- **Key features and advanatages of Lua:**
  - The compact C implementation makes Lua fast and memory-efficient.
  - It's code can be built and run across various platforms like Windows, Linux, macOS, iOS, Android etc. This flexibility is powered by its ANSI C codebase.
  - It features incremental garbage collection which makes memory management seamless.
- **Limitations of Lua:**
  - It has a very small community of users, still relatively unknown next to Perl and Python languages.
  - The Lua does not support inheritance in coding while we have to use Meta table for using inheritance functionality.
  - It is rarely used for standalone programming language, usually it is used as embedded scripting language for individual programs.

Analysis of Declarative paradigm

* **Strengths:**
  - Declarative programming can be simpler and more straightforward than any paradigms, as it focuses on describing what should happen rather than how it should happen. Example: SQL commands which was discusses earlier.
  - Declarative code is often easier to read and understand, as it contains codes with more implementation details.
  - In declarative programming optimization is easy as its implementation is controlled by an predefined algorithm.
  - Using the declarative programming style, programmers can make code that they can use for different purposes in the future.
* **Weakness:**
  - In declarative programming, it is tough to personalize codes as per requirement as it depends on the implementation of an algorithm. Thus, it is very difficult to modify the programs.
  - Sometimes, declarative programming becomes very difficult to understand for beginners.
  - It is less efficient than imperative programming, as it can involve more steps to accomplish a particular task.
  - Debugging declarative programs can be more challenging than debugging other programs as tracing the flow of execution can be less straightforward, making it harder to identify and fix bugs.
* **Notable features:**
  - **Specifies style** - It focuses on expressing the desired result, allowing us to declare what needs to be done rather than specifying the step-by-step instructions on how to achieve it.
  - **Improves code readability** - It's code tends to be more readable and expressive. By removing the low-level details of implementation, the code becomes more closer to natural language making it easier to understand.
  - **Provides more abstraction** - Declarative languages provides higher levels of abstraction, allowing developers to work with concepts and entities at a more abstract level.
  - **Uses immutable data** - It increases the use of immutable data structures, rather than modifying existing data, this results in creating new data structures, which can lead to better program stability and easier reasoning about code.

Analysis of HiveQL

- **HiveQL:**
  - **Strengths:**
    - HiveQL is easy to use application for both beginners and experts in programming. Anyone who is familiar with SQL can work with Hive easily.
    - It supports various types of file formats such as textfile, ORC, Parquet, LZO Compression, etc.
    - It is used to manage the very large datasets that are stored in the Hadoop Distributed File System.
  - **Weakness:**
    - Hive Query Language does not support the transaction processing feature.
    - Latency of HiveQL is generally very high.
    - HiveQL uses table form to store data which always process structured data. Even if the unstructured data is written using SQL queries, HiveQL cannot support them.
  - **Notable features:**
    - **Syntax like SQL** - HiveQL has a syntax similar to SQL, making it accessible and familiar to users who are already good in SQL. This helps in quickly transitioning users from traditional relational databases to Hive.
    - **DML and DDL support** - It supports standard SQL operations for defining and manipulating data. This includes statements for creating and altering tables (DDL) as well as querying and updating data (DML).
    - **Large Built-in functions** - It provides a variety of built-in functions for performing operations on data like mathematical operations, string manipulation, date and time functions, etc.
    - **Supports subqueries** - It supports subqueries, allowing us to nest queries within other queries to express more complex logic and perform advanced data manipulations.

Analysis of Scripting paradigm

- **Scripting paradigm:**
  - **Strengths:**
    - Scripting languages are embeddable, that mean they can be easily embedded into existing components or applications such as Microsoft Word, etc.
    - As most scripting languages use an interpreter so no executable files are stored. Therefore, they require less memory.
    - They are portable, that means they can run on any operating system.
    - The coding of scripting language is easy to use and learn so it is very beginner friendly language.
  - **Weakness:**
    - The main disadvantage of scripting languages is that they are slower than compiled languages because the interpreter in scripting languages need to read and analyze each statement line by line during the execution.
    - Every time the interpreter detects an error during the execution of program, it stops further execution until the error is resolved.
    - It can lack the coding standards that are followed by the developers.
  - **Notable features:**
    - **Provides automation** - Scripting languages are commonly used for automating repetitive tasks, system administration, and workflow automation. They allow users to create scripts that perform a series of commands or tasks with less effort.
    - **Platform independent** - Many scripting languages are platform-independent(i.e, The scripts that are written once can be executed on different operating systems without modification).
    - **Used as Glue code** - They are often used as "glue code" to integrate different components in application. They can facilitate communication between various software components written in different languages.

Analysis of Lua

- **Lua:**
  - **Strengths:**
    - Lua has a fast and efficient interpreter, which contributes to its high performance. It is often used in performance-critical applications such as game development.
    - It has automatic memory management through garbage collection, it simplifies the memory management for developers and helps to prevent memory leaks.
    - It's small size and high performance make it easy to integrate into applications.
    - It supports all versions of Unix and Windows, mobile devices, embedded microprocessors, and IBM mainframes.
  - **Weakness:**
    - Lua's standard libraries are relatively minimal compared to some other languages.
    - Limited error handling support can lead to longer debuging times to identify the exact errors and correct it.
    - All variables in lua are created as global variables (global scope), which can lead to errors in variable assignments.
    - It is rarely used for standalone programming language, it is used as embedded scripting language for individual program.
  - **Notable features:**
    - **Embeddability** - Lua is designed to be easily embedded into other applications, providing a simple and clean API for integration. This feature makes it a popular choice for extending the functionality of existing software.
    - **Portability** - It is highly portable and can run on various platforms, including Windows, Linux, macOS, and embedded systems. This makes it a best choice for cross-platform development.
    - **Lightweight** - It is designed to be lightweight and fast, with a small memory usage. This makes it suitable for embedded systems and environments with limited resource constraints.
    - **Garbage collection** - It includes automatic memory management through a garbage collector, reducing explicit memory management. It helps to prevent memory leaks.

Declarative vs Scripting Paradigm

- **Similarities:**
  - **Readability:**
    - Both paradigms emphasize on readable code.
    - Declarative focuses on expressing what should happen.
    - Scripting languages focus on human-readable syntax like natural language.
  - **Abstraction:**
    - Both Declarative and Scripting paradigm provide a level of abstraction by simplifying complex operations for users.
  - **User-friendly:**
    - Both paradigms strive to make programming accessible, either by simplifying code (declarative) or using an easily understandable syntax (scripting).

Declarative vs Scripting Paradigm (Contd.)

- **Differences:**
  - **Focus of the language:**
    - Declarative focuses on "what" needs to be achieved without specifying the step-by-step process.
    - Scripting focuses on automation and scripting tasks, detailing "how" a task is accomplished through a series of commands.
  - **Efficient execution:**
    - In declarative codes are generally optimized through predefined algorithms, allowing for efficient execution.
    - In scripting, the execution speed is slower due to interpretation of commands line by line.
  - **Portability:**
    - In declarative, code may be less portable compared to scripting.
    - In scripting, it is more platform-independent; codes are easily portable on multiple platforms.
  - **Use cases:**
    - Declarative programs are commonly used in database queries, configuration files, etc.
    - Scripting files are used for automation, integration, and used as "glue code" to connect different components.

HiveQL vs Lua

- **Similarities:**
  - **Abstraction:**
    - Both HiveQL and Lua provide a level of abstraction, allowing users to work at a higher conceptual level without dealing with low-level details.
    - Also, abstraction enhances code readability, maintenance, and reduces complexity in different domains.
  - **Scripting capabilities:**
    - Scripting features provide flexibility and are valuable for various applications, from data processing to game development.
    - So, both HiveQL and Lua have scripting capabilities, enabling users to automate tasks and execute sequences of commands.

HiveQL vs Lua (Contd.)

- **Differences:**
  - **Domain:**
    - HiveQL is specifically designed for querying and managing large-scale distributed data in the context of Apache Hive, often used in big data processing and analytics.
    - While Lua is a lightweight scripting language used in a variety of domains, including game development and embedded systems.
  - **Scaling and optimization:**
    - HiveQL is optimized for distributed data processing in large-scale environments.
    - While Lua is efficient for various applications, it is not optimized for large-scale distributed processing like HiveQL.
  - **Ecosystem:**
    - HiveQL is a part of the Apache Hive ecosystem, integrating with Hadoop for distributed computing.
    - While Lua is known for its embeddability and extensibility, used as a scripting language in applications and games.
  - **Uses-cases:**
    - HiveQL is used to write queries to analyze and process large datasets stored in distributed environments.
    - While Lua is a scripting language used to write game logic, configure applications, and embed scripts in various software.

Conclusion

- Quick summary obtained by exploring the programming paradigms and its associated languages are:
  - **Declarative vs Scripting:**
    - **Similarities:** Both paradigms prioritize readable code, abstraction, and user-friendliness.
    - **Differences:**
    - Declarative focuses on "what," scripting on "how."
    - Declarative is more efficient, but less portable while scripting is less efficient but portable.
    - Declarative suits database queries while scripting for automation and integration.
  - **HiveQL vs Lua:**
    - **Similarities:** Both languages offer abstraction and scripting capabilities.
    - **Differences:**
    - HiveQL is used for big data processing while Lua is used for low level games, embedded systems, etc.
    - HiveQL is optimized for large-scale distributed processing while Lua is efficient but not optimized for large-scale.
    - HiveQL integrates with Hadoop while Lua is embeddable and extensible.
    - HiveQL used to query large datasets while Lua for game logic and software scripting.
- In conclusion, declarative and scripting paradigms share foundational principles but differ in focus and efficiency. Similarly, HiveQL and Lua offer abstraction and scripting features but differ in domains, scalability, ecosystems, and use cases.

References

- FreeCodeCamp - An Introduction to Programming Paradigms
- GeeksforGeeks - Introduction of Programming Paradigms
- Programming Paradigms by Van Roy
- GeeksforGeeks - Difference Between Imperative and Declarative Programming
- TechTarget - Declarative Programming Definition
- Wikipedia - Declarative Programming
- GeeksforGeeks - Difference Between SQL and HiveQL
- Scaler - Hadoop HiveQL
- Medium - What is HiveQL?
- Wikipedia - Scripting Language

References (Contd.)

- MRCET - Scripting Languages Digital Notes
- Rock Content - Scripting Languages
- JavaTpoint - What is a Scripting Language?
- Lua Official Website
- TutorialsPoint - Lua Programming Tutorial
- BMC - Lua Programming Language
- Programming in Lua (First Edition)
- Codecademy - What is Lua Programming Language Used For?
- LinkedIn - Lua Programming Language by Bikash Kumar Sundaray