Programming Language Paradigms + Scripting Languages

Programming Language Theory

Topics

- PL Paradigm Overview
- Scripting Languages

PL Paradigm Overview

PL Paradigms

- Principles and Strategies which a programming language follows.
  - e.g.) Procedural, Imperative, OOP, Functional, Logic, etc.
- One programming language may follow multiple paradigms.
  - Java - Imperative and Object-Oriented.

Language Paradigms

- So many ways to group or categorize programming languages.
- **Imperative**
  - procedural (Fortran, Pascal, Basic, C)
  - object-oriented (Smalltalk, C++, Java)
  - scripting languages (Perl, Python, JavaScript)
- **Declarative**
  - functional programming (Scheme, ML, Haskel)
  - logic programming (Prolog, Datalog)

Imperative Languages

- *imperative*: *giving an authoritative* ***command (or statements)****.*
- Using ***assignments*** to change a program's state.
- Focus on ***how*** a program operates (algorithms).
- In low-level, machine code (instructions) is to modify the program state (i.e. register/memory).
- In high-level, program states are described by variables, and statements are the machine instructions.

Declarative Languages

- Declarative languages are often defined as ***non-imperative*** languages.
- Instead of how to do, it describes ***what to do***, or ***what a program should be***.
- Programs describe ***desired results***, rather than provide commands which should be performed by a computer.
- *Functional* and *Logic* languages.
- However, ***there is no clear distinction*** between Imperative and Declarative languages.

Example: Euclidean Algorithm

- **Imperative: C++**
- Start from main(), the program is a series of commands.
- We can follow the program step by step.

```cpp
int gcd(int a, int b) {
    if(b == 0)
        return a;
    else
        return gcd(b, a%b);
}

int main() {
    cout << gcd(10, 5);
    return 0;
}
```

Example: Euclidean Algorithm

- **Imperative, OOP: Java**
- We can also follow the program step by step with main() part.
    - Imperative.
- The program defines a class GCD, and hides its information inside, provides operations for the class.
    - OOP.

```java
public class GCD {
    private int a, b;
    public GCD(int a, int b) {
        this.a = a;
        this.b = b;
    }
    public int compute() {
        return a >= b ? gcd(a, b) : gcd(b, a);
    }
    private int gcd(int a, int b) {
        if(b == 0)
            return a;
        else
            return gcd(b, a%b);
    }
    Run | Debug
    public static void main(String[] args) {
        System.out.println(new GCD(12, 4).compute());
    }
}
```

Example: Euclidean Algorithm

- **Declarative, Functional: Scheme**
- It simply defines what is a function gcd.
  - Declarative.
- Still, it has a command to print something to screen.
  - Imperative.

```scheme
(define (gcd a b)
    (if (equal? b 0)
        a
        (gcd b (modulo a b))))
(display (gcd 12 4))
```

Example: Euclidean Algorithm

- **Declarative, Functional: Scala**
- It defines what is a function gcd.
  - Declarative.
- It also contains Java-like features such as object GCD, main() method.

```scala
object GCD {
  def main(args: Array[String]): Unit = {
    println(gcd(12, 4))
  }
  def gcd(a: Int, b: Int): Int =
    if (a%b==0)
      b
    else {
      var r: Int = a%b
      gcd(b, a%b)
    }
}
```

Example: Euclidean Algorithm

- **Declarative, Logic: Prolog**
- It describes ***Facts*** and ***Rules*** about gcd.
    - Declarative.
- You can ask ***questions (or queries)*** based on the facts and rules.

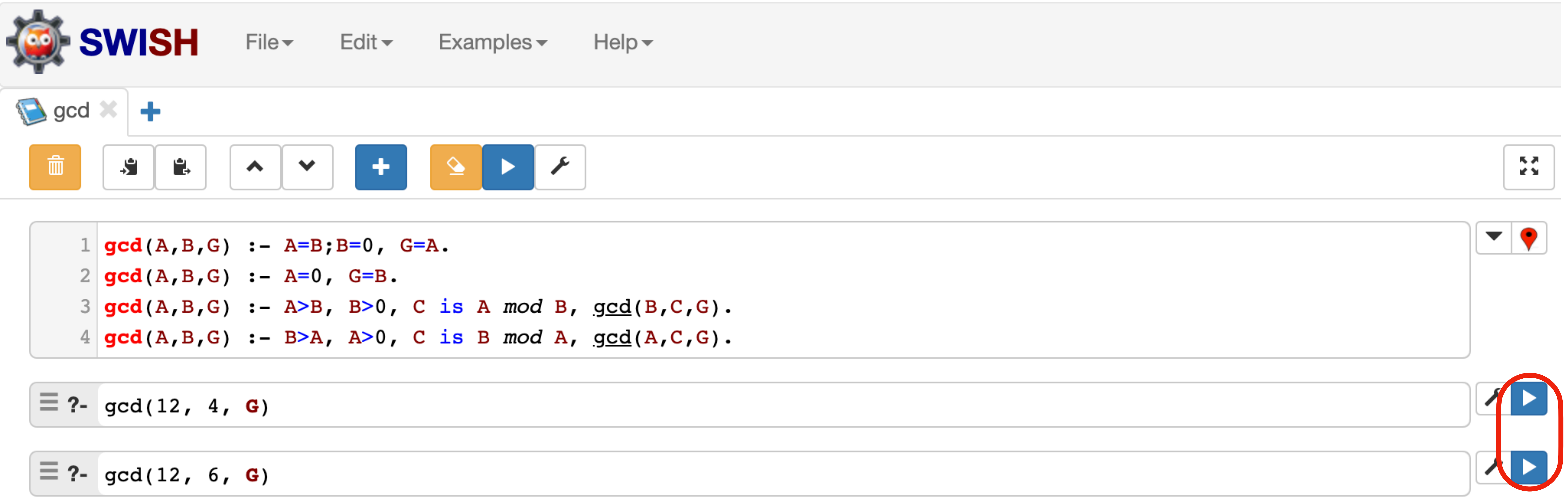
```
gcd(A,B,G) :- A = B;B = 0, G = A.
gcd(A,B,G) :- A = 0, G = B.
gcd(A,B,G) :- A > B, C is A mod B, gcd(C,B,G).
gcd(A,B,G) :- A < B, C is B mod A, gcd(C,A,G).
```

```
?- gcd(12, 4, G).
G = 4 .

?- gcd(12, 6, G).
G = 6 .
```

Example: Euclidean Algorithm

https://swish.swi-prolog.org/p/gcd.swinb



Scripting Languages

What is Scripting Language?

- Actual use of a computer often requires to combine multiple programs.
  - e.g.) Print a certain type of error messages from all the log files in a directory.
  - A: List up all the log files in a directory.
  - B: Read each log file from the list.
  - C: Find error messages of the type.
  - D: Print the found messages in a specific format.

Glue Language

- Scripting languages are often called ***Glue Languages***.
- Glue multiple programs together to achieve a goal.
- Two Ancestors: Shells/Terminals(sh, bash) + Text Processing (sed, awk).
- General purpose scripting languages.
    - Perl, Python, Ruby, PowerShell, AppleScript, etc.
- For Web.
    - PHP, JSP, Ruby on Rails, JavaScript, etc.

Common Characteristics

- Usually provide both ***batch and interactive*** mode.
- *More easy to write - simple expressions*.
  - Hello World

```java
public class HelloWorld {
    Run | Debug
    public static void main(String[] args) {
        System.out.println("Hello World!");
    }
}
```

```python
print("Hello World!")
```

Common Characteristics

- *Simple Scoping Rules with Optional Declarations.*
  - Often consider all names as global or local.
  - Declarations are not mandatory.
  - In Python,

```python
a = 10 + 3
b = a + 2
print(a, b)
```

Common Characteristics

- *Flexible and Dynamic Typing.*
  - Mostly employ dynamic type checking.
  - One variable is used as different types in different contexts.
  - In JavaScript,

```
a = 3
str = "string"
c = str + 3
```

Common Characteristics

- *Good for Pattern Matching and String Manipulation.*
- *High-level Data Structures.*
    - Tuples, List, Dictionaries.
    - In Python, they are supported by basic language features.
    - In C++ or Java, they are supported by standard libraries (i.e., extension or pre-implemented libraries).

Problem Domains

- **Shell Scripts**
  - Manipulating files and directories.
  - Interactively glue unix commands.
- **Text Processing and Report Generation**
  - Support of pattern matching and string manipulation.
  - Perl: **P**ractical **E**xtraction and **R**eport **L**anguage
    - Used in Bio Informatics - Gene Sequence Analysis.

Problem Domains

- **Mathematics and Statistics**
  - Easy to write, easy manipulation of data.
  - R and Python are popularly used in this area.
- **General Purpose Glue Language**
  - You can connect or redirect one programs output to another programs input.

Problem Domains

- **Extension Language**
  - Scripting languages are often used to add more useful features (such as new commands) to existing programs.
  - Lua: Heavily used in Game Industry.
    - Quest, Skill, Item, Monster Specifications.
    - Add-on development.
- **Web Applications**
  - Used both in server and client side.

Python Scope Rules

- Python has distinctive, interesting scope rules.
- A variable is assumed to be local, unless it is explicitly declared.
- A variable that is only read, but not written in a block can be found in the closest enclosing scope contains the write.

```python
i = 1; j = 3 #these are global
def outer():
    def middle(k):
        def inner():
            global i
            i = 4
        inner()
        return i, j, k
    i = 2
    return middle(j)

print(outer()) #(i, j, k)
print(i, j) #1, 3 -> 4, 3
```

```
(2, 3, 3)
4 3
```

Python Scope Rules

- `outer()` doesn't read `i`, but write a new value to `i`.
- It reads `j` and passes it to `middle()`.
- `middle()` reads both `i` and `j`.
- `inner()` writes global `i`.

```python
i = 1; j = 3 #these are global
def outer():
    def middle(k):
        def inner():
            global i
            i = 4
        inner()
        return i, j, k
    i = 2
    return middle(j)

print(outer()) #(i, j, k)
print(i, j) #1, 3 -> 4, 3
```

```
(2, 3, 3)
4 3
```

Task: Print Error Logs

- A series of log files are stored in a directory.
- Log file name contains its date.
- In each file, a log's type is represented by its header.
  - e.g.) [build], [error], [normal]
- Store only [error] logs to a new file.

```
[build] build task xxx started.
[normal] /usr/local/bin/python3 ...
[error] cannot resolve dependencies...
[normal] /usr/local/bin/python3 ...
[error] failed to compile XXX.
[error] cannot execute task YYY.
[build] build task xxx completed.
```

Task: Print Error Logs

- With Java:
- Reading a file itself is not a convenient task if you're only using standard libraries.
- Writing code ➞ compile ➞ execution.
    - These steps are also not easy if you're not using IDE.

```java
public static String getContent(File file, String charset)
    StringBuffer sb = new StringBuffer();
    String content;
    BufferedReader br = getBufferedReader(file, charset);
    char[] cbuf = new char[500];
    int len = 0;
    while((len=br.read(cbuf))>-1){
        sb.append(cbuf, 0, len);
    }
    br.close();
    content = sb.toString();
    return content;
}
```

Task: Print Error Logs

- With Python:
- Several lines of code are enough to satisfy the requirements.
- Using REPL, more easily write and execute necessary code.

```python
errors = []
with open('xxx.log') as f:
    lines = f.readlines()
    for line in lines:
        if line.startswith('[error]'):
            errors.append(line)
print(errors)
```

```
Python 3.9.6 (v3.9.6:db3ff7
[Clang 6.0 (clang-600.0.57)
Type "help", "copyright", "
>>> f = open('xxx.log')
>>> lines = f.readlines()
>>> lines[0:1]
['[error] xxxxx\n']
>>>
```

Summary

- PL Paradigms
  - Imperative vs. Declarative: Examples.
- Scripting Languages
  - Common Characteristics.
  - Problem Domains.