5.7 MULTI-PARADIGM LANGUAGES

Multi-Paradigm Languages are languages that can use multiple Programming paradigm together. Many programming languages support Muli Pradigum programming, for example, C++ is both an imperative and an object-oriented language, Parlog is both a parallel and a logic programming language.

Concept of Multi-Paradigm programming?

The concept of multi-paradigm programming is very simple to understand, When a programming language use more than one programming paradigm it know as multi-paradigm language, and the concept of combining these paradigms are Multiparadigm programming.

In OOPs, programmers can think of a program as a collection of interacting objects, while in functional programming a program can be thought of as a sequence of stateless function evaluations. When programming computers or systems with many processors, process-oriented programming allows programmers to think about applications as sets of concurrent processes acting upon logically shared data structures

Some languages are designed to support one particular paradigm (Smalltalk supports object-oriented programming, Haskell supports functional programming), while other programming languages support multiple paradigms

Many programming paradigms are as well known for what techniques they forbid as for what they enable. For instance, pure functional programming disallows the use of Side effects while Structured programming disallows the use of the goto statement. Partly for this reason, new paradigms are often regarded as doctrinaire or overly rigid by those accustomed to earlier styles.

PYTHON

- PYTHON was designed in the early 1990s by Guido van Rossum.
- PYTHON borrows ideas from languages as diverse as PERL, HASKELL, and the object oriented languages, skillfully integrating these ideas into a coherent whole.
- PYTHON scripts are concise but readable, and highly expressive

Python is extensible: if we invoke how to program in C, it is easy to add new built in function or module to the interpreter, either to perform critical operations at maximum speed of to link python programs to libraries that may only be available in binary form

Python has following characteristics.

- Easy to learn and program and is object oriented.
- Rapid application development
- Readability is better
- It can work with other languages such as C,C++ and Fortran
- Powerful interpreter

Extensive modules support is available

Values and types

- PYTHON has a limited repertoire of primitive types: integer, real, and complex Numbers.
- It has no specific character type; single-character strings are used instead.
- Its boolean values (named False and True) are just small integers.
- PYTHON has a rich repertoire of composite types: tuples, strings, lists, dictionaries, and objects.

Variables, storage, and control

- PYTHON supports global and local variables.
- Variables are not explicitly declared, simply initialized by assignment.
- PYTHON adopts reference semantics. This is especially significant for mutable values, which
- can be selectively updated.

Primitive values and strings are immutable; lists, dictionaries, and objects are mutable; tuples are mutable

if any of their components are mutable

 PYTHON's repertoire of commands include assignments, procedure calls, con-ditional (ifbutnotcase-) commands, iterative (while- and for-) commands, and exception-handling

commands.

Dynamically typed language:

Python is a dynamically typed language. Based on the value, type of the variable is during the execution

of the program.

Python (dynamic)

C = 1

C = [1,2,3]

C(static)

Double c; c = 5.2;

C = “a string….”

Strongly typed python language:

Weakly vs. strongly typed python language differs in their automatic conversions.

Perl (weak)

$b = `1.2`

$c = 5 * $b;

Python (strong)

b =`1.2`

c= 5* b;

PYTHON if- and while-commands are conventional

Bindings and scope

• A PYTHON program consists of a number of modules, which may be grouped into packages.

• Within a module we may initialize variables, define procedures, and declare classes

• Within a procedure we may initialize local variables and define local procedures.

• Within a class we may initialize variable components and define procedures (methods).

• PYTHON was originally a dynamically-scoped language, but it is now statically scoped

In python, variables defined inside the function are local to that function. In order to change them as global variables, they must be declared as global inside the function as given below.

```
S = 1

Def myfunc(x,y);

Z = 0

Global s;

 S = 2

Return y-1 , z+1;
```

Procedural abstraction

• PYTHON supports function procedures and proper procedures.

• The only difference is that a function procedure returns a value, while a proper procedure

returns nothing.

Since PYTHON is dynamically typed, a procedure definition states the name but not the type of each formal parameter

Python procedure

```
 Eg :Def gcd (m, n):

 p,q=m,n

 while p%q!=0:

 p,q=q,p%q

 return q
```

Python procedure with Dynamic Typing

```
Eg: def minimax (vals):
```

```
min = max = vals[0]
for val in vals:
if val < min:
min = val
elif val > max:
max = val
return min, max
```

Data Abstraction

• PYTHON has three different constructs relevant to data abstraction: packages ,modules , and Classes

Modules and classes support encapsulation, using a naming convention to distinguish between public and private components.

• A Package is simply a group of modules

• A Module is a group of components that may be variables, procedures, and classes

• A Class is a group of components that may be class variables, class methods, and instance methods.

• A procedure defined in a class declaration acts as an instance method if its first formal parameter is named self and refers to an object of the class being declared. Otherwise the procedure acts as a class method.

Separate Compilation

• PYTHON modules are compiled separately.

• Each module must explicitly import every other module on which it depends

• Each module's source code is stored in a text file. E g: program.py

• When that module is first imported, it is compiled and its object code is stored in a file named program.pyc

• Compilation is completely automatic

• The PYTHON compiler does not reject code that refers to undeclared identifiers. Such code simply fails if and when it is executed

• The compiler will not reject code that might fail with a type error, nor even code that will certainly fail, such as:

```
Def fail (x):
 Print x+1, x[0]
```

Module Library

• PYTHON is equipped with a very rich module library, which supports string handling, markup, mathematics, and cryptography, multimedia, GUIs, operating system services, internet services, compilation, and so on.

• Unlike older scripting languages, PYTHON does not have built-in high-level string processing or GUI support, so module library provides it.