Motivation

A4M36TPJ, 2015/2016

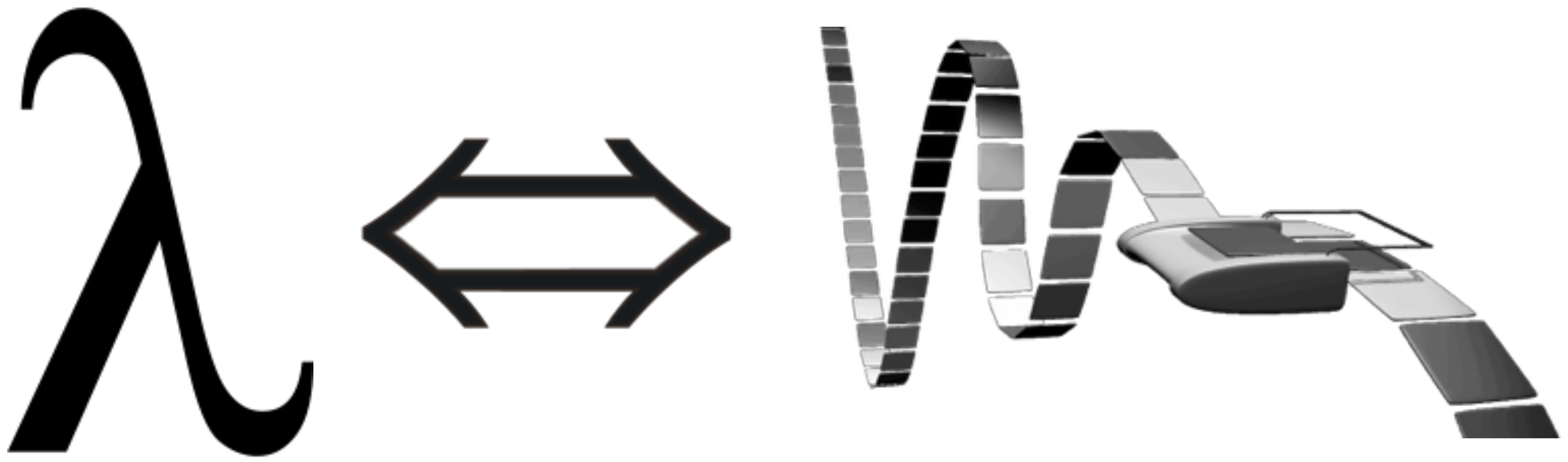
What is….

- Q: What is a **programming language**?
- A: A **programming language** is a formal constructed language designed to communicate instructions to a machine, particularly a computer. Programming languages can be used to create programs to control the behavior of a machine or to express algorithms. [wikipedia]

History

Computational Models

- 1936 - A. Church invented Lambda Calculus
- 1936 - A. Turing invented Turing Machine



History…

- First programming languages **50s** to **60s**:
  - 1951 - Regional Assembly Language
  - 1952 - Autocode
  - 1954 - IPL (forerunner to LISP)
  - 1955 - FLOW-MATIC (led to COBOL)
  - 1957 - FORTRAN (First compiler)
  - 1957 - COMTRAN (precursor to COBOL)
  - 1958 - LISP
  - 1958 - ALGOL 58
  - 1959 - FACT (forerunner to COBOL)
  - 1959 - COBOL
  - 1959 - RPG
  - 1962 - APL
  - 1962 - Simula
  - 1962 - SNOBOL
  - 1963 - CPL (forerunner to C)
  - 1964 - BASIC
  - 1964 - PL/I
  - 1966 - JOSS
  - 1967 - BCPL (forerunner to C)

Fortran

```
PROGRAM Compute_Factorial
IMPLICIT NONE

INTERFACE
   FUNCTION Factorial(n)
     INTEGER :: Factorial
     INTEGER, INTENT(IN) :: n
   END FUNCTION Factorial
END INTERFACE

! Declare local variables
INTEGER :: n

!  Prompt user for radius of circle
write(*, '(A)', ADVANCE = "NO") "Enter n for computing n!:  "
read(*,*) n

! Write out value of factorial using function call
write(*,100) n, "factorial is ", Factorial(n)
100 format (I3, 2x, A, 2x, I12)

END PROGRAM Compute_Factorial

RECURSIVE FUNCTION Factorial(n)  RESULT(Fact)

IMPLICIT NONE
INTEGER :: Fact
INTEGER, INTENT(IN) :: n

IF (n == 0) THEN
   Fact = 1
ELSE
   Fact = n * Factorial(n-1)
END IF

END FUNCTION Factorial
```

LISP

```
(defun factorial (N)
  "Compute the factorial of N."
  (if (= N 1)
      1
    (* N (factorial (- N 1)))))
```

BASIC

```
10 PRINT "Enter a number to see the factorial"
20 INPUT N
30 LET F = 1
40 FOR L = 1 to N
50 LET F = F * L
60 NEXT L
70 PRINT N; "! = "; F
80 PRINT "Would you like to see another
factorial? 1=YES ANY(other)=NO"
90 INPUT RESPONSE
100 IF RESPONSE = 1 THEN GOTO 10
110 PRINT "Thanks! Goodbye."
120 END
```

History…

- The period from the late 1960s to the late 1970s brought a major flowering of programming languages:
    - **Simula**, invented in the late 1960s by Nygaard and Dahl as a superset of Algol 60, was **the first language designed to support object-oriented programming**.
    - **C**, an early **systems programming language**, was developed by Dennis Ritchie and Ken Thompson at Bell Labs between 1969 and 1973.
    - **Smalltalk** (mid-1970s) provided a complete ground-up design of an **object-oriented language**.
    - **Prolog**, designed in 1972 by Colmerauer, Roussel, and Kowalski, was **the first logic programming language**.
    - **ML** built a **polymorphic type system** (invented by Robin Milner in 1973) on top of Lisp, pioneering statically typed functional programming languages.

Prolog

```
fact1(0,Result) :-

    Result is 1.

fact1(N,Result) :-

    N > 0,

    N1 is N-1,

    fact1(N1,Result1),

    Result is Result1*N.
```

ML

```
fun fac (0 : int) : int = 1
  | fac (n : int) : int = n * fac (n - 1)
```

C using loop

```
#include <stdio.h>

int main()

{

  int c, n, fact = 1;

  printf("Enter a number to calculate it's factorial\n");

  scanf("%d", &n);

  for (c = 1; c <= n; c++)

    fact = fact * c;

  printf("Factorial of %d = %d\n", n, fact);

  return 0;

}
```

C using recursion

```
#include<stdio.h>

long factorial(int);

int main()

{

  int n;

  long f;

  printf("Enter an integer to
find factorial\n");

  scanf("%d", &n);

  f = factorial(n);

  printf("%d! = %ld\n", n, f);

  return 0;

}


long factorial(int n)

{

  if (n == 0)

    return 1;

  else

    return(n * factorial(n-1));

}
```

History …

The 1980s were years of relative consolidation in imperative languages. Rather than inventing new paradigms, all of these movements elaborated upon the ideas invented in the previous decade. C++ combined object-oriented and systems programming.

- 1980 - C++ (as C with classes, renamed in 1983)
- 1983 - Ada
- 1984 - Common Lisp
- 1984 - MATLAB
- 1985 - Eiffel
- 1986 - Objective-C
- 1986 - Erlang
- 1987 - Perl
- 1988 - Tcl
- 1988 - **Mathematica**
- 1989 - FL (Backus)

Ada

```
function Factorial(N : Positive) return Positive is

   Result : Positive := 1;

begin

   if N > 1 then

      Result := N * Factorial(N - 1);

   end if;

   return Result;

end Factorial;
```

Erlang

```
-module(math).

-export([factorial/1]).

factorial(0) ->
  1;
factorial(X) when X > 0 ->
  X * factorial(X-1).
```

Mathematica

```
fact[x_] := x!

fact[10]
```

History…

The rapid growth of the Internet in the mid-1990s was the next major historic event in programming languages. By opening up a radically new platform for computer systems, the Internet created an opportunity for new languages to be adopted. In particular, the JavaScript programming language rose to popularity because of its early integration with the Netscape Navigator web browser. Various other scripting languages achieved widespread use in developing customised applications for web servers such as PHP.

- 1990 - **Haskell**
- 1991 - **Python**
- 1991 - Visual Basic
- 1993 - **Ruby**
- 1993 - Lua
- 1994 - CLOS (part of ANSI Common Lisp)
- 1995 - Ada 95
- 1995 - **Java**
- 1995 - Delphi (Object Pascal)
- 1995 - **JavaScript**
- 1995 - PHP
- 1996 - WebDNA
- 1997 - Rebol
- 1999 - D

Haskell

```
factorial n = if n < 2 then 1 else n *
factorial (n-1)
```

```
factorial 0 = 1
factorial n = n * factorial (n - 1)
```

Python

```
def factorial(n):

    if n == 0:
        return 1
    else:
        return n * factorial(n-1)
```

JavaScript

```
function rFact(num)

{

     if (num === 0)
       { return 1; }
     else
       { return num * rFact( num - 1 ); }
}
```

Statistics

- Q: Which is the most popular programming language?

- A: See http://langpop.com/ or http://githut.info

Programming paradigms

- Procedural languages (Imperative) - BASIC, C, ALGOL, COBOL, …
- Object-Oriented languages - Smalltalk, C++, Java, C#, …
- Declarative languages
  - Functional languages
  - Logic languages
- Multi-Paradigm languages - C#, Scala

Elements

Elements

- All programming languages have some primitive building blocks for the **description of data and the processes or transformations applied to them** (like the addition of two numbers or the selection of an item from a collection).

- These primitives are defined by **syntactic** and **semantic rules** which describe their structure and meaning respectively.

Syntax

- The syntax of a language describes the possible combinations of symbols that form a syntactically correct program.

- LISP syntax (RegEx and BNF):

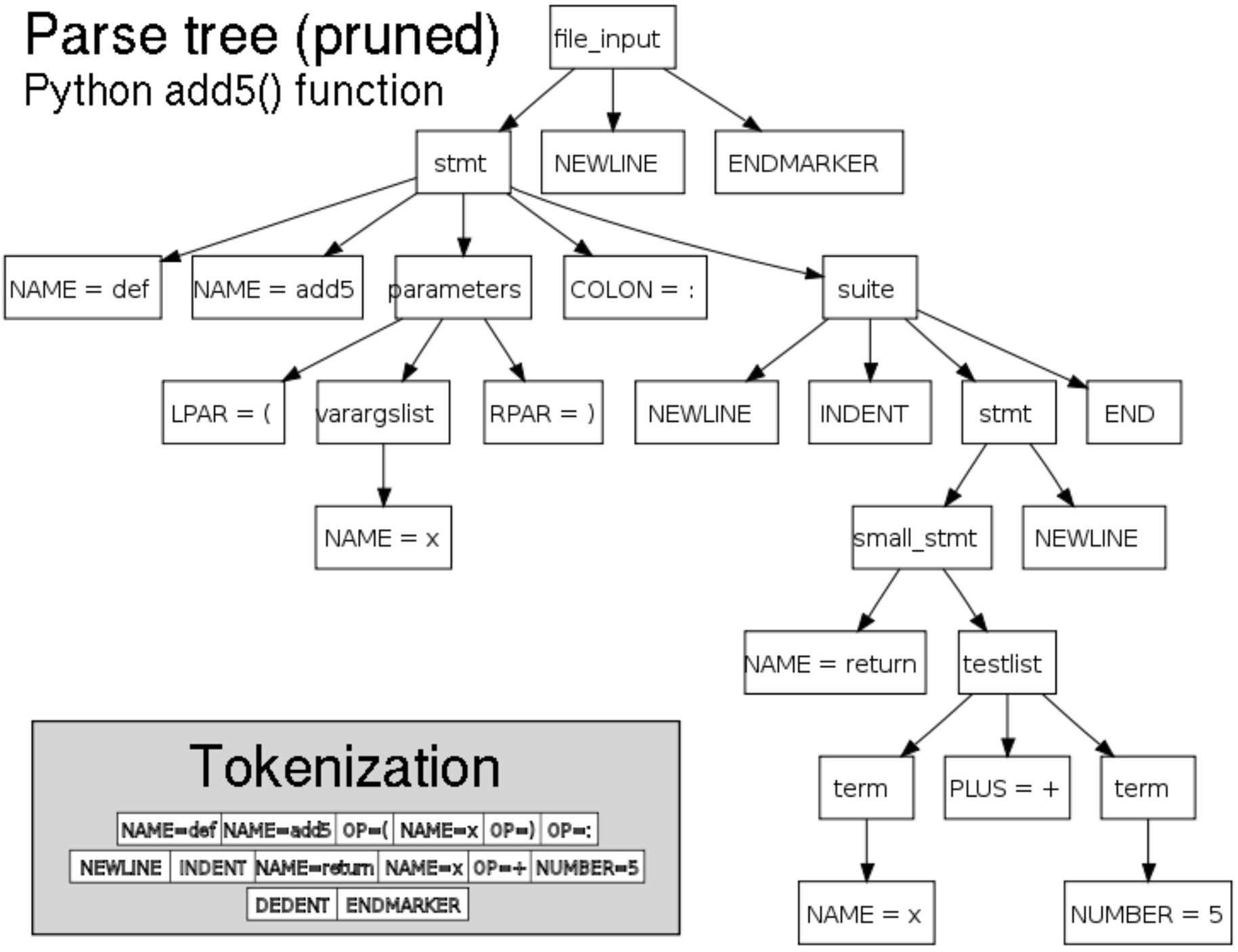
```
expression ::= atom | list

atom       ::= number | symbol

number     ::= [+-]?['0'-'9']+

symbol     ::= ['A'-'Z''a'-'z'].*

list       ::= '(' expression* ')'
```



"Python add5 parse". Via Wikipedia - http://en.wikipedia.org/wiki/File:Python_add5_parse.svg

Semantics

- The term Semantics refers to the **meaning** of languages, as opposed to their form (syntax).
    - Static semantics
        - Type system
        - Argument checking, Local variable checking (duplicity etc…)
    - Dynamic semantics
        - Operational semantics
        - Denotational semantics
        - Axiomatic semantics