LANGUAGE DESIGN PRINCIPLES

Based on Louden Lambert
Companion slides

EFFICIENCY

Execution efficiency

Static data types allow efficient allocation and access

Data declaration and subroutine calls known at compile time

Manual memory management avoids overhead of “garbage collection”

Constant sized array

Non recursive subroutine calls

Simple semantics allow for simple structure of running programs

PROGRAMMER'S EFFICIENCY

Writability or expressiveness (ability to express complex processes and structures)

- Structured control statements of Algol

Conciseness of syntax

- No explicit types
- Recursion
- Dynamic memory allocation

Dynamic data structures provide an extra layer of abstraction between the programmers and the machine

PROGRAMMER’S EFFICIENCY

Depends on the ease with which errors are detected and new features are added

```
if x>0:
     numsolns=2
     r1=sqrt(x)
     r2=-r1
elif x=0.0:
      numsolns=1
      r1=0.0
else
      numsolns=0
```

NOTE CONFLICTS WITH EFFICIENCY

Writability, expressiveness: no static data types (variables can hold anything, no need for type declarations). [harder to maintain]

Reliability, writability, readability: automatic memory management (no need for pointers). [runs slower]

- Isolate the modifications
- Easily spot and remove erroneous behavior

Maintainability

INTERNAL CONSISTENCY OF A LANGUAGE DESIGN: REGULARITY

Regularity is a measure of how well a language integrates its features, so that there are no unusual restrictions, interactions, or behavior.

- Fewer unusual restrictions on the use of particular constructs
- Fewer strange interactions between constructs
- Fewer surprises in general in the way language features behave

GENERALITY DEFICIENCIES

- avoids special cases whenever possible
- combines closely related constructs
- Procedures and functions   f(5)   16
- Operators
- Constants

In pascal, procedures can be passed as parameters, but no procedure variable.

Pascal has no variable length arrays –length is defined as part of definition (even when parameter)

```
Int f(int i) {
  const  int a;
   a=10;
   int b=a+i
    return (b);
}
```

```
Int j=f(5);
Class a,b;
A=b;
```

ORTHOGONALITY

Allows constructs to be combined in any meaningful way without any unusual restrictions or any unexpected behaviour arising as a result of the interaction between the constructs or the context of use

Closely related to simplicity - the more orthogonal, the fewer rules to remember.

- ❑ Function and Return types
- ❑ Placement of variable declarations
- ❑ Primitive vs reference types

FOR EXAMPLES OF NON-ORTHOGONALITY CONSIDER C++:

- We can convert from integer to float by simply assigning a float to an integer, but not vice versa. (not a question of ability to do – generality, but of the way it is done)
- Arrays are pass by reference while integers are pass by value.
- A switch statement works with integers, characters, or enumerated types, but not doubles or Strings.

JAVA VS PYTHON

Java	Python
Scalar types and object types in java	Object types

UNIFORMITY

- refers to the consistency of appearance and the behavior of language constructs
- Similar things look similar and dissimilar things look different
- Extra semicolon in C++
- Class A {….} a,b;
- Int a,b, c;
- Int fn(…){…}

- Using assignments to return a value in Pascal
  - Function f:Boolean;
  - begin
  - f:=true;
  - end;
- * in C

REGULARITY EXAMPLES FROM C++

Functions are not general(simplicity of environment).

Declarations are not uniform: data declarations must be followed by a semicolon, function declarations must not.

Lots of ways to increment – lack of uniformity (++i, i++, i = i+1)

i=j and i==j look the same, but are different. Lack of uniformity

WHAT ABOUT JAVA?

Are function declarations non-general?

- There are no functions, so a non-issue. (static methods)

JAVA REGULARITY, CONTINUED

Are some parameters references, others not?

- Yes: objects are references, simple data are copies.
- This is a result of the non-uniformity of data in Java, in which not every piece of data is an object.
- The reason is efficiency: simple data have fast access.

What is the worst non-regularity in Java?

- arrays. But there are excuses.

SECURITY

- closely related to reliability
- a language designed with an eye toward security both discourages programming errors and allows errors to be discovered and reported
- Security concern led to types, type checking, variable declarations
  - Idea is to “maximize the no. of errors that could not be made”
- too much security may compromise the expressiveness and conciseness of a language

SECURITY

- Security is to be balanced by expressiveness and generality
  - ML, HASKELL are functional in approach,
  - Allow multityped objects
  - do not require declarations
  - Yet perform static type checking
- strong typing, static or dynamic typing are only one component of type safety

TYPE SAFETY

Java/Python?LISP

- Semantically safe
  - Prevents programmers from compiling/executing any expressions that may violate the definition of the language
  - ArrayIndexOutOfBounds
- Automatic garbage collection prevents memory leaks

C/C++

- Not semantically safe
- Array index out of bounds may go unnoticed
- failure to recycle dynamic storage may result in memory leaks

OTHER DESIGN PRINCIPLES

Extensibility: allow the programmer to extend the language through adding features

- Types, operators
- Functions and procedures
- Packages and modules

Built-in features are also extended through new releases

- New features should be backward compatible

- LISP allows to extend the syntax and semantics of a language through macro

LISP

```
(do()
   ((= 0 b))
(let ((temp b))
    (setf b (mod a b))
     (setf a temp))

(while(> b 0)
   (let ((temp b))
      (setf b (mod a b))
       (setf a temp)))
```

```
(defmacro while (condition
&rest body)
   `(do()
         ((not, condition))
           , @body))
```

OTHER DESIGN PRINCIPLES (CONT.)

Preciseness: having a definition that can answer programmers and implementors questions. (Most languages today, but only one has a mathematical definition: ML).

If it isn't clear, there will be differences.

Example: Declaration in local scope (for loop) unknown/known after exit

Example: implementation of switch statement

Example: constants – expressions or not?

Example: how much accuracy of float?

FLON’S AXIOM

- An equal possibility of misusing language constructs
- Nested blocks, goto, procedures with too many parameters
- Low cohesion within a module and high coupling between modules
- Unrestricted use of pointers
- Programming languages may also help to stop misuse
  - Java –Pointers
  - Python indentation

TURING TAR-PIT

All computing languages or computers can compute anything in theory but nothing of practical interest is easy

- It is a place where a program has become so powerful, so general that the effort to configure it to solve a specific problem matches or exceeds the effort to start over and write a program that solves a specific problem

PYTHON

Bridging the gap between shell scripting and programming

- ❑ Simplicity
  - ❑ Small set of primitive operations and data types
- ❑ Regularity
  - ❑ Reference semantics
  - ❑ Each new unit of abstraction such as, functions, class, modules retains a simple regular syntax
  - ❑ Reduces the cognitive load on the programmer
- ❑ Extensibility
  - ❑ Libraries
- ❑ Static vs Dynamic Typing

PYTHON

Interactivity and Portability

- Short development cycle that provides immediate feedback with low I/O overhead
- Expressions and statements can be run in a Python shell
- Longer scripts are saved in a file
  - this experimental style supports iterative growth of reasonably large scale systems
- diversity of audience
  - byte code- Python Virtual Machine
  - Application specific libraries

DISCUSSION TOPICS

Provide examples of one feature that promotes and one feature that violates each of the design principles

- efficiency
- extensibility
- regularity
- security

- C, C++, Java, Python, Javascript

DISCUSSION TOPICS

- Difference between data structure and abstract data types
- An abstraction allows programmers to say more with less in their code. Discuss about its pros and cons with examples