Types

COP-3402 Systems Software
Fall 2018
Paul Gazzillo

Why Use Types?

To prevent errors during runtime

Typed vs Untyped

A type is

- a set of values
- and operations on those values

Typed languages restrict variable's range of values (Python, C, Java, etc)

Untyped languages do not (Lisp, assembly)

Safe vs Unsafe

Runtime errors are

- Trapped
  - terminated by machine, e.g., NULL-pointer error, divide-by-zero
- Untrapped
  - program continues, e.g.,

Safe languages prevent untrapped (and some trapped) errors

Static vs Dynamic Checking

When do checks happen

- Compile-time (static): C, Java
- Run-time (dynamically): Python, Java(?)

Weak vs Strong

Forbidden errors: all untrapped errors and some trapped errors

Good behavior: a program has no forbidden behaviors

- Strongly-checked: all programs have good behavior
- Weakly-checked: some programs violate safety

Table 1. Safety

	Typed	Untyped
Safe	ML, Java	LISP
Unsafe	C	Assembler

http://lucacardelli.name/Papers/TypeSystems.pdf

Demo: Python vs C

Static Type Checking

- Record (or infer) types of identifiers in symbol table
- Traverse tree (AST)
- Check identifiers used in
  - Arithmetic operators
  - Function calls
  - Assignments
- Lookup type in symbol table

Demo: Static Checking an AST