Programming languages Overview

* Brief History of:

Lisp, Cobol, Algol, APL, PL/I, Snobol, Simula, Pascal, C, Prolog, Small talk, C++, Java, C#, Perl, Python, Ruby, Javascript, Go, Rust, Swift

* Universal Client Language

- as browsers become universal client, programming languages used in most browsers Javascript is becoming a universal client language.

* Universal Target Architectures

- Previously used general purpose hardware architectures are shrinking: x86, ARM, RISC-V, etc (compiled in C)

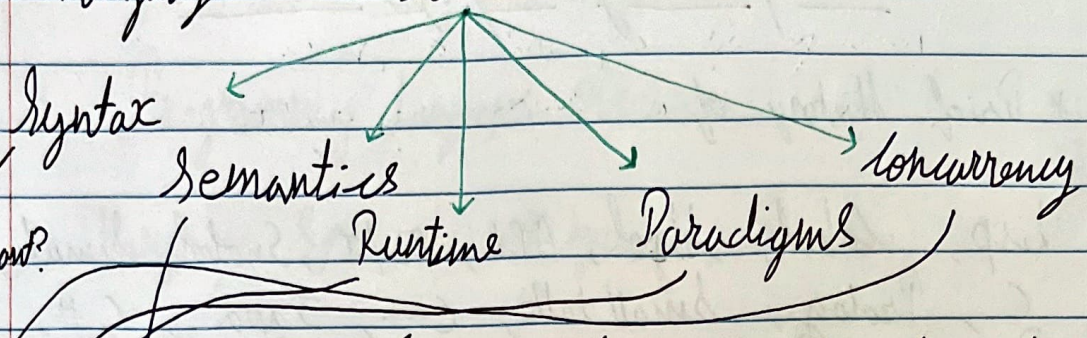
- Current new languages often target

1. Common-Language Runtime CLR on Windows
   Eg: F#, Iron Ruby, Iron Python

2. JVM - Java Virtual Machine
   Eg: Scala, Jruby, Jython, Clojure

- Browser VMs targeted using Emscripten or Zig

* Language dimensions



White space significant?

→ is the language type safe? arithmetic safe? scoping rule? support mutability? first class? encapsulation? memory visible?

Runtime → does it provide automatic memory management? (garbage collection)
is it tied to a particular OS architecture?
does it have extended library support?
can it replace a running program
Eg: f1.java
eval ("1+2")
→ 3 "string is being evaluated"

Paradigms → most languages are imperative (includes OOP lang)
declarative paradigms include:
Functional, Logic based, constraint based, Dataflow
(Haskell) (prolog) (spreadsheet)

Concurrency → is it part of lang. or only supported via library?
what are safety and liveness properties?
Does the lang. support distributed prog. across network?

3.

* C Compilers
  - usually set up as a driver program which runs other seprate programs

For eg: With Gcc

1. Compiler cc1 : compiled *.c file to *.s assembly file

2. Assembler : assembles *.s file to *.o object file (certain import symbols)

3. Linker ld : links several object files together with libraries to produce an executable

Static linker
- links in all code
produces fully self contained executable

Dynamic linker
- links in all object files but only links in references to the libraries (at runtime)

- Most modern systems use Dynamic linked libraries
  dis adv : DLL Hell
  adv : Smaller executables & possibility of having multiple concurrently executing programs share the same library code in memory (Dynamic shared objects)

4.

* Demo program
  - main.c → converts string to int
    f1.c → returns a+b
    f2.c → returns a * b

Compiler Flow