Programming Languages (5) Using Libraries

Kenjiro Taura

Concepts to learn

using a library entails different procedures depending on how "embedded" it is into the language

- some libraries are *"builtin"*
  - automatically available in every program
- some libraries are *"standard"*
  - you need to master how to refer to names in it
  - you say "import" or "use" it and/or use prefixes to refer to names in it
  - installed with the language
- some libraries are *"external"*
  - you may have to install it
  - you may have to tell the compiler where it is

Importing a library to your program

- OCaml :
  - *module-name.name-in-the-module*
  - `open` *module-name* and *name-in-the-module*
- Julia :
  - `import` *module-name* and *module-name.name-in-the-module*
  - `using` *module-name* and *name-in-the-module*
- Go :
  - `import` "*module-name*" and *module-name.name-in-the-module*
- Rust :
  - *crate-name::module-name::module-name::...::name-in-the-module*
  - `use` *crate-name*`::`*module-name*`::`*module-name*`::...::`*name-in-the-module* and *name-in-the-module*
  - anywhere between the two

Build system

many languages have “build system” to help you use external libraries

- OCaml : dune `https://dune.build/`
- Go : go itself is a build system
- Rust : cargo

Repository of libraries

- master how to get information you need (names of functions, their types, etc.) from those repositories
- is it builtin? standard? external?
- OCaml : opam `https://opam.ocaml.org/`
- Julia : Julia packages `https://julialang.org/packages/`
- Go : `https://pkg.go.dev/`
- Rust : `https://crates.io/`