100 days of code. Day 1 with Julia

Sergio-Feliciano Mendoza-Barrera

December 2, 2020

The today's topic is metaprogramming basics.

Metaprogramming in Julia

Julia code can be represented as a data structure of the language itself. This allows a program to transform and generate its own code (Lauwens & Downey, 2020, p. 204).

Expressions

Every Julia program starts as a string

```
prog = "1 + 2"
ex = Meta.parse(prog)
```

we can get the type of the variable with `typeof` as usual and `dump` the tree structure[1].

[1] The `dump` function displays `expr` objects with annotations.

```
typeof(ex)
dump(ex)
```

```
Expr
Expr
  head: Symbol call
  args: Array{Any}((3,))
    1: Symbol +
    2: Int64 1
    3: Int64 2
```

Expressions can be constructed directly by prefixing with : inside parentheses or using a `quote` block

```
ex = quote
    1+2
end
```

Now, *Julia can evaluate an expression object* using `eval`

```
eval(ex)
```

```
3
```

Every module has its own `eval` function that evaluates expressions in its scope[2].

[2] *WARNING*: When you are using a lot of calls to the function `eval`, often this means that something is wrong. `eval` is considered *evil*.

Bibliography

Lauwens, B. & Downey, A. B. (2020). *Think Julia: How to Think Like a Computer Scientist* (1st ed.). O'Reilly. Retrieved from https://benlauwens.github.io/ThinkJulia.jl/latest/book.html.