Week 12 Studio
MeTaCiRcUlAr EvAlUaToR

CS1101S AY21/22 Semester 1
Studio 05E

Yan Xiaozhi (David)
@david_eom
yan_xiaozhi@u.nus.edu

1 Nov 2021

Admin

- Contact tracing (QR code + class photo)
- End of syllabus! (but PE and finals are coming)
- Mastery Check 2
  - Environment model, memoisation, streams, preferably after 8 Nov (Mon)
- Week 13 Studio plans
  - Prepare questions!
- Miscellaneous: CP3108, avenger…

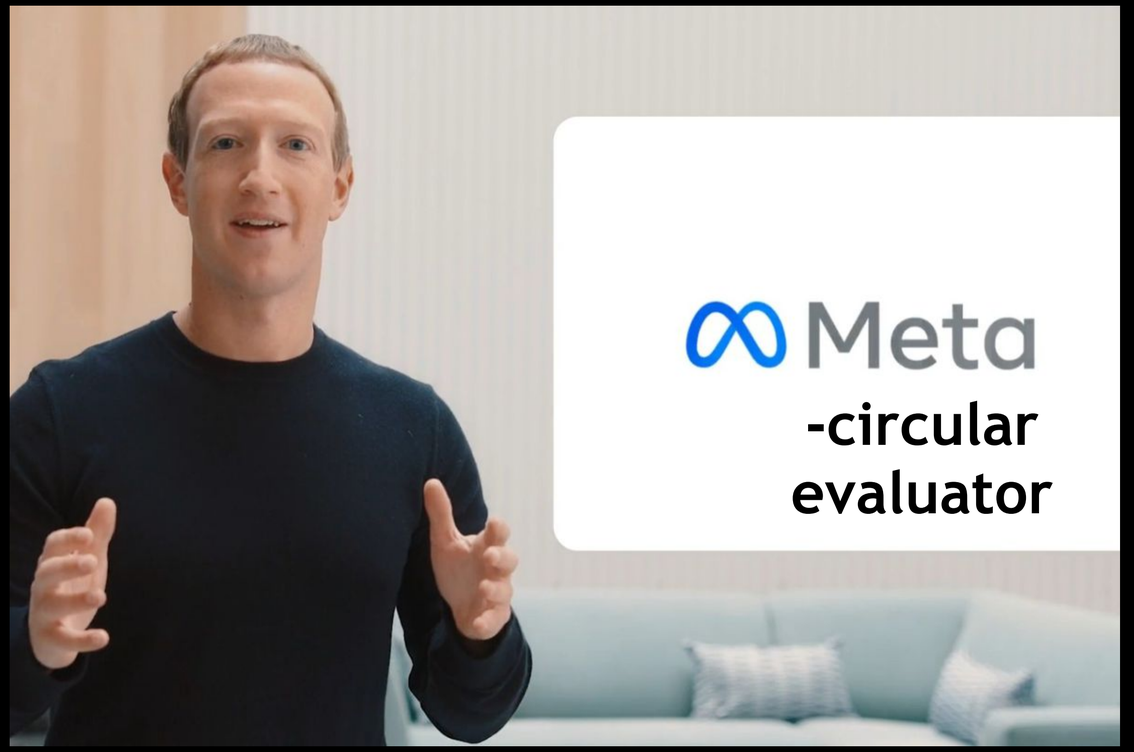
Recap

I am a bit lost 😩
(but tbh since when am I not)

Metacircular Evaluator

What Is It?

- A Source programme
- That runs a Source programme 🌚
- Allow us to better understand Source and how it works
- CS4215 Programming Language Implementation



eval(*expression*[, *globals*[, *locals*]])

The arguments are a string and optional globals and locals. If provided, *globals* must be a dictionary. If provided, *locals* can be any mapping object.

The *expression* argument is parsed and evaluated as a Python expression (technically speaking, a condition list) using the *globals* and *locals* dictionaries as global and local namespace. If the *globals* dictionary is present and does not contain a value for the key `__builtins__`, a reference to the dictionary of the built-in module `builtins` is inserted under that key before *expression* is parsed. That way you can control what builtins are available to the executed code by inserting your own `__builtins__` dictionary into *globals* before passing it to `eval()`. If the *locals* dictionary is omitted it defaults to the *globals* dictionary. If both dictionaries are omitted, the expression is executed with the *globals* and *locals* in the environment where `eval()` is called. Note, *eval()* does not have access to the nested scopes (non-locals) in the enclosing environment.

The return value is the result of the evaluated expression. Syntax errors are reported as exceptions. Example:

```
>>> x = 1
>>> eval('x+1')
2
```

Metacircular Evaluator

What Do You Have To Do?

- Define the structure of the language (already done for you ✅)
  - Decide on the syntax
- Parse the programme (already done for you ✅)
  - Convert your programme string into an abstract syntax tree
- Define a method to process the tree structure (Your job!!!)
  - Understand the structure of parsing
  - Decide on changes to make to the MCE

Parsing

What Is It?

- A parser is a compiler or interpreter component that breaks data into smaller elements for easy translation into another language.
- `parse()` in Source
- `display_list(parse("const foo = x => x + 1;”));`
- ```
  list("constant_declaration",
    list("name", "foo"),
    list(“lambda_expression",
      list(list("name", "x")),
      list("return_statement",
        list(“binary_operator_combination",
          "+", list("name", "x"), list("literal", 1)))))
  ```
  ```

Metacircular Evaluator

Tags

- First item of the list is always a tag
  - literal
  - name
  - assignment
  - return statement
  - sequence
  - compound function
  - ...

- `is_<tag>(component)`
- String as head, relevant information as tail
- Tags are used to:
  - Specify what the tail list is
  - What is the structure of the list
  - How to evaluate this list

Metacircular Evaluator

What Should I Do?

- Are you really gonna read through the 500 LoC?
  - Not a lot TBH
- Brownfield project for software engineering
- When you do your internship at look at the company codebase
- Make use of your “Ctrl+B” / “Cmd+B”!

Metacircular Evaluator

Why Do I Need To Learn This?

- Have to admit it’s not tested (esp online)
- It’s quite fun
- Maybe you wanna try out programming language focus area?
- All programming languages are just programmes that can run some other programmes!
  - Python is written in C
  - Essentially a C programme of MCE that evaluates Python strings

Any Questions?
(hopefully not)