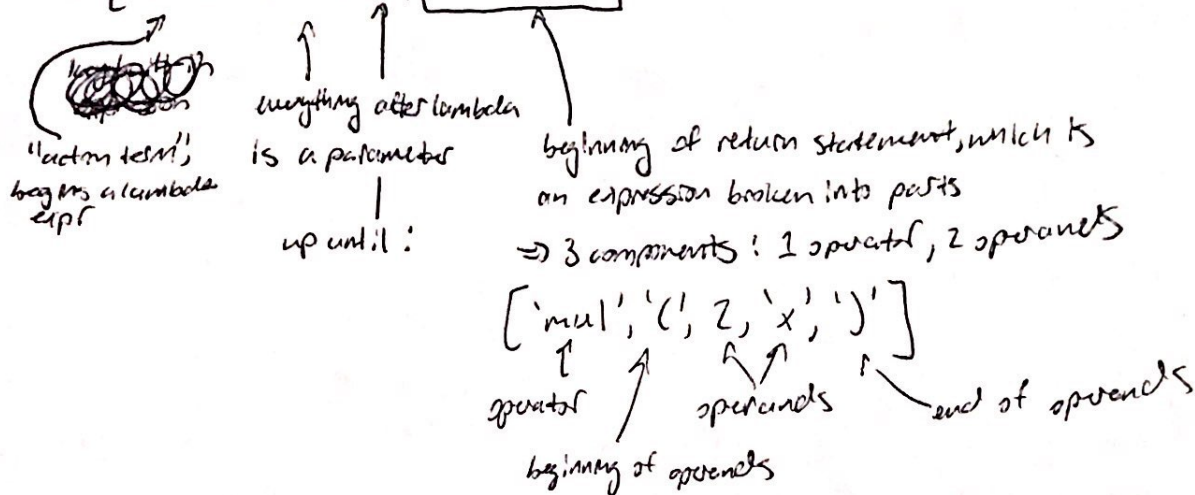


Read

lexer: turns string into "tokens"

- takes a string and breaks it into all its "parts"

Ex: `lambda x: mul(2, x)`
⇒ `['lambda', 'x', ':', 'mul', '(', '2', 'x', ')']`



parser: takes all the key values and turns them into data structures so it's easy to evaluate them

i.e. lambda means create a lambda function

- unique eval method
- functions apply things together

i.e. `x` is a name, so we need to define how to know what it represents, lookup in environments to evaluate it

i.e. `mul` is a primitive function

- eval takes in 2 arguments
- apply multiplies them

need to store these effects into data structures (expr.py)

Eval

eval methods in expr.py (classes)

- diff eval for names, lambda expr., & call expr.

Print

print return values

Loop

implemented in repl.py