UNIVERSITY OF
Southampton
School of Electronics
and Computer Science

COMP2212
PROGRAMMING LANGUAGE CONCEPTS

Dr Julian Rathke

LEXING CONCEPTS

BASIC CONCEPTS

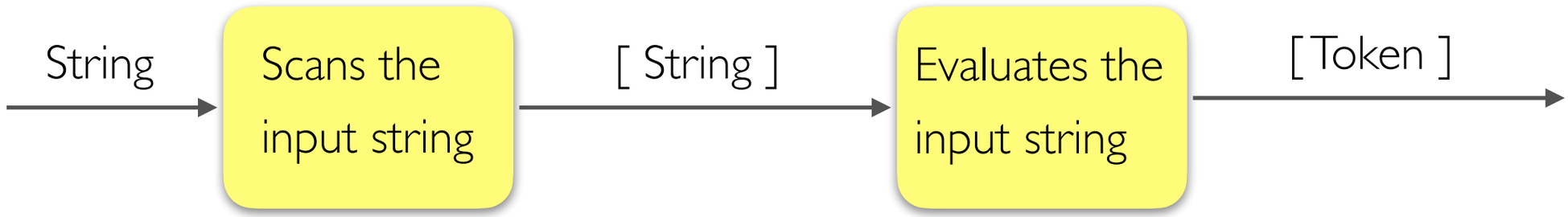
- We've learned that lexical analysis or **lexing** is the process of converting a source input string in to a sequence of tokens
- A **lexeme** is a pattern of characters in the input string that are considered meaningful in the given programming language
  - these may be defined by regular expressions
- A **token** is a lexeme that has been identified and "tagged" with a meaning and possibly including a value
  - e.g. **`while`** is a lexeme from the characters 'w', 'h', 'i', 'l', 'e' identified as the while-command token. Or **`true`** is a boolean token with value "true".

SCANNING AND EVALUATION

- Lexemes are identified use a **scanner** that does the pattern matching using "*maximal munch*"
  - For example, for a pattern [1-9]+ and the input " … 456,234 …"
    - there are lexemes "456" and "234" matching that pattern
  - For the pattern [1-9]+,[1-9]+ we would have "456,234" as a lexeme
- Tokens are created using an **evaluator** whose job it is to analyse the lexemes, tag them appropriately and identify any associated value.
  - For example, for a pattern [1-9]+ and the input " … 42 … ", as above we have the lexeme "42" and the evaluator would identify this as a "integer" token with value being the actual integer 42.
- In Haskell, the function `read` is very useful for evaluation

HOW TO WRITE A LEXER : DON'T

- In order to write a lexer then we need to have code that



- This code can be reasonably generic if
  - We know what the lexemes look like
  - Know which lexemes correspond to which tokens and how to associate values with them
- If we have a means of describing these things then we could just automatically generate the code to do the scanning and evaluating
- This is great for code re-use leading to productivity gains and robustness of code

LEXER GENERATORS

- A lexer generator is a software tool that, given an input that describes the lexemes and what tokens they produce, will generate code for you that performs lexical analysis.
- Lexer generators are widely and freely available for most programming languages
- The majority of them are based around the C/Unix based tools called lex and the variant of this called flex.
- e.g, JLex is a Java version of lex, ply.lex is a Python version etc
- The input languages for these versions are mostly similar to lex so learning to use one version will make it easy to learn a version in another language also.
- We will be using the Haskell version of lex, which is called "Alex"