# Programming Language (7)
lexers and parsers

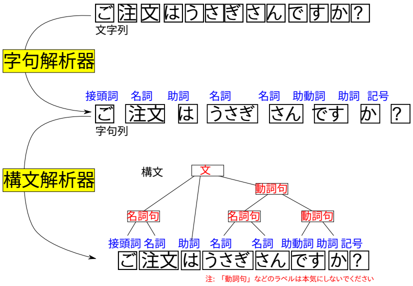
田浦

## Introduction

- All programming language implementations first read a program and check its grammar
  - lexical analyzer ( *“lexer”* or *“tokenizer”*)
  - syntax checker ( *“parser”*)
- they are necessary not only in programming language implementations but in many other circumstances
  - web pages (HTML or XML)
  - CSV, SVG, . . . files . . .
  - config files of software . . .
- it’s an important skill to be able to make them quickly
  - you’d better not process strings in an ad-hoc manner
  - there are useful tools to make them (*parser generators*)
  - it never hurts to have an experience with them

# Lexer and parser

- lexer ≈
  - converts a sequence of "characters" → a sequence of "tokens" (≈ words)
  - rejects when characters do not constitute a valid token
- parser ≈
  - converts a sequence of "tokens" → a "sentence" (expression, statement, whole program, etc.)
  - rejects tokens that constitute a valid sentence



ご注文はうさぎさんですか?
文字列
字句解析器
接頭詞 名詞 助詞 名詞 名詞 助動詞 助詞 記号
ご 注文 は うさぎ さん です か ?
字句列
構文解析器
構文
文
動詞句
名詞句
名詞句
動詞句
接頭詞 名詞 助詞 名詞 名詞 助動詞 助詞 記号
ご 注文 は うさぎ さん です か ?
注: 「動詞句」などのラベルは本気にしないでください

## How to define a token and a sentence?

- normally, we define
  - tokens: by *regular expression (regex)*
  - sentences: by *context free grammar (CFG)*
- there are tools that generate lexers and parsers from their declarative descriptions (*lexer/parser generators*)
- “practice makes perfect.” Let’s see it working

## lexer/parser generators

- there are many tools for many languages
  - C/C++ : lex (flex) and yacc (bison)
  - OCaml : ocamllex and ocamlyacc (menhir)
  - Python : a whole bunch of tools, e.g., in `https://wiki.python.org/moin/LanguageParsing` and `https://tomassetti.me/parsing-in-python/`
- I will give you a parser code that converts source language into XML, which you can then read using the XML library in the language you are using
- the parser will be written in Python using Tatsu
- details to be announced later (hopefully in a few days . . . )