# Lark Cheat Sheet

by erezsh via cheatography.com/61630/cs/15903/

## Lark Options

| | |
|---|---|
| `parser ="ea rle y"` | **Earley** - default |
| `parser ="la lr"` | **LALR(1)** |
| `debug=True` | Enable debug prints |
| `lexer="standard"` | Revert to simple lexer |
| `ambiguity='explicit'` | Return all derivations for Earley |
| `start= " foo "` | Set starting rule |
| `cache=True` | Enable grammar caching |
| `transf orm er=...` | Apply transformer to tree (for LALR) |
| `propagate_positions` | Fill tree instances with line number information |
| `maybe_placeholders` | `[]` returns `None` when not matched |
| `keep_a ll_ tok ens = True` | Don't remove unnamed terminals |
| `postlex` | Provide a wrapper for the lexer |
| `tree_class` | Provide an alternative for `Tree` |
| `regex=True` | Use the `regex` module |

## Tree Reference

| | |
|---|---|
| `tree.data` | Rule name |
| `tree.c hildren` | Rule matches |
| `tree.meta` | Positional information, if enabled |
| `print( tre e.p ret ty())` | |
| `tree.i ter _su btr ees ()` | Iterate all subtrees |
| `tree.find_data("foo")` | Find by rule |
| `tree.f ind _pr ed(...)` | Find by predicate |
| `tree1 == tree2` | |

## Token Reference

| | |
|---|---|
| `token.type` | Terminal name |
| `token.v alue` | Matched text |
| `token.p os _in _stream` | Index in source text |
| `token.line` | |
| `token.c olumn` | |
| `token.e nd _line` | |
| `token.end_column` | |
| `token.e nd_pos` | |
| `len(token)` | |

Tokens inherit from `str`, so all string operations are valid (such as `t oken.u pper()`).

## Grammar Definitions

| | |
|---|---|
| `rule: ...` | Define a rule |
| `TERM: ...` | Define a terminal |
| `rule.n: ...` | Rule with priority n |
| `TERM.n: ...` | Terminal with priority n |
| `// text` | Comment |
| `%ignore ...` | Ignore terminal in input |
| `%import ...` | Import terminal from file |
| `%declare TERM` | Declare a terminal without a pattern (used for postlex) |
| `t{p1, p2}: ...` | Define template |
| `rule: t{foo, bar}` | Use template |

**Rules** consist of values, other rules and terminals.
**Terminals** only consist of values and other terminals.

By **erezsh**
cheatography.com/erezsh/

Published 26th May, 2018.
Last updated 29th September, 2020.

# Lark Cheat Sheet

by erezsh via cheatography.com/61630/cs/15903/

## Grammar Patterns

| | |
|---|---|
| `foo bar` | Match sequence |
| `(foo bar)` | Group together (for operations) |
| `foo \| bar` | Match one or the other |
| `foo?` | Match 0 or 1 instances |
| `[foo bar]` | Match 0 or 1 instances |
| `foo*` | Match 0 or more instances |
| `foo+` | Match 1 or more instances |
| `foo~3` | Match exactly 3 instances |
| `foo~3..5` | Match between 3 to 5 instances |

## Terminal Atoms

| | |
|---|---|
| `" str ing "` | String to match |
| `" str ing "i` | Case-insensitive string |
| `/regexp/` | Regular Expression |
| `/re/imslux` | Regular Expression with flags |
| `" a".." z "` | Literal range |

## Tree Shaping

| | |
|---|---|
| `rule: " foo " BAR` | "foo" will be filtered out |
| `!rule: " foo " BAR` | "foo" will be kept |
| `rule: /foo/ BAR` | /foo/ will be kept |
| `_TERM` | Filter out this terminal |
| `_rule` | Always inline this rule |
| `?rule: ...` | Inline if matched 1 child |
| `foo bar -> new_name` | Rename this derivation |

**Rules** are a branch (node) in the resulting tree, and its children are its matches, in the order of matching.
**Terminals** (tokens) are always values in the tree, never branches.
**Inlining rules** means removing their branch and replacing it with their children.

## Examples

```
// Define template for comma-separated list
cs_lis t{i tem}: item ("," item)*
// Use template to make a list of numbers
number _list: cs_list{ number }
// Example of a terminal for a Python comment
PY_COM MENT: /#[^\n]*/
// Example of a terminal for C comment
C_COMMENT: "/" /.?/s " */"
```