木兰编程语言工具设计与实现中的巧思

功能设计

```
usage: ulang-0.2.2.exe [-apbcidsDth] input_file
Options and arguments:
 --dump-ast,          -a   dump ast info
 --dump-python,       -p   dump python source code
 --dump-blockly,      -b   dump blockly xml (experimental)
 --dump-bytecode,     -c   dump donsok bytecode (experimental)
 --python-to-ulang,   -s   convert python to ulang
 --debug,             -D   debug with Pdb (experimental)
 --interact,          -i   inspect interactively after running script
 --disassemble,       -d   disassemble the python bytecode
 --exec-code=<code>   -e   run code from cli argument
 --show-backtrace,    -t   show backtrace for errors
 --version,           -v   show the version
 --help,              -h   show this message
```

功能设计

```
usage: ulang-0.2.2.exe [-apbcidsDth] input_file
Options and arguments:
 --dump-ast,          -a   dump ast info
 --dump-python,       -p   dump python source code
 --dump-blockly,      -b   dump blockly xml (experimental)
 --dump-bytecode,     -c   dump donsok bytecode (experimental)
 --python-to-ulang,   -s
 --debug,             -D
 --interact,          -i
 --disassemble,       -d
 --exec-code=<code>   -e
 --show-backtrace,    -t
 --version,           -v
 --help,              -h
```

```
使用方法：木兰 [-树p码兰调交反执溯版助] 源码文件

选项：
  --语法树              -树    将木兰源码转换为 Python 语法树
  --木兰变python        -p     将木兰源码转换为 Python 源码
  --生成字节码          -码    将木兰源码转换为 donsok 字节码（实验性）
  --python变木兰        -兰    将 Python 源码转换为木兰源码
  --调试                -调    使用 Pdb 环境调试代码
  --交互                -交    以交互式审查脚本
  --反汇编              -反    将生成的 Python 字节码反汇编
  --执行代码=<代码>     -执    执行来自命令行参数的代码
  --显示回溯            -溯    显示异常的栈回溯信息
  --版本                -版    显示版本
  --帮助                -助    显示帮助信息
```

语法设计一二：多句 lambda

- 感谢第一位合作者，第一时间指出这一特色

```
最近 = min(map(
        数 -> {
            差 = 数 - $想的
            return 差 > 0 ? 差 : -差
        }, $历史))
```

可用 $ 代替 self

```
type 代码段 {
  func $代码段(词性, 行, 起, 止) {
    self.词性 = 词性
    self.行 = 行
    self.起, self.止 = 起, 止
  }

  func 开始(self) {
    return str(self.行) + "." + str(self.起)
  }

  func 结束(self) {
    return str(self.行) + "." + str(self.止)
  }

  func __repr__(self) {
    return "[" + self.词性 + " 行:" + str(self.行) + " 列:" + str(self.
起) + "~" + str(self.止) + "]"
  }
}
```

```
type 代码段 {
  func $代码段(词性, 行, 起, 止) {
    $词性 = 词性
    $行 = 行
    $起, $止 = 起, 止
  }

  func $开始 {
    return str($行) + "." + str($起)
  }

  func $结束 {
    return str($行) + "." + str($止)
  }

  func $__repr__ {
    return "[" + $词性 + " 行:" + str($行) + " 列:" + str($起) + "~" + s
tr($止) + "]"
  }
}
```

省略乘号设计及副效应

- 示例:

```
> 边长=5; 周长=4边长; println(周长)
20
```

- 副效应1

```
> a=1; println(a[0])
😨 整数变量不支持按索引取项，见第1行
> println(1[0])
[0]
```

- 副效应2

```
> println(0xf)
15
> println(0xg)
😨 请先定义‘xg’再使用，见第1行
```

牵一发动全身：大括号与冒号

- {} 代替冒号加缩进 Python

```
if True:
    print()
```

木兰

```
if true { println() }
{ println() } if true
```

- 匿名函数体支持多句，用 ->

```
最近 = min(map(
        数 -> {
            差 = 数 - $想的
            return 差 > 0 ? 差 : -差
        }, $历史))
```

- 支持 ? : 三段表达式
- 冒号指定类型

```
type 猜数字 : cmd.Cmd {
```

- 字符串插值用反引号而非大括号 Python

```
a=3; print(f'{a}小')
```

木兰

```
a=3; println(`a`小')
```

- 空字典初始化用 {:}

实现

- 用 RPly 实现？：三段表达式

```
@分析器母机.语法规则(语法.三元表达式.成分(语法.表达式, 问号, 语法.表达式, 冒号, 语法.表达式))
def 三元表达式(self, 片段):
    return 语法树.如果表达式(
        条件=片段[0],
        主体=片段[2],
        否则=片段[-1],
        片段=片段)
```

- 性能潜力待研究 https://zhuanlan.zhihu.com/p/601321957

需求揣摩

- 功能虽有缺陷，基本满足入门教学需要
- 功能需求之外，还有哪些非功能需求？