Pythy语言项目概述

——软件工程选题报告

2017HYSE04小组

一. Pythy语言文法的BNF定义

为了在表达式比较清晰，下面的定义不考虑表达式运算符的优先级。

```
//语句
Statement ->
            Where_Statement
           |If_Statement
           |Else_Statement
           |Assign_Statement
           |Get_Statement
           |Echo_Statement
//where
Where_Statement ->
            <Where> Expression ":"

//if
If_Statement ->
            <If> Expression ":"

Else_Statement ->
            <Else> ":"

//赋值
Assign_Statement ->
            <Name> "=" Expression

//读取数据
Get_Statement ->
            Console_Get_Statement
           |File_Get_Statement
           |File_Get_Declaration
           |File_Get_Have

//从标准输入中读
Console_Get_statement ->
            <Get> <Name>

//从文件中读一行数据，作用和get一样
File_Get_Statement ->
            <Special_Get> <Name>

//声明一个文件迭代器
File_Get_Declaration ->
            <Special_Get> <From> <FileName>
```

```
//判断文件迭代器是否结束
File_Get_Have ->
            <Have> <Special_Get>

//表达式
Expression ->
            Logical_Expression
           |Operational_Expression
           |Primary_Expression
           |"(" Expression ")"     //括号
//运算表达式
Operational_Expression ->
            Operational_Expression "+" Operational_Expression
           |Operational_Expression "-" Operational_Expression
           |Operational_Expression "*" Operational_Expression
           |Operational_Expression "/" Operational_Expression
           |Operational_Expression "//" Operational_Expression
//逻辑运算表达式
Logical_Expression ->
            Expression "and" Expression
           |Expression "or" Expression
           |"not" Expression
           |Expression "==" Expression
           |Operational_Expression ">" Operational_Expression
           |Operational_Expression "<" Operational_Expression
//基本类型
Primary ->
            <Integer>
           |<Float>
           |<Name>
           |<Bool>

//token定义
<token> Get         : "get"
<token> From        : "from"

<token> Where       : "where"
<token> If          : "if"
<token> Else        : "else"

<token> Integer     : [0-9][0-9]*
<token> Float       : [0-9][0-9]* "." [0-9]*
<token> Name        : [a-zA-Z][a-zA-Z0-9_]*
<token> Bool        : "False" | "True"

<token> Have        : "have"
<token> Special_Get : "get_" [a-zA-Z0-9]*  //Special_Get语句定义的读文件迭代器
<Token> FileName    : " (.)* "    //一个被双引号引起来的字符串，里面是文件名
```

二. 需求分析之编程语言属性解析

1. 基本数据类型

Pythy语言目前支持的数据类型有三种，分别是整数，浮点和布尔型。

布尔型在赋值时直接写True和False即可

```
>>> x = True
```

由于Pythy没有类型申明语句，我们在进行整数和浮点的赋值时，要特别写明右值

```
>>> x = 1    #  1::int
>>> y = 1.0  #  1.0::float
```

---

2. 运算表达式及逻辑表达式

Pythy语言中，表达式分为运算表达式和逻辑表达式两种。

- **运算表达式**

  运算表达式实现的功能就是四则运算，包括 + , - , * , / , // 。其中 / 代表整数除法，即其运算结果一定是整数，如果是非整数的结果会被强制转换为整数。//代表浮点数除法，即其结果一定是浮点数。

- **基本逻辑表达式**

  基本的逻辑表达式实现的功能就是逻辑判断，包括not, == , < , >，它们的结果**一定**是一个布尔类型的。

  首先，Pythy语言和Python一样，任何的数据都具有一个对应的布尔值,对于整型和浮点，只有0的布尔值是False，其他的都是True. 类似的如果Pythy之后实现数组或串，那么只有空数组和空串的布尔值是False。

  因此有了上面的基础，not运算的结果就是先对一个数据求布尔值，然后再取非：

  ```
  >>> not 0
  True

  >>> not 1.0
  False
  ```

  ==运算判断左右两式是否相等，若相等则返回真，否则返回假;这里说的相等可能是类型转换后的结果。

  ```
  >>> 1 == 2
  False

  >>> 1 == 1
  True
  ```

```
>>> x = 1
>>> x == 1
True

>>> 1 == 1.0
True
```

< 判断左式是否小于右式，若小于则返回真，否则返回假；> 的运算逻辑则和 < 相反。

```
>>> 1 < 2
True
>>> 1 > 2
False
```

- **特殊的逻辑表达式and和or**

  需要特别说明的是and与or运算。为了保持和Python的一致，我们定义and与or运算的运算结果如下：

  - 设有两变量X、Y。
  - 在表达式X and Y中，若X与Y皆为真，则返回Y；否则返回第一个判断为假的变量的值。
  - 在有表达式X or Y中，若X与Y皆为假，则返回Y；否则返回第一个判断为真的变量的值。

```
>>> 0 and 1
0

>>> 1 and 2
2

>>> 1 or 0
1

>>> 0 or 1
1
```

---

3. 条件判断,循环，赋值语句

- **Pythy遵循缩进规则**

  Pythy和Python,使用缩进的方式表示if-else while等语句的**语句体(body)**

  为了解释缩进，一个用来求两数最小值的Pythy程序如下：

```
while True:
    if x < y:
        res = x
    else:
```

```
        res = y

echo res
```

接下来具体地说明这些语句的执行逻辑，以便开发时参考

- **条件判断：if-else语句**

  if语句的功能是实现一个条件分支，后面需要接上一个条件表达式。如果表达式运算结果（的布尔值，下同）为真，则顺序执行后续语句并忽略else；如果结果为假，则跳转到对应的else位置执行后续语句。

- **循环：while语句**

  while语句的功能是实现条件循环，每次执行while语句时，先对括号内的条件表达式求真假值，如果为真，则顺序执行直到while循环体结束，再回到while重新判断条件；如果为假，则忽略整个while循环体执行后续语句。

- **赋值：=**

  赋值语句实现的是将等号右边的表达式的值复制给等号左边的变量。

---

4. 输入输出

由于在本阶段的Pythy语言没有实现函数，因此我们规定了输入输出语句

- **输出：echo语句**

  echo语句类似命令行中的echo,先看一个小例子：

```
>>> echo 1+3
4

>>> x = 1 + 3
>>> echo x
4
```

  可以看到echo语句在执行过程中先对表达式求值，接下来将求出来的值在标准输出（Console）中打印出来。

- **控制台输入：get语句**

  get我们用get语句来从用户标准输入中获得一个值，并且将他赋值给一个变量。下面是get语句的一个例子

```
>>> get x
1            //这里是用户输入
>>> echo x
1
```

- **文件输入：Special Get**

  这里的文件输入是我们的一个创新点，我们希望Pythy语言不仅仅是一个小的玩具，而是能在生活中给我们带来一些小小的便利，这里用一个计算小明同学上学期加权平均分的Pythy程序为例来解释文件输入：

  首先小明同学有5门课的成绩，按照每行一个成绩存在score.txt中：

  ```
  90
  97
  89
  85
  93
  ```

  同样，他的这五门课的对应学分存在credit.txt

  ```
  4
  3
  5
  2
  2
  ```

  这是对应的Pythy程序：

  ```
  get_score from "score.txt"  #定义文件迭代器
  get_credit from "credit.txt"

  sum = 0             #总加权分数
  total_credit = 0    #总学分数

  while have get_score:  #have get_score 判断文件是否还有内容

      get_score score   #文件迭代器从文件中获取一行并赋值
      get_credit credit

      sum = sum + score * credit        #增量
      total_credit = total_credit + credit

  echo sum     #输出
  ```

三. 需求分析之集成开发环境

1. 编辑功能

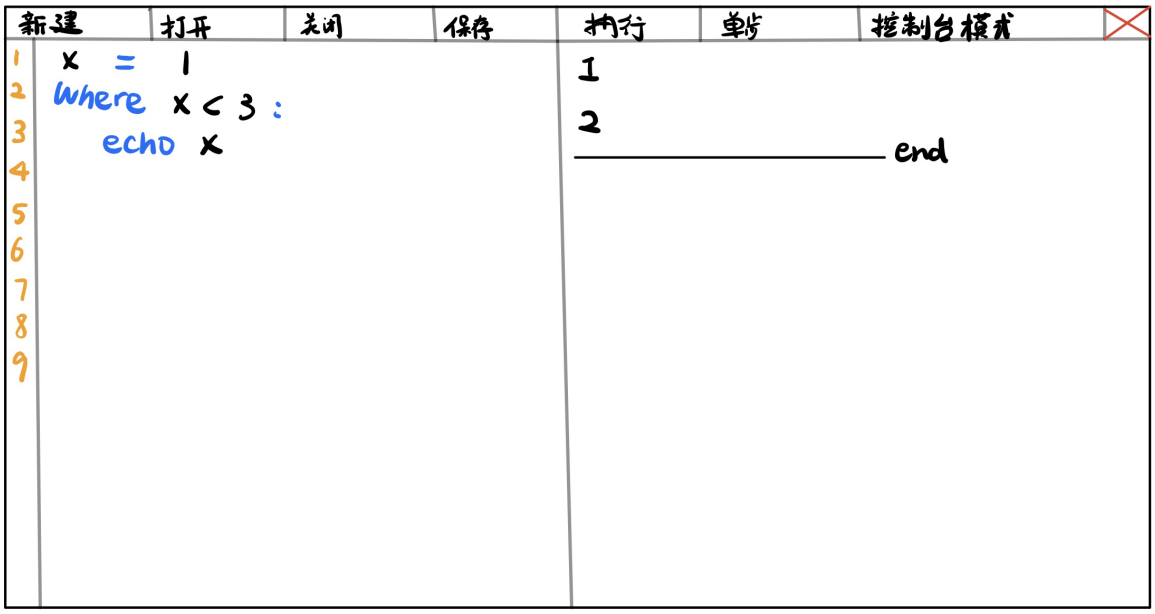
集成开发环境的编辑功能需要实现基本的文件操作，即新建、打开、关闭、保存，并且实现和记事本类似的编辑功能。在此基础上，我们希望能够实现关键词高亮。

2. 调试功能

调试方面希望同时实现普通执行以及单步调试。单步调试时，用户能看见当前执行到的语句行数，同时还可以查看当前的运行环境，对于Pythy语言而言，主要就是查看符号表的绑定情况。

3. 简易效果图

正常执行中查看控制台



单步调试中查看运行环境