PikuLang Documentation

Table of Contents

Introduction

PikuLang is a purely functional programming language implemented in the Go programming language in March of 2025 by Bora Yılmaz. It aims to be as close as possible to the mathematical field of Lambda Calculus. It has no support for comments.

Data Types

PikuLang has only 3 data types: functions, integers and lists. Functions are defined using the "func" command. Please see the func part of the Commands section for more information. The integers are written without the sign and negated using the "neg" command if needed. Please see the neg part of the Commands section for more information on the neg command. Lists represent ordered lists of any type and are defined using the "list" command. See the list part of the Commands section for more information on the list command. Indexing lists is done by the "index" command. See the index part of the Commands section for more information on the index command.

Differences from other languages

As mentioned in the introduction, PikuLang has no form of comments built in. Every number is an integer where literals cannot have a negative sign and the only other data types are the function and the list. Functions can only have arguments/parameters and a return value. There is also no form of scope differentiation. Every single value including function parameters are defined in the global scope. Lists represent ordered arrays of items.

Commands

1. call

Calls a function passed in as the first argument with the other command arguments as the function arguments and returns the result

```
[call add2nums 3 4]
```

2. set

Defines a function or integer value to be a specific variable.

```
[set x 10]
[set f [func [x] [mul x 2]]]
```

3. echo

Prints an integer value to the console. Warns when trying to print a function.

```
[echo 142]
```

4. func

Creates an inline function. The first argument is a list of arguments and the second one is the return value.

```
[echo [call [func [x] [mul x 2]] 5]]
```

5. Mathematical Commands

A. add

Adds 2 values and returns the result.

B. sub

Subtracts 2 numbers and returns the result.

C. mul

Multiplies 2 numbers and returns the result.

D. div

Divides 2 numbers and returns the result.

E. mod

Gets the remainder of the division between 2 numbers and returns the result.

F. neg

Negates a number.

```
[echo [add 5 2]]
[echo [sub 5 2]]
[echo [mul 5 2]]
[echo [div 5 2]]
[echo [mod 5 2]]
[echo [neg 5]]
```

Result:

```
7
3
10
2
1
-5
```

6. import

Runs another file inline without creating a new environment.

module.pi:

```
[set a 5]
[set b 10]
```

main.pi

```
[import module]
[echo [add a b]]
```

Prints:

```
15
```

7. if

Returns the second argument if the first argument is positive or a function. If not, the third argument is returned.

```
[echo [if 5 1 0]]
[echo [if 0 1 0]]
[echo [if [neg 5] 1 0]]
[echo [if [fn [] 1] 1 0]]
```

Prints:

```
1
0
0
1
```

8. list

Represents an ordered list of items of any type.

```
[echo [list 1 2 3]]
```

Prints:

```
[ list 1 2 3 ]
```

9. index

Returns the item at a specified index of a list.

```
[set li [list 1 2 3]]
[echo [index li 1]]
```

Prints:

```
2
```

10. range

Returns a part of a list denoted by the start and end indices. The start index is included but the end index isn't. If the end index is 0, everything after the start index is returned.

```
[set li [list 1 2 3]]
[echo [range li 1 0]]
```

Prints:

```
[ list 2 3 ]
```

11. edit

Sets a specified index of a list to a specified value.

```
[set li [list 1 2 3]]
[edit li 0 5]
[echo li]
```

Prints:

```
[ list 5 2 3 ]
```

12. printchar

Prints a character to the console using the provided integer as a unicode codepoint.

```
[printchar 65]
```

Prints:

```
A
```

13. print

Prints a string consisting of characters with unicode codepoints from the provided list to the console.

```
[print [list 65 66 67]]
```

Prints:

```
ABC
```

14. newline

Prints a newline to the console.

```
[newline]
```

No output example is provided because it is not a visible character.