2. More Details About Pyth

Now that you have the programming environment set up, let's learn some more about the language.

2.1. Pyth Uses Prefix Notation

Try entering `2+2`. It'll be 4 right?:

```
2
Traceback (most recent call last):
  File "safe_pyth.py", line 300, in <module>
  File "<string>", line 5, in <module>
TypeError: plus() missing 1 required positional argument: 'b'
```

Oh noes! An error! What went wrong? Well, Pyth doesn't use Infix notation (the operator goes between the operands) like most languages do. Instead, Pyth uses Prefix (aka Polish) notation, which means the operator goes *before* the operands. This has the benefit of not requiring parenthesis, making programs shorter and freeing up operators for other uses. Let's try that math problem again:

```
+2 2

Output:

4
```

Now it is working as expected! Notice the space between the `2`'s so the parser doesn't interpret them as a `22`.

2.2. Pyth Has Many, Many Operators

The addition operator we just saw doesn't even begin to scratch the surface of Pyth's rich variety of operators. As well as addition, Pyth has all the customary arithmetic operators - `-` for subtraction, `*` for multiplication, `/` for division, `%` for modulo, and `^` for exponentiation.

Integers are defined as you would expect and Floats with the decimal point. Ex:

```
0
1
18374
2.5
```

However, negative numbers do not use a unary version of the subtraction symbol since all operators have a fixed arity (more on arity later). Instead, negative numbers use the unary reversal operator: `_`:

```
Invalid: -25

Valid: _25
```

Pyth also has many predefined variables:

```
Z=0
T=10
k=""
d=" "
b="\n"
```

All operators, variables, and control flow keywords, are always one character long to reduce the size of the program.

2.3. All Operators in Pyth Have a Fixed Arity

The arity of an operator or function (according to Wikipedia) is *the number of arguments or operands the function or operation accepts*. Most programming languages allow functions to accept different numbers of arguments, but doing so requires parenthesis. Pyth instead has a fixed number of operands per operator, allowing it to do away with parenthesis or any other delimiter.

2.4. Operators Mean Different Things in Different Contexts

To further increase the number of functions available with only one character operators, operators operate differently depending on the values passed to it. For example, we saw the `+` operator adds numbers, but if `+` gets two sequences (e.g. strings, lists) as operands, it concatenates them:

```
+2 T -> 12 (remember that T=10)

+"Hello ""World!" -> Hello World!
```

Letting `+` mean both concatenation and addition is pretty common, but Pyth takes this to another level, with most operators having 2 or more meanings.

2.5. The Pyth Code is Compiled to Python Code Then Run

Pyth is technically a JIT compiled language since the Pyth code is converted to Python through a series of rules defined in the file `data.py` and then run with the variables and functions defined in `macros.py`. You can select the debug button in the online interpreter or pass the `-d` flag to the local one to see the compiled Python code. Here is what comes out as debug from `^T6` which evaluates to a million:

```
==================================================
^T6
==================================================
Pprint("\n",Ppow(T,6))
==================================================
1000000
```

As you can see, the exponentiation call is translated to a call to the function `Ppow` and is implicitly printed through a custom print function called `Pprint`. The debug option can be very helpful with seeing what at first glance looks like a bunch of random characters does.

Previous
Next