8. The Language Specification - Control Flow

This section of the language specifications deals with control flow. It contains the keywords and the operators that affect which parts of the programs are run.

8.1. "#" - Exception Loop

Arity: Unbounded

This is the only form of error handling available in Pyth. It runs an infinite while loop until an error is reached, then breaks out of the loop.

Ex:

```
==================== 11 chars ====================
#/100T=T-T1
==================================================
while True:
 try:
  imp_print(div(100,T))
  assign('T',minus(T,1))
 except Exception:
  break
==================================================
10
11
12
14
16
20
25
33
50
100
```

8.2. ")" - Close Parenthesis

This ends one function or statement. Control flow like `if` or `for` all open up an unbounded arity and this closes one of them. Also useful for tuple and list constructors.

Ex:

```
==================== 28 chars ====================
I>5T"Hello")"Bye"[0 1 2 3 4)
==================================================
if gt(5,T):
 Pprint("\n","Hello")
Pprint("\n","Bye")
Pprint("\n",Plist(0,(1),(2),(3),(4)))
==================================================
Bye
[0, 1, 2, 3, 4]
```

8.3. ";" - End Statement

This is effectively an infinite amount of close parenthesis. This closes how many ever arities are needed to start completely afresh.

Ex:

```
==================== 16 chars ====================
V5I>5T[1 2;"Bye"
==================================================
for N in urange(5):
 if gt(5,T):
  Pprint("\n",Plist(1,(2)))
Pprint("\n","Bye")
==================================================
Bye
```

8.4. "B" - Break

This translates into the break keyword in Python. It is used to break out of both for and while loops (and also the infinite error loop). Pyth does not have a continue statement. Break automatically puts a close parenthesis after itself.

Ex:

```
==================================================
#ZI>ZTB~Z1
==================================================
while True:
 try:
  Pprint("\n",Z)
  if gt(Z,T):
   break
  Z+=1

 except:
  break
==================================================
0
1
2
3
4
5
6
7
8
9
10
11
```

8.5. ".?" - The Else Statement

Arity: Unbounded

This is the else part of the if-else construct. It is pretty self explanatory and works like it would in any programing language. This can also be used as part of a for-else or while-else construct. The If still needs a close parenthesis after it.

Ex:

```
==================================================
I>5T"It's greater").?"It's less than"
==================================================
if gt(5,T):
 Pprint("\n","It's greater")
else:
 Pprint("\n","It's less than")
==================================================
It's less than
```

8.6. "F" - The For Loop

Arity: Variable, Sequence, Unbounded

This is the ubiquitous for loop. It works like it does in Python, iterating through a sequence.

Ex:

```
==================================================
FNU5N
==================================================
for N in urange(5):
 Pprint("\n",N)
==================================================
0
1
2
3
4
```

8.7. "I" - The If Statement

Arity: Boolean, Unbounded

This is the If statement from Python. If the first argument is truthy, it executes the code, else it does nothing.

Ex:

```
==================================================
I>5T"The Universe Has Exploded"
==================================================
if gt(5,T):
 Pprint("\n","The Universe Has Exploded")
==================================================
```

8.8. "V" - Unary-Range-Loop

Arity: Integer, Unbounded

It is the shortest way to do a for loop. It is equivalent to the characters `FNU`. This makes it execute the following code a number of times equal to the input, with `N` being the loop variable. If a sequence is given as input, it is converted to an integer via its length.

Ex:

```
==================================================
VT*NN
==================================================
for N in urange(T):
 Pprint("\n",times(N,N))
==================================================
0
1
4
9
16
25
36
49
64
81
```

8.9. "W" - While Loop

Arity: Boolean, Unbounded

This the while loop construct from Python. It executes the following code until the condition becomes False.

Ex:

```
==================================================
W<lYT~Y]5;Y
==================================================
while lt(Plen(Y),T):
 Y+=[5]
Pprint("\n",Y)
==================================================
[5, 5, 5, 5, 5, 5, 5, 5, 5, 5]
```

8.10. & - Logical And

Arity: 2

This the logical `and` operator. It returns the first falsy value of its inputs, or the last value if all are truthy. It is shortcircuiting, just like Python's `and`.

Ex:

```
==================================================
&Z1&1T
==================================================
Pprint("\n",(Z and 1))
Pprint("\n",(1 and T))
==================================================
0
10
```

8.11. "|" - Logical Or

Arity: 2

This is the logical `or` operator. It returns the first truthy value of the input, or the last value if all are falsy. It is shortcircuiting, just like Python's `or`.

Ex:

```
==================================================
|Z1|ZZ
==================================================
Pprint("\n",(Z or 1))
Pprint("\n",(Z or Z))
==================================================
1
0
```

8.12. "?" - Logical If Else

Arity: 3

This is Pyth's ternary. Like most languages, but unlike Python, the conditional is the first input. The second input is executed and returned if the conditional is truthy, and the third input is executed and returned if the conditional is falsy. It is shortcircuiting, just like Python's `if else`.

Ex:

```
==================== 8 chars =====================
?T1 3?Z1 3
==================================================
Pprint("\n",(1 if T else 3))
Pprint("\n",(1 if Z else 3))
==================================================
1
3
```

Previous
Next