SquidScript Syntax Proposal

Table of Contents

- Variable Initialization/Declaration
- Variable Assignment/Updating
- If Statements/Switch Cases
- Loops
- Function Declaration
- Comments

Variable Initialization and Declaration

We decided that SquidScript will be explicitly and statically typed. We also agreed that the standard "=" assignment was a poor design choice, leading to confusion with new programmers. To deal with that we decided to adopt the "walrus" operator ":=" and the keyword let to denote the initialization of a new variable. This gives us the following:

```
let <type> <variable name> := <value of the same type>
```

ex:

```
let int x := 3
```

Assignment is treated a little differently, we instead use the keyword set:
`set x := 2`

Updating: we are doing something very different, instead of using an operator (+, -, *, etc.) in conjunction with =, we are going to be using them inconjunction with ":"

```
set x :+ 3
```

is equivalent to

```
set x := x + 3
```

Guards

We took some time to think of something that would be intuitive to the new programmer, and we settled on lock and key statements.

```
key (month) // a key is tested against "locks" until one "unlocks"
{
    lock 2 {print("February");}
}
```

or

```
key (x, y) // In this example we pass in 2 keys
{
```

```
  lock x = y
      {
        set x := 5;
      }

  lock x > y
    {//do stuff}

  lock else // instead of lock else -> unlocked?
    {//do stuff}
}
```

We decided to call them lock and key because, to a new programmer, this concept of finding a lock for a given key is intuitive to explain.

Loops

Squidscript will support both While loops and for loops. We are discarding the c-style for loop in favor of the "`for in`" structure found in Rust and Python. The idea of iterating through a list of items makes more sense in this context instead of relying on an interator integer you use for indexes. (You can of course still use "for in" on a range of numbers to achieve the same effect.)

```
for <variable iterable> in <range, list, or array of elements>
{
    perform action
}
//ex
for num in 1..100
{
    key(num)
    {
        lock (num%2 = 0){print(num)}
    }
}

while <conditional>
{
    perform action
}
```

Functions

Functions in Squidscript are fairly average, with the main difference being the order in which we define typing. We define the return type after declaring the

input types in an attempt to make it clearer to a new programmer what a function does.

```
func <funcName> (<param type> <params if any>) returns <output type>
        {
            //code
        }

//ex

func add(int x, int y) returns int
{
    return x + y;
}
```