Verbose

Overview

Verbose is a descriptive-syntax language where the keywords and grammar form highly readable code.

By avoiding keyword obscurity, uncommon formatting, and using familiar terminology, the language is easier to understand and learn. While Verbose can be used to write incredibly powerful programs, its descriptive nature allows readers and writers to more easily understand what the program is doing with each line of code. This language can be used to teach the concepts of programming to beginners, while also providing powerful capabilities for larger projects.

Design

Syntax Examples

```
-- Printing
print value of "Hello, World!"

-- Instantiating variables
create variable x with no value
create variable y with value of 1
create list list_2 with variables 1, 2, "dog", "cat"

-- Assigning values
assign value of 2 to variable x
assign value of x + 1 to variable x
assign value of "tiger" to variable 3 in list list_2

-- Condional/Loop Structures
if value of i is less than 5 do:
        print value of i + " is less than 5"

while value of i is less than 5 do:
        print value of i + " is less than 5"
        assign value of i + 1 to variable i
```

Semantics

Data Types

Variables are implicitly typed, with common types of float, integer, string, and list.

Data Scoping and Lifetimes

Variable scopes are global, except for the *for* and *for each* loop variables, which live until the end of the loop.

Execution Model

Verbose is an interpreted language.
Programs written in Verbose are text files with the `.verb` suffix, and are interpreted by a Verbose interpreter.

Features

Common programming paradigms for conditions and loops are supported, similar to what's available with Python. See *More Syntax Examples*.
A standout feature of Verbose is the ability to reference variables of a list as by which number it is (1, 2, 3), or by their index (0, 1, 2) by

using the `at index` key phrase. All lists in Verbose are 0-indexed but for the sake of learning indexing, allow for this more familiar way as well.

Sample Programs

Sample Program 1 - Hello World

```
print value of "Hello, World!"
```

Sample Program 2 - Creating and Re-assigning Variables

```
create variable num1 with value of 20, num2 with value of 22
assign value of num1 + num2 to variable num2
print value of "The addition of num1 and num2 is: " + num2
```

Sample Program 3 - FizzBuzz

```
for variable num with values from 1 to 100 do:
        if value of num % 15 is equal to 0 do:
                print value of "FizzBuzz"
        else if value of num % 3 is equal to 0 do:
                print value of "Fizz"
        else if value of num % 5 is equal to 0 do:
                print value of "Buzz"
        else do:
                print value of num
```

Ecosystem

I would like to also develop a syntax highlighter, most likely for VS Code, my current editor of choice. Visualizing the difference between key words, variables, and lists will also help with the readability and writability of Verbose.
If I can, as a stretch goal, I'd like to implement support for using Python libraries within Verbose programs. A part of this is also building support for functions, function calling, and probably a pre-processor that can handle import statements.

Challenges

- To allow syntax to be *verbose*, the grammar needs to allow for many different permutations of a statement, which will likely lead to a quite large parser
- The line between verbose and overcomplicated is fuzzy, so a large time-consuming challenge of building Verbose will be the syntax design
    - Statements can be verbose, while still making sense and feeling familiar to all other types of statements
- I've also never built any sort of syntax analyzer/highlighter, and I feel like a large part of the readability will come from that

More Syntax Examples

Printing a value

```
print value of "Hello, World!"
print value of "Hello, World!" without newline
print value of "Hello, " + "World!"
print values of "Hello, ", "World!"
```

Instantiating a variable

```
create variable x with no value
create variable y with value of 1
create constant variable z with value of 100
create list list_1 with no variables
create list list_2 with variables 1, 2, "dog", "cat"
create list list_3 with variables from list_1, List_2
```

Assigning a value to a variable

```
assign value of 2 to variable x
assign value of x + 1 to variable x
assign value of z to variable y
assign value of "tiger" to variable 3 in list list_2
assign value of "tiger" to variable at index 2 in list list_2
assign values of 3, 4 to variables 1, 2 in list list_2
```

Loop structures

```
if value of i is less than 5 do:
        print value of i + " is less than 5"

while value of i is less than 5 do:
        print value of i + " is less than 5"
        assign value of i + 1 to variable i

for variable i with values from 1 to 100 do:
        print value of i

for each variable in list list_1 as item do:
        print value of item
```

Comment

```
-- This is a comment

---
This is a
multi-line comment
---

print value of "Hello, World!" -- This is an inline comment
```