1 Introduction and Overview

Meowlang is an esoteric programming language inspired by LOLCODE, a language created by Adam Lindsay based on internet lolspeak. Our goal is to push the boundaries of language design in a creative, humorous way while still providing functionality and usability. Considering the nature of internet lolspeak (deliberately misspelled natural language), Meowlang improves on the safety and functionality of LOLCODE. Firstly in safety, unlike LOLCODE’s dynamic typing, Meowlang enforces strong static typing to reinforce the integrity of user’s code. Secondly in functionality, Meowlang provides an implementation of arrays and limited support for object-oriented programming, both which LOLCODE currently lacks.

Both goals for improving on LOLCODE’s functionality (arrays and limited OOP) involve implementing entirely new features to the original language. We hope to implement arrays because they are necessary, versatile building blocks and to implement OOP because it illustrates the creativity of the esolang. We would greatly appreciate feedback on the feasibility of this plan.

TLDR: Meowlang is an imperative, high-level, object-oriented programming language with static scoping, strong static typing and strict left-to-right evaluation order.

2 Language Paradigm

Meowlang is a pared-down version of LOLCODE, sharing similar syntax, but with some changes to its language paradigm. Meowlang shares similarities with both C and Java, with its lack of garbage collection and object-oriented programming features, respectively. Meowlang will have object-oriented features like classes, class variables, class methods/functions, instance variables, instance methods/functions, constructors, and destructors. Code blocks are delineated with keywords HAI and KEYE in substitution of braces as well as with indentation such as in Python.

3 Language Details

3.1 Data Types and Basic Operations

Four data types will be supported by Meowlang out of the box:
1. Strings - *YARN*, demarcated by quotation marks
2. Integers - *NUMBR*
3. Floats - *NUMBAR*
4. Boolean - *BOO*, with options *AYE*(true) and *NAY*(false)

New variables can be declared with the keyword phrase *ITZ ME* followed by the variable type and name:

- *ITZ ME* <variable type><identifier>

Variable identifiers are made up of lowercase letters and underscores, and must begin with a letter. No spaces, dashes or symbols are allowed.

The original LOLCODE language has a number of built in operators that support basic math, boolean expressions and comparisons. These operators rely on prefix notation, taking the form:

- <operator><expression1>
- <operator><expression1>AN<expression2>

Our implementation of Meowlang will follow LOLCODE syntax, as specified at: https://github.com/justinmeza/lolcode-spec/blob/master/v1.2/lolcode-spec-v1.2.md

### 3.2 Comments

The PSST keyword precedes every comment and continues until the end of the line, for single line comments and comments inline with code. There are no multi-line comments.

### 3.3 Keywords

Keywords will be denoted in all capital letters, for examples: *ITZ ME, IZ, HAI, NEW* and *WIT*.

### 3.4 Modules/Library

Libraries can be imported with the keyword *GIMME* and will end with the character “?”.

- *GIMME* <module_name>?

The STDIO library will be implemented, which will include functions for printing (*MEOW*) and retrieving input from users (*SCAN*). This module is imported in the code sample below.

### 3.5 Functions

Functions will have a fixed number of arguments and return types; arguments will be optional. Functions are defined using the following syntax structure:

- *ITZ ME* <return type>FUNC <name>WIT <arg type><arg name>AN <arg2 type><arg2 name>

To return a value from a function, use the keyword *GIVE*. Functions are called using the keyword *PURR*:

- *PURR* <function name >WITH <arg1>AN <arg2>
3.6 Arrays

Arrays are called **BUCKETS** in Meowlang. Meowlang supports fixed-length arrays with size determined at compile time. To create a new **BUCKET**:

\[
\text{ITYZ ME \langle variable type \rangle BUCKET OF \langle array size (natural number) \rangle \langle array name \rangle}
\]

\[
\text{WIT \langle element at index [0] \rangle}
\]

\[
\text{WIT \langle element at index [1] \rangle}
\]

\[
\vdots
\]

**BUCKET** elements may be accessed with \langle bucket name \rangle[0], \langle bucket name \rangle[1], etc.

3.7 Object Oriented Programming

Unlike the original LOLCODE, which does not have support for custom objects, Meowlang will allow programmers to define their own classes with instance variables and methods. A new object can be created with the **MAEK** keyword, and the use of **NEW** allows memory to be allocated on the heap. Meowlang will not support automatic garbage collection, so memory must be freed using the **BLEEP** keyword:

- Creation: **MAEK** \langle object name \rangle **NEW** \langle object type \rangle
- Deletion: **BLEEP** \langle object name \rangle

3.8 Code Example

Files containing Meowlang code end in **.meow**.

---

Listing 1: **mouse.meow**

1  GIMME STDIO?  PSST Import the Standard Library
2  
3  PSST Create a custom MOUSE class, class convention is in all-caps
4  HAI ITZ ME MOUSE
5  
6  ITZ ME NUMBR cookies IZ 0  PSST This is an instance variable
7  
8  ITZ ME FUNC squeek  PSST This is a class method
9  MEOW "Eeeeeek!"
10  
11  ITZ ME NUMBR count_cookies
12  GIVE cookies
13  
14  ITZ ME FUNC give_cookie WIT NUMBR count_cookies
15  
16  PSST This uses the SUM prefix operator
17  cookies IZ SUM OF cookies AN count_cookies
18  
19  KBYE
HAI ITZ ME YARN FUNC Chase WIT YARN bad_cat AN YARN poor_mouse
    GIVE bad_cat CAT " chases " CAT poor_mouse
KBYE

HAI ITZ ME FUNC MAIN

PSST Declaring a bunch of variables...

ITZ ME YARN cat_name IZ "Silvester" PSST declaration of string
ITZ ME YARN mouse_name IZ "Gary"
ITZ ME NUMBR num IZ 2 PSST declaration of int
ITZ ME NUMBAR value IZ 2.0 PSST declaration of float
ITZ ME BOO mouse_is_small IZ AYE PSST declaration of boolean
ITZ ME BOO cat_is_friendly IZ NAY

ITZ ME YARN BUCKET OF 2 names
    WIT cat_name
    WIT mouse_name

MEOW PURR Chase WIT names[0] AN names[1] PSST Prints "Silvester chases Gary"

MAEK gus NEW MOUSE PSST Create a object using 'NEW' keyword
BLEEP gus PSST Release memory associated with gus
KBYE