H** Programming Language Reference Manual

Group: Dominance
Shubh Agarwal, Pulkit Gautam, Manav Parmar, Aman Singh

January 22, 2024

1 Introduction

H** is a minimalistic imperative programming language designed for simplicity and ease of use. It incorporates basic types, compound types, conditionals, loops, functions, closures, mutable variables, exceptions, and more.

2 Syntax

2.1 Variables and Assignment

```
const x = 5;                    % Constant Variable
var y = 10;                     % Non-Constant Variable

% Mutable variable assignment
y = 15;

% Multiple assignments
var a, b = 2, 3;
```

2.2 Basic Types

```
var num = 42;                   % Number
var isTrue = true;              % Boolean
var text = "Hello";             % String
```

2.3 Compound Types

```
tuple t = [1, "two", true];     % Tuple
list l = [1, 2, 3, 4];          % List
arr a = [1, 2, 3, 4];           % Arr
```

2.4 Conditionals

```
if (condition) {
    % code block
} elif (condition) {
    % code block
} else {
    % code block
}
```

2.5 Loops

```
while (condition) {
    % code block
}

for (var i = 0; i < 5; i++) {
    % code block
}

do {
    % code block
} while (condition)
```

3 Functions and Closures

3.1 Function Definition

```
% Function definition
func add(var x, var y) {
    return x + y;
}
```

3.2 Function Invocation

```
% Function call
var result = add(3, 4);
```

3.3 Closures

```
function outerFunction() {
    var outerVar = "I'm from outer!";

    function innerFunction() {
        print(outerVar);
    }

    return innerFunction;
}

% Creating a closure
var closure = outerFunction();

% Executing the closure
closure();  % Outputs: I'm from outer!
```

4 Exception Handling

```
try {
    % code block
} catch (ExceptionType e) {
    % handle exception
} finally {
    % optional: code block to execute regardless of exception
}

throw ExceptionType("An error occurred");
```

5 Printing

```
print("Hello, World!");
```

6 Additional Features

6.1 Unary and Binary Operators

```
var a = 5;
a++;
print(a); % Outputs 6
a+=1;
print(a) % Outputs 7
```

```
var b = 5
a = a + b
print(a) % Outputs 12
```

6.2 Boolean Type

```
var a = false;
if (!a) {
  print("Hello");     % Prints Hello
}
```

6.3 Comparators and if-else

```
if (a >= c) {
  print("Hello");
} elif (a == c) {
  print("World");
} else (a != c) {
  print("Worlds");
}
```

6.4 Strings

```
% Concatenation
var a = "Hello";
var b = "World";
var c = a + "-" + b;
print(c); % Outputs "Hello-World"

% Slicing
var a = "Hello-World";
var c = a.slice(0, 5);
print(c); % Outputs "Hello"
```

6.5 List and Arrays

```
list l = [2, 3, 4];
l.append(5);
print(l[3]); % Outputs 5, list is variable in size

arr a = [2, 3, 4];
a.append(5); % Throws an error, array is fixed size
```

```
print(a.length()); % Outputs 3, same for list
print(a[0]);  % Prints first element, same for list
print(a[a.length() - 1]); % Prints last element, same for list
```

6.6 Assignment and Let Expressions

```
const a = 5; % Variable a cannot be changed after assignment
var b = 5; % Variable can be changed after assignment
```