

WESS Programming Language

WESS is a C-like programming language that is designed to be easy to learn and use. Also it must be your first choice if you want to hack NASA.

Table of Contents:

- [Installation:](#)
- [How to use \(For Phase 1\):](#)
- [Language Features & Syntax:](#)
- [Samples:](#)
- [Contrinutors:](#)
- [License:](#)

Installation:

install ply:

```
pip install ply
```

clone the repo:

```
git clone https://github.com/EssamWisam/WESS-Lang
```

How to use (For Phase 1):

write your code in `code.txt` file, then run `parser.py`:

```
python Parser.py
```

If there is no syntax error messages, your code is parsed successfully. (For Phase one)

Language Features & Syntax:

The language has syntax similar to C with some differences. The main difference is that WESS is a dynamic language, so you don't need to declare the type of the variable. Also WESS is a case-sensitive language, so `x` and `X` are two different variables.

Let's get started!

Comments:

To add a comment, use `'#'` for single line comments.

```
# This is a comment
```

Data Types:

- **Integer:**

```
var x = 5;  
var y = -99 - 1; # y = -100
```

- **Float**

```
var x = 5.5;
```

- **Boolean (True or False)**

```
var x = True;  
var y = False;
```

- **String**

```
var x = "Hello World!";
```

Constants:

```
const PI = 3.14;
```

Variables:

```
var x;
```

and you can assign the value at the time of declaration:

```
var x = 5;  
var y = (x + 2.1) * 3;  
var z = "Hello World!";
```

Enum:

```
enum Colors {
    RED,
    GREEN,
    BLUE
};
enum Colors c;
c = RED;
```

Operators:

Arithmetic Operators:

Operator	Description
+	Addition
-	Subtraction
*	Multiplication
/	Division
//	Integer Division
%	Modulus
()	Parenthesis

Examples:

```
var x = (5 + 2) * 3;
```

Assignment Operator:

Operator	Description
=	Assigns the value of the right operand to the left operand

Examples:

```
var x = 5;
x = 10;
```

Comparison Operators:

Operator	Description
<code>==</code>	Equal to
<code>!=</code>	Not equal to
<code>></code>	Greater than
<code><</code>	Less than
<code>>=</code>	Greater than or equal to
<code><=</code>	Less than or equal to

Examples:

```
var x = 3 > 5;
```

Logical Operators:

Operator	Description
<code>and</code>	Logical AND
<code>or</code>	Logical OR
<code>not</code>	Logical NOT

Examples:

```
var x = 10 >= 5;  
var y = x and (7 != 8);
```

```
var x = True or False;
```

Expressions:

There are two types of expressions:

- **Arithmetic Expressions:**

```
(5 + 1) / 3;
```

- **Logical Expressions:**

```
True or False;  
(x and True) or False;
```

Conditional Statements:

If Statement:

```
if (expression) {  
    # code  
}
```

If-Else Statement:

```
if (expression) {  
    # code  
} else {  
    # code  
}
```

Loops:

While Loop:

```
while (expression) {  
    # code  
}
```

do-While Loop:

```
do {  
    # code  
} while (expression);
```

For Loop:

```
for (initialization; condition; increment) {  
    # code  
}
```

```
}
```

Examples:

```
for (var i = 0; i < 10; i = i + 1) {  
  # code  
}
```

Switch Statement:

```
switch (expression) {  
  case value1:  
    # statements  
    break;  
  case value2:  
    # statements  
    break;  
  default:  
    # statements  
}
```

Functions:

```
function functionName(parameter1, parameter2, ...) {  
  # code  
  return expression;  
}
```

Blocks:

```
{  
  # code  
}
```

Samples:

```
var i;  
var x;  
for (i = 1; i <= 10; i = i + 1) {  
  if (i / 2 == 0) {  
    x = i;  
  }  
}
```

```
    } else {  
        x = i + 1;  
    }  
}
```

```
function factorial(n) {  
    if (n == 1) {  
        return 1;  
    }  
    return n * factorial(n-1);  
}  
var x;  
x = factorial(5);
```

Contrinutors

- Ahmed Waleed
- Essam Wisam
- Mohamed Saad
- Mohamed Salama

License

[MIT](#)