TITANIUM PROGRAMMING LANGUAGE

⚠️ **TITANIUM IS STILL IN BETA AND, AS SUCH, IT MIGHT STILL CONTAIN LOTS OF BUGS AND SOME FEATURES MIGHT BE MISSING.**

Titanium is a minimalist interpreted fictional programming language designed for one of my stories (actually, it was initially planned to be fictional, but I want to take it a step further, if possible). It’s been developed by the fictional characters Matheus, Gustavo N, and Léo Andrew.

The language itself and its online playground (which is currently the only way you can run Titanium) are being written in TypeScript and then compiled to JS. Python and Lua have heavily inspired Titanium’s syntax and its goal is to be easy to read and understand, clear, and concise.

Since its goal was to be a very minimalist programming language, it contains few keywords and functionalities, but you should be able to declare variables and constants, print them to the screen, get input from the user, generate random numbers, use boolean, use the ternary operator, and even converting decimal numbers to binary and hexadecimal (will be added later).

The language, however, comes neither with IF, FOR, and SWITCH control structures nor with more complex things, such as functions and the object-oriented paradigm. Since its goal is to be as minimalist and as simple as possible, I probably won’t be adding those features to the language.

However, I might create a superset of it in the future which would include those aforementioned things (just like TypeScript is a superset of JavaScript). Also, two of Titanium’s outstanding characteristics are the use of capitalized keywords, inspired by COBOL, and the use of short keywords, with an average of three or four letters. It’s also worth mentioning that spaces in Titanium are mandatory, in order to keep the code readable and organized.

Titanium Playground - Codename Gusleothew

https://redwars22.github.io/Titanium/titanium/playground.html

GitHub - Redwars22/Titanium: Titanium is a minimalist

Titanium is a minimalist interpreted fictional programming la

https://github.com/Redwars22/Titanium

🔡 CONSTANTS AND VARIABLES

Constants are defined with the DEF keyword and can't be changed later on. Variables are defined with the DECL keyword.

```
DEF name = "Matheus"
DECL age = 17
```

Titanium currently supports the following data types: STRING, BOOLEAN and NUMBER. Strings should be encapsulated between "" and boolean values can be either TRUE, YES, FALSE, or NO. Titanium also supports positive, negative, integer, float and double numbers. You can also assign NULL or UNDEF to them.

#️⃣ COMMENTS

Titanium supports both inline and multiline comments. Single-line comments are declared by using – before the line you want to turn into a comment. A multiline comment begins and ends with $$. i.e.:

```
--This is a single-line comments

$$
This is a comment
that spaws over several lines
$$
```

🔢 MATH AND LOGIC EXPRESSIONS

You can assign math expressions such as `2 + 2` or `9 % 2` to variables and constants. You can also pass them as arguments to the print function to get their result printed to the screen. You can also evaluate logic expressions such as `"a" == "b"` or `9 != 10`. They will return either TRUE or FALSE.

```
print(9 < 20) --this should return true
print(2 + 2) --this should return 4
```

❓ TERNARY OPERATOR

Titanium also supports the ternary operator. You can assign it to either variables or constants.

```
DEF x = 10
DEF y = 9
DEF result = x < y ? TRUE : FALSE
```

💽 INPUT AND OUTPUT

Titanium has some functions to handle input and output of data:

- `print()`: outputs something to the console. It accepts variables, constants and math and logic expressions as its parameter.

- `printLine()`: it prints an empty line.
- `get()`: it asks the user for input. Its argument is the variable where the data should be stored.
- `clear()`: it clears the console.

You can concatenate two or more values in the print function by using the `&&` operator: `print("Your name is " && username && " and your userID is " && id)`

🏁 ENDING THE EXECUTION OF THE PROGRAM

You can end the execution of a Titanium program in two ways: by using the `EXIT` keyword or by using `RET` followed by the return value, which should be a number, a math expression or a string.

🔢 ARRAYS

You can declare an array in Titanium by using the ARR keyword. You should put all the data and use colons to separate the elements.

`ARR names = ["Karl", "Doug", "Matthew", "Andrew"]`

You can print all the elements in the array by using the `print()` function and passing the name of the array as its argument. To print only one of its elements, just put the index of the element you want to access inside square brackets. Example:

`print(names[2])`

IMPORTANT!: The first index of an array is 0, just like in most programming languages, such as C, JavaScript, and so on!

Also, before trying to access a specific index, please double-check if it exists, or else Titanium will throw an error. To get the length of an array, you can use the `MAX` or `LEN` word instead of the number of the index. Example:

```
ARR names = ["Karl", "Doug", "Matthew", "Andrew"]
print(names[MAX])
```

```
ARR names = ["Karl", "Doug", "Matthew", "Andrew"]
print(names[LEN])
```

You can push a new element to an array from the user input by using the get function and passing the name of the array you want to push an element to as an argument. For example:

```
ARR names = ["Matthew"]
get(names[])
```

➕ INCREMENT AND DECREMENT

You can increment or decrement the value of a variable by 1 by using the `INC` or `DEC` keywords respectively and the name of the variable you want to manipulate. Titanium doesn’t allow incrementing or decrementing constants.

```
INC x
DEC x
```

🗑️ DELETING VARIABLES

You can delete variables, constants and arrays in Titanium using the DEL keyword followed by the identifier of the thing you want to delete. For example, if you want to delete the constant `username`:

```
DEF username = "André Pereira"
DEL username
--this line below should give you an error, since username no longer exists
print(username)
```

📏 MATH FUNCTIONS

You can also do things like generating a random number in a specific range, getting the hexadecimal or binary representation of a decimal number, getting the squareroot or the sin of a number, and so much mure with Titanium’s built-in math functions.

Their structure is simple: they begin with the MATH keyword followed by the math operation you want to execute, then the name of the variable where you want to store the result and, finally, the arguments required by the math operation you had chosen.

For example, if you want to get the square root of the number 25 and store it in the variable `result`, you can use `MATH SQRT result 25`. Here’s a table with all the math operations Titanium already supports:

DESCRIPTION	KEYWORD	ARGUMENT 1	ARGUMENT 2
Gets the number without the sign	ABS	number	NOT REQUIRED
Gets the binary representation of a decimal number	BIN	decimal number	NOT REQUIRED
Gets the cosin value of a number	COS	number	NOT REQUIRED
Gets the hexadecimal representation of a decimal number	HEX	number	NOT REQUIRED
Generates a random number in the specified range	RAND	The min value	The max value

DESCRIPTION	KEYWORD	ARGUMENT 1	ARGUMENT 2
Rounds the number to the nearest integer	ROUND	number	NOT REQUIRED
Gets the sin value of a number	SIN	number	NOT REQUIRED
Gets the square root of a number	SQRT	number	NOT REQUIRED
Gets the tangent of a number	TG	number	NOT REQUIRED

🆕 UPCOMING FEATURES IN THE NEXT VERSIONS

- `SLEEP x` - delays the execution of the program for the specified amount of time, in miliseconds.
- Add the `PI` constant to the math functions.
- Add the `POW` operation to the math functions.