Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science



A11

Language Specification

Team:

[Youssef Hamzo] - Id: [041024593] / [Akshit Sabharwal] - Id: [041025056]

Language Name [Viper]

***This template is suggested (not mandatory) to answer A11 Specification.***

|  |  |
| --- | --- |
| **Part**  **1** | **Language User Reference** |

**EXPLANATION**

This document defines our new Python-like language Viper and describes its basic structure, datatypes and functions or variable implementation or syntax.

* 1. **User Manual**

**Element 1: Name / Extension**

***Language Name: Viper***

*Extension:* ***.vip***

*Based on: Python* ***/ boa***

**Element 2 – Comments**

*That’s our own feature*

*# Your comment here*

// single comments

/\* blocked commenting\*/

**Element 3 – Keywords**

If, else, elif, while do, break, return, true, false, def, print, const, input

**Element 4 – Variables and Datatypes**

*Num (number): for any numbers like integer, float (32-bits)*

*String: for characters (32-bits)*

**Element 5 – Variables and Datatypes**

*variableName = 4;* *type : number*

*variableName = 4.25; type: number*

*variableName = “YO” type: str*

*Number (num), String (str)*

**Element 6 - Commands**

Variable declaration

Y = 3

X = 2

We will allow casting in our language. Our language will automatically convert one datatype to another.

Ex: a = 7

print(type(a))

Here our language will automatically convert a to int.

Regarding math in our language: For Addition: we use the “ **+** ” sign

For Subtraction: we use the “ **-** ” sign

For Multiplication: we use the “ **\*** “ sign

For Division: we use the “ **/** “ sign

For remainder: we use “**%**” sign

We can declare constant by using const

const VARIABLE\_NAME = “Something”;

Our language **will support** string **concatenation** using the **+** operator, for example:

X = ‘My age is’

Y = 19

print(X + Y) will output: My age is 19.

Selection (if-else condition):

We will do our conditionals in the following way:

We will have a certain expression. If this expression is true then a statement will be executed, if its false this statement will be skipped and not executed.

Ex: a = 8

b = 2

If a > b:

Print(‘true’)

The program will print true.

If a < b:

print(‘true’)

Here the statement will be skipped, and nothing will be printed since the expression is false.

For Boolean operators, in our language we can write them in the following way:

For "or” we use: **or** ex: a || b

For "and" we use: **and** ex: a && b

For “not” we use: **not** ex: !a

For “greater than” we use: **>** ex: a > b

For “less than” we use: **<** ex: a < b

For “not equal: we use: **!=** ex: a != b

for “equal” we use: **=** ex: a == b

**Interaction (loops)**:

An example of a do while in our language is:

do{

loop statement to be executed

}

while(condition)

In the above snippet, the statement inside the do will keep executing whenever the condition in the while is true.

while(condition){

//do something

} //loop will keep executing till the condition is true.

for (i=0; i<5; i++) {

//do something

} //this for loop will execute five times

* ***Input****: How does your program get input from the keyboard**? (Strings are easiest.)*

In our language we will get input from the keyboard using the following way:

Ex:

>>> name = input(‘What is your name? ”)

What is your name? Paulo Sousa

* ***Output****: What would a programmer type to put output on the screen**? What sort of variables or data will your code take?*

To print the above input example in our language, you can use the print( ) method.

Ex:

>>> print(“Name: Paulo Sousa”)

Name: Paulo Sousa

* ***Functions****:*

Our function definition will look like this:

def functionName(p1,p2) {

variableName = p1+p2

return variableName

}

**Element 7 – Proper elements**

**Number**: this datatype will be used to represent integers and decimal numbers as well.

**Comments**: we have single line (//) and block commenting (/\* \*/).

**Constant**: we will be introducing the const keyword to declare constant in our language.

**Lesser code for following like function:**

**def myFunction(x) {**

**return x\*4**

**}**

A function with only a return statement could be written as well:

**my\_function(x) -> x\*4**

**In-built Function:** **validator:** to validate the input for the number, will check if the user enters the number or not.

|  |  |
| --- | --- |
| **Part**  **2** | **Examples** |

**Option 1: Python-like**

**Hello World**

|  |  |  |
| --- | --- | --- |
|  | [Your Code here]  main() {  print(‘ Hello World! ‘)  } |  |

**Sphere Volume Expression (or any other example)**

|  |  |  |
| --- | --- | --- |
|  | [Your Code here]  main() {  pi = 3.1415926535897931  r= 6.0  V= 4.0/3.0\*pi\* r\*3  print('The volume of the sphere is: ',V)  } |  |

|  |  |
| --- | --- |
| **Part**  **3** | **Architectural Aspects** |

**Advantages**

Our goal is to create a python-like language for beginners who are new to the programming world. Learning your first programming language can be a hassle as there are lots of options available in the market. Why not create a simple and easy language to start with? As our language will focus on implementing simple things for beginners to learn about like how variables, function definition and implementation look in programming.

**Strategy: C Implementation**

Language implementation in C

We have C datatypes like int, float, and double to implement our Number datatype for language.

For String, we are thinking of using an array of chars.

For our in-built function (validator), we can use isdigit() function from C to implement ours.

As our logical operators remain consistent with C so we should be able to implement them easily.

Ideas about how to identify elements from the language

The print function in our language will take any number or string and print it on one text line. The direct literal like *"this is going to get printed"* will be shown as an error or will be ignored by our compiler for our language if it is not used for variable declaration such as:

This is text //error or will be ignored

Var = “This is text” //will be compiled successfully.

print(“This is text”) // will be printed to console/output

Code Scope :

In our language to control what portion of code gets looped through / for function scope we will be using { and } literals, anything between them will have scope only between these two literals just like C:

def functionName {

//do something

return “anything”

}

**Basic ideas about C implementation**

*The hardest part we believe for parsing our language:*

**Lesser code for following like function:**

**def myFunction(x) {**

**return x\*4**

**}**

A function with only a return statement could be written as well:

**my\_function(x) -> x\*4**

The above shorter code thing that we are thinking of implementing is going to be one of the hardest things to implement. For now, we have just an idea about this feature, we will have to figure out its implementation in future assignments.

***Note 1: C Datatypes***

*Remember that you are implementing your language in ANSI C. For this reason, you cannot create arbitrarily your language (from scratch). You need to use what is already provided by C Compiler. For this reason, think about using and defining the language obeying the datatypes.*

**Problems when using the C implementation**

As there is no garbage collector in C so memory allocation is going to be one of the main problems. We will have freed up the memory on our own. Casting the variable on its own can be also a great problem as there is no such thing in C, we will have to C’s inbuilt function to do that job for our language.

**References**

# References

|  |  |
| --- | --- |
| [1] | B. Lee, "How to create your own programming language," Software for Enterprise , [Online]. Available: https://www.softwareforenterprise.us/how-to-create-your-own-programming-language/. [Accessed 15 September 2022]. |

Algonquin College

Fall, 2022