A11

Language Specification

Team:

Shivam Patel - Id: 041043423 / Nathan Chen - Id: 041000672

Vision (Vision Serpent)

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

**EXPLANATION**

* The purpose of this project is to invent a new language that adapts ‘BOA’ to be compatible with ‘Python’. The syntax of the language will be familiar to ‘Python’ but will have additional similar structures to other languages like ‘C’.

**User Manual**

**Element 1: Name / Extension**

We decide to name our language called ‘Vision’. We get inspire from the name ‘Vision Serpent’. File extension is ‘.vs’. We are changing BOA language to python, that’s why our compiler language will be very similar to python.

**Element 2 – Comments**

* Commenting is very important in every language. In python in-line comments are done by “#”.
* In python multi line comments are not allowed, but our compiler allow to put multi-line comments just by putting “/#....................#/”.

**Element 3 – Keywords**

|  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- |
| and | as | assert | break | class | continue | def |
| del | elif | else | except | False | finally | for |
| from | global | if | import | in | is | lambda |
| None | nonlocal | not | or | pass | raise | return |
| True | try | while | with | yield |  |  |

**Element 4 – Variables and Datatypes**

|  |  |  |  |
| --- | --- | --- | --- |
|  | **Datatype** | **Bytes** | **Range** |
| **Text Type** | str | 8 bytes | 0 to 255 |
| **Numeric Types** | int | 4 bytes | -2,147,483,648 to +2,147,483,647 |
| float | 4 bytes | ±3.4E-38 and ±3.4E38 |
| **Boolean Type** | bool | 2 bytes |  |

**Element 5 – Variables and Datatypes**

|  |  |
| --- | --- |
| **Integer** | x = 64 |
| **Floating Point** | y = 3.14 |
| **String** | z = ‘Hello World’ |

**Element 6 - Commands**

* The language will allow programmers to assign values to a variable by adding an equal sign in between them. (e.g., x = 32)
* The language will allow programmers to handle math by adding an operator between two integer and/or floating-point values. (e.g., y = 4 + 3.14)
* The language will let programmers to handle concatenation by adding a plus between two strings. (e.g., z = ‘Hello ’ + ‘World’)
* The language will allow programmers to do if-style logic by having an ‘if’ and can be followed by an ‘elif’ and/or ‘else’. After adding an if-style logic it will be followed by a variable, then a logical condition, another variable, and finally a colon ‘:’.

|  |  |  |
| --- | --- | --- |
|  | a = 1  b = 2  if a > b:  print(‘a is greater than b’)  elif a == b:  print(‘a is equal to b’)  else:  print(‘a is less than b’) |  |

|  |  |  |
| --- | --- | --- |
| **Conditional Operators** | | |
| **Equals** | a == b | If ‘a’ is equal to ‘b’ |
| **Not Equals** | a != b | If ‘a’ is not equal to ‘b’ |
| **Less Than** | a < b | If ‘a’ is less than ‘b’ |
| **Less Than or Equal to** | a <= b | If ‘a’ is less than or equal to ‘b’ |
| **Greater Than** | a > b | If ‘a’ is greater than ‘b’ |
| **Greater Than or Equal to** | a >= b | If ‘a’ is greater than or equal to ‘b’ |

|  |  |  |
| --- | --- | --- |
| **Logical Operators** | | |
| **Or** | a > b or a > c | If ‘a’ is more than ’b’ or ‘a’ is more than ‘c’ |
| **And** | a > b and a > c | If ‘a’ is more than ’b’ and ‘a’ is more than ‘c’ |
| **Not** | not a > b | If ‘a’ is not more than ‘b’ |

* The language can handle looping by using a ‘while’ loop. The loop can be performed by using ‘while’, followed by a variable, a conditional operator, followed by another variable, and finally a semicolon.
* The language can also loop by using a ‘for’ loop. The loop should be written as ‘for’, followed by a variable, then ‘in’, and then ‘range()’.

|  |  |  |
| --- | --- | --- |
|  | while x < 6:  print(x)  x += 1  for x in range(6):  print(x) |  |

* The language will get user input by using “input()” method. In the brackets you can give a message to the user in double quotes.

|  |
| --- |
| something = input(“Hi! Just write something: ”) |

* Programmers can put output on the screen by using “print()” method. What message user but in double quote inside brackets, compiler will treat it as a string and if it is not double quote compiler will think it’s a data type and look out in the code to find that datatype variable and print it out on screen.

|  |  |
| --- | --- |
|  | Z = 23  print(“Hello, I am”, Z, “old”) |

* The syntax for creating a function will include the keyword ‘def’, followed the name of the function, then succeeded by brackets, and ended with a semicolon. The user can call the function by writing the function’s name with brackets.

|  |  |  |
| --- | --- | --- |
|  | def function\_name():  print(‘Hello World’)  function\_name() |  |

* The language will take parameters by including a variable name in the brackets of the function. The user can give a value to the variable in the brackets, or when calling the function insert a value in the brackets.

|  |  |  |
| --- | --- | --- |
|  | def function\_name(world):  print(‘Hello ’ + world)  function\_name(‘World’) |  |

* The language can return results by using ‘return’ followed by a value after a function.

|  |  |  |
| --- | --- | --- |
|  | def function\_name():  return ‘Hello World’  print(function\_name()) |  |

**Element 7 – Proper elements**

* + We are including the feature to add multi-line commenting, since Python does not allow it.
  + We may include the use of curly brackets when making a function, since Python uses indents.

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

**Option 1: Python-like**

**Hello World**

|  |  |  |
| --- | --- | --- |
|  | [Your Code here]  def main():  print(“Hello World”)  main() |  |

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

|  |  |  |
| --- | --- | --- |
|  | [Your Code here]  def sphereVolume():  print("Enter a radius")  r = input()  r = int(r)  pi = 3.14  v = 4.0/3.0\*pi\*(r\*r\*r)  return v    print("The sphere volume is",sphereVolume()) |  |

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

**Advantages**

* Our language is made to be simple, and for fun. We are trying to implement new features in Python and make the user experience enjoyable and easy to use. It won't be useful to scientists, but more useful for students who want to learn a language that is familiar to other languages, while being easy to learn. Python syntax are simplest of all the other languages, and we try to add some more simple features.

**Strategy: C Implementation**

* Our compiler language in C will parse the variable name and it is assigned value. Then our compiler language will put datatype like int, float, char and double, in front of the variable name and then decide which datatype compiler should provide to variable by using “LHS = RHS” method.
* To read commands or the user input from console, programmers need to use ‘scanf()’ or “scanfln()”. “scanf();” and “scanfln();” will be the inbuild function which deals with console output. The difference between both is if user input is of only one word, then “scanf()” can handle it. But if user input is more than one word, the “scanfln()” can only handle it.
* In our language, which portion of code to get looped through will be decided based on ‘\t’. While analyzing the loop code, the compiler will loop through the code up until it detects ‘\t’. When the compiler will not detect ‘\t’ in the code, it will think the loop code is ended.

**Basic ideas about C implementation**

* We believe that we would struggle in modifying “scanf()” and creating new “scanfln()” because in C language scanf bug is one of the confusing concepts to understand. While creating “scanfln()” command we are not sure how we will deal with scanf bug.

**References**

*[Include eventual references used here]*

* <https://www.w3schools.com/python/python_while_loops.asp>
* <https://www.w3schools.com/python/python_for_loops.asp>
* <https://www.w3schools.com/python/python_functions.asp>
* <https://www.w3schools.com/python/ref_keyword_return.asp>
* <https://www.educative.io/answers/how-to-concatenate-strings-in-python>
* <https://www.techopedia.com/definition/3853/parse#:~:text=To%20parse%2C%20in%20computer%20science,transform%20it%20into%20machine%20language.>
* <https://www.informit.com/articles/article.aspx?p=2469149&seqNum=5>

Algonquin College

Fall, 2022