Rules:

1. Variable and function names cannot consist of the following:

1. Numbers (e.g. one, 2, ten) [Consider using: First, second, third instead]
2. Boolean (e.g. true, false)
3. Variable types (e.g. String, character, integer)
4. Comparison operators (e.g. less than, greater than, less than equal, not equal)
5. Logical operators (e.g. and, or)
6. Unary operators (“minus minus”, “plus plus”, “not”, “negative”)
7. Following keywords: “declare”, “array”, “size”, “with”, “equal”, “equals”, “index”, “if”, “then”, “else”, “end”, “create”, “function”, “return”, “type”, “parameter”, “call”, “for”, “while”, “switch”, “case”, “dot”, “symbol”, “begin”, “point”

2. Variable and function names can be spelled out in words or spelled by characters. (e.g. “circle” vs “c i r c l e”)

3. “ampersand” will be read as “&”, “backslash” will be read as “\”

Structured Language:

|  |  |  |  |
| --- | --- | --- | --- |
| Programming Constructs | Proposed Structured Language | Example speech | Example code |
| Verbalizing words of different forms | Integer: just verbalize the number | Ten  One hundred and thirty four  NOTE: we do not accept “one three four” | 10  134 |
| Float: verbalize the number and use point for the decimal place | Thirty four point two  NOTE: we do not accept “three four point two” | 34.2 |
| String: String (string content) | x equal string nine  x equal string true | x = “nine”  x = “true” |
| Character: character (character content) | x equal character x | X = ‘x’ |
| Boolean: verbalize true or false  [only for languages with Boolean] | is done equal true | isDone = true |
| Symbols: symbol (word) | Symbol percent  Symbol dollar | %  $ |
| if-else conditions | begin if – then – else - end if  (When dealing with nested if-loops, use end if to close the if loop) | begin if A then B begin if C then D else F end if G end if | if (A) {  B  if (C) {  D  } else {  F  }  G  } |
| Variable declaration | declare (variable type) (variable name) end declare  [Optional: equal (literal)] | declare integer x end declare | int x |
| declare float tax rate equal one point zero seven end declare | float taxRate = 1.07 |
| Array declaration | declare (variable type) array (variable name) size (number) end declare  [Optional: with] | declare integer array sequence size ten end declare  declare integer array sequence with size ten end declare | int sequence[10] |
| Variable assignment | (variable name) equal (literal) end equal  [can use equals instead of equal]  [Note: we do not allow equal to as to can be commonly misread as 2] | x equal ten end equal  x equals ten end equal | x = 10 |
| Array assignment | (variable name) array index (index) equal (literal) | sequence array index three equal four | sequence[3] = 4 |
| Function declaration | create function (function name) return type (variable type) parameter (variable type) (variable name) begin – end function  [Optional: with] | create function search return type boolean parameter integer lower parameter integer higher begin A end function  create function search with return type Boolean with parameter integer lower with parameter integer higher begin A end function | bool search(int lower, int higher) {  A  } |
| Function call | call function (function name) parameter (variable name) end function  [Optional: with]  call function (function name) parameter (variable name) end function | call function search parameter lower parameter higher end function  call function search with parameter lower with parameter higher  call function search with parameter lower with parameter higher end function | search(lower, higher) |
| For loops | for loop condition  (variable name) equal (literal)  condition  (variable name) (comparison operator) (literal)  condition  (variable name) (operator) begin – end for loop | for loop condition i equal one condition i less than sum condition i plus plus begin end for loop | for (i=1; i<sum; i++) {  } |
| switch - case | switch (variable name) case (literal) – end switch | switch alphabet case character a X case character b Y end switch | switch(alphabet) {  case a:  X  case b:  Y  } |
| While loops | While (variable name/ true / false) begin – end while | while true begin A end while  while A begin B end while | while(true) {  A  }  while (A) {  B  } |
| Dot operator | dot | human dot call function eat  human dot height | human.eat()  human.height |

Basket of words for testing:

Google Cloud gets better results when preferred\_phrases contain the whole phrase

|  |  |  |  |
| --- | --- | --- | --- |
| Words | Google Results | Google Cloud Results | Microsoft Bing Results |
| Equal | Correct | Correct | Correct |
| If then else | Correct though not 100% | Correct though not 100% | A lot of misreading (best case : “then else”) |
| End if | Correct though not 100% | Correct | A lot of misreading (best case : “end is”) |
| Declare integer | Correct though not 100% | Correct though not 100% | Correct though not 100% |
| Size | Correct | Correct | Correct |
| Index | Correct | Correct | Correct |
| Create function | Correct though not 100% | Correct | Correct |
| Return type | Correct | Correct | Correct |
| Parameter | Correct | Correct | A lot of misreading (best case : “Perimetre” and “para meter”) |
| Call | Correct | Correct | Correct |
| For | Misread as “fall” and “4” | Correct though not 100% | Misread as “fall” and “4” |
| Plus plus | Correct though not 100% (commonly read as “place place”) | Correct though not 100% (perform better than google results) | Correct |
| while | Could not recognize (commonly read as “wild”, “wall”, “Wall-E”) | Could not recognize (commonly read as “wild”, “wall”, “Wall-E”) | Could not recognize (commonly read as “wow”) |
| Switch case end switch | Correct though not 100% and “end” commonly spelled as "and" | Correct though not 100% | Correct though not 100% (“switch” misspelled as “which” sometimes , “end” misread as “and” ) |
| Dot | Read as ‘.’ | Correct | Correct though not 100% (commonly read as “don’t) |

Main problems:

|  |  |  |
| --- | --- | --- |
| Problem | Explanation | Example(s) |
| Homophones | Words that sound similar but have different spelling and meaning | “four” vs “for”  “and” vs “n” |
| Noise or stop words | Words with no meaning | “erm”, “uh” |
| Capitalization | Which alphabet to apply capitalization on | “printStackTrace” vs “PrintStackTrace” |
| Spaces | Whether space is required between words | “Device waterMeter” vs  “deviceWaterMeter” |
| Punctuations | Dot operator, braces, semicolon | “.” , “{” , “}” , “;” |

Sample Program:

pip install SpeechRecognition  
pip install pyaudio

pip install pocketsphinx # only if using recognize\_sphinx  
// python -m speech\_recognition

pip install word2number # for converting words to numbers  
pip install num2words

Problems:

Background noise (especially if great variance)

Homophones

Test Program:

int findMaximum(int numbers[], int length) { // create function find maximum with return type integer with parameter integer array numbers with parameter integer length begin

int max = numbers [0]; // declare integer max equal numbers array index zero

int i; // declare integer i

for (i = 1; i < length; i++) { // for i equal one i less than length i plus plus begin

if (numbers [i] > max) { // if numbers array index i greater than max then

max = numbers [i]; // max equal numbers array index i

} // end if

} // end for

return max; // return max

} // end function

Script for Test Program:

create function find maximum with return type integer with parameter integer array numbers with parameter integer length begin

declare integer max equal numbers array index zero

declare integer i

for i equal one i less than length i plus plus begin

if numbers array index i greater than max then

max equal numbers array index i

end if

end for

return max

end function

|  |
| --- |
| Equal |
| If then else |
| End if |
| Declare integer |
| Size |
| Index |
| Create function |
| Return type |
| Parameter |
| Call |
| For |
| Plus plus |
| while |
| Switch case end switch |
| Dot |

Record:

Week 2: Sample program from speech to text using Google Speech Recognition API with SpeechRecognition python library + structured language defining

Week 3: Sample program using Google Cloud Speech Recognition API with preferred phrases and Microsoft Bing API + structured language redefining (with more natural language and rules)

Week 4: Record 10 audio clips for different voice/tone and try against basket of words, and quantify the results set. Look at Martin FYP AST and try to understand Java. Write a sample program to find max in array using structure language and record as well to test, thereafter quantify results - find how many words are correct out of the total number of words in the script

Week 6: End-to-end program for sample code + improve accuracy

Week 7: Program to accept voice input line by line + parser api to parse

Week 8: Context free grammar to restructure parsing (e.g. for loop --- end for), detect first word to detect parsing, added partial parsing to predicted code

Week 9: