## IS 214 - Assignment 3

Specifications for INTERPOL (Interpreter)

### **Introduction**

INTERPOL, which stands for Integer Program Oriented Language, is a language specifically designed for IS 214 students. Compared to its IS 214 predecessors, INTERPOL is the simplest of them all. What makes it “integer oriented” is that the PL primarily deals with integers and consequently, no floating-point values are allowed. This implies that the results of arithmetic operations are automatically converted to integers. Strings are also allowed for printing purposes.

### **Objectives**

The goal of this programming assignment is to implement an INTERPOL interpreter using Python, which you have learned at the beginning of the semester. Your interpreter should be able to perform the following:

* Accept expressions from the user.
* Parse the code entered by the user and check for lexical and syntax errors.
* Properly execute INTERPOL expressions by the user.

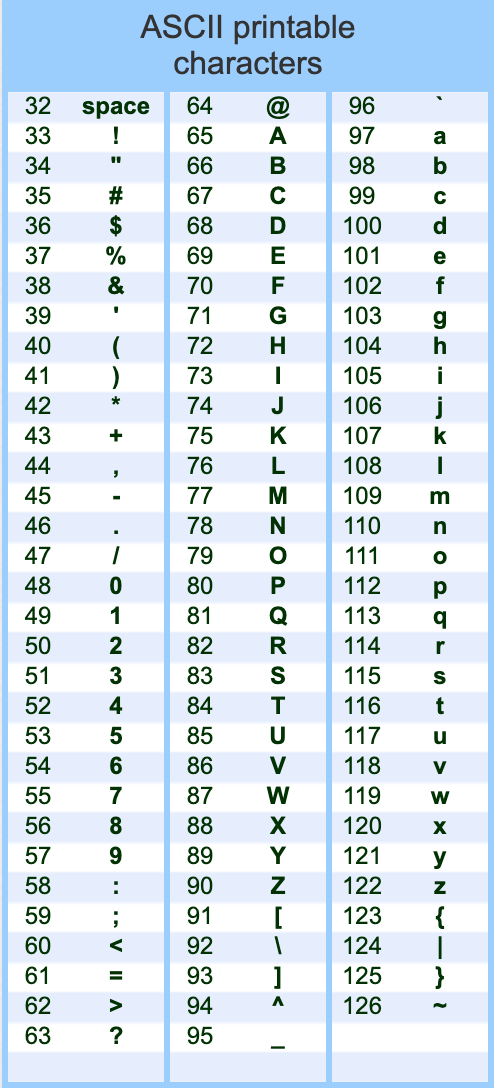
### **Formatting**

* Spaces are used to demarcate tokens in the language.
* Multiple spaces and tabs are treated as single spaces and are otherwise considered irrelevant.
* Indentation is also irrelevant.
* INTERPOL is case sensitive.

### **Syntactic Elements**

1. **Character Set:** set of symbols used in the programming language

Interpol is able to process all ASCII printable characters.



1. **Identifiers:** strings used to name variables, data objects, procedures/functions, etc.

* Should be less than 50 characters
* Should start with a character in the alphabet.
* Can be uppercase/lowercase

1. **Operator Symbols:** symbols used to represent the primitive operations in the language

Interpol does not use operator symbols. (See the section on [Operations](#_bzslychi3m9r) for more details)

1. **Keyword:** an identifier used as a fixed part of the syntax.

**Reserved words**: keywords that may not be used as a programmer-chosen identifier.

In INTERPOL, **all keywords are reserved words.**

1. **Comments:** words ignored during translation

All lines starting with # are considered comments.

1. **Delimiters:** used to mark the beginning or end of some syntactic constructs

In INTERPOL, a new line is a delimiter.

1. **Field Format**

INTERPOL uses a free-field format where program statements can be written anywhere on an input line without regard for positioning.

1. **Expressions**

An expression can be a literal/value, a variable, or it can be composed of several adjacent operators. Hence, nested expressions should be successfully implemented as well in your interpreter.

1. **Statement**

Using the correct syntax, each keyword in INTERPOL can be considered a statement.

1. **Overall Structure**

Programs should begin with the CREATE keyword and end with the RUPTURE keyword.

### **Section: CREATE & RUPTURE**

INTERPOL only has one section. The section begins with the CREATE keyword and ends with the RUPTURE keyword. All coding - variable declarations and actual instructions - are to be placed between these keywords.

|  |  |
| --- | --- |
| **Example:**  CREATE  DSTR stringmo WITH [I have a pen]  GIVEYOU!! stringmo  DINT x WITH PLUS 1 MINUS 9 9  DINT y  STORE 100 IN y  GIVEYOU!! x  GIVEYOU! y  RUPTURE | **Expected Output:**  I have a pen  1  100 |

### **Variables and Data Types**

* There are only 2 main data types: integer and string.
* The interpreter shall follow the principles of strong typing. Therefore, intermixing of integers and strings in arithmetic operations is not allowed.
* All variables are declared and are global in scope.
* Any attempts to use floating-point values should result to an error.
* A string is enclosed by square brackets.

#### How Do You Declare Variables?

|  |  |
| --- | --- |
| Declaring an integer variable without an initial value | DINT <variable\_name> |
| Declaring an integer variable with an initial value | DINT <variable\_name> WITH <expression> |
| Declaring a string variable without an initial value | DSTR <variable\_name> |
| Declaring a string variable with an initial value | DSTR <variable\_name> WITH <expression> |

### **User Input and Output**

|  |  |
| --- | --- |
| Asking input from the user | GIVEME? <variable\_name> |
| Printing values, variables, and expressions | GIVEYOU! <expression> |
| Printing values, variables, and expressions with a new line affixed at the end | GIVEYOU!! <expression> |

### **Operations**

|  |  |  |
| --- | --- | --- |
| **Assignment Statement** | Assignment | STORE <expression> IN <variable> |
| **Basic Arithmetic Operations** | Addition  Subtraction  Multiplication  Division  Modulo | PLUS <expression1> <expression2>  MINUS <expression1> <expression2>  TIMES <expression1> <expression2>  DIVBY <expression1> <expression2>  MODU <expression1> <expression2> |
| **Advanced Arithmetic Operations** | Exponentiation  Nth Root of a No.  Average  Distance between two points | RAISE <expression> <exponent>  ROOT <N> <expression>  MEAN <expr1> <expr2> <expr3> … <exprn>   * This is the only operation that accepts an unlimited number of parameters.     DIST <expr1> <expr2> AND <expr3> <expr4>   * The 2 points are separated by ‘AND’ * The coordinates of the first point are <expression1> and <expression2> * The coordinates of the second point are <expression3> and <expression4> |

**Important Note:** An expression can be a literal/value, a variable, or it can be composed of several adjacent operators. Hence, nested expressions should be successfully implemented as well in your interpreter.

### **Interface and Flow**

* Do not use graphical user interface, databases (eg. MySQL), and other unnecessary libraries in Python. Focus on what’s core in this assignment – the interpreter itself.
* Your interface should be command line.
* Implement your interpreter with the following use case in mind:
  1. The user writes his INTERPOL source code using any text editor. Take note that the source code has “.ipol” extension.
  2. From the command prompt, the user runs your interpreter. Your interpreter then asks for the file name of the .ipol code to be executed.
  3. Still in the command prompt, the following outputs are displayed to the user:
     + Actual output of the source code
       - If there’s an error, the user should be informed of the error and possibly, the source of the error.
     + Symbol table
     + Lexemes and tokens table

### **Submission**

* Submit your programming assignment on or before Wednesday, 11 December 2019 at 8:59:59 am (GMT+8).
* Your assignment should be in a single Python source code that contains your implementation of the INTERPOL interpreter.
  + Make sure to use Python 3+!
  + Name your file as first and middlename initials-lastname)-03.py
  + Example: Since my name is Ria Mae H. Borromeo, the filename of my assignment should be rmh-borromeo-03.py
  + Submit your file in the submission bin
  + Please don’t forget to document your code properly by using comments.

## Error Messages

|  |  |
| --- | --- |
| Invalid syntax | Syntax is invalid |
| Variable is not declared | A variable must be declared before use |
| Invalid arithmetic operation | General error for |
| Invalid expression | Invalid expression |
| Invalid data type | Invalid data type (i.e. using float) |
| Duplicate variable declaration | Variable name has been declared before |
| Incompatible data type | If an operation on incompatible data types is done |
| Invalid data type input | If the data type inputted is invalid (i.e. float) |
| Invalid end of file | If there is no RUPTURE |
| File is empty | File is empty |
| File not found | File does not exist |
| Invalid file | Input file does not end with .ipol |

## Sample Files and Output

### 1. test01.ipol

**input**

|  |
| --- |
| CREATE  DSTR stringmo WITH [I have a pen]  GIVEYOU!! stringmo  DINT x WITH PLUS 1 MINUS 9 9  DINT y  STORE 100 IN y  GIVEYOU!! x  GIVEYOU! y  RUPTURE |

**output**

|  |
| --- |
| $ python3 interpol.py  ======== INTERPOL INTERPRETER STARTED ========  Enter INTERPOL file (.ipol): test01.ipol  ================ INTERPOL OUTPUT ================  ---------------- OUTPUT START ---------------->  I have a pen  1  100  <----------------- OUTPUT END -------------------  ========= INTERPOL LEXEMES/TOKENS TABLE =========  LINE NO. TOKENS LEXEMES  1 PROGRAM\_CREATE CREATE  1 END\_OF\_STATEMENT EOS  2 DECLARATION\_STRING DSTR  2 IDENTIFIER stringmo  2 DECLARATION\_ASSIGN\_WITH\_KEY WITH  2 STRING I have a pen  2 END\_OF\_STATEMENT EOS  3 OUTPUT\_WITH\_LINE GIVEYOU!!  3 IDENTIFIER stringmo  3 END\_OF\_STATEMENT EOS  4 DECLARATION\_INT DINT  4 IDENTIFIER x  4 DECLARATION\_ASSIGN\_WITH\_KEY WITH  4 BASIC\_OPERATOR\_ADD PLUS  4 NUMBER 1  4 BASIC\_OPERATOR\_SUB MINUS  4 NUMBER 9  4 NUMBER 9  4 END\_OF\_STATEMENT EOS  5 DECLARATION\_INT DINT  5 IDENTIFIER y  5 END\_OF\_STATEMENT EOS  6 ASSIGN\_KEY STORE  6 NUMBER 100  6 ASSIGN\_VAR\_KEY IN  6 IDENTIFIER y  6 END\_OF\_STATEMENT EOS  7 OUTPUT\_WITH\_LINE GIVEYOU!!  7 IDENTIFIER x  7 END\_OF\_STATEMENT EOS  8 OUTPUT GIVEYOU!  8 IDENTIFIER y  8 END\_OF\_STATEMENT EOS  9 PROGRAM\_RUPTURE RUPTURE  10 END\_OF\_FILE EOF  ================= SYMBOLS TABLE =================  VARIABLE NAME TYPE VALUE  stringmo STRING I have a pen  x INTEGER 1  y INTEGER 100  ======== INTERPOL INTERPRETER TERMINATED ======== |

### 2. test02.ipol

input

|  |
| --- |
| CREATE  DSTR stringmo WITH "Enter an integer: "  GIVEME!! stringmo  RUPTURE |

output

|  |
| --- |
| $ python3 interpol.py  ======== INTERPOL INTERPRETER STARTED ========  Enter INTERPOL file (.ipol): test02.ipol  ================ INTERPOL OUTPUT ================  ---------------- OUTPUT START ---------------->  Invalid character at line number [ 2 ]  ----> DSTR stringmo WITH " |

### 3. test03.ipol

input

|  |
| --- |
| CREATE  DSTR stringmo WITH [Enter an integer: ]  GIVEYOU!! stringmo  DINT var  GIVEME? var  GIVEYOU!! MINUS 2 2  GIVEYOU!! TIMES 1 5  GIVEYOU!! DIVBY 1 1  GIVEYOU!! MODU 5 4  GIVEYOU!! RAISE 2 16  GIVEYOU!! ROOT 16 2  GIVEYOU!! MEAN 1 2 3 4 5  GIVEYOU!! DIST 1 2 AND 3 4  RUPTURE |

output

|  |
| --- |
| $ python3 interpol.py  ======== INTERPOL INTERPRETER STARTED ========  Enter INTERPOL file (.ipol): test03.ipol  ================ INTERPOL OUTPUT ================  ---------------- OUTPUT START ---------------->  Enter an integer:  12  0  5  1  1  65536  1  3  2  <----------------- OUTPUT END -------------------  ========= INTERPOL LEXEMES/TOKENS TABLE =========  LINE NO. TOKENS LEXEMES  1 PROGRAM\_CREATE CREATE  1 END\_OF\_STATEMENT EOS  2 END\_OF\_STATEMENT EOS  3 DECLARATION\_STRING DSTR  3 IDENTIFIER stringmo  3 DECLARATION\_ASSIGN\_WITH\_KEY WITH  3 STRING Enter an integer:  3 END\_OF\_STATEMENT EOS  4 OUTPUT\_WITH\_LINE GIVEYOU!!  4 IDENTIFIER stringmo  4 END\_OF\_STATEMENT EOS  5 DECLARATION\_INT DINT  5 IDENTIFIER var  5 END\_OF\_STATEMENT EOS  6 INPUT GIVEME?  6 IDENTIFIER var  6 END\_OF\_STATEMENT EOS  7 END\_OF\_STATEMENT EOS  8 OUTPUT\_WITH\_LINE GIVEYOU!!  8 BASIC\_OPERATOR\_SUB MINUS  8 NUMBER 2  8 NUMBER 2  8 END\_OF\_STATEMENT EOS  9 OUTPUT\_WITH\_LINE GIVEYOU!!  9 BASIC\_OPERATOR\_MUL TIMES  9 NUMBER 1  9 NUMBER 5  9 END\_OF\_STATEMENT EOS  10 OUTPUT\_WITH\_LINE GIVEYOU!!  10 BASIC\_OPERATOR\_DIV DIVBY  10 NUMBER 1  10 NUMBER 1  10 END\_OF\_STATEMENT EOS  11 OUTPUT\_WITH\_LINE GIVEYOU!!  11 BASIC\_OPERATOR\_MOD MODU  11 NUMBER 5  11 NUMBER 4  11 END\_OF\_STATEMENT EOS  12 OUTPUT\_WITH\_LINE GIVEYOU!!  12 ADVANCED\_OPERATOR\_EXP RAISE  12 NUMBER 2  12 NUMBER 16  12 END\_OF\_STATEMENT EOS  13 OUTPUT\_WITH\_LINE GIVEYOU!!  13 ADVANCED\_OPERATOR\_ROOT ROOT  13 NUMBER 16  13 NUMBER 2  13 END\_OF\_STATEMENT EOS  14 OUTPUT\_WITH\_LINE GIVEYOU!!  14 ADVANCED\_OPERATOR\_AVE MEAN  14 NUMBER 1  14 NUMBER 2  14 NUMBER 3  14 NUMBER 4  14 NUMBER 5  14 END\_OF\_STATEMENT EOS  15 OUTPUT\_WITH\_LINE GIVEYOU!!  15 ADVANCED\_OPERATOR\_DIST DIST  15 NUMBER 1  15 NUMBER 2  15 DISTANCE\_SEPARATOR AND  15 NUMBER 3  15 NUMBER 4  15 END\_OF\_STATEMENT EOS  16 END\_OF\_STATEMENT EOS  17 PROGRAM\_RUPTURE RUPTURE  18 END\_OF\_FILE EOF  ================= SYMBOLS TABLE =================  VARIABLE NAME TYPE VALUE  stringmo STRING Enter an integer:  var INTEGER 12  ======== INTERPOL INTERPRETER TERMINATED ======== |

### 4. test04.ipol

input

|  |
| --- |
| CREATE  DSTR inputString WITH [I have a pen]  GIVEYOU!! inputString  GIVEYOU! [Enter year of birth: ]  DINT birthyear  GIVEME? birthyear  DSTR lit WITH [You are ]  DINT year WITH 2019  GIVEYOU! lit  GIVEYOU!! MINUS year birthyear  DINT x WITH PLUS 100 MINUS 9 7  DINT y  STORE 100 IN y  GIVEYOU! [x = ]  GIVEYOU!! x  GIVEYOU! [y = ]  GIVEYOU!! y  DINT m  STORE 2 IN m  GIVEYOU! [ROOT RAISE 2 2 2 = ]  GIVEYOU!! ROOT RAISE m m m  GIVEYOU! [MEAN 1 2 3 4 5 6 = ]  GIVEYOU!! MEAN 1 2 3 4 5 6  RUPTURE |

output

|  |
| --- |
| $ python3 interpol.py  ======== INTERPOL INTERPRETER STARTED ========  Enter INTERPOL file (.ipol): test03.ipol  ================ INTERPOL OUTPUT ================  ---------------- OUTPUT START ---------------->  I have a pen  Enter year of birth: 1986  You are 33  x = 102  y = 100  ROOT RAISE 2 2 2 = 1  MEAN 1 2 3 4 5 6 = 3  <----------------- OUTPUT END -------------------  ========= INTERPOL LEXEMES/TOKENS TABLE =========  LINE NO. TOKENS LEXEMES  1 PROGRAM\_CREATE CREATE  1 END\_OF\_STATEMENT EOS  2 END\_OF\_STATEMENT EOS  3 DECLARATION\_STRING DSTR  3 IDENTIFIER inputString  3 DECLARATION\_ASSIGN\_WITH\_KEY WITH  3 STRING I have a pen  3 END\_OF\_STATEMENT EOS  4 OUTPUT\_WITH\_LINE GIVEYOU!!  4 IDENTIFIER inputString  4 END\_OF\_STATEMENT EOS  5 END\_OF\_STATEMENT EOS  6 OUTPUT GIVEYOU!  6 STRING Enter year of birth:  6 END\_OF\_STATEMENT EOS  7 DECLARATION\_INT DINT  7 IDENTIFIER birthyear  7 END\_OF\_STATEMENT EOS  8 INPUT GIVEME?  8 IDENTIFIER birthyear  8 END\_OF\_STATEMENT EOS  9 DECLARATION\_STRING DSTR  9 IDENTIFIER lit  9 DECLARATION\_ASSIGN\_WITH\_KEY WITH  9 STRING You are  9 END\_OF\_STATEMENT EOS  10 DECLARATION\_INT DINT  10 IDENTIFIER year  10 DECLARATION\_ASSIGN\_WITH\_KEY WITH  10 NUMBER 2019  10 END\_OF\_STATEMENT EOS  11 OUTPUT GIVEYOU!  11 IDENTIFIER lit  11 END\_OF\_STATEMENT EOS  12 OUTPUT\_WITH\_LINE GIVEYOU!!  12 BASIC\_OPERATOR\_SUB MINUS  12 IDENTIFIER year  12 IDENTIFIER birthyear  12 END\_OF\_STATEMENT EOS  13 END\_OF\_STATEMENT EOS  14 DECLARATION\_INT DINT  14 IDENTIFIER x  14 DECLARATION\_ASSIGN\_WITH\_KEY WITH  14 BASIC\_OPERATOR\_ADD PLUS  14 NUMBER 100  14 BASIC\_OPERATOR\_SUB MINUS  14 NUMBER 9  14 NUMBER 7  14 END\_OF\_STATEMENT EOS  15 DECLARATION\_INT DINT  15 IDENTIFIER y  15 END\_OF\_STATEMENT EOS  16 ASSIGN\_KEY STORE  16 NUMBER 100  16 ASSIGN\_VAR\_KEY IN  16 IDENTIFIER y  16 END\_OF\_STATEMENT EOS  17 OUTPUT GIVEYOU!  17 STRING x =  17 END\_OF\_STATEMENT EOS  18 OUTPUT\_WITH\_LINE GIVEYOU!!  18 IDENTIFIER x  18 END\_OF\_STATEMENT EOS  19 OUTPUT GIVEYOU!  19 STRING y =  19 END\_OF\_STATEMENT EOS  20 OUTPUT\_WITH\_LINE GIVEYOU!!  20 IDENTIFIER y  20 END\_OF\_STATEMENT EOS  21 END\_OF\_STATEMENT EOS  22 DECLARATION\_INT DINT  22 IDENTIFIER m  22 END\_OF\_STATEMENT EOS  23 ASSIGN\_KEY STORE  23 NUMBER 2  23 ASSIGN\_VAR\_KEY IN  23 IDENTIFIER m  23 END\_OF\_STATEMENT EOS  24 END\_OF\_STATEMENT EOS  25 OUTPUT GIVEYOU!  25 STRING ROOT RAISE 2 2 2 =  25 END\_OF\_STATEMENT EOS  26 OUTPUT\_WITH\_LINE GIVEYOU!!  26 ADVANCED\_OPERATOR\_ROOT ROOT  26 ADVANCED\_OPERATOR\_EXP RAISE  26 IDENTIFIER m  26 IDENTIFIER m  26 IDENTIFIER m  26 END\_OF\_STATEMENT EOS  27 END\_OF\_STATEMENT EOS  28 OUTPUT GIVEYOU!  28 STRING MEAN 1 2 3 4 5 6 =  28 END\_OF\_STATEMENT EOS  29 OUTPUT\_WITH\_LINE GIVEYOU!!  29 ADVANCED\_OPERATOR\_AVE MEAN  29 NUMBER 1  29 NUMBER 2  29 NUMBER 3  29 NUMBER 4  29 NUMBER 5  29 NUMBER 6  29 END\_OF\_STATEMENT EOS  30 END\_OF\_STATEMENT EOS  31 PROGRAM\_RUPTURE RUPTURE  32 END\_OF\_FILE EOF  ================= SYMBOLS TABLE =================  VARIABLE NAME TYPE VALUE  inputString STRING I have a pen  birthyear INTEGER 1986  lit STRING You are  year INTEGER 2019  x INTEGER 102  y INTEGER 100  m INTEGER 2  ======== INTERPOL INTERPRETER TERMINATED ======== |