Programming language is like telling a story.

How variables are defined: A <type> called <identifier>

<https://morsecode.scphillips.com/morse2.html>

BNF

**<Program> ::= -... . --. .. -. / { <dataDefinition> } \***

**{ (( <FUNCTIONDefinition> )) }\***

**<dataDefinition> ::= <variableDefinition> | <constantDefinition>**

**<variableDefinition> ::= <datatype> <identifier> [ [ <LBUBRange> ] ]**

**<datatype> ::= ( .. -. - | -.. -... .-..**

**| ... - .-. | -... --- --- .-..**

**| -.-. .... .-.)**

**<FUNCTIONDefinition> ::= -...- <identifier> < ( <formalParameter> {, <formalParameter> }\* ) >**

**{ <dataDefinition> }\***

**{ <statement> }\***

**.-.-.-**

**<formalParameter> ::= [ <identifier> <datatype> [ | {,}\* | ] ]**

**<statment> ::= { <assertion> }\***

**(( <PRINTStatement> | <INPUTStatment>**

**| <assignmentStatement> | <IFStatement>**

**| <FORStatement> | <DOWHILEStatement>**

**| <CALLStatement> | <RETURNStatement>**

**)) { <assertion> }\***

**<assertion> ::= <expression>**

**<PRINTStatment> ::= .--. .-. .. -. - / (( <string> | <expression> )) { --..-- / (( <string> | <expression> )) }\* / -.-**

**<INPUTStatment> ::= .. -. .--. ..- - / <string> / <variable> / -.-**

**<IFStatment> ::= .. ..-. / < <expression> > /**

**{ <statment> }\* / .-.-.-**

**<DOWHILEStatement> ::= -.. --- /**

**| { <statement> }\* |**

**/ .-- .... .. .-.. . / < <expression> > / -.-**

**<FORStatement> ::= ..-. --- .-. / < <expression> / -.-**

**/ <expression> / -.-**

**/ <expression> > /**

**| { <statement> }\* |**

**/ -.-**

**<CALLStatement> ::= <identifier> / -.-**

**<RETURNStatement> ::= .-. - .-. -. / [ ( <expression> ) ] /** **-.-**

**<expression> ::= <conjuction> { (( | | )) }\***

**<conjuction> ::= <negation> / { (( --- .-. | -. --- - ))**

**/ <negation> }\***

**<negation> ::= [ -. --- - ] <comparison>**

**<comparison> ::= <comparator> [ / (( )) / <comparator> ]**

**<comparator> ::= <term { (( + | - )) <term> }\***

**<term> ::= <factor> { (( –- | -.. )) <factor> }\***

**<factor> ::= [ .-.-. | -....- ] <secondary>**

**<prefix>**

**<identifier> ::= (( <letter> )) { (( <letter> | <digit> | \_ )) }\***

**<literal> ::= <integer> | <double> | <boolean> | <string>**

**<integer> ::= <digit> { <digit> }\***

**<float> ::= <integer> , <integer>**

**<variable> ::= <identifier> [[ <expression> ]]**

**<boolean> ::= - .-. ..- . | ..-. .- .-.. … .**

**<string> ::= \* / { <ASCIICharacter> }\* / \***

**<letter> ::= A | B | ... | Z | a | b | ... | z**

**<digit> ::= 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9**

**<ASCIICharacter> ::= || Every printible ASCII character in range**

**[‘ ’,’~’]**

**<comment> ::= << { <ASCIICharacter> }\* >>**

**KEYWORDS**

**BEGIN (BEGIN): -... . –-. .. -.**

**INTEGER (INT): .. -. -**

**DOUBLE (DBL): -.. -... .-..**

**STRING (STR): ... - .-.**

**BOOLEAN (BOOL): -... --- --- .-..**

**CHARACTER (CHR): -.-. .... .-.**

**END STMT (K): -.-**

**FUNCTION (BT): -...-**

**END FUNCTION (PERIOD): .-.-.-**

**COMMA: --..--**

**IF (IF):** **.. ..-.**

**DO (DO): -.. ---**

**WHILE (WHILE): .-- .... .. .-.. .**

**FOR (FOR): ..-. --- .-.**

**PRINT (PRINT): .--. .-. .. -. -**

**INPUT (INPUT): .. -. .--. ..- -**

**RETURN (RTRN): .-. - .-. -.**

**TRUE (TRUE): - .-. ..- .**

**FALSE (FALSE):** **..-. .- .-.. … .**

**OR (OR): --- .-.**

**AND (AND): .- -. -..**

**NOT (NOT): -. --- -**

**MULTIPLY (M): --**

**DIVIDE (D): -..**

**ADD (+): .-.-.**

**MINUS (-): -....-**

**QUOTE (“): .-..-.**