Programming Language (PlainEnglish)

Authors

Asit Singh and Joe Hudson
Joe Tasks: Justification, Program
Asit Tasks: Grammar, Definitions
The rest were a collective effort.

Extension

.pe -> so for example a file written using the PlainEnglish language will be called, foo.pe

Reason/Justification

Most educational platforms that teach programming languages are either very simple, such as in the form of a video game that tries to imitate a programming language, or just complicated tutorials that are not the best way to learn for absolute beginners, especially students who are trying to learn at an early age, such as 5.

We came with a programming language that is very similar to plain english, and therefore called PlainEnglish, which uses simple functions that could be used to code and teach how coding works without learning complicated syntaxes and the theory behind various data structures and algorithms.

To act as an introduction to coding and will help young children to see what it is like to code without getting intimidated by it. Additionally, it will help students who do not enjoy coding, learn that fact about themselves without wasting a lot of time and mental energy. So PlainEnglish, in the long run will be able to save people time and because “Time is money” it will save the world money as well.

We also tried to make this language turing complete while keeping it very-simple to use. This language can implement loops, via using a combination of if and goto, it can implement basic arithmetic functions, such as addition, subtraction, multiplication and division, and it can also implement logic functions, such as equal to, greater/smaller than, if/else and great/smaller than or equal to. We implemented loops using if and goto because we believe it is more intuitive and easier to use/implement for a simple educational language like PlainEnglish. Although the definition of Turing Complete is a little vague and the fact that this language is still in its development phase, we do not think this language is Turing Complete yet, but has the potential to be. We plan on actually implementing the whole language and providing support for it as we do believe that this language could be very useful for students who are trying to learn how to code!

Goal and Target audience

Provide a super simple programming language to be used for introducing coding languages to young children (between ages of 3-10) who are absolute beginners.

Definitions

Variable types supported:

Int: Supports all arithmetic {Add(), Subtract(), Multiply(), Divide(), Mod()} and logical operations {Greater(), GreaterOrEqual(), Smaller(), SmallerOrEqual(), Equal()}
String: Supports the operation Add()
Float: Supports all arithmetic {Add(), Subtract(), Multiply(), Divide(), Mod()} and logical operations {Greater(), GreaterOrEqual(), Smaller(), SmallerOrEqual(), Equal()}
Bool: Supports the logical operation Equal()

Declare(int, a):

Used for Declaring a new variable. Takes two arguments as parameters, first one to define the type of the variable; second one for defining the name of the variable. Initiates the new variable as nil but retains the type as defined.

Assign(a, 5):

Used to assign value to a variable. Takes two arguments as parameters, first one is the variable that the value is supposed to be assigned to; second parameter provides the value that the variable is to be assigned.

Add(5, 6) or Add(a, 5) or Add(a, b):

Used to increment a variable by a certain amount. Takes two arguments as parameters, first one is the variable name to be incremented; second one is the value that the variable is going to be incremented by.

Subtract(5, 6) or Subtract(a, 5) or Subtract(a, b):

Used to decrement a variable by a certain amount. Takes two arguments as parameters, first one is the variable name to be decremented; second one is the value that the variable is going to be decremented by.

Multiply(5, 6) or Multiply(a, 5) or Multiply(a, b):

Used to multiply a variable by a certain amount. Takes two arguments as parameters, first one is the variable name to be multiplied; second one is the value that the variable is going to be multiplied by.

Divide(5, 6) or Divide(a, 5) or Divide(a, b):

Used to divide a variable by a certain amount. Takes two arguments as parameters, first one is the variable name to be divided; second one is the value that the variable is going to be divided by.

Mod(5, 6) or)Mod(a, 5) or Mod(a, b):

Used to find the remainder of the division performed. Takes two arguments as parameters, first one is the variable name to be divided; second one is the value that the variable is going to be divided by.

Greater(5, 6) or Greater(a, b) or Greater (a, 6):

Used to decide which variable is greater. Takes two arguments as parameters, first one is the first variable name to be checked; second one is the second variable name to be used for comparison. Returns a boolean value.

GreaterOrEqual(5, 6) or GreaterOrEqual(a, b) or GreaterOrEqual(a, 6):

Used to decide whether the variable is greater than or equal to a given variable. Takes two arguments as parameters, first one is the first variable name to be checked; second one is the second variable name to be used for comparison. Returns a boolean value.

Smaller(5, 6) or Smaller(a,b) or Smaller(a, 5):

Used to decide which variable is smaller. Takes two arguments as parameters, first one is the first variable name to be checked; second one is the second variable name to be used for comparison. Returns a boolean value.

SmallerOrEqual(5, 6) or SmallerOrEqual(a, b) or SmallerOrEqual(a, 6):

Used to decide whether the variable is smaller than or equal to a given variable. Takes two arguments as parameters, first one is the first variable name to be checked; second one is the second variable name to be used for comparison. Returns a boolean value.

Equal(5, 6) or Equal(a,b) or Equal(a, 5):

Used to decide whether the variables are equal. Takes two arguments as parameters, first one is the first variable name to be checked; second one is the second variable name to be used for comparison. Returns a boolean value.

Print(a) or Print("Hello World"):

Used to print the argument that is passed. Takes one argument as a parameter which is the object that is supposed to be printed.

DeclareAndAssign(int, a, 5):

Used to declare and assign a new variable. Takes three arguments as parameters, first one defines the type of the variable; second one defines the name of the variable; third one provides the value to be assigned to the newly created variable.

If(Greater(a, b)) or If(Smaller(a,b)) or If(Equal(a,b)) or If() or If(GreaterOrEqual(a, b)) or If(SmallerOrEqual(a, b)):

Used to check whether the condition described within the outside parentheses is true or false. Takes a logic statement as input and returns either True or False. Returns a boolean value.

Else():

Used when the logic statement, If(), returns False.

GoTo(here):

Used to loop back to a certain line in order to re-execute everything after here until GoTo again. Takes one parameter which shows where to loop back from.

End():

Used to end the Program. Nothing after this statement will be executed.

Grammar

DeclareAndAssign => Variable VARNAME EQ Variable
Declare => variable VARNAME
Assign => VARNAME EQ Variable
Equal => VARNAME EQUALITY VARNAME
Add => Variable ADDI Variable | VARNAME ADDI Variable | VARNAME ADDI VARNAME
Subtract => Variable SUBI Variable | VARNAME SUBI Variable | VARNAME SUBI VARNAME
Multiply => Variable MULT Variable | VARNAME MULT Variable | VARNAME MULT VARNAME
Divide => Variable DIV Variable | VARNAME DIV Variable | VARNAME DIV VARNAME
Mod => Variable MOD Variable | VARNAME MOD Variable | VARNAME MOD VARNAME
If => IF Greater | IF Smaller | IF Equal | IF GreaterOrEqual | IF SmallerOrEqual
Else => ELSE
Greater => Variable GRET Variable | VARNAME GRET VARNAME | VARNAME GRET NUM
GreaterOrEqual => Variable GRET EQUALITY Variable | VARNAME GRET EQUALITY VARNAME | VARNAME GRET EQUALITY Variable
Smaller => Variable LESST Variable | VARNAME LESST VARNAME | VARNAME LESST NUM
SmallerOrEqual => Variable LEEST EQUALITY Variable | VARNAME LESST EQUALITY VARNAME | VARNAME LESST EQUALITY Variable
Variable => INT | STRING | FLOAT | BOOL
Print => PRINT Variable | PRINT VARNAME
GoTo => GOTO HERE
End => END

Terminals

VARNAME => “Name”
INT => [0-9]+
STRING => [a-zA-Z]*
FLOAT => [0.00000000000000-9.9999999999999999]+
BOOL => “True” | “False”
ASSIGN => “=”
EQUALITY => “==”
ADDI => “+”

SUBI => “-”
MULT => “*”
DIV => “/”
MOD => “%”
GRET => “>”
LESST => “<”
IF => “if”
ELSE => “else”
PRINT => “print”
HERE => “here”
GOTO => “goto”
END => “end”

Special Conditions

As the main purpose of the language is to introduce coding to young children (between the ages of 3-10) who are absolute beginners, the language does not support nesting functions as it is more complex than we want this language to be. Although, we thought everyone should still be introduced to how nesting functions would work and therefore we are going to give the If() function a super-power! Called Nestification!

Examples

Program 1: (This program shows how any PlainEnglish language will end after the End() statement, without executing anything after the End() statement)

```
Declare(int, a)
Assign(a, 5)
DeclareAndAssign(int, b, 6)
Print(a)
Print(b)

Print("Hello World")

End()

If(Smaller(a, b))
        Print(“a is bigger than b”)
else()
        Print(“a is smaller than b”)
```

Program 2: (This program shows the implementation for every function that has bee implemented)

```
Declare(int, a)
Declare(string, f)
Assign(f, hello)
Declare(string, d)
Assign(d, world)
Assign(a, 5)
DeclareAndAssign(int, b, 6)
Declare(bool, c)
Assign(c, True)

Add(f, d)
Add(a, 5)
Add(a, b)
Add(5, 6)
Subtract(a, 5)
Subtract(a, b)
Subtract(5, 6)
Multiply(a, 5)
Multiply(a, b)
Multiply(5, 6)
Divide(a, 5)
Divide(a, b)
Divide(5, 6)
Mod(a, 5)
Mod(a, b)
Mod(5, 6)
Greater(a, 5)
Greater(a, b)
Greater(5, 6)
Smaller(a, 5)
Smaller(a, b)
Smaller(5, 6)
GreaterOrEqual(a, 5)
GreaterOrEqual(a, b)
GreaterOrEqual(5, 6)
SmallerOrEqual(a, 5)
SmallerOrEqual(a, b)
SmallerOrEqual(5, 6)

End()
```

Program 3: (This program shows how the loop would function in PlainEnglish)

```
Declare(int, a)
Assign(a, 6)
DeclareAndAssign(int, b, 7)
```

```
here
Subtract(a, 1)
If(GreaterOrEqual(a, 0))
        GoTo(here)
End()
```

Program 4: (This program shows the implementation of some more If() and Else() statements)

```
Declare(int, a)
Assign(a, 6)
DeclareAndAssign(int, b, 7)

If(Greater(a, b))
        print("a is greater than b and PlainEnglish is awesome!")

If(Smaller(a, b))
        print("a is smaller than b and PlainEnglish is awesome!")

If(Equal(a, b))
        print("a is equal to b and PlainEnglish is awesome!")

Else()
        print("This language sucks and does not work!")

End()
```

Program 5: (This program shows our implementation for the error handling capabilities for our program)

```
Declare(char, a)
Assign(z, hello)
DeclareAndAssign(int, b, hello)

Add(a, z)
Subtract(5)
Greater(5)
Print(1)

End()
```

Future Growth Potential

1) New function support:

DeclareFun(“FunName(x)”, “Adds 1 to the integer passed”):

Further, once natural language processing is advanced enough, we can use that to implement the DeclareFun() function which can declare a function as defined using the parameters sent. It will take two parameters as arguments; the first parameter defines the function name and how many parameters the newly defined function is going to receive as arguments. The second parameter would describe what the newly defined function does, in plain english, which a natural language processor would read and implement, kind of similar to what OpenGPT 3 does but it has to be a little more advanced to be fully functional. Although, GitHub CoPilot could also be used to implement this and we thought of a way in which it could be made possible but we did not have enough time to implement it or get the required paperwork done in order to not do anything illegal.

2) Add support for more experienced users of the language. In other words, provide another bridge between PlainEnglish and other more popularly used languages such as Python.
    a) Add support for nesting more functions together for an advanced mode version of the language. That will increase capabilities and make coding in this language easier for the more experienced coders.

3) Add logical operation support for the string variable, such as Greater(), Smaller(), etc.
4) Add the logical operation Equal() support for boolean
5) Implement a more robust error handling
6) Add looping abilities by using If() and GoTo()

Implementation

Disclaimer:
Due to time constraints we were only able to implement some of the functions of the language PlainEnglish. The language is still in beta phase and therefore is buggy. We have tried our best to fix all the bugs that we could find but there are still certain bugs in the implementation that need to be addressed. We plan on continuing development of the project as we truly believe that this language could actually benefit children who want to learn how to code!

Following are the bugs that we are aware of and are working on fixing:
1) Only the Program 2 works fully as If, Else, GoTo and here were not implemented.
2) Loop has not been implemented as If, Else, GoTo has not been implemented.
3) Error handling works but not everytime
4) DeclareAndAssign() function does not work well everytime.

Functions Implemented:
Declare(), Assign(), DeclareAndAssign() Add(), Subtract(), Multiply(), Divide(), Mod(), Greater(), GreaterOrEqual(), Smaller(), SmallerOrEqual(), Equal(), Print() and End()

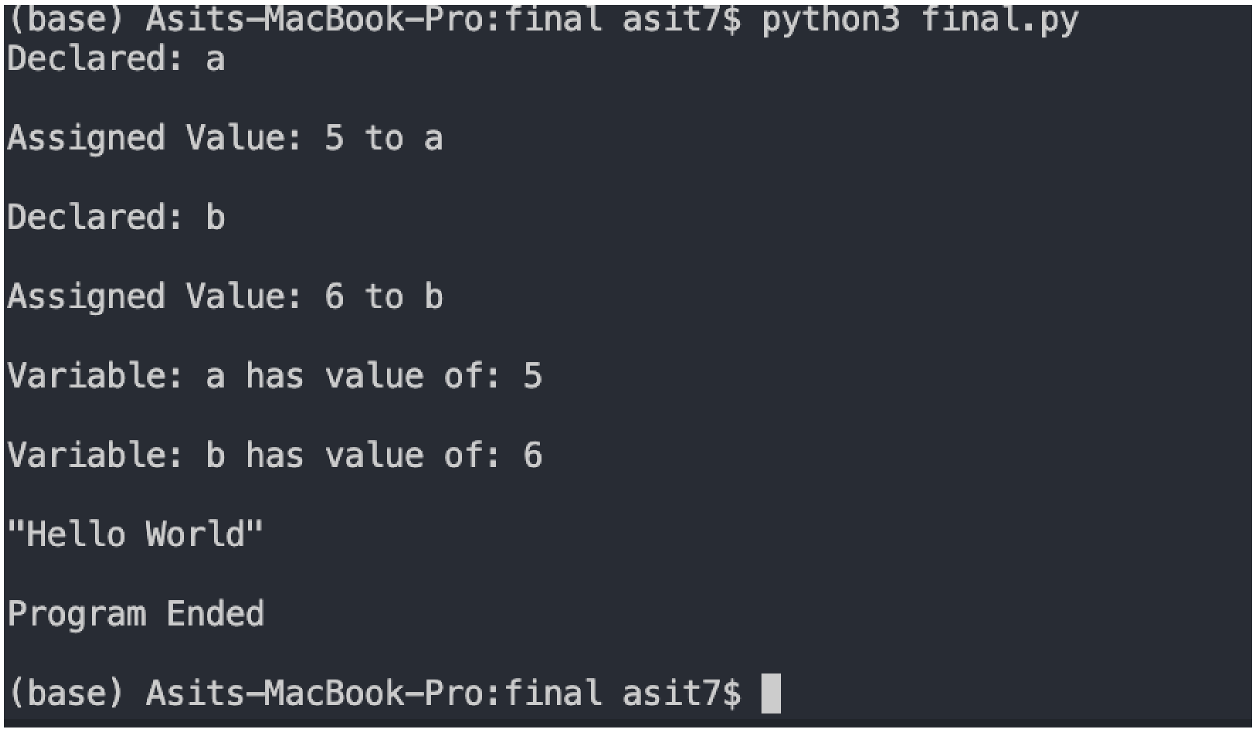
Functions Not Implemented:
If(), Else() and GoTo()

Programs Implemented:
Program1.pe, Program2.pe and Program5.pe

Programs Not Implemented:
Program3.pe and Program4.pe

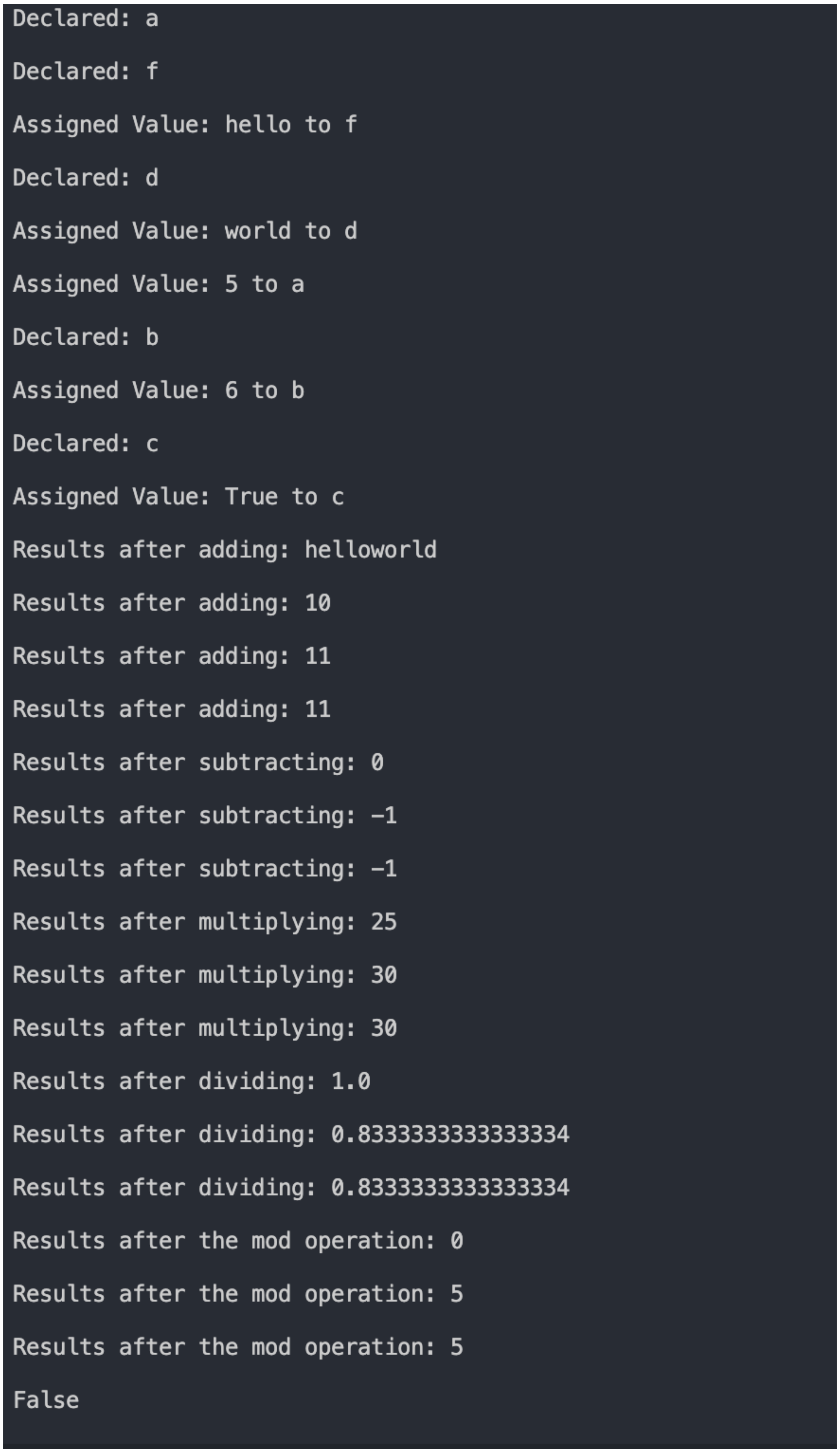
Output of the programs 1, 2 and 5:

Program 1:

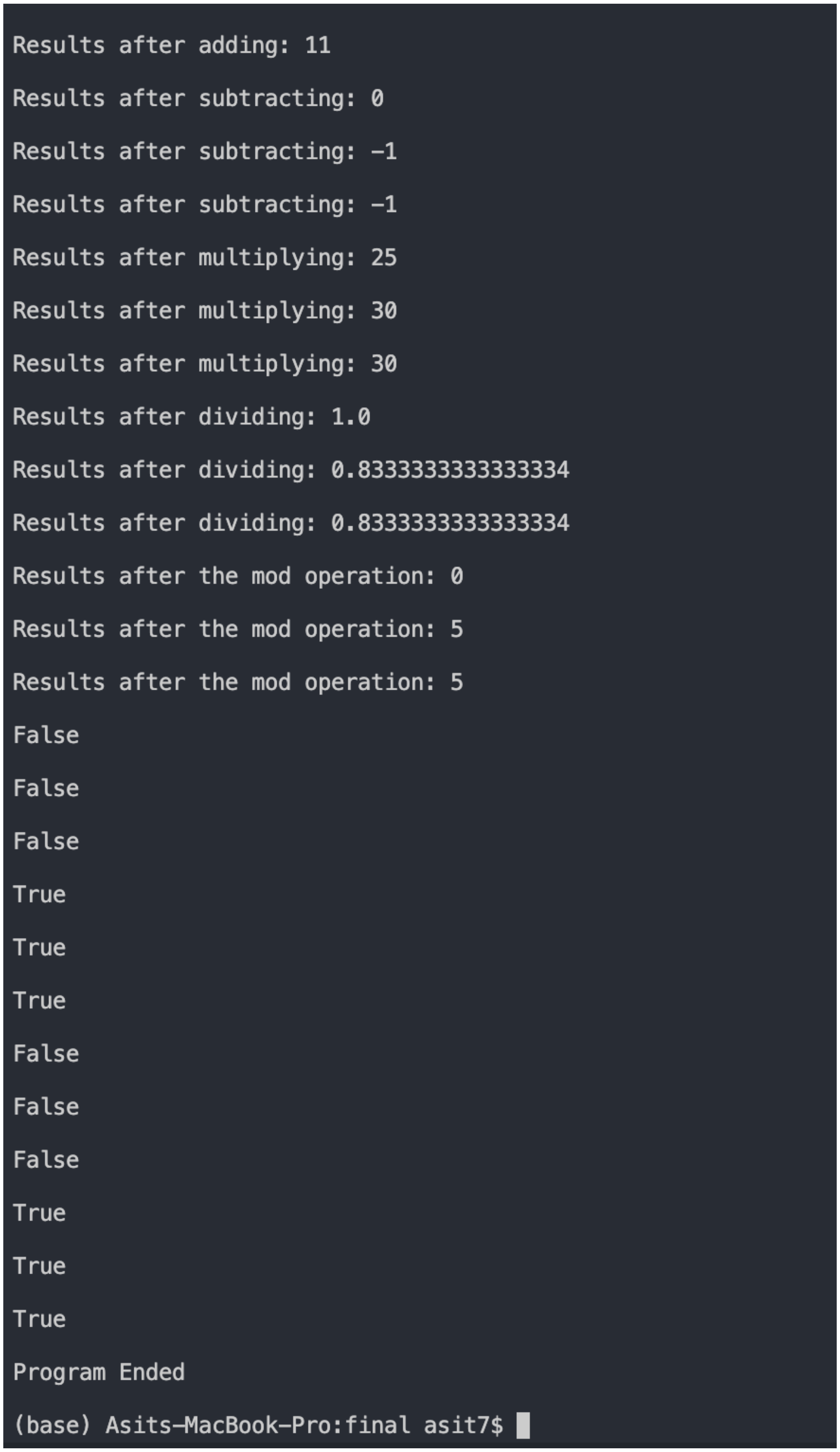


```
(base) Asits-MacBook-Pro:final asit7$ python3 final.py
Declared: a

Assigned Value: 5 to a

Declared: b

Assigned Value: 6 to b

Variable: a has value of: 5

Variable: b has value of: 6

"Hello World"

Program Ended

(base) Asits-MacBook-Pro:final asit7$ 
```

Program 2:

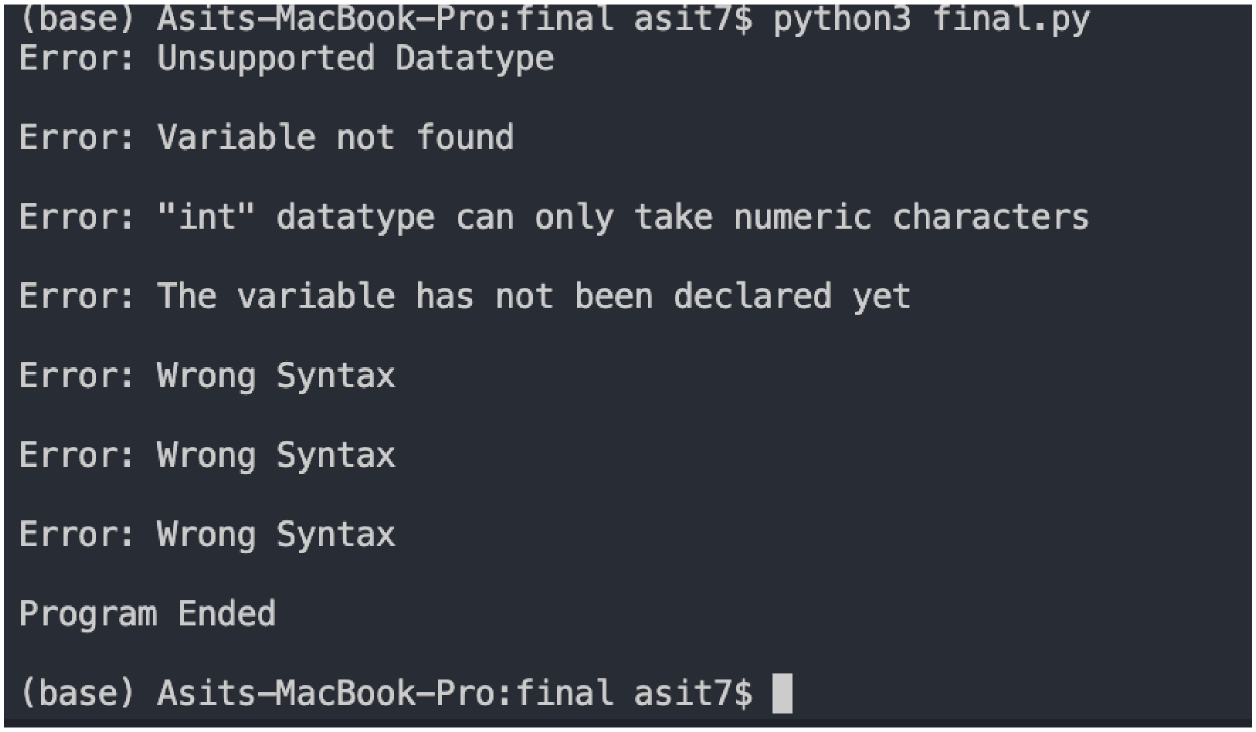


```
Declared: a

Declared: f

Assigned Value: hello to f

Declared: d

Assigned Value: world to d

Assigned Value: 5 to a

Declared: b

Assigned Value: 6 to b

Declared: c

Assigned Value: True to c

Results after adding: helloworld

Results after adding: 10

Results after adding: 11

Results after adding: 11

Results after subtracting: 0

Results after subtracting: -1

Results after subtracting: -1

Results after multiplying: 25

Results after multiplying: 30

Results after multiplying: 30

Results after dividing: 1.0

Results after dividing: 0.8333333333333334

Results after dividing: 0.8333333333333334

Results after the mod operation: 0

Results after the mod operation: 5

Results after the mod operation: 5

False
```



```
Results after adding: 11

Results after subtracting: 0

Results after subtracting: -1

Results after subtracting: -1

Results after multiplying: 25

Results after multiplying: 30

Results after multiplying: 30

Results after dividing: 1.0

Results after dividing: 0.8333333333333334

Results after dividing: 0.8333333333333334

Results after the mod operation: 0

Results after the mod operation: 5

Results after the mod operation: 5

False

False

False

True

True

True

False

False

False

True

True

True

Program Ended

(base) Asits-MacBook-Pro:final asit7$ 
```

Program 5:



```
(base) Asits-MacBook-Pro:final asit7$ python3 final.py
Error: Unsupported Datatype

Error: Variable not found

Error: "int" datatype can only take numeric characters

Error: The variable has not been declared yet

Error: Wrong Syntax

Error: Wrong Syntax

Error: Wrong Syntax

Program Ended

(base) Asits-MacBook-Pro:final asit7$ 
```

Conclusion

In conclusion we believe that the language PlainEnglish is a very good language that could be beneficial for young students who are trying to get into programming and learn to code. The language is still in development but consider this project our 1st version that shows that it could work and be useful. We will continue working on the language to make it better by fixing bugs and adding support for more functions across more datatypes. Thank you for this project, it has been truly a pleasure working on this project and we plan on continuing support for this language and make it a fully functional language that is very easy to learn and use to write fun and interesting programs!