```
###

  What follows are tiny CoffeeScript programs.
  Each program is separated by: ######################################## P

  The programs in this document introduce:

    • Basic commands.
    • Basic operators.
    • Primitive datatypes.
    • Assignment and Variables

  To use this document:

    1. Download and open Framer (or go to http://prototyp.in )
       If using framer, turn autorefresh OFF. View > Auto Refresh > Off

    2. Type out, by hand, each program, by itself. Refresh (command+r) and compare the results to what's written

       (You can ignore any lines with instructions or commentary.)

    3. Make notes where anything is the least bit confusing. Experiment with specific example to explore how it works.

###
```

```
1  ######################################## P1
2  # Some words are like magic spells.
3  # We cast the spell by using a keyword, followed by parenthesis.
4  # In programming we call spells "functions".
5  # Casting them is called "calling" them.
6  # Below, we "call" the "function" print.
7
8  print()  # A panel should pop up on the right side.
9
10
11
```

```
12  ######################################## P2
13  # If you try to use a function that doesn't exist, you'll get an error.
14
15  explode()
16
17
18
```

```
######################################## P3
# Some functions accept additional details.
# We call these details "arguments".

print(100)
print(100,200,300)

# When we call functions with arguments we sometimes say that we're: "passing" data to the function.
# Above, we're passing 100 to the print function. Next we pass three numbers to print.
# Note: The program runs the code from top to bottom and then stops.
```

```
######################################## P4
# Computers do math.
# +,-,/,* are called "operators" or "operations".
# A math expression is called an "expression".
# Expressions are always solved BEFORE function calls.
# For example, 'print 10+10' becomes 'print 20' and then the computer runs that code.

print(10+10)    # becomes print(20)
print(10-10)    # becomes print(0)
print(10*10)    # becomes print(100)
print(10/10)    # becomes print(1)
```

```
46  ######################################## P5
47  # Data comes in different types, or datatypes.
48  # Three basic datatypes in CoffeeScript/Javascript are Number, String, and Boolean
49  # Numbers are numbers.
50  # String stands for string of characters. They are always in "quotes"
51  # The quotes allow us to distinguish between reserved keywords and words.
52  # Boolean values are the two logic values true or false
53  # Technically, function is a datatype too. But that's another story...
54
55  # Print a Number
56  print(1000.0)   # 1000.0
57
58  # Print a String
59  print("cat")    # "cat"
60
61  # Print the word "print"
62  print("print")  # "print"
63
64  # Print the actual function that the keyword print refers to.
65  print(print)    # ...
66
67  # Print a boolean value
68  print(true)     # true
69
70
```

```
71  ######################################## P6
72  # In JavaScript and CoffeeScript, datatypes affect how operations work.
73  # Technically, the computer will try to convert the data to something that makes sense.
74
75  print(2+2)      # 4
76  print("2"+"2")  # "22"  When using + and strings. The computer combines (concatenates) the strings.
77  print("2"+2)    # "22"  The expression is ambiguous and the computer converts the 2 to "2"
78  print("2"*2)    # 4     Can't multiply strings so the computer tries to convert the string to a number.
79  print("cat"*"dog")  # NaN  literally means Not a Number which figuratively means Hot Garbage.
80
81
```

```coffeescript
######################################## P7
# Data can be stored in memory.
# There are two basic places: long term (hard drive) and short term (RAM).
# Data in long term storage will remain even if the program stops. It is a topic for another day.
# Short term storage disappears after the program stops.
# Short term storage is both easy and essential. It works almost the same way in almost every language.
# In CoffeeScript, this is how it's done:

myData = "SomeData"
amIReallyAProgram = true
VALUE_FOR_PI = 3.141598
_A2d3f_ = "Bad Name"
herp_derp = 1

# The word on the left is called a variable, the equal sign is called the assignment operator.
# The space before and after the = (the assignment operator) is for readability.
# Variable can be named whatever you like but they can't contain spaces, operators, or symbols like {}<>.
# Variable names can include numbers but can't start with them.
# The naming convention for variables in most languages is "camelCase". First word lowercase, next words uppercase.

# Tech Note: For those who know JavaScript, 'var' is not allowed in CoffeeScript.
```

```
105 ######################################## P8
106 # Variables can be used wherever you would use data
107
108 a = 1
109 print(a)
110
111
```

```
112 ######################################## P9
113 # When variables are used with a math operator, the variable names are replaced with their data before the math is resolved.
114
115 a = 2
116 print(a+a)      # This becomes 'print 2+2' which becomes 'print 4'
117 print((a+a)*a)  # This becomes 'print (2+2)*2' which becomes 'print 8'
118
119
```

```
120 ######################################## P10
121 # The same applies during assignment:
122
123 a = 2
124 b = a+a+a      # This becomes 'b = 2+2+2' which becomes 'b = 6'
125 print(b + b)   # 'print 6+6' becomes 'print 12'
126
127 # In other words, when you use a variable on the right side of an equal, you are referring to the data in that variable.
128
129
```

```
######################################## P11
# When primitive data is assigned to a variable, the variable's old data is overwritten.

a = 1
a = 2      # The 1 in memory is overwritten with the number 2.
a = a + a  # becomes 'a = 2 + 2', then becomes 'a = 4'. Finally, the 2 that was in memory is overwritten with the number 4

print(a)   # 4
```

```
######################################## P12
# Look very carefully. This part is tricky.


a = 1
b = a     # This line becomes 'b = 1'
          # There are now two places in memory (labeled 'a' and 'b').
          # 'a' and 'b' each has it's own unique number '1'.
          # The data for variable 'a' was duplicated. The copy was placed into variable 'b'.
          # The variable 'b' does NOT point to the same memory that 'a' points to.
a = 2     # Changing the number in 'a' will not affect the data in 'b' or vice versa.

print(a) # 2
print(b) # 1

# It is important to understand that 'b = a' does _NOT_ mean 'make b always equal to a'.

# Note: This behavior holds for almost any programming language an interaction designer will encounter.
# Note: There is a huge exception to this rule that we will discuss in the section on Objects.
```

```
########################################
######################################## [Optional] Additional operators
########################################

######################################## P13
# CoffeeScript supports common assignment shortcuts or "mutating operators"

a = 1

a += 2    # add to self, e.g. a = a + 2, a = 1 + 2, a = 3
print(a)  # 3

a -= 2    # subtract from self: a = a - 2, a = 3 - 2, a = 1
print(a)  # 1

a++       # increment by one, e.g. a = a + 1, a = 1 + 1, a = 2
print(a)  # 2

a--       # decrement by one, e.g. a = a - 1, a = 2 - 1, a = 1
print(a)  # 1

a /= 2    # divide self by value, e.g. a = a/2, a = 1/2, a = 0.5
print(a)  # 0.5

a *= 2  # multiply self by value, e.g. a = a * 2
print(a)

```

```coffeescript
########################################
######################################## [Optional] CoffeeScript function shortcut
########################################

######################################## P14
# In CoffeeScript, if we "call" a function "with arguments", we can leave out the parenthesis.

print 100          # same as print(100)
print 100,200,300  # same as print(100,200,300)

# The spaces between the function name and the argument are meaningful symbols to CoffeeScript, just like '=' or '(' or ","
# This is called "significant whitespace".
# Python is another common language that uses significant whitespace.
# Most languages are NOT like this though (C,C++,C#,Java,Javascript,Swift,Objective-C)
# Significant whitespace is a polarizing paradigm among developers.

######################################## P15
# If we ever call a function WITHOUT arguments, we MUST use '()'

print()  # Function call. Ok.
print "ok"  # Function call with arguments. Ok.
print  # No arguments, no function call. Nothing happens.


```

```
214 ########################################
215 ######################################## Note
216 ########################################
217 #
218 # The print function is part of the FramerJS library.
219 # Using JavaScript, you'll normally use console.log() for the same end.
220 # Everything else in this document works anywhere you use CoffeeScript.
221 #
222 # End!
223 #
224 #
225 #
```