

Flow Control Statements

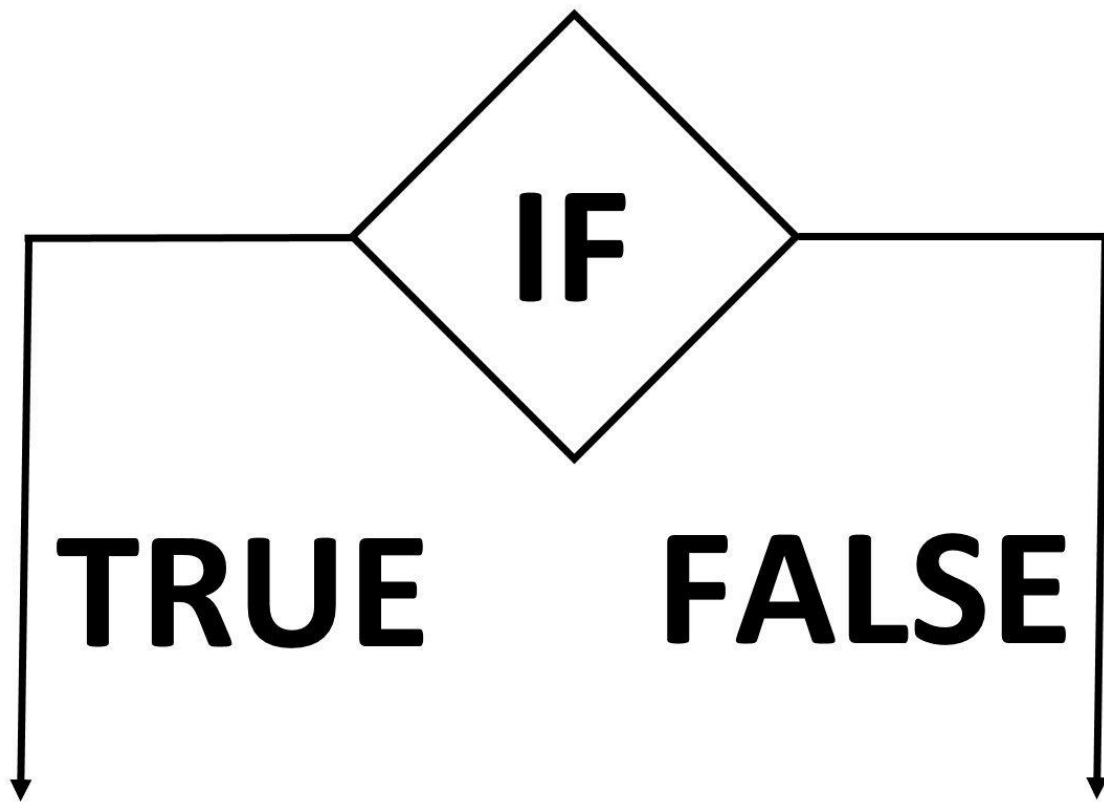


Figure 1: I drew the above segment of a Flowchart Algorithm in Microsoft Word.

In programming, statements such as:

if

, which introduce a condition, are known as:

flow-control statements

One way to conceive of Computer Algorithms, is to represent them as Flowcharts. The:

if

statement *alters* or *controls* the *flow* of the algorithm.

In the above depicted example¹, if the condition tested by the:

if

statement should be found to be *true*, then the logical execution of the algorithm will *flow down* the left-hand side of the page.

In the above depicted example, if the condition tested by the:

if

statement should be found to be *false*, then the logical execution of the algorithm will *flow down* the right-hand side of the page.

When we introduce a logical split into our algorithm, then this is termed:

‘branching’

.

The true and false tributaries of the depicted flow-chart algorithm are termed:

‘branches’

.

¹ i.e. the example depicted in **Figure 1**.

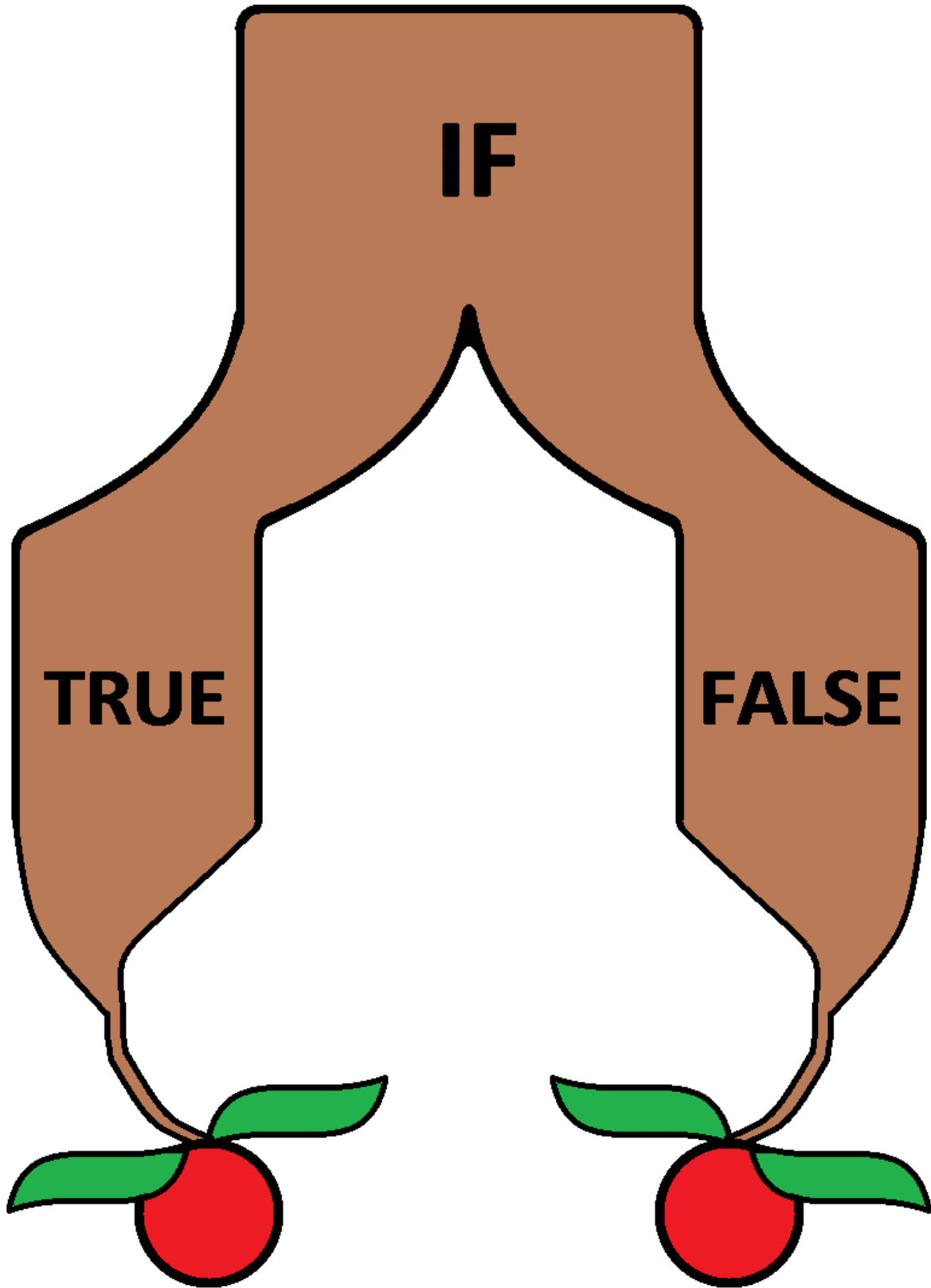


Figure 2: The algorithm *branches*. We can instruct the computer to do different tasks depending upon whether the logical condition tested by the if statement be found true or false.

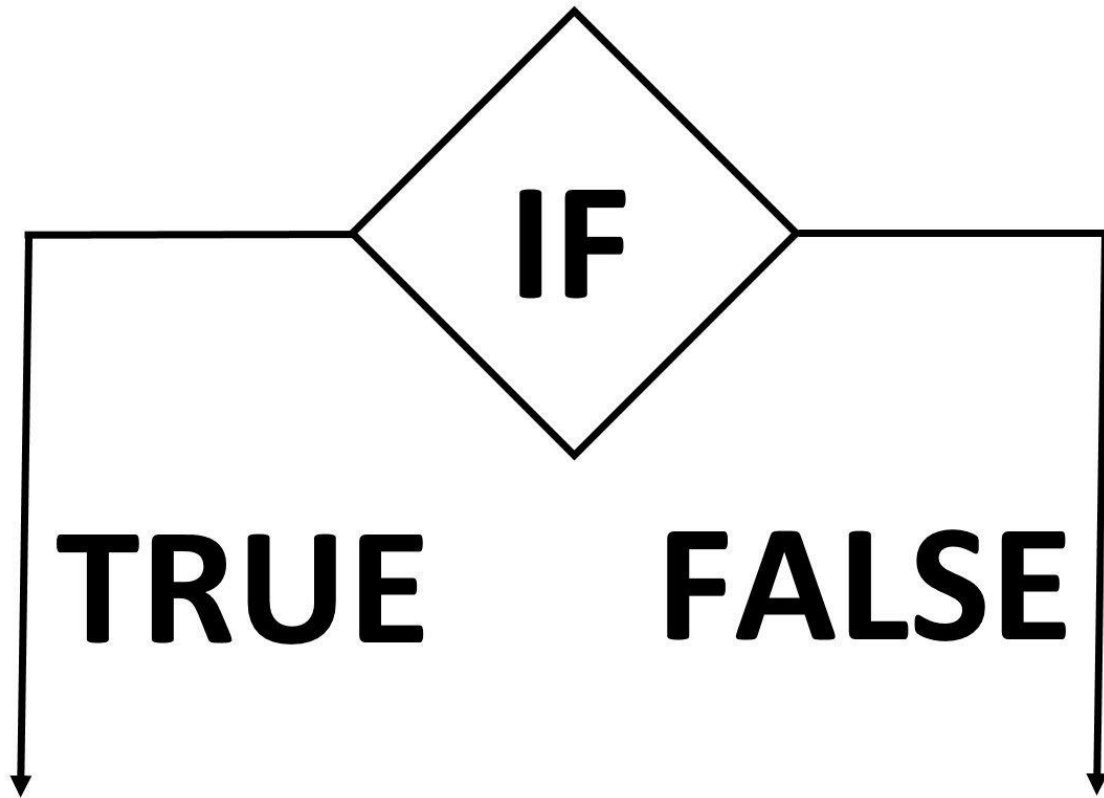
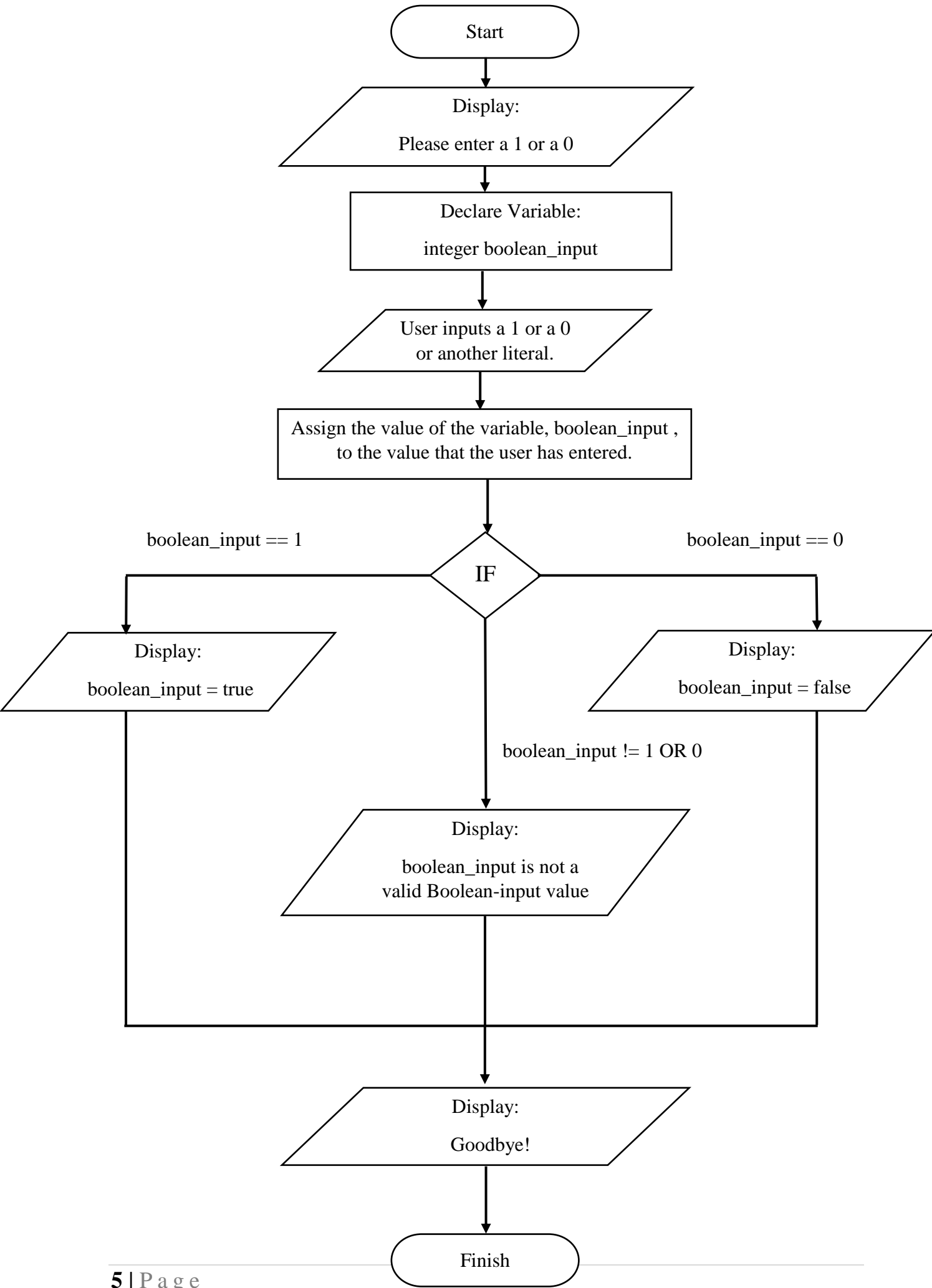


Figure 3: In this segment of a Flowchart Algorithm, we can see that it *branches* after we test a logical condition with an *if statement*. In the depiction, above, we can observe the *true* branch of the Algorithm, and the *false* branch of the Algorithm.



On the previous page, we have a flow-chart algorithm that describes a program that takes an integer – either a 1 or a 0 – inputted by the user, and which outputs a string contingent upon what the user has inputted.

The above algorithm solves a computational problem. The computational problem that the above algorithm solves may be stated as:

How can we test a literal inputted by a user so as to see if it should equate to
Boolean True or Boolean False?

Mutual Exclusion:

The:

true

and:

false

branches of this algorithm are termed:

‘mutually exclusive’

.

A logical test is performed, and *if* that which is tested *be true* then that excludes the possibility of its *being false*.

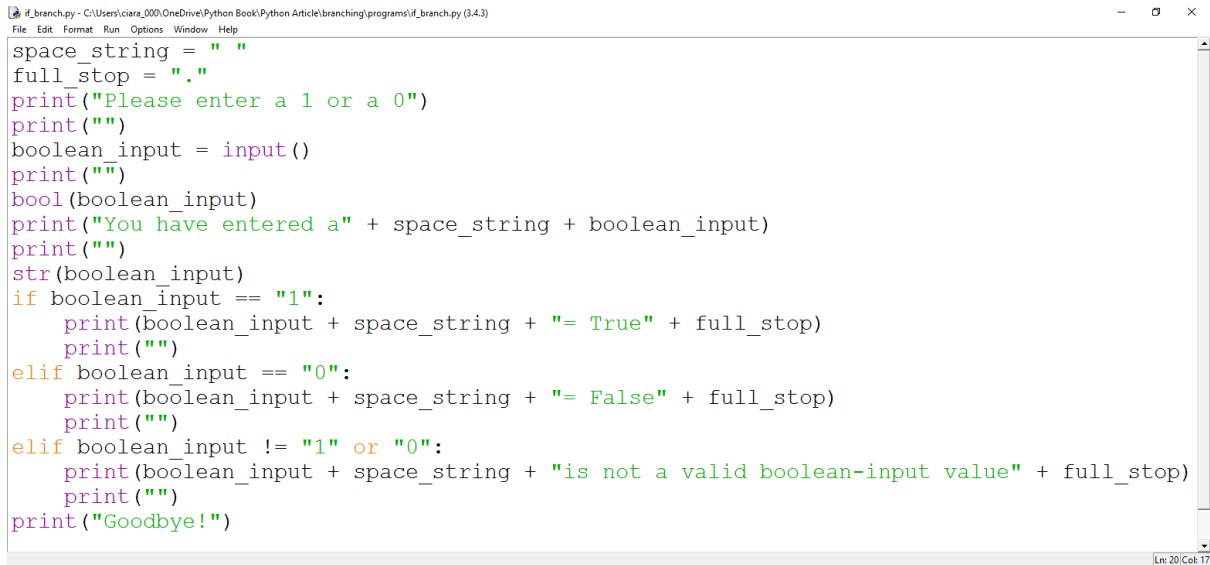
A logical test is performed, and *if* that which is tested *be false* then that excludes the possibility of its *being true*.

If the true branch of the algorithm be executed, then the false branch of the algorithm will not be executed.

If the false branch of the algorithm be executed, then the true branch of the algorithm will not be executed.

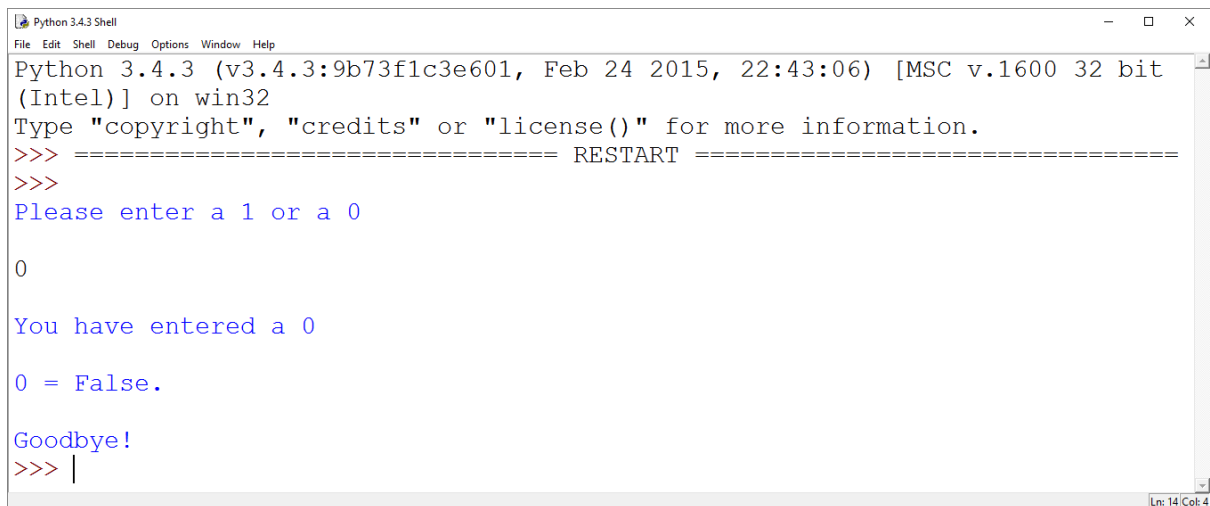
Branching in Python:

We shall now write a program in Python that corresponds to the algorithm depicted on **Page 5**.



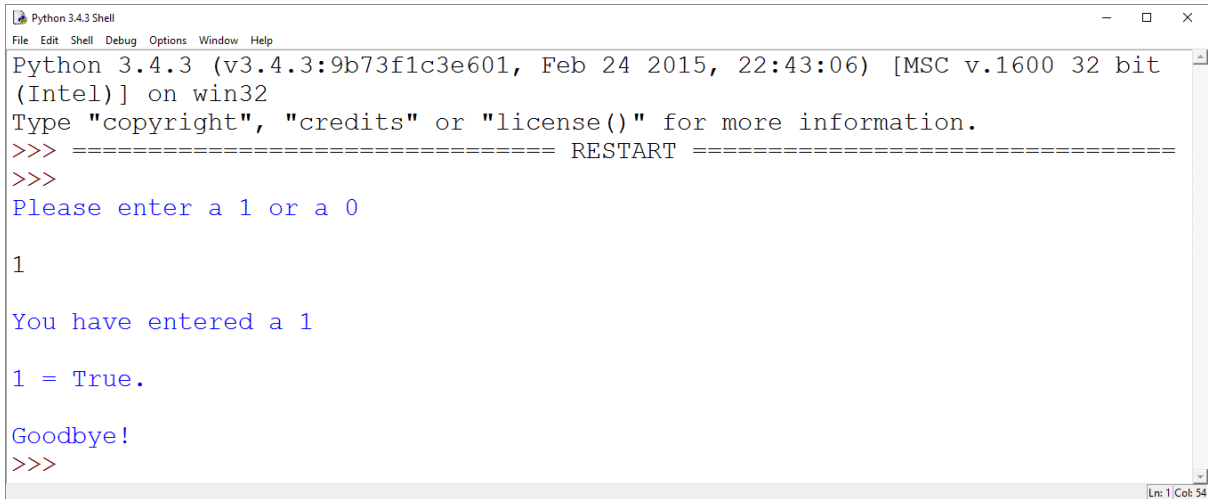
```
if_branch.py - C:\Users\ciara_000\OneDrive\Python Book\Python Article\branching\programs\if_branch.py (3.4.3)
File Edit Format Run Options Window Help
space_string = " "
full_stop = "."
print("Please enter a 1 or a 0")
print("")
boolean_input = input()
print("")
bool(boolean_input)
print("You have entered a" + space_string + boolean_input)
print("")
str(boolean_input)
if boolean_input == "1":
    print(boolean_input + space_string + "= True" + full_stop)
    print("")
elif boolean_input == "0":
    print(boolean_input + space_string + "= False" + full_stop)
    print("")
elif boolean_input != "1" or "0":
    print(boolean_input + space_string + "is not a valid boolean-input value" + full_stop)
    print("")
print("Goodbye!")
Ln: 20 Col: 17
```

Figure 4: This is the python program that corresponds to the algorithm on Page 5.



```
Python 3.4.3 Shell
File Edit Shell Debug Options Window Help
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Please enter a 1 or a 0
0
You have entered a 0
0 = False.
Goodbye!
>>> |
Ln: 14 Col: 4
```

Figure 5: This is what is outputted by the Python program depicted in Figure 4 should the user input the value, 0.



```
Python 3.4.3 Shell
File Edit Shell Debug Options Window Help
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Please enter a 1 or a 0

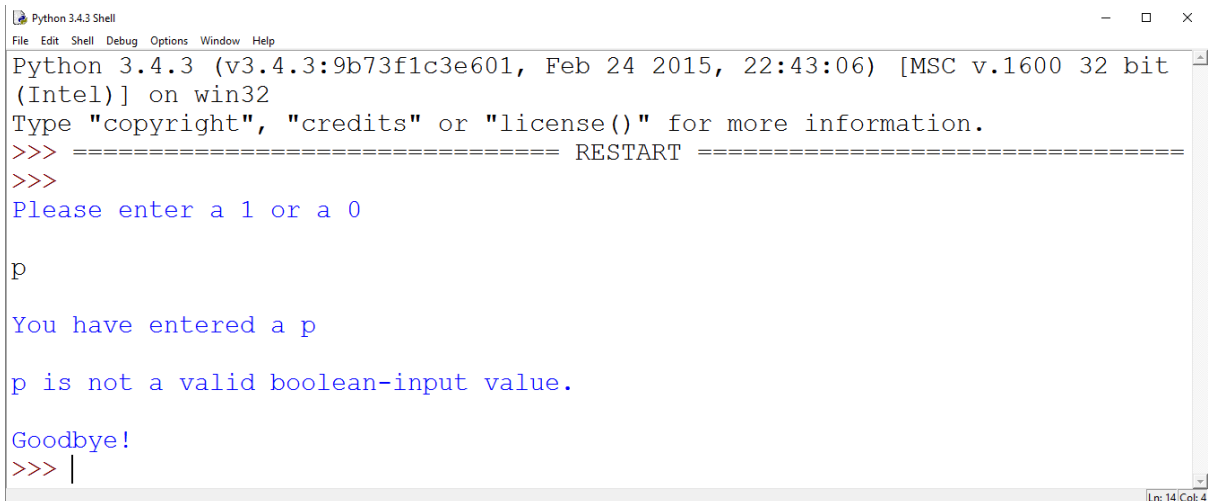
1

You have entered a 1

1 = True.

Goodbye!
>>>
```

Figure 6: This is what is outputted by the Python program depicted in **Figure 4** should the user input the value, 1.



```
Python 3.4.3 Shell
File Edit Shell Debug Options Window Help
Python 3.4.3 (v3.4.3:9b73f1c3e601, Feb 24 2015, 22:43:06) [MSC v.1600 32 bit
(Intel)] on win32
Type "copyright", "credits" or "license()" for more information.
>>> ===== RESTART =====
>>>
Please enter a 1 or a 0

p

You have entered a p

p is not a valid boolean-input value.

Goodbye!
>>> |
```

Figure 7: This is what is outputted by the Python program depicted in **Figure 4** should the user input a literal that is not a 1 or a 0.

More on Branching in Algorithms in General:

As we can see from the algorithm depicted on **Page 5**, the:

true

and:

false

branches of the algorithm *converge* or *attain a confluence* prior to the:

“Goodbye!”

string’s being outputted.

The:

“Goodbye!”

string will be outputted regardless of the result of the logical condition tested by the:

if

statement.

Back to If Statements in Python:

One quintessential piece of Python syntax is the colon. The colon is used to declare that what follows will be an indented code block.



Figure 8: In Python, if statements are always terminated by colons. In Python, the colon always declares that the preceding code block will be indented. The code block that follows the colon that terminates the if statement is indented².

² In Python style, an indent is worth 4 spaces.

Back to Branching in Algorithms in General:

Trees are not the only things that branch. Rivers also branch into tributaries. Rivers also *flow downwards*³, and so it is an excellent analogy so as to conceive of branching in algorithms.

³ As does a *flow-chart* algorithm.

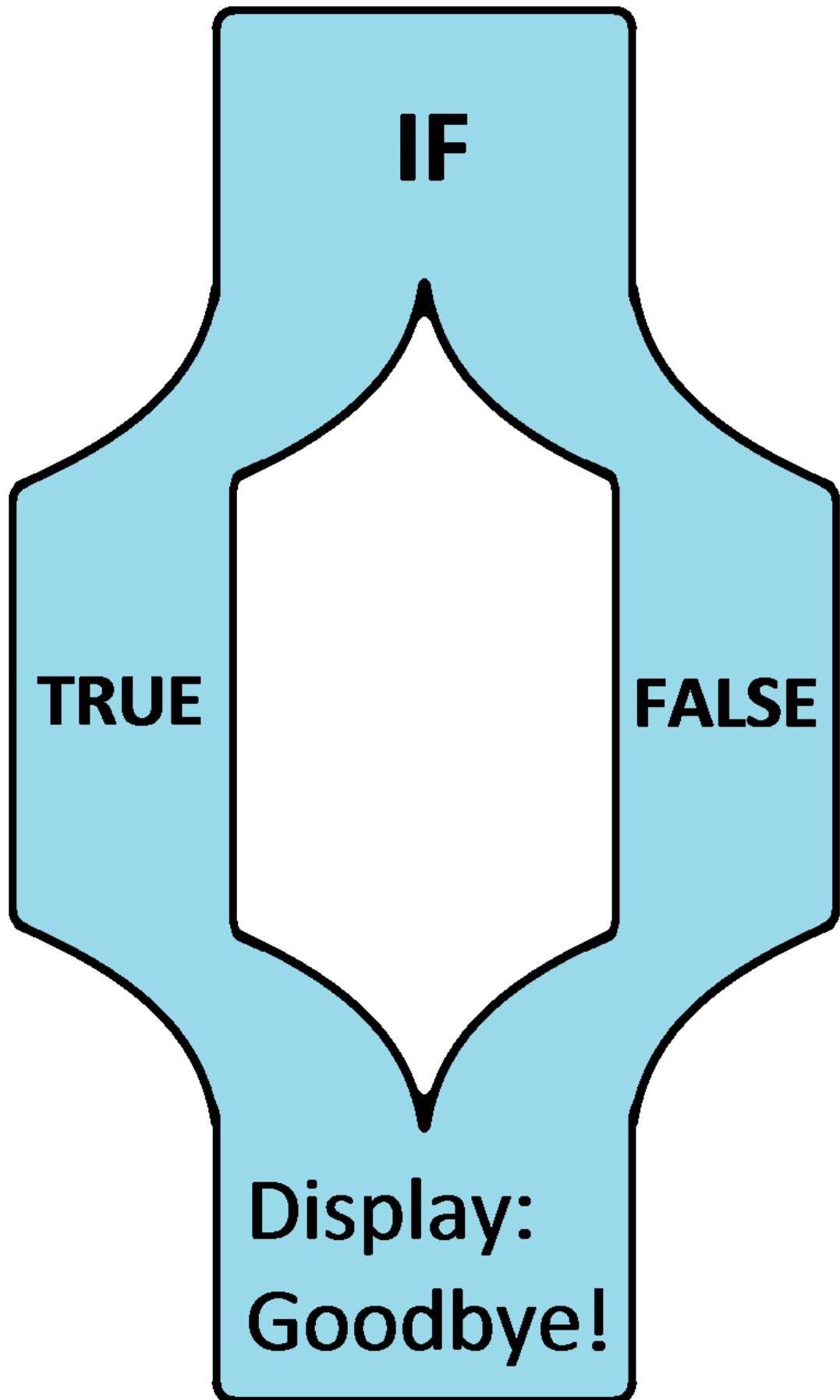


Figure 9: Another way to conceive of *branching* in flow-chart algorithms: the flow-chart algorithm *branches* into true and false code blocks after a logical condition is tested, before re-attaining a confluence prior to “Goodbye” being printed. The two tributaries of the flow-chart algorithm *merge* together again prior to “Goodbye” being printed. Regardless of whether the true code block or the false code block be executed, “Goodbye” will nonetheless be printed.

What is the Purpose of Writing an Algorithm prior to Writing a Program?

An algorithm is imperative⁴ knowledge. It tells one *how* to do something. In computing, an algorithm tells one *how to solve a computational problem*.

In computing, an algorithm is a series of commands that solves a computational problem.

There are two approaches to programming:

Seat-of-the-Pants Method:

With this method, the programmer just dives into writing the program. However, the programmer still composes an algorithm, only this time, the algorithm is *mental*. At each stage of his writing a program, the programmer still must imagine what he must *command* the computer to do for it to solve a computational problem. The programmer just does not take the time to write this *series of commands* or *algorithm* down.

Write-the-Algorithm-First Method:

With this method, the programmer *solves the computational problem first* prior to his commencing writing the program. He does this by *writing an algorithm*.

The advantage of writing an algorithm is that it does not limit the programmer to a solution in a single language such as Python. Should the programmer take the time to write out the algorithm first, then it will allow him to easily compose a program that corresponds to that algorithm not only in Python, but in whatever programming language that he should so choose.

⁴ From the Latin 1st-conjugation verb, ‘imperō, imperāre, imperāvī, imperātum,’ which means: ‘to command,’ ‘to order.’ Cp. *Latin English Lexicon: Optimized for the Kindle*, Thomas McCarthy, (Perilingua Language Tools: 2013) Version 2.1 Loc 46105.

Writing a Program that Corresponds to our Algorithm in

C:

On **Page 5**, we wrote an algorithm that solved a computational problem. The computational problem that was solved by the algorithm depicted on **page 5** can be stated as:

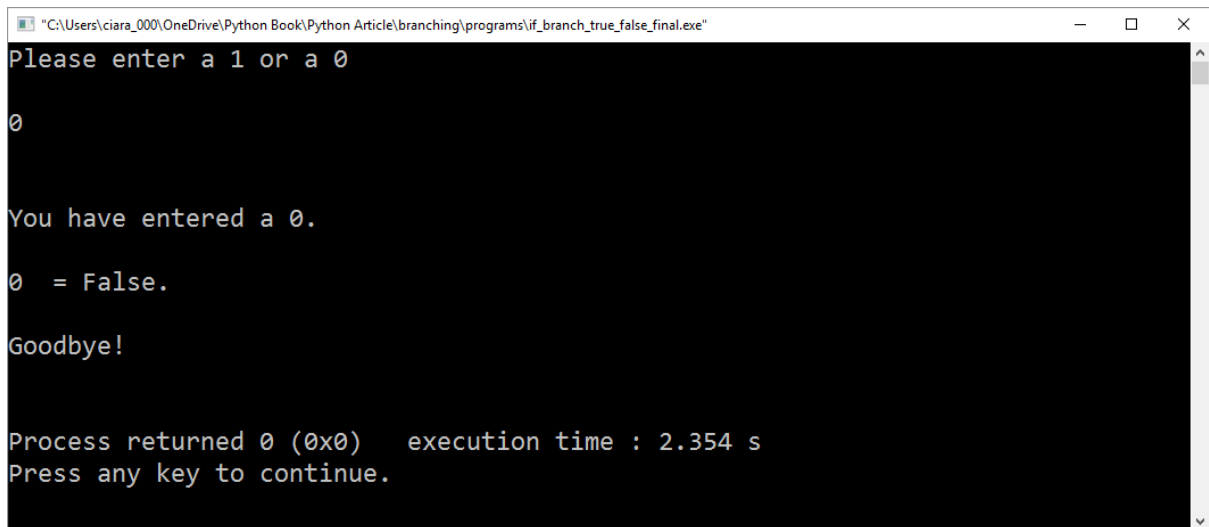
How can we test a litteral inputted by a user so as to see if it should equate to

Boolean True or Boolean False?

With the above-stated computational problem solved, we can now easily write a program that corresponds to the algorithm, not only in Python syntax, but in C syntax, as well.

```
if_branch_true_false_final_char_version.c x
1  #include <stdio.h>
2
3  int main () {
4
5      printf("Please enter a 1 or a 0");
6      printf("\n\n");
7
8      char boolean_input;
9      scanf("%c", &boolean_input);
10
11     printf("\n\n");
12     printf("You have entered a %c." , boolean_input);
13     printf("\n\n");
14
15     if (boolean_input == '1')
16     {
17         printf( "%c = True.", boolean_input);
18         printf("\n\n");
19     }
20     else
21     {
22
23         if (boolean_input == '0')
24         {
25
26
27             printf( "%c = False.", boolean_input);
28             printf("\n\n");
29         }
30         else
31         {
32
33             if (boolean_input != '1' || '0')
34             {
35                 printf("%c is not a valid boolean-input value.", boolean_input );
36                 printf("\n\n");
37             }
38         }
39     }
40     printf("Goodbye!");
41     printf("\n\n");
42
43     return 0;
44 }
45
```

Figure 10: The C program that corresponds to the algorithm depicted on Page 5.



```
"C:\Users\ciara_000\OneDrive\Python Book\Python Article\branching\programs\if_branch_true_false_final.exe"
Please enter a 1 or a 0
0

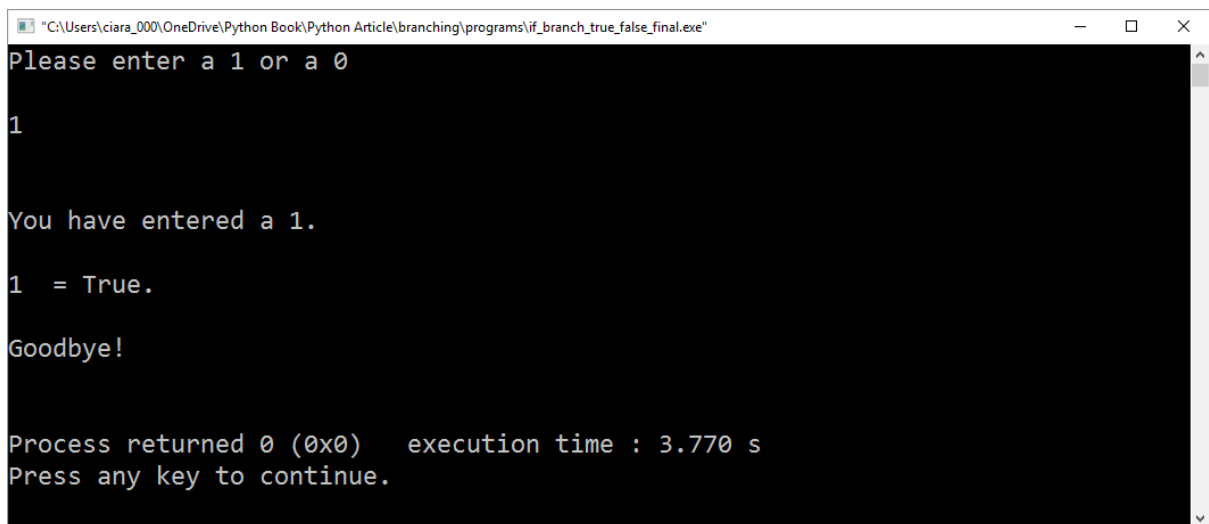
You have entered a 0.

0 = False.

Goodbye!

Process returned 0 (0x0)   execution time : 2.354 s
Press any key to continue.
```

Figure 11: What the C program depicted in **Figure 10** outputs should the user input a 0.



```
"C:\Users\ciara_000\OneDrive\Python Book\Python Article\branching\programs\if_branch_true_false_final.exe"
Please enter a 1 or a 0
1

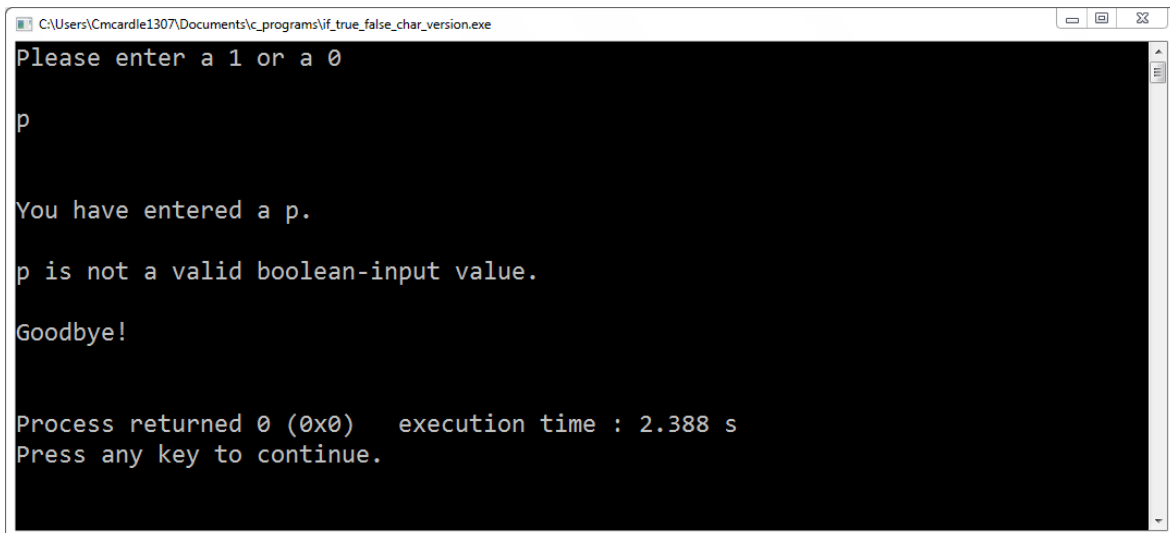
You have entered a 1.

1 = True.

Goodbye!

Process returned 0 (0x0)   execution time : 3.770 s
Press any key to continue.
```

Figure 12: What the C program depicted in **Figure 10** outputs should the user input a 1.



```
C:\Users\Cmcardle1307\Documents\c_programs\if_true_false_char_version.exe
Please enter a 1 or a 0
p
You have entered a p.
p is not a valid boolean-input value.
Goodbye!
Process returned 0 (0x0)   execution time : 2.388 s
Press any key to continue.
```

Figure 13: What the C program depicted in **Figure 10** outputs should the user input a literal that is neither a 0 or a 1.

Glossary:

confluence

- **noun.** the junction of two rivers, especially rivers of approximately equal width.
 - an act or process of merging: *a major confluence of the world's financial markets.*

<**ORIGIN**> late Middle English: from late Latin *confluentia*, from **Latin** *confluere* ‘flow together’ (see CONFLUENT).⁵

<**ETYMOLOGY**> From the Latin 1st-declension feminine noun, ‘confluentia, confluentiae,’ which means ‘a flowing together.’⁶ From the Latin preposition, ‘cum,’ which means ‘together;’ and the Latin 3rd-conjugation verb, ‘fluō, fluere, fluxī, fluxum,’ which means ‘to flow;’ and the Latin 1st-declension nominal suffix, ‘-tia, -tiae,’ which denotes a state of being. A confluence, therefore, etymologically, is ‘a flowing together.’

As regards algorithms, by way of an analogy, a confluence can be said to describe the merging of two or more branches of a flow-chart algorithm.

⁵ Oxford University Press. *Oxford Dictionary of English* (Electronic Edition). Oxford. 2010. Loc 146068.

⁶ Cp. *Latin English Lexicon: Optimized for the Kindle*, Thomas McCarthy, (Perilingua Language Tools: 2013) Version 2.1 Loc 23064.

confluent

- *adjective.* flowing together or merging.

<**ORIGIN**> late 15th century: from Latin **confluent-** ‘flowing together’, from *confluere*, from *con-* ‘together’ + *fluere* ‘to flow’.⁷

<**ETYMOLOGY**> From the Latin 3rd-declension masculine noun, ‘cōnfluēns, cōnfluētis,’ ‘which means ‘confluence.’ ‘flowing together.’ From the Latin preposition, ‘cum,’ which means ‘together;’ and the Latin present active participle, ‘fluēns, fluētis,’ which means ‘flowing.’

As regards algorithms, by way of an analogy, two or more branches of an algorithm can be said to be *confluent* when they merge together.

⁷ Oxford University Press. *Oxford Dictionary of English* (Electronic Edition). Oxford. 2010. Loc 146082.

imperative

- *adjective.*
 1. of vital importance; crucial: *immediate action was imperative* | [with *clause*] *it is imperative that standards are maintained.*
 2. giving an authoritative command; peremptory: *the bell pealed again, a final imperative call.*
 - [GRAMMAR] denoting the mood of a verb that expresses a command or exhortation, as in *come here!*
- *noun.*
 1. an essential or urgent thing: *free movement of labour was an economic imperative.*
 - a factor or influence making something necessary: *the biological imperatives which guide male and female behaviour.*
 2. [GRAMMAR] a verb or phrase in the imperative mood.
 - (**the imperative**) the imperative mood.

<DERIVATIVES> **imperatival** *adjective.* **imperatively** *adverb.* **imperativeness** *noun.*

<ORIGIN> late Middle English (as a grammatical term): from Late Latin *imperativus* (literally ‘specially ordered’, translating Greek *prostatikē enklisis* ‘imperative mood’), from **imperare** ‘to command’, from **in-** ‘towards’ + **parare** ‘make ready’⁸.

<ETYMOLOGY> from the Latin 1st-and-2nd-declension adjective, ‘*impērātīva, impērātīvus, impērātīvum,*’ which means ‘pertaining to the command;’ ‘of the command.’ From the Latin 1st-conjugation verb, ‘*imperō, imperāre, imperāvī, imperātum,*’ which means ‘to command,’ ‘to order,’ and the Latin 1st-and-2nd-declension adjectival suffix ‘*-īva, -īvus, -īvum,*’ which means ‘of,’ ‘concerning,’ ‘pertaining to.’ From the Latin prefix ‘*in-*’ which expresses the concept of ‘unto,’ ‘toward,’ and the Latin 1st-conjugation verb, ‘*parō, parāre, parāvī, parātum,*’ which means ‘to make ready,’ ‘to prepare.’ The etymological sense, therefore, of the English adjective, ‘imperative’ is: ‘concerning the command;’ ‘pertaining to the command;’ ‘of the command;’ ‘concerning the order;’ ‘pertaining to the order;’ ‘of the order;’ ‘concerning the making ready of;’ ‘pertaining to the making ready of;’ ‘of the making ready of;’ etc.

⁸ *ibid.* Loc 345790

As regards algorithms, 'imperative' denotes the type of knowledge expressed by a series of *commands*, as opposed to declarative knowledge.