ICS3U Case Study: Value Range

While small programs may be fairly easy to write without much analysis, there are some situations where putting some thought into the design of an algorithm is necessary. For example, a process that involves several complicated calculations might first need to be verified mathematically before it can be written as a program. In other cases, software may need to be written according to specific guidelines that include analysis as part of the process.

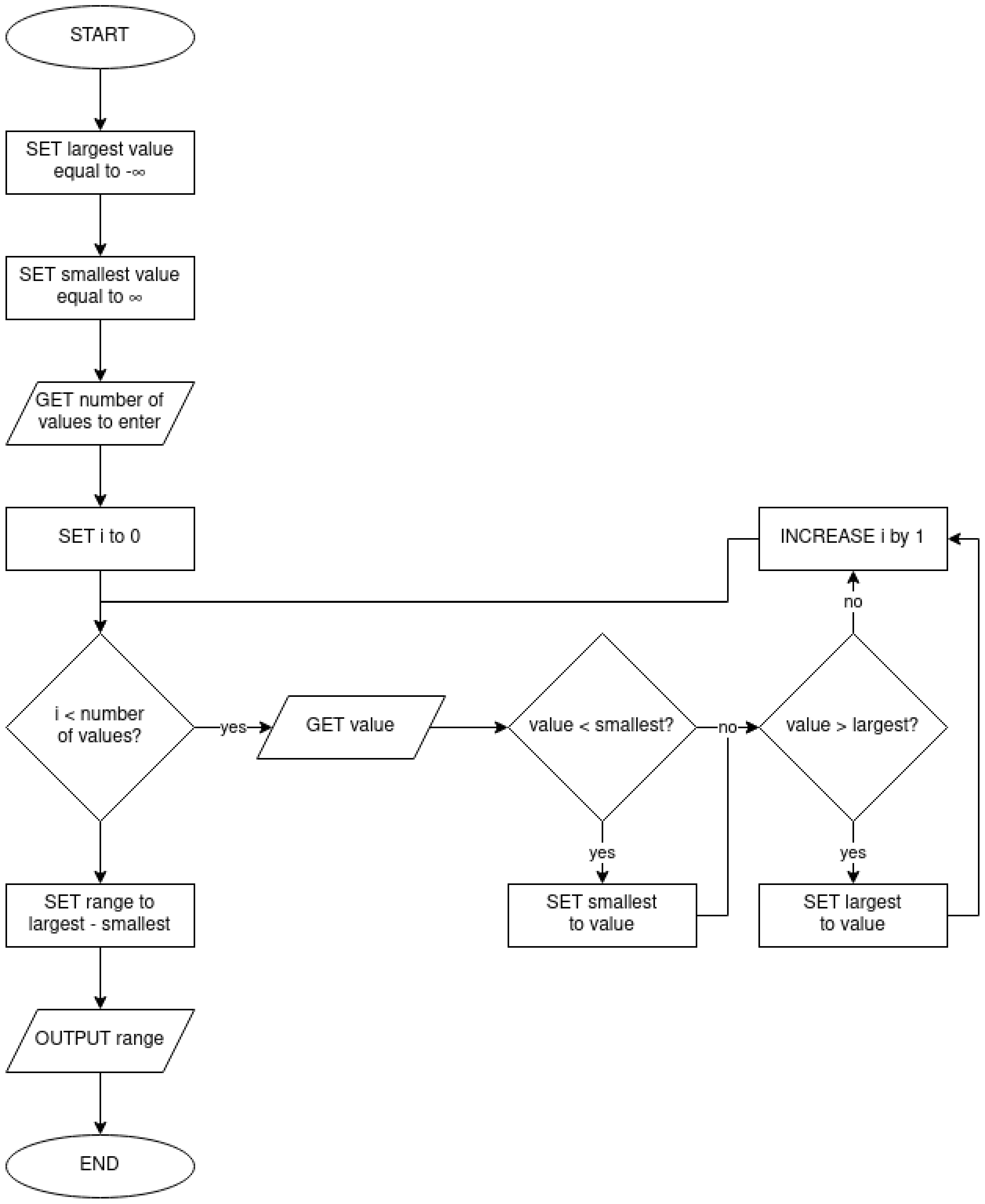
Consider the problem of determining the *range* of a set of values. The range is the difference between the largest and the smallest value. For this scenario, we will write code that reads an arbitrary number of integers from the user and determines the range of the values. Since we do not know how many values will be entered (or what those values will be), we cannot take advantage of the `max` and `min` functions in Python (at least, not with our current knowledge). Instead, we will monitor the values as they are entered, and keep track of the largest and smallest values ourselves.

Pseudocode for the algorithm may look something like below.

```
SET largest ← -∞
SET smallest ← ∞
GET num values
SET i ← 0
WHILE i < num values
    GET value
    IF value > largest
        SET largest ← value
    IF value < smallest
        SET smallest ← value
    SET i ← i + 1
SET range ← largest - smallest
OUTPUT range
```

Note that we are not actually assigning “infinity” to anything here; this is just notation to assign a very large (or very small) value as a placeholder until an actual value takes its place.

Translating the pseudocode to a flowchart is a simple matter of associating each step with a particular symbol (e.g. a parallelogram for input/output, a diamond for a decision, and so on). While the layout can vary, the overall steps should be similar. It may look something like the diagram below.



Now that the algorithm has been created, we can attempt to write code to implement it. Our first attempt might appear as follows.

```
# Determine the range (largest - smallest) of a set of integer values.
import math
smallest = math.inf
largest = -math.inf
num_values = int(input("Enter how many values? "))
for count in range(num_values):
    value = int(input("Value: "))
    if value < smallest:
        smallest = value
    elif value > largest:
        largest = value
value_range = largest - smallest
print("The range is", value_range)
```

Note that while our pseudocode and flowchart use a condition to control the loop, we have opted for a `for` loop in our code, since it will be iterating a fixed number of times. If we wanted to, we could use a `while` loop instead; however, this would require that we modify the counter variable manually.

```
count = 0
while count < num_values:
    value = int(input("Value: "))
    # ... MORE CODE ...
    count += 1
```

In order to test the program, we try some sample input.

```
Enter how many values? 3
Value: 2
Value: 9
Value: 4
The range is 7
```

This checks out, since 9 – 2 = 7. What if all values are the same?

```
Enter how many values? 3
Value: 5
Value: 5
Value: 5
The range is 0
```

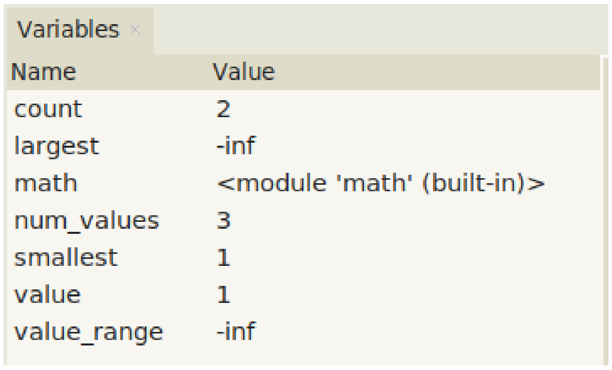
Again, everything appears to be working as intended; however, there are some situations where the program will *not* produce the correct output.

```
Enter how many values? 3
Value: 8
Value: 3
Value: 1
The range is -inf
```

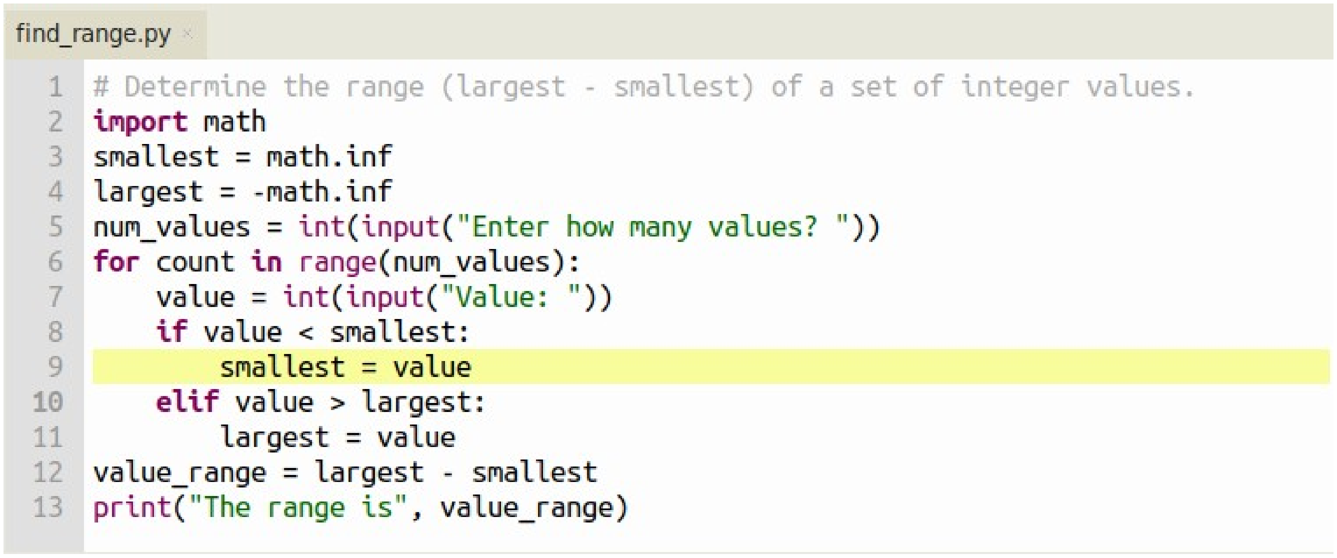
We *should* obtain a range of 8 – 1 = 7 in this scenario. To diagnose where the problem lies, we can use the debugger to step through the program.

Running the program with the values 8, 3 and 1 shows us that the value of `largest` is never updated. That suggests that the condition `value > largest` is never `True`. We can use the debugger to trace the flow of the program to see why this is happening.

As we step through, we eventually reach line 8 after entering the first value (8). Since `smallest` has a value of `math.inf`, then `value < smallest` is `True`, and we proceed to line 9 as shown.

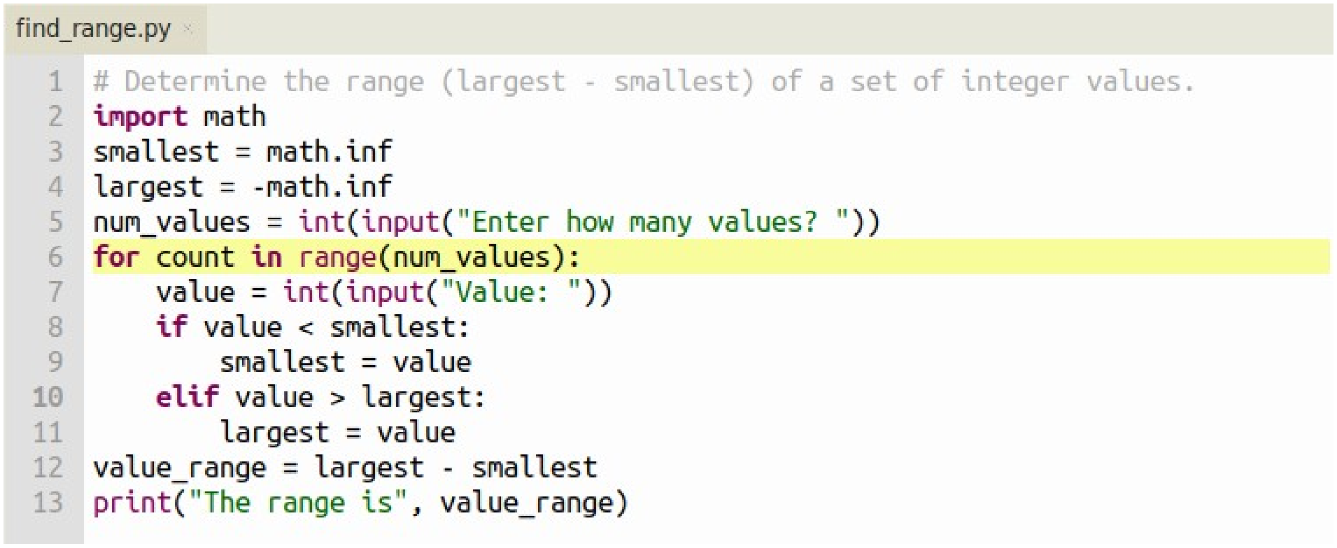


Name	Value
count	2
largest	-inf
math	<module 'math' (built-in)>
num_values	3
smallest	1
value	1
value_range	-inf



```
 1  # Determine the range (largest - smallest) of a set of integer values.
 2  import math
 3  smallest = math.inf
 4  largest = -math.inf
 5  num_values = int(input("Enter how many values? "))
 6  for count in range(num_values):
 7      value = int(input("Value: "))
 8      if value < smallest:
 9          smallest = value
10      elif value > largest:
11          largest = value
12  value_range = largest - smallest
13  print("The range is", value_range)
```

When we move on to the next step, the program proceeds to line 6 again.



```
 1  # Determine the range (largest - smallest) of a set of integer values.
 2  import math
 3  smallest = math.inf
 4  largest = -math.inf
 5  num_values = int(input("Enter how many values? "))
 6  for count in range(num_values):
 7      value = int(input("Value: "))
 8      if value < smallest:
 9          smallest = value
10      elif value > largest:
11          largest = value
12  value_range = largest - smallest
13  print("The range is", value_range)
```

This means that the value of `largest` remains `-math.inf`. When the first value is entered, however, it should be both the smallest *and* the largest value. The problem arises because we are using `elif` instead of `if`. Changing this fixes the error, at the expense of a slightly longer execution time due to checking both conditions each time through the loop.

```
# Determine the range (largest - smallest) of a set of integer values.
import math
smallest = math.inf
largest = -math.inf
num_values = int(input("Enter how many values? "))
for count in range(num_values):
    value = int(input("Value: "))
    if value < smallest:
        smallest = value
    if value > largest:
        largest = value
value_range = largest - smallest
print("The range is", value_range)
```

```
Enter how many values? 3
Value: 8
Value: 3
Value: 1
The range is 7
```

Exercise

Follow the same process here to write pseudocode, a flowchart, and program code for the following scenarios:

1. Read a positive integer and determine the largest digit (e.g. for 17235 the largest digit is 7).
2. Read a sequence of integers and determine how many of them are even and how many are odd.