Unit 2

Computing Practice and Programming

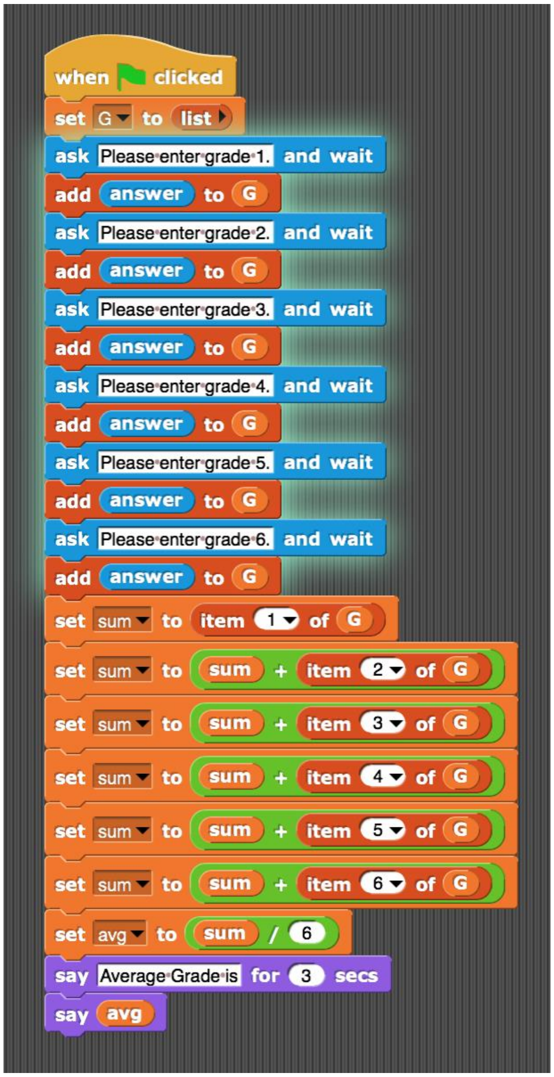
Part 2: Connect the Development Cycle of Program Construction to Problem Solving

Operations in Program

We will show how to implement the sequencing, selection, and iteration algorithms from Unit 3 using SNAP.

Sequencing

A sequencing algorithm follows a set list of steps in order to produce a result. In this SNAP program, the user is asked for six grades and then the average of those grades is calculated. This is a sequence because the program follows each step of collecting grades to put into a list, adding those grades from the list, and then calculating the average in that specific order each time.

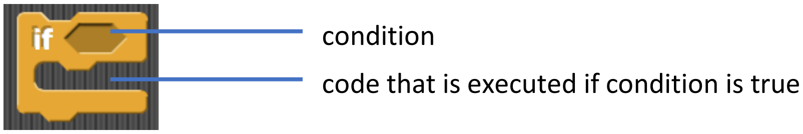


```
when ⚑ clicked
set G to list ▸
ask Please enter grade 1. and wait
add answer to G
ask Please enter grade 2. and wait
add answer to G
ask Please enter grade 3. and wait
add answer to G
ask Please enter grade 4. and wait
add answer to G
ask Please enter grade 5. and wait
add answer to G
ask Please enter grade 6. and wait
add answer to G
set sum to item 1 of G
set sum to sum + item 2 of G
set sum to sum + item 3 of G
set sum to sum + item 4 of G
set sum to sum + item 5 of G
set sum to sum + item 6 of G
set avg to sum / 6
say Average Grade is for 3 secs
say avg
```

Selection

Selection algorithms allow the user to choose a different outcome based on information defined within the program. A typical way of implementing a selection algorithm is through an if or if-else statement.

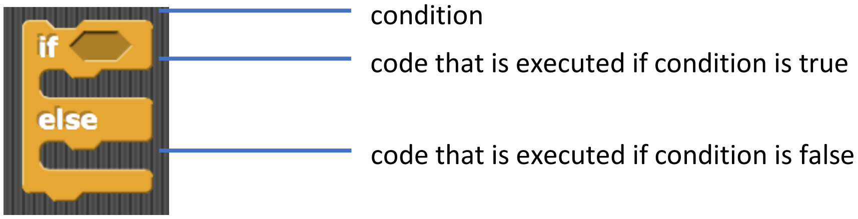
An if statement in SNAP looks like this:



condition

code that is executed if condition is true

An if-else statement in SNAP looks like this:

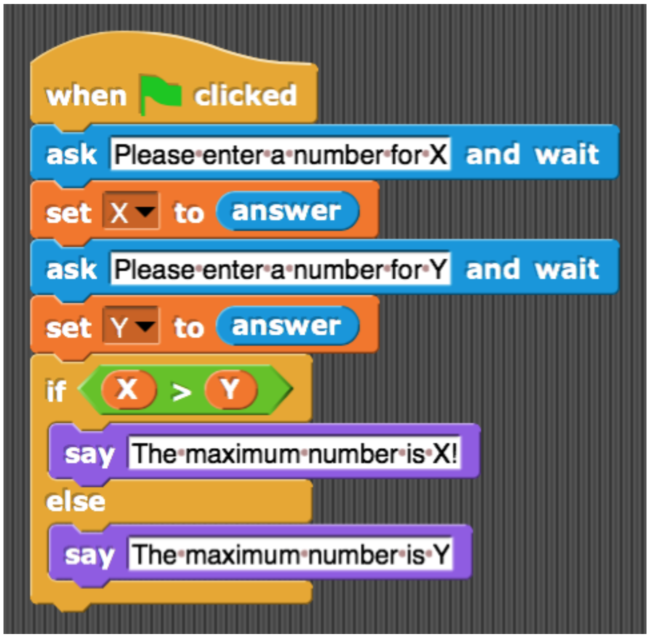


condition

code that is executed if condition is true

code that is executed if condition is false

The following is an implementation of the example selection algorithm from Unit 3. In this example, the user is asked for two numbers. The selection algorithm is performed based on which number is larger using an if-else statement. If the first number is larger, the condition is true, and the first code is executed. However, if the second number is larger, the statement is false and the code under the else is executed.

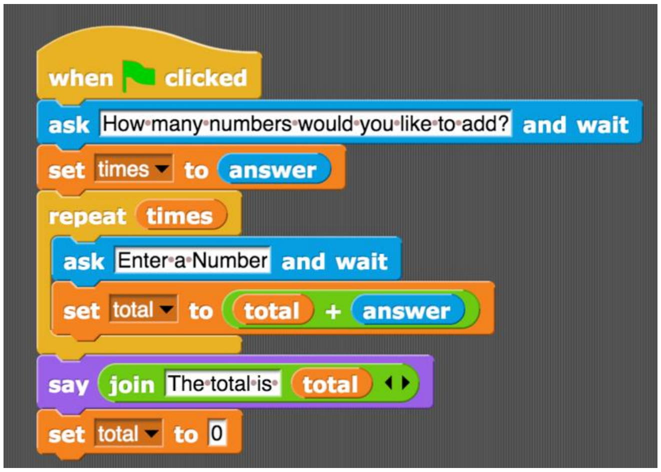


```
when ⚑ clicked
ask [Please enter a number for X] and wait
set [X] to (answer)
ask [Please enter a number for Y] and wait
set [Y] to (answer)
if <(X) > (Y)>
    say [The maximum number is X!]
else
    say [The maximum number is Y]
```

Iteration

Iteration algorithms allow the same code to be repeated several times to accomplish a task. In SNAP, this iteration can be accomplished with the repeat control.

The following is an implementation of the example iteration algorithm from Unit 3. In this program, the user is asked for how many numbers they would like to add. That number is then passed into the repeat control to continue asking the user for a number to add until it reached that number. Finally, their result is printed, and the total is cleared.



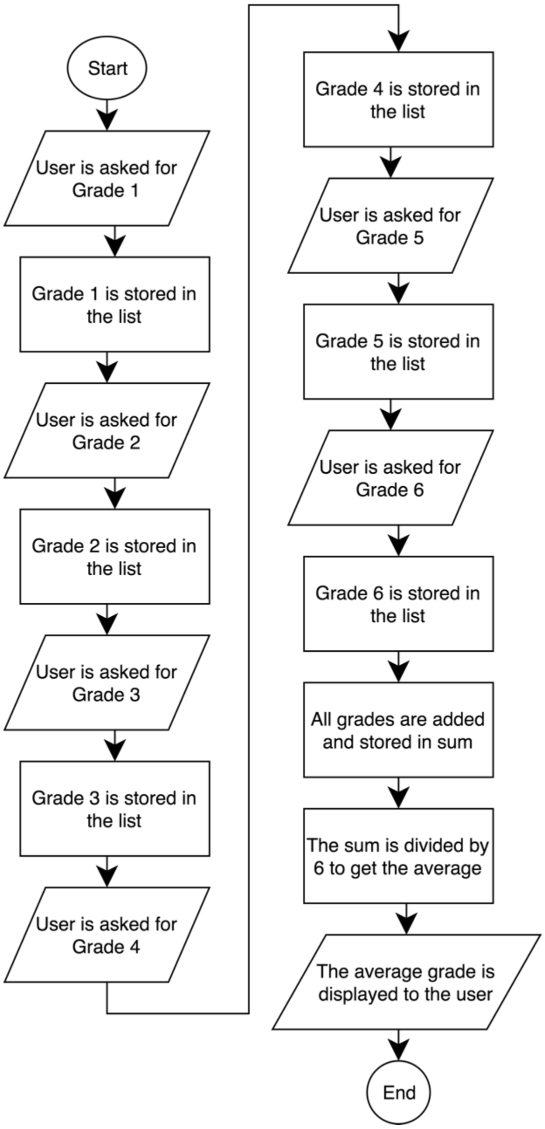
```
when ⚑ clicked
ask [How many numbers would you like to add?] and wait
set [times ▾] to (answer)
repeat (times)
    ask [Enter a Number] and wait
    set [total ▾] to ((total) + (answer))
say (join [The total is ] (total) ◂▸)
set [total ▾] to [0]
```

Flow Charts

Flowcharts are important in programming for understanding how the code works. Any program can be represented with a flow chart to see how data is moving throughout the process of running the code. Each of the three algorithms demonstrated above can also be shown in a flowchart that looks very similar to the SNAP program that is being executed.

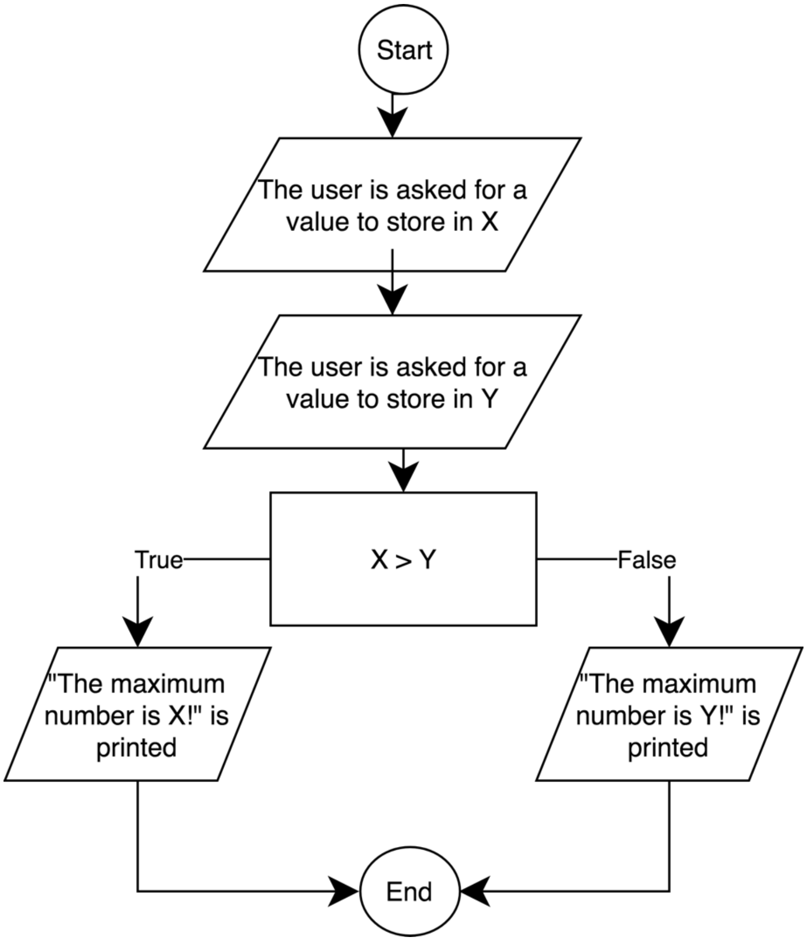
Sequence Flowchart

In the sequencing algorithm, since the data flows the same way each time, it can be represented in a straight flow chart.


Start
User is asked for Grade 1
Grade 1 is stored in the list
User is asked for Grade 2
Grade 2 is stored in the list
User is asked for Grade 3
Grade 3 is stored in the list
User is asked for Grade 4
Grade 4 is stored in the list
User is asked for Grade 5
Grade 5 is stored in the list
User is asked for Grade 6
Grade 6 is stored in the list
All grades are added and stored in sum
The sum is divided by 6 to get the average
The average grade is displayed to the user
End

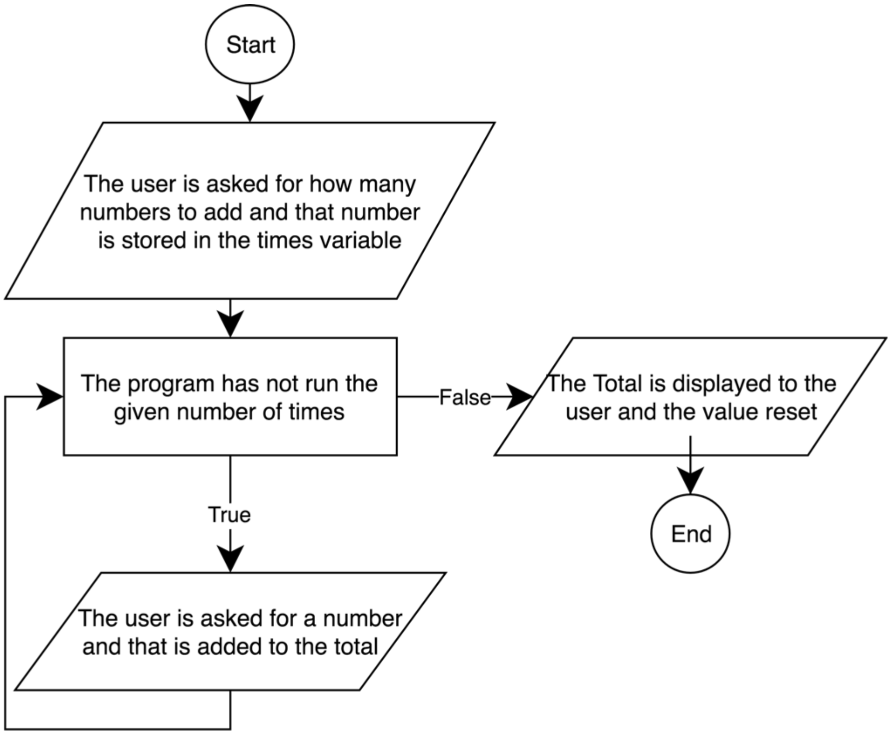
Selection Flowchart

A selection flowchart looks slightly different as there are multiple outcomes for the program to reach. As shown here, if X is greater than Y, the option on the left is chosen. However, if X is not greater than Y, the option on the right is chosen.


Start
The user is asked for a value to store in X
The user is asked for a value to store in Y
X > Y
True
False
"The maximum number is X!" is printed
"The maximum number is Y!" is printed
End

Iteration Flowchart

Through iteration, the flowchart shows that code is being run multiple times. In our example, the code continues asking for numbers and adding them until it reaches the number of times specified by the user at the beginning of the program.


Start
The user is asked for how many numbers to add and that number is stored in the times variable
The program has not run the given number of times
False
The Total is displayed to the user and the value reset
End
True
The user is asked for a number and that is added to the total