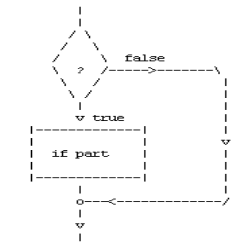
CS 8
Assignment 1:

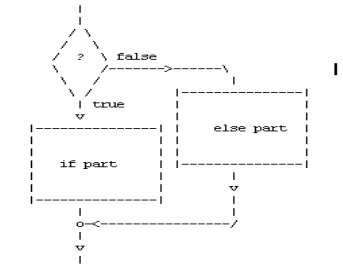
In this assignment, you will implement two algorithms as three complete Python programs. The Python statements: **if**, **for**, **if – else,** and **while** are outlined below.

if – statement

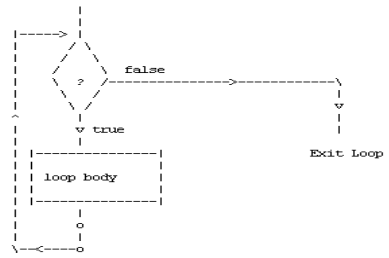


Use these to recognize the Python statement needed to translated the algorithm into a complete program!

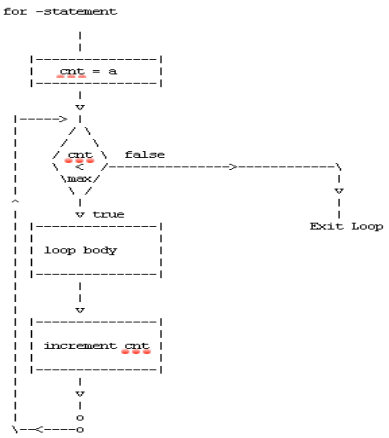
if – else statement



while – statement

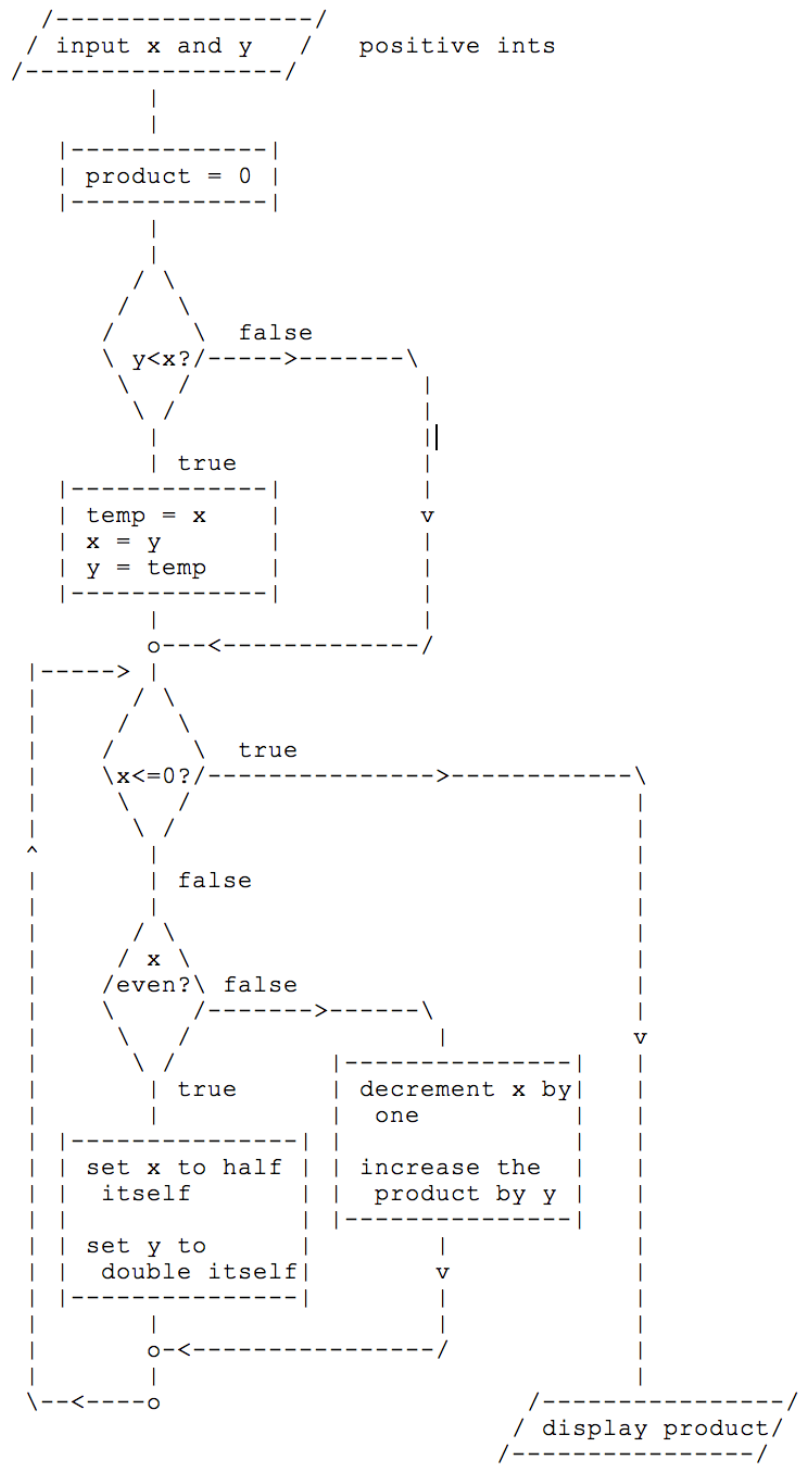


for –statement



Algorithm 1

Compute the product of two positive integers using the multiplication algorithm of the ancient Egyptians.



Three Sample Runs
Enter x: 14
Enter y: 12
14 * 12 = 168

Enter x: 4891
Enter y: 52
4891 * 52 = 254332

Enter x: 72890346
Enter y: 35619
72890346 * 35619 = 2596281234174

Final Touches
In order to show explicitly what this algorithm is doing (those ancient Egyptians were busy), we shall use formatting to display x, y, and product each time through the loop to produce a table with borders of three columns illustrated below. Here are two runs:

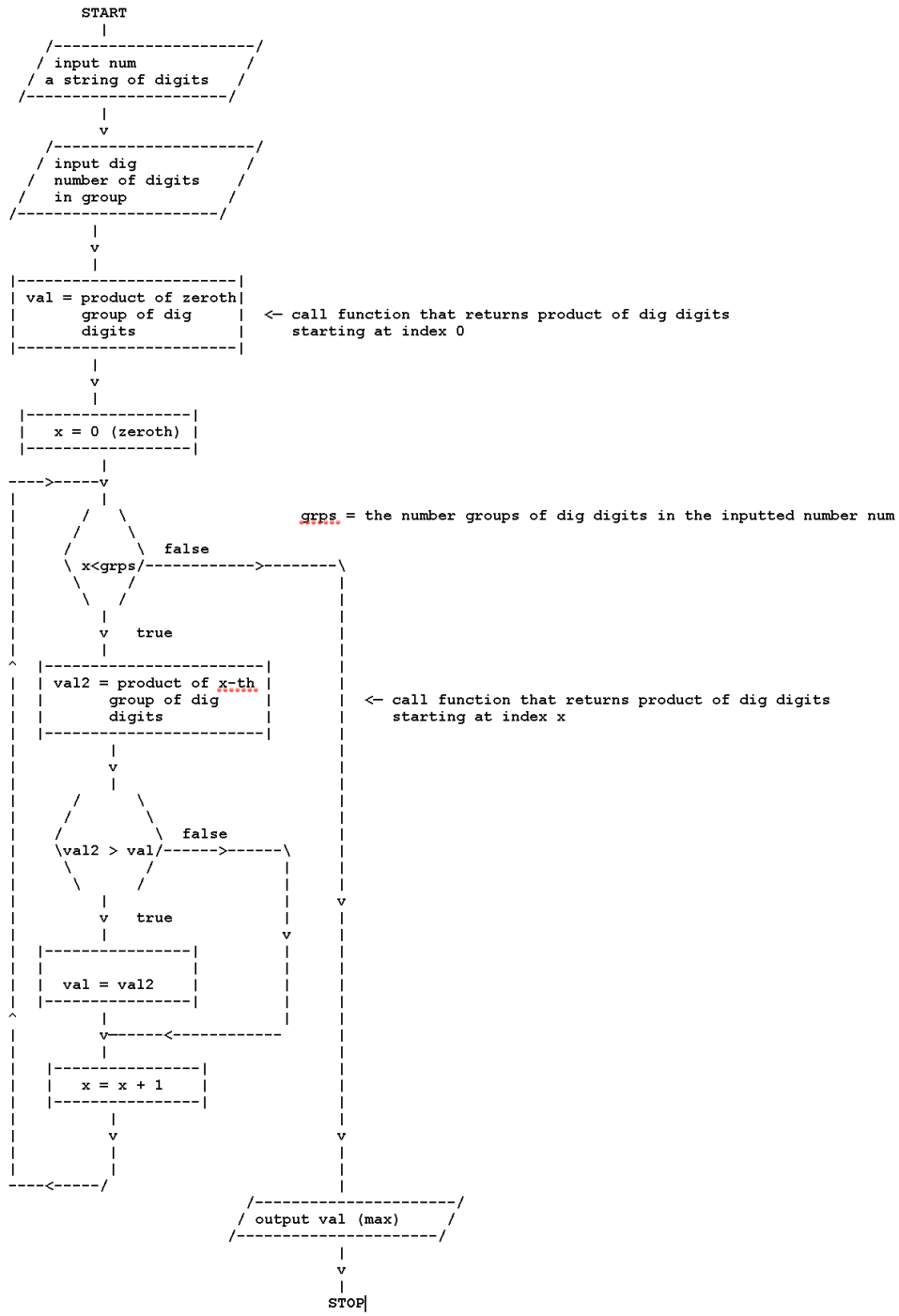
```
Enter x: 14
Enter y: 12
-----------------------------------------------------
              x              y                product
-----------------------------------------------------
             14             12                    0
              6             28                    0
              3             56                    0
              2             56                   56
              1            112                   56
              0            112                  168
-----------------------------------------------------
14 * 12 = 168

Enter x: 72890346
Enter y: 35619
-----------------------------------------------------
              x              y                product
-----------------------------------------------------
       72890346          35619                    0
          35618       72890346             72890346
          17809      145780692             72890346
          17808      145780692            218671038
           8904      291561384            218671038
           4452      583122768            218671038
           2226     1166245536            218671038
           1113     2332491072            218671038
           1112     2332491072           2551162110
            556     4664982144           2551162110
            278     9329964288           2551162110
            139    18659928576           2551162110
            138    18659928576          21211090686
             69    37319857152          21211090686
             68    37319857152          58530947838
             34    74639714304          58530947838
             17   149279428608          58530947838
             16   149279428608         207810376446
              8   298558857216         207810376446
              4   597117714432         207810376446
              2  1194235428864         207810376446
              1  2388470857728         207810376446
              0  2388470857728        2596281234174
-----------------------------------------------------
72890346 * 35619 = 2596281234174
```

Algorithm 2

Input a string of digits (e.g., '72890346') and an integer representing the number of digits in each group (e.g., 3). From among the consecutive products of each group of three, determine the maximum.

7 x 2 x 8 = 112
2 x 8 x 9 = **144** <- maximum
8 x 9 x 0 = 0
9 x 0 x 3 = 0
0 x 3 x 4 = 12
3 x 4 x 6 = 72



You will need to create a user-defined function **product**(*dig*, *indx*, *snum*) that computes the product of *dig* consecutive digits, starting at index *indx* in the string of digits *snum*. For example, if *indx* = 5, *dig* = 3, and snum = '72890346', the function call **product**(3, 5, '72890346') returns 144, the product of 3 x 4 x 6 back to the caller.

Three Sample runs:
Enter an integer: 491773
Number of digits in a product: 2
The max product of 2 consecutive digits in 491773 is 49

Enter an integer: 72890346
Number of digits in a product: 3
The max product of 3 consecutive digits in 72890346 is 144

Enter an integer: 8458015616609791913387549 92
Number of digits in a product: 6
The max product of 6 consecutive digits in 8458015616609791913387549 92 is 90720

Finishing Touches
Modify your program to allow the user (you) to enter the number of **runs** you wish to execute. Here is a single execution of this program.

Enter the number of runs: 4
Enter an integer: 2315891
Number of digits in a product: 3
The max product of 3 consecutive digits in 2315891 is 360 (run # 1)

Enter an integer: 5640912587
Number of digits in a product: 6
The max product of 6 consecutive digits in 5640912587 is 5040 (run # 2)

Enter an integer: 9991
Number of digits in a product: 1
The max product of 1 consecutive digits in 9991 is 9 (run # 3)

Enter an integer: 72890346
Number of digits in a product: 4
The max product of 4 consecutive digits in 72890346 is 1008 (run # 4)

Program Execution Complete

Big Inputs
Try this integer if you dare:

```
731671765313306249192251196744265747423553491949 34
969835203127745063262395783180169848018694788518 43
858615607891129494954595017379583319528532088055 11
125406987471585238630507156932909632952274430435 57
668966489504452445231617318564030987111217223831 13
622298934233803081353362766142828064444866452387 49
303589072962904915604407723907138105158593079608 66
701724271218839987979087922749219016997208880937 76
657273330010533678812202354218097512545405947522 43
525849077116705560136048395864467063244157221553 97
536978179778461740649551492908625693219784686224 82
839722413756570560574902614079729686524145351004 74
821663704844031998900088952434506585412275886668 81
164271714799244429282308634656748139191231628245 86
178664583591245665294765456828489128831426076900 42
242190226710556263211111093705442175069416589604 08
071984038509624554443629812309878799272442849091 88
845801561660979191338754992005240636899125607176 06
058861164671094050775410022569831552000559357297 25
716362695618826704282524836008232575304207529634 50
```