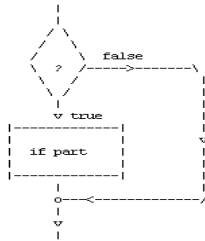


CS 8

Assignment 1:

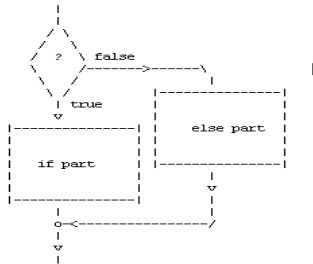
In this assignment, you will implement two algorithms as three complete Python programs. The Python statements: **if**, **for**, **if - else**, and **while** are outlined below.

if - statement

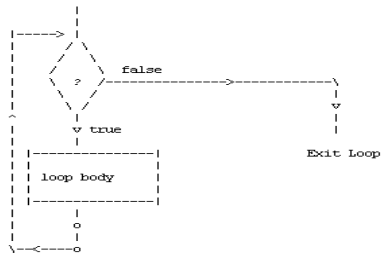


Use these to recognize the Python statement needed to translated the algorithm into a complete program!

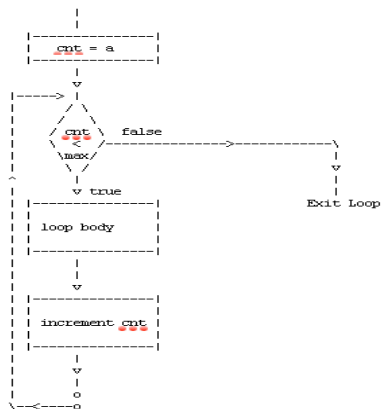
if - else statement



while - statement

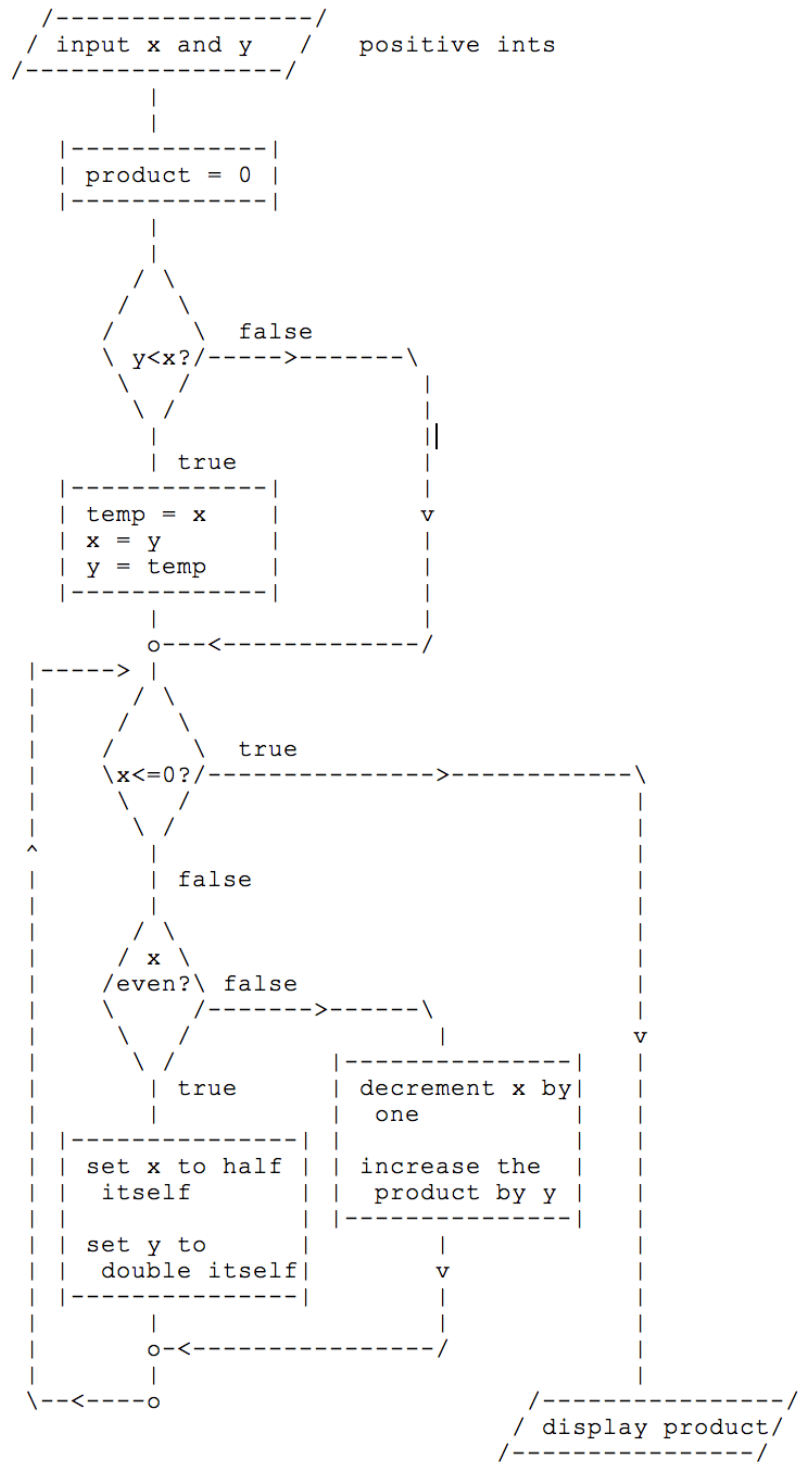


for -statement



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
12	14	0
6	28	0
3	56	0
2	56	56
1	112	56
0	112	168

12 * 14 = 168

Enter x: 72890346

Enter y: 35619

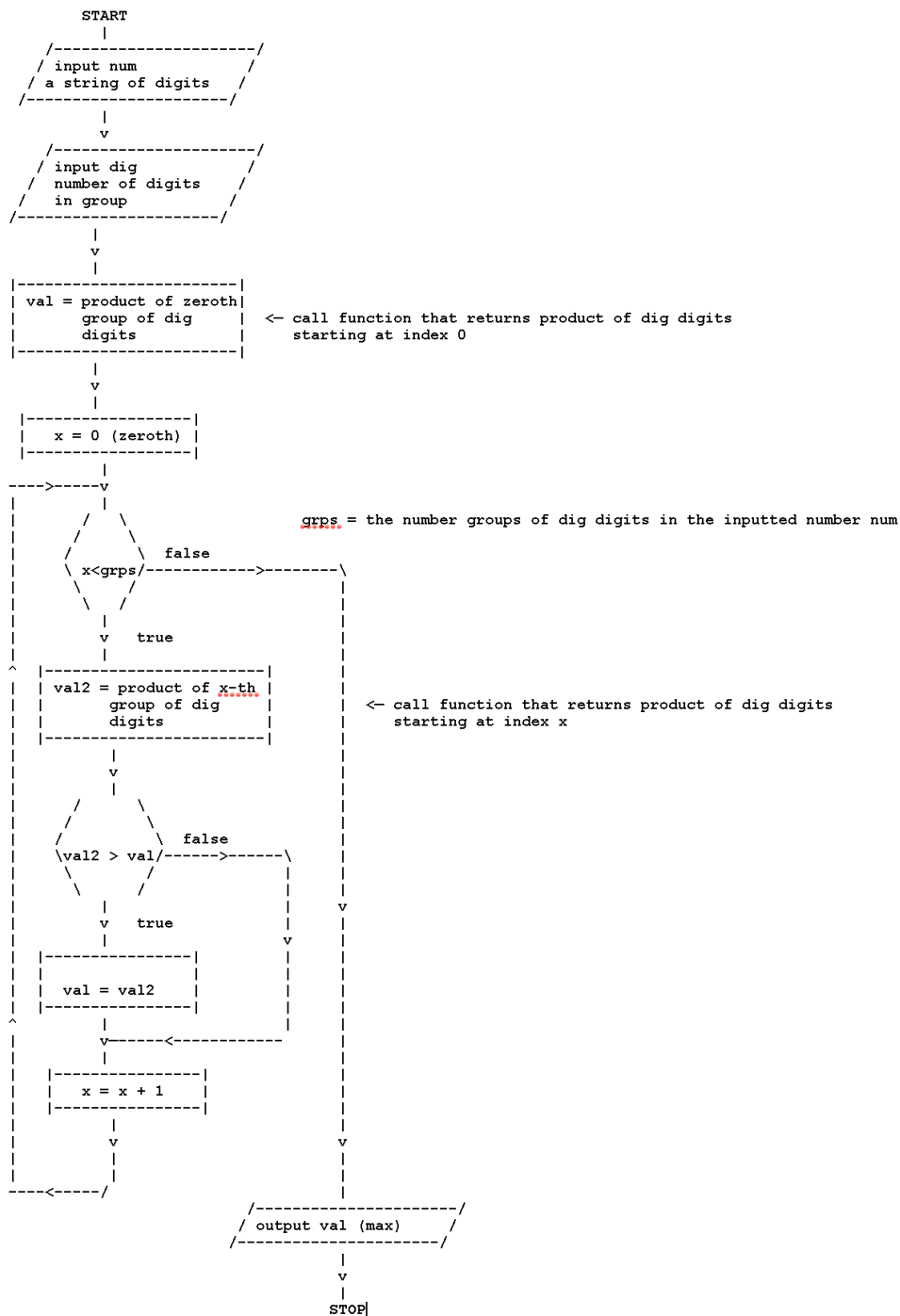
x	y	product
35619	72890346	0
35618	72890346	72890346
17809	145780692	72890346
17808	145780692	218671038
8904	291561384	218671038
4452	583122768	218671038
2226	1166245536	218671038
1113	2332491072	218671038
1112	2332491072	2551162110
556	4664982144	2551162110
278	9329964288	2551162110
139	18659928576	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

35619 * 72890346 = 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.

$7 \times 2 \times 8 = 112$
 $2 \times 8 \times 9 = 144$ <- maximum
 $8 \times 9 \times 0 = 0$
 $9 \times 0 \times 3 = 0$
 $0 \times 3 \times 4 = 12$
 $3 \times 4 \times 6 = 72$



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 72, 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: 845801561660979191338754992

Number of digits in a product: 6

The max product of 6 consecutive digits in 845801561660979191338754992 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:

```
73167176531330624919225119674426574742355349194934
96983520312774506326239578318016984801869478851843
85861560789112949495459501737958331952853208805511
12540698747158523863050715693290963295227443043557
66896648950445244523161731856403098711121722383113
62229893423380308135336276614282806444486645238749
30358907296290491560440772390713810515859307960866
70172427121883998797908792274921901699720888093776
65727333001053367881220235421809751254540594752243
52584907711670556013604839586446706324415722155397
53697817977846174064955149290862569321978468622482
83972241375657056057490261407972968652414535100474
82166370484403199890008895243450658541227588666881
16427171479924442928230863465674813919123162824586
17866458359124566529476545682848912883142607690042
24219022671055626321111109370544217506941658960408
07198403850962455444362981230987879927244284909188
84580156166097919133875499200524063689912560717606
05886116467109405077541002256983155200055935729725
71636269561882670428252483600823257530420752963450
```