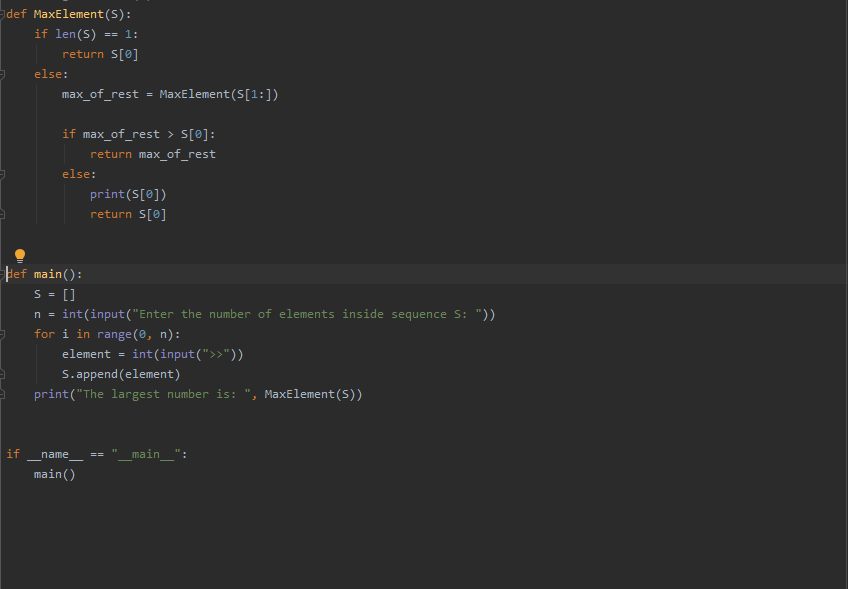
**Task 1:** (R-4.1) Describe a recursive algorithm for finding the maximum element in a sequence, S, of n elements. What is your running time and space usage?





Time complexity: O(n)

Recursive trace and Algorithm Explain:

Number of elements input: 4

Sequence S = [6,1,9,8]

MaxElement([6,1,9,8]) = return 9

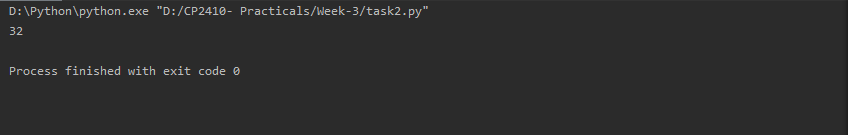
MaxElement([1,9,8]) = return 9

MaxElement([9,8]) = return 9

MaxElement([8]) = return 8

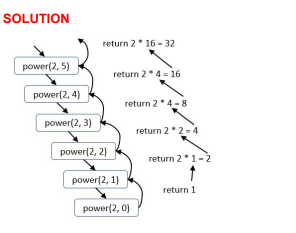
**Task 2:** (R-4.2) Draw the recursion trace for the computation of power(2,5), using the traditional function implemented below:





Result: 32

Recursion trace:



power (2, 5) = return 2 x 2 x 2 x 2 x 2 = 32

power (2, 4) = return 2 x 2 x 2 x 2 = 16

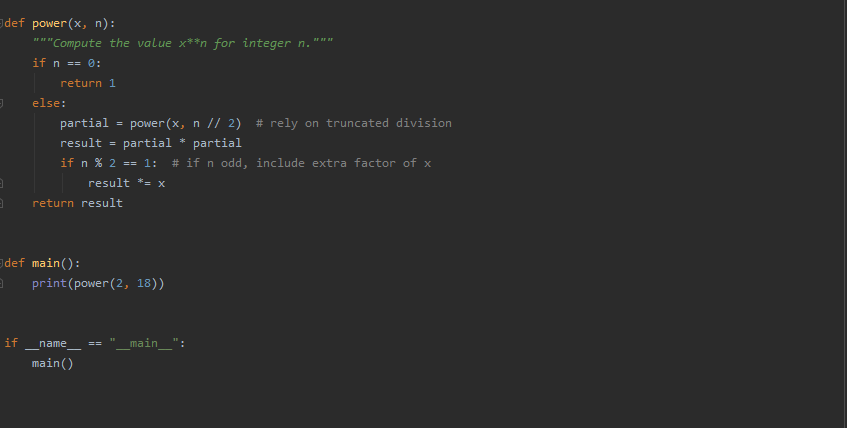
power (2, 3) = return 2 x 2 x 2 = 8

power (2, 2) = return 2 x 2 = 4

power (2, 1) = return 2 x 1 = 2

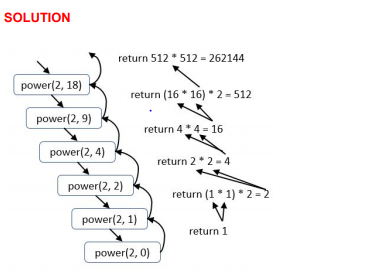
power (2, 0) = return 1

**Task 3:** (R-4.3) Draw the recursion trace for the computation of power(2,18), using the repeated squaring algorithm, as implemented below:





Recursion trace:



power (2, 8) = return 512\*512= 262144

power (2, 9) = return (16\*16) \*2 = 512 // n =9, which is odd

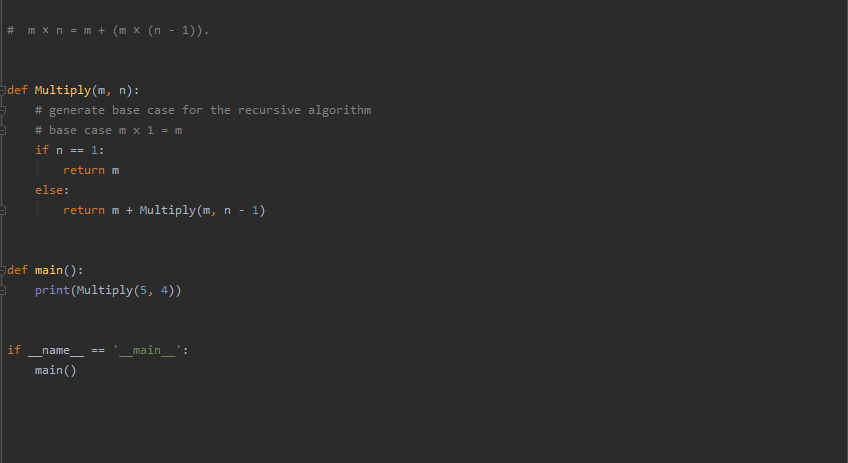
power (2, 4) = return 4\*4 = 16

power (2, 2) = return 2 \* 2 = 4

power (2, 1) = return (1\*1) \*2 =2 // n =1 which is odd

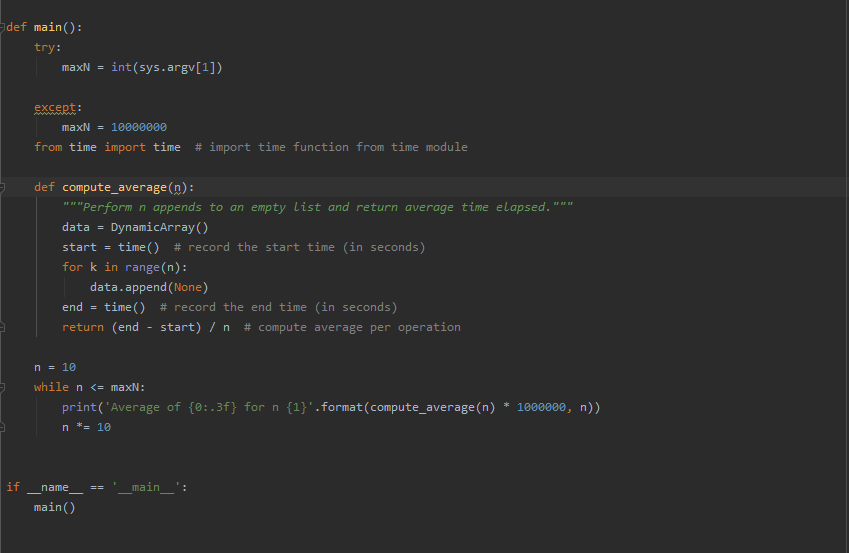
power (2, 0) = return 1

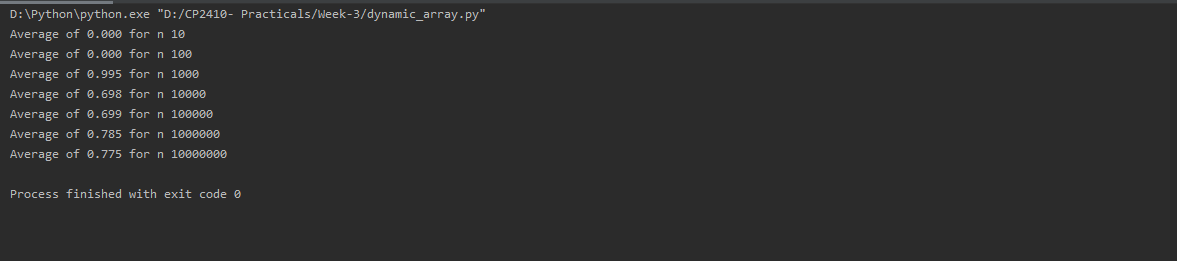
**Task 4:** (C-4.12) Give a recursive algorithm to compute the product of two positive integers, m and n, using only addition and subtraction.





**Task 5:** Modify ch05/experiment\_list\_append.py to investigate the time taken by append operations for DynamicArray (ch05/dynamic\_array.py).





**Task 6:** Create a modified version of DynamicArray (ch05/dynamic\_array.py) that takes a parameter, resize\_factor, which it uses to determine the new size (rather than doubling in the original code - self.\_resize(2 \* self.\_capacity)). Using different values of resize\_factor, examine if and how the average time to append changes.

