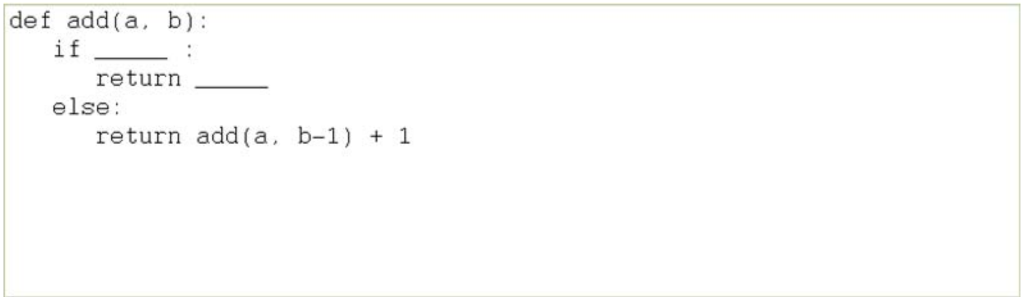
Problem Wk.2.3.4: Introduction to Recursion

Part 1: Add

In a recursive procedure definition we have one or more base cases and one or more recursive cases. Base cases terminate the recursion and return a value without calling the recursive procedure again. Recursive cases call the procedure again, but with an argument that is getting smaller, in some sense.

Here is a recursive definition of addition, using only the operation of adding and subtracting 1. Supply the base case (when b is zero) by replacing the underscores with the appropriate Python expressions.



```
def add(a, b):
   if _____ :
      return _____
   else:
      return add(a, b-1) + 1
```

Part 2: Execution

Consider the `add` procedure above.

1. What conditions must be true of `a` and `b` for the procedure to terminate? Options:

   a and b can be any number
   a can be any number and b must be an integer
   a must be an integer and b can be any number
   a can be any number and b must be a non-negative integer
   a must be a non-negative integer and b can be any number
   a and b must be integers
   a and b must be non-negative integers
   a must be an integer and b must be a non-negative integer
   a must be a non-negative integer and b must be an integer

2. In order to compute `add(5, 2)`, what recursive calls are made to `add` (in sequence)? Enter the values of a and b and enter None if there are too many entries.

   add(5, 2)

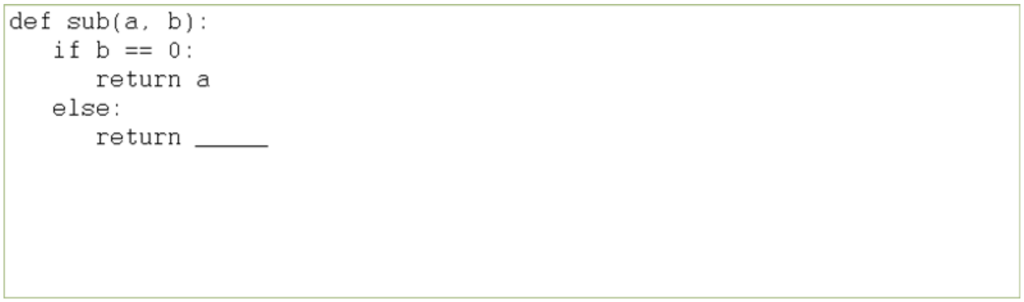
   add(□,□)

   add(□,□)

add(□,□)

Part 3: Sub

Here is a recursive definition of subtraction, using only the operation of adding and subtracting 1. Supply the recursive case by replacing the underscores with the appropriate Python expressions.



```
def sub(a, b):
   if b == 0:
      return a
   else:
      return _____
```