Name-Amrit Aanand

Reg no.-RA1911003010917

1.Write a Python class to find a pair of elements (indices of the two numbers) from a given  array whose sum equals a specific target number.

Code-

class py\_solution:

def twoSum(self, nums, target):

lookup = {}

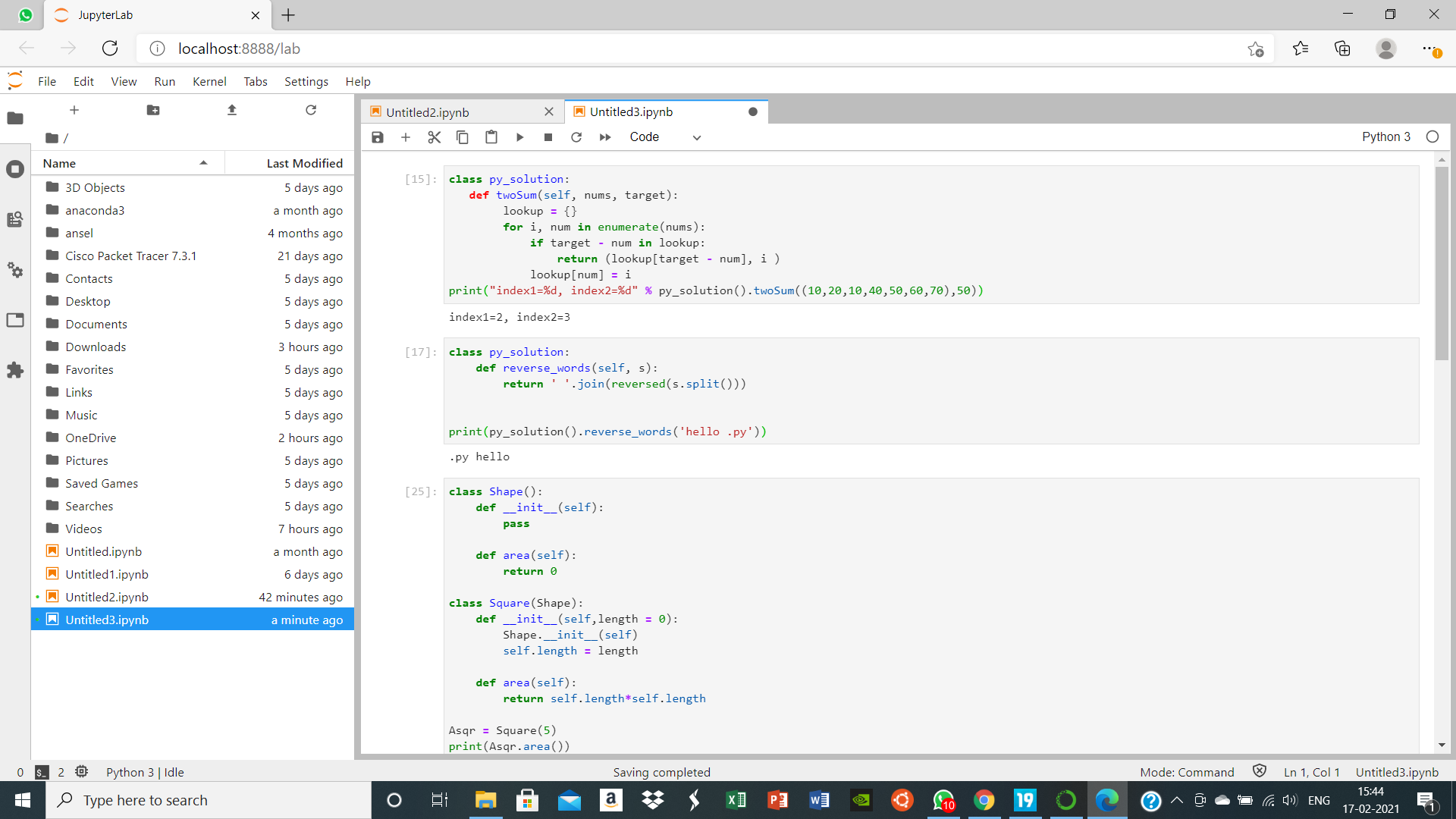
for i, num in enumerate(nums):

if target - num in lookup:

return (lookup[target - num], i )

lookup[num] = i

print("index1=%d, index2=%d" % py\_solution().twoSum((10,20,10,40,50,60,70),50))



1. Write a Python class to reverse a string word by word.

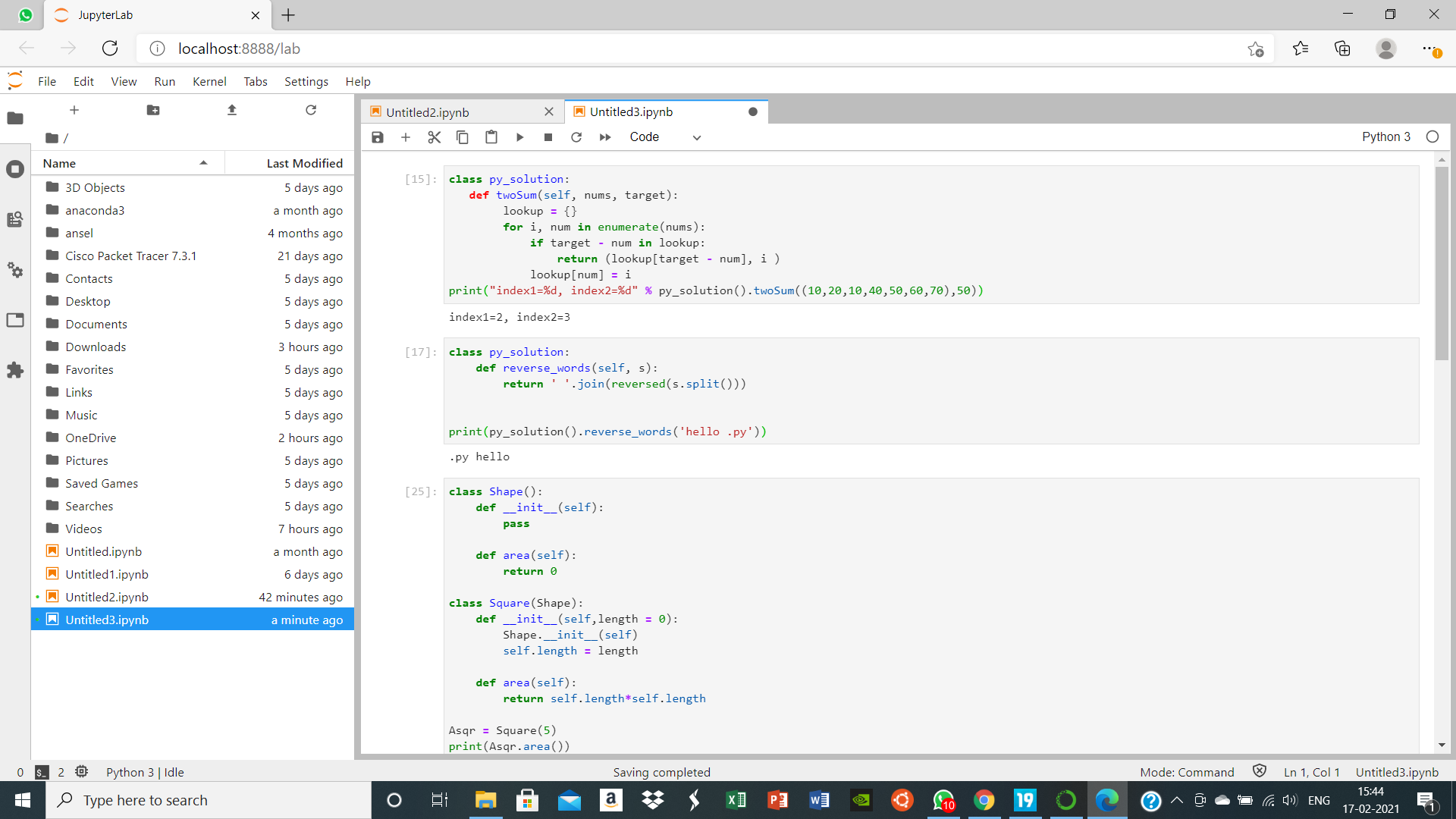
Code-

class py\_solution:

def reverse\_words(self, s):

return ' '.join(reversed(s.split()))

print(py\_solution().reverse\_words('hello .py'))



3. Define a class named Shape and its subclass Square. The Square class has an init function  which takes a length as argument. Both classes have a area function which can print the area  of the shape where Shape's area is 0 by default.

Code-

class Shape():

def \_init\_(self):

pass

def area(self):

return 0

class Square(Shape):

def \_init\_(self,length = 0):

Shape.\_init\_(self)

self.length = length

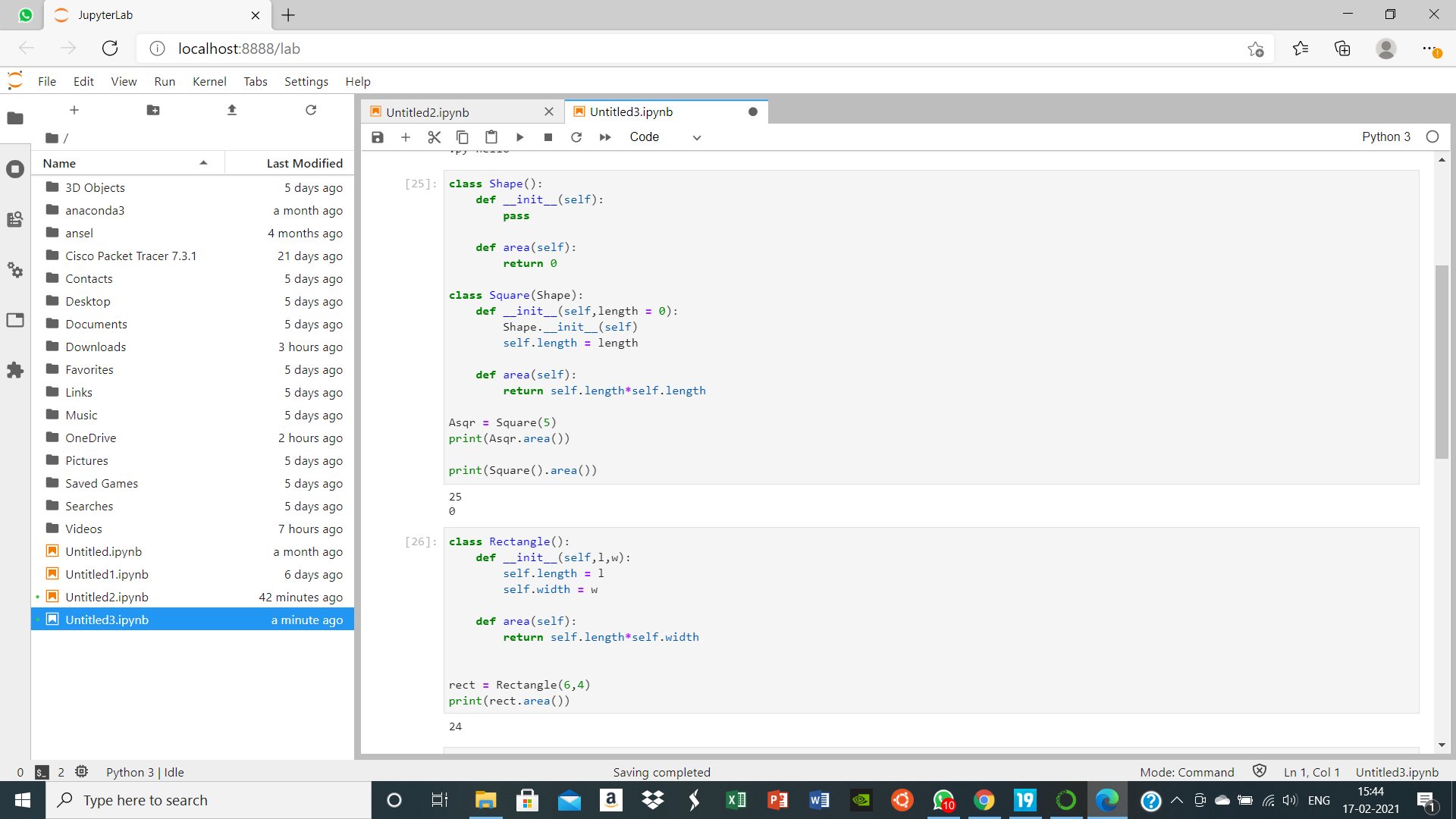
def area(self):

return self.length\*self.length

Asqr = Square(5)

print(Asqr.area())

print(Square().area())



4.Define a class named Rectangle which can be constructed by a length and width. The  Rectangle class has a method which can compute the area.

Code-

class Rectangle():

def \_init\_(self,l,w):

self.length = l

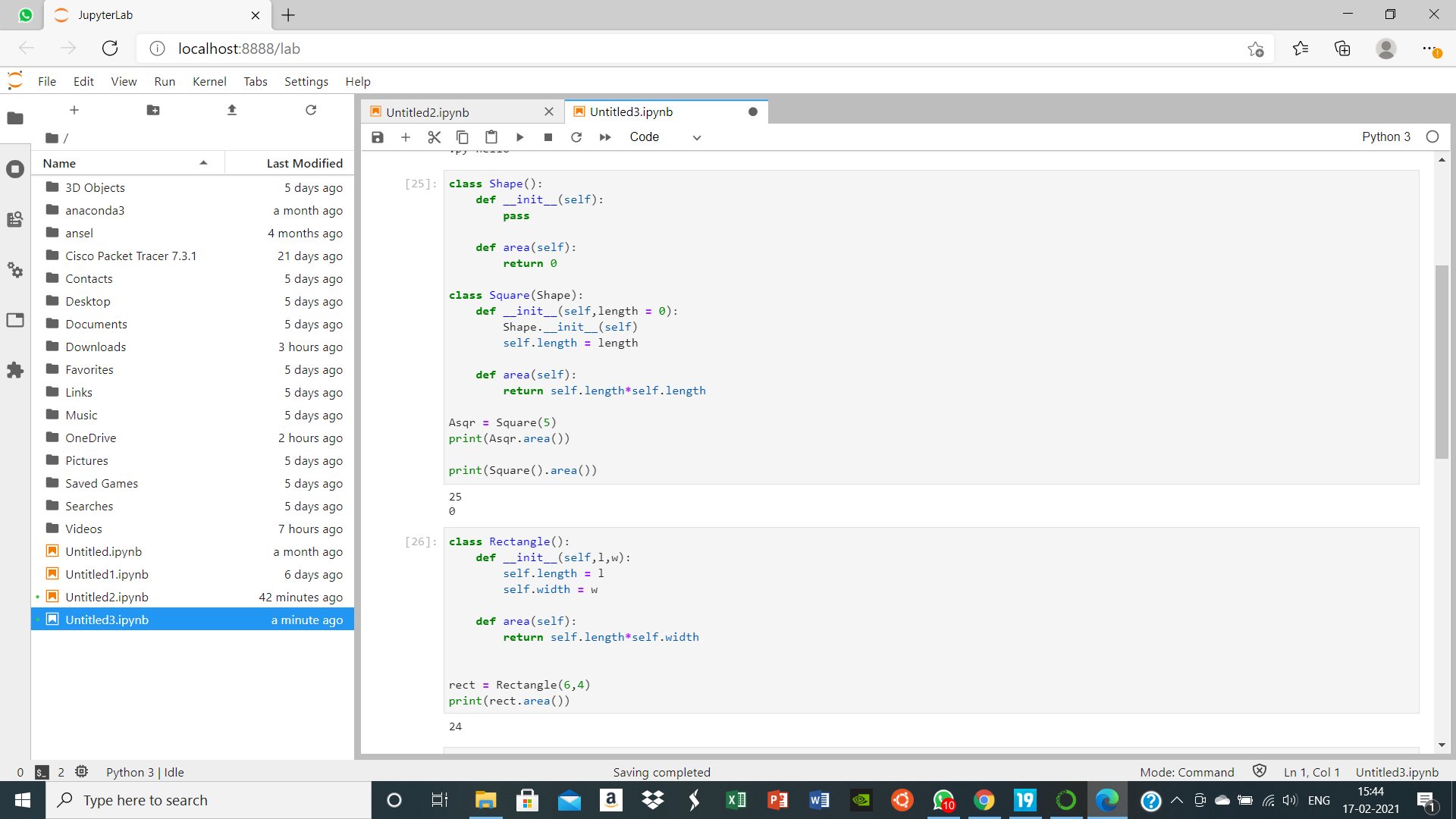
self.width = w

def area(self):

return self.length\*self.width

rect = Rectangle(6,4)

print(rect.area())



5.Write a Python class to find validity of a string of parentheses, '(', ')', '{', '}', '[' and ']. These  brackets must be close in the correct order, for example "()" and "()[]{}" are valid but "[)",  "({[)]" and "{{{" are invalid.

Code-

class py\_solution:

def is\_valid\_parenthese(self, str1):

stack, pchar = [], {"(": ")", "{": "}", "[": "]"}

for parenthese in str1:

if parenthese in pchar:

stack.append(parenthese)

elif len(stack) == 0 or pchar[stack.pop()] != parenthese:

return False

return len(stack) == 0

print(py\_solution().is\_valid\_parenthese("(){}[]"))

print(py\_solution().is\_valid\_parenthese("()[{)}"))

print(py\_solution().is\_valid\_parenthese("()"))

