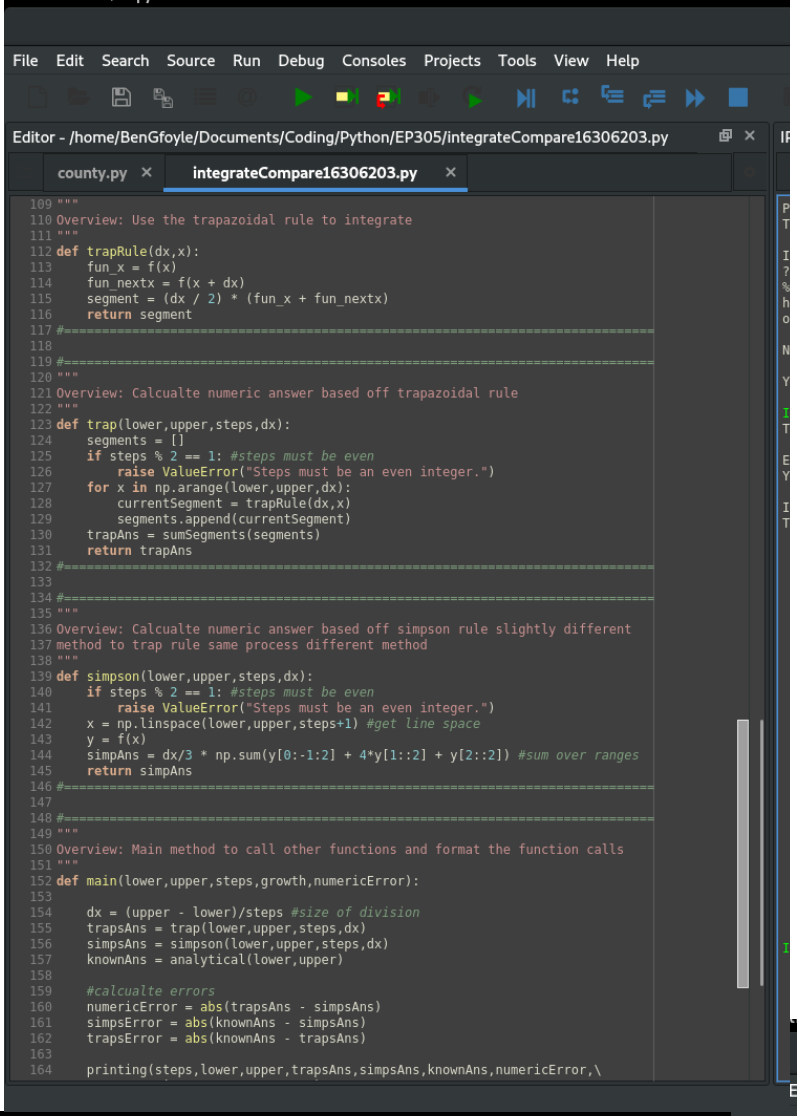
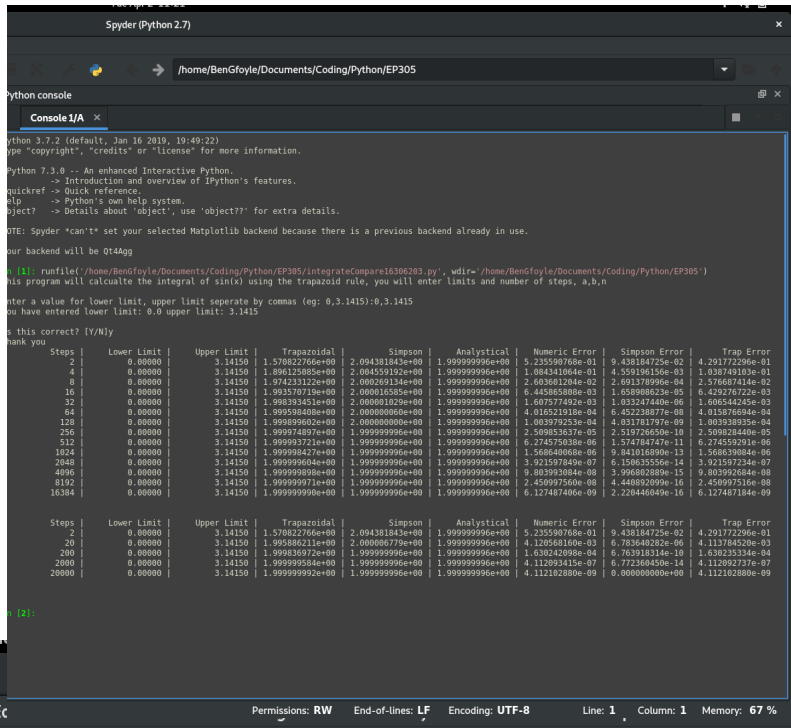
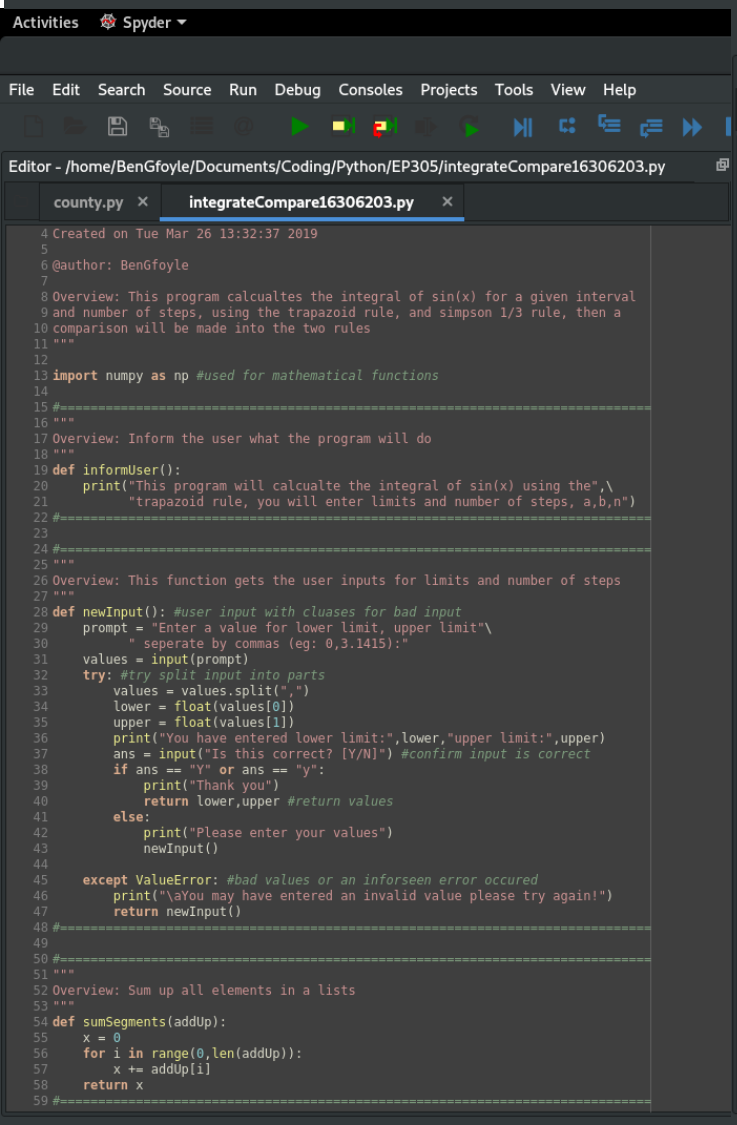
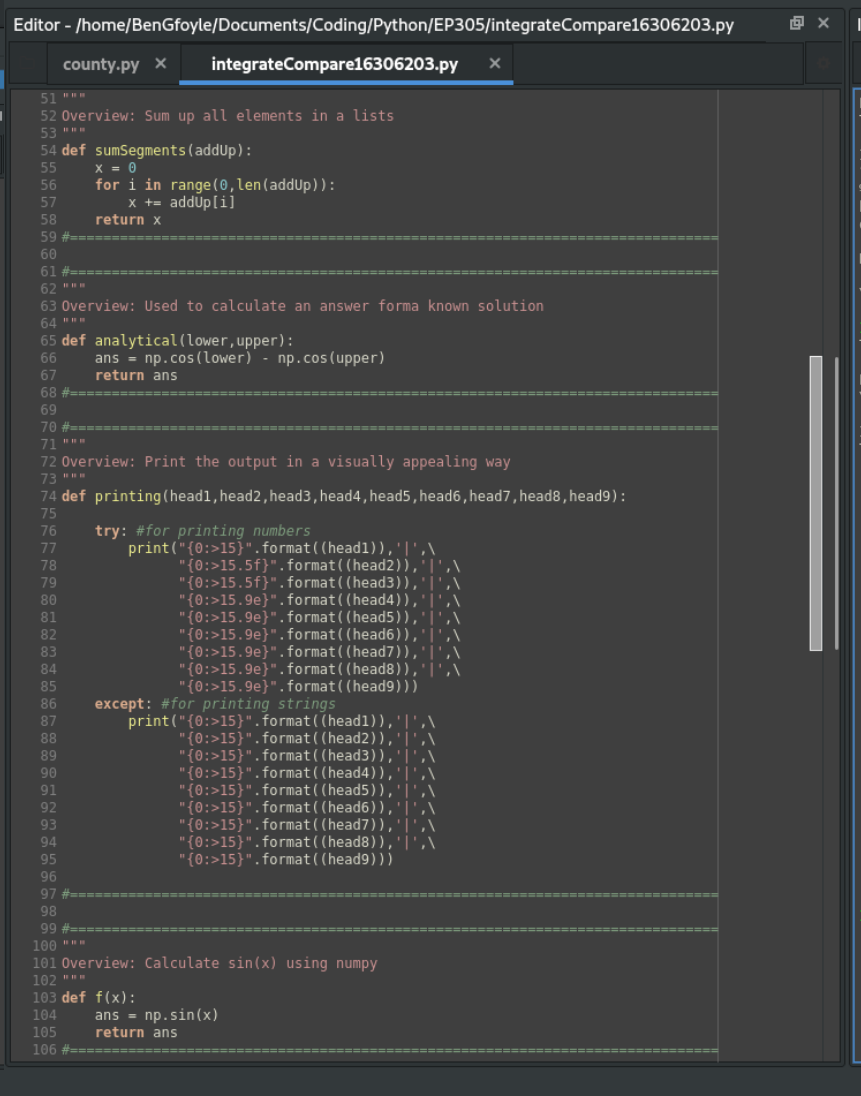



```
"""
Overview: Use the trapazoidal rule to integrate
"""
def trapRule(dx,x):
    fun_x = f(x)
    fun_nextx = f(x + dx)
    segment = (dx / 2) * (fun_x + fun_nextx)
    return segment
#==========================================================================


#==========================================================================
"""
Overview: Calcualte numeric answer based off trapazoidal rule
"""
def trap(lower,upper,steps,dx):
    segments = []
    if steps % 2 == 1: #steps must be even
        raise ValueError("Steps must be an even integer.")
    for x in np.arange(lower,upper,dx):
        currentSegment = trapRule(dx,x)
        segments.append(currentSegment)
    trapAns = sumSegments(segments)
    return trapAns
#==========================================================================


#==========================================================================
"""
Overview: Calcualte numeric answer based off simpson rule slightly different
method to trap rule same process different method
"""
def simpson(lower,upper,steps,dx):
    if steps % 2 == 1: #steps must be even
        raise ValueError("Steps must be an even integer.")
    x = np.linspace(lower,upper,steps+1) #get line space
    y = f(x)
    simpAns = dx/3 * np.sum(y[0:-1:2] + 4*y[1::2] + y[2::2]) #sum over ranges
    return simpAns
#==========================================================================


#==========================================================================
"""
Overview: Main method to call other functions and format the function calls
"""
def main(lower,upper,steps,growth,numericError):

    dx = (upper - lower)/steps #size of division
    trapsAns = trap(lower,upper,steps,dx)
    simpsAns = simpson(lower,upper,steps,dx)
    knownAns = analytical(lower,upper)

    #calcualte errors
    numericError = abs(trapsAns - simpsAns)
    simpsError = abs(knownAns - simpsAns)
    trapsError = abs(knownAns - trapsAns)

    printing(steps,lower,upper,trapsAns,simpsAns,knownAns,numericError,\
```

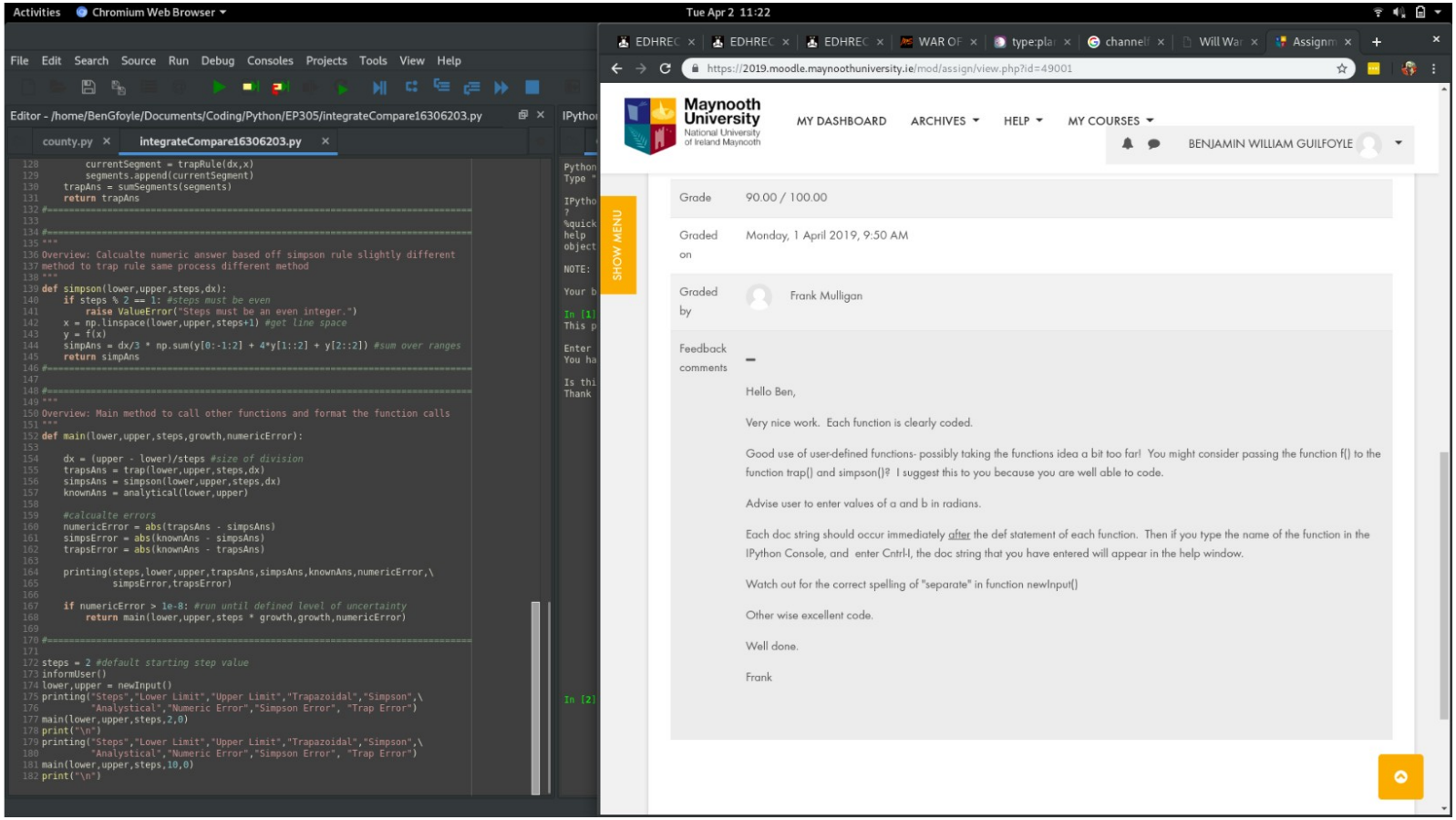





```
Created on Tue Mar 26 13:32:37 2019

@author: BenGfoyle

Overview: This program calcualtes the integral of sin(x) for a given interval
and number of steps, using the trapazoid rule, and simpson 1/3 rule, then a
comparison will be made into the two rules
"""

import numpy as np #used for mathematical functions

#==========================================================================
"""
Overview: Inform the user what the program will do
"""
def informUser():
    print("This program will calcualte the integral of sin(x) using the",\
          "trapazoid rule, you will enter limits and number of steps, a,b,n")
#==========================================================================


#==========================================================================
"""
Overview: This function gets the user inputs for limits and number of steps
"""
def newInput(): #user input with cluases for bad input
    prompt = "Enter a value for lower limit, upper limit"\
          " seperate by commas (eg: 0,3.1415):"
    values = input(prompt)
    try: #try split input into parts
        values = values.split(",")
        lower = float(values[0])
        upper = float(values[1])
        print("You have entered lower limit:",lower,"upper limit:",upper)
        ans = input("Is this correct? [Y/N]") #confirm input is correct
        if ans == "Y" or ans == "y":
            print("Thank you")
            return lower,upper #return values
        else:
            print("Please enter your values")
            newInput()

    except ValueError: #bad values or an inforseen error occured
        print("\aYou may have entered an invalid value please try again!")
        return newInput()
#==========================================================================


#==========================================================================
"""
Overview: Sum up all elements in a lists
"""
def sumSegments(addUp):
    x = 0
    for i in range(0,len(addUp)):
        x += addUp[i]
    return x
#==========================================================================
```




```
"""
Overview: Sum up all elements in a lists
"""
def sumSegments(addUp):
    x = 0
    for i in range(0,len(addUp)):
        x += addUp[i]
    return x
#==========================================================================


#==========================================================================
"""
Overview: Used to calculate an answer forma known solution
"""
def analytical(lower,upper):
    ans = np.cos(lower) - np.cos(upper)
    return ans
#==========================================================================


#==========================================================================
"""
Overview: Print the output in a visually appealing way
"""
def printing(head1,head2,head3,head4,head5,head6,head7,head8,head9):

    try: #for printing numbers
        print("{0:>15}".format((head1)),'|',\
              "{0:>15.5f}".format((head2)),'|',\
              "{0:>15.5f}".format((head3)),'|',\
              "{0:>15.9e}".format((head4)),'|',\
              "{0:>15.9e}".format((head5)),'|',\
              "{0:>15.9e}".format((head6)),'|',\
              "{0:>15.9e}".format((head7)),'|',\
              "{0:>15.9e}".format((head8)),'|',\
              "{0:>15.9e}".format((head9)))
    except: #for printing strings
        print("{0:>15}".format((head1)),'|',\
              "{0:>15}".format((head2)),'|',\
              "{0:>15}".format((head3)),'|',\
              "{0:>15}".format((head4)),'|',\
              "{0:>15}".format((head5)),'|',\
              "{0:>15}".format((head6)),'|',\
              "{0:>15}".format((head7)),'|',\
              "{0:>15}".format((head8)),'|',\
              "{0:>15}".format((head9)))

#==========================================================================


#==========================================================================
"""
Overview: Calculate sin(x) using numpy
"""
def f(x):
    ans = np.sin(x)
    return ans
#==========================================================================
```


Activities
Chromium Web Browser
Tue Apr 2 11:22
File Edit Search Source Run Debug Consoles Projects Tools View Help
Editor - /home/BenGfoyle/Documents/Coding/Python/EP305/integrateCompare16306203.py
county.py
integrateCompare16306203.py
https://2019.moodle.maynoothuniversity.ie/mod/assign/view.php?id=49001
Maynooth University
National University of Ireland Maynooth
MY DASHBOARD
ARCHIVES
HELP
MY COURSES
BENJAMIN WILLIAM GUILFOYLE
SHOW MENU
Grade
90.00 / 100.00
Graded on
Monday, 1 April 2019, 9:50 AM
Graded by
Frank Mulligan
Feedback comments
Hello Ben,
Very nice work. Each function is clearly coded.
Good use of user-defined functions- possibly taking the functions idea a bit too far! You might consider passing the function f() to the function trap() and simpson()? I suggest this to you because you are well able to code.
Advise user to enter values of a and b in radians.
Each doc string should occur immediately after the def statement of each function. Then if you type the name of the function in the IPython Console, and enter CntrI-I, the doc string that you have entered will appear in the help window.
Watch out for the correct spelling of "separate" in function newInput()
Other wise excellent code.
Well done.
Frank