Exam 3

This exam is confidential and is not to be shared with anyone not in ChEn 263 Fall 2021

Problem 1 (3 points)

At the beginning of the Lesson 14 notes posted on Learning Suite (entitled *quad.ipynb*) is some boilerplate code to import needed functionality. Do the following:

- Take those lines of code and make them into their own Python module. Except do not include in the module the line of code that begins with `%` : it won't work.
- Store that module file in the working directory on your computer
- Using a command below, import all the contents from that module into this notebook, such that prefixes are not necessary to access the contents of the module (hint: the command involves an `*`)
- Next you can include below the line of code that begins with `%`
- Prove that the import worked by executing the command `'quad' in dir()` and generating a `True` result

In [1]:
```
from all_imports import *
%matplotlib inline

'quad' in dir()
```

Out[1]:
```
True
```

Problem 2 (8 points)

Do the following:

- Make a tuple called `yes` with values 5 and 10
- Make a numpy array called `I` with values 5 and 10
- Make a dictionary called `can` with any values of your choice
- Make a Python list called `pod` that contains the above three objects
- Using array slicing, print out the contents of the last two elements of `pod`

In [2]:
```python
# Define tuple
yes = 5, 10

# Define numpy array
I = np.array([5,10])

# Define dictionary
can = {0:'zero', 1:'one', 2:'two', 3:'three', 4:'four', 5:'five', 6:'six', 7:'seven', 8:'eight', 9:'nine', 10: 'ten'}

# Combine data strucutres in a list
pod = [yes, I, can]

# Print out the last two data strucutres
print("Last two elements of pod:\n{P}".format(P=pod[-2:]))
```

```
Last two elements of pod:
[array([ 5, 10]), {0: 'zero', 1: 'one', 2: 'two', 3: 'three', 4: 'four', 5: 'five', 6: 'six', 7: 'seven', 8: 'eight', 9:
'nine', 10: 'ten'}]
```

Problem 3 (20 points)

In Homework 10 and Exam 2 you were asked to generate series solutions to the Gaussian error function, which is defined as

$$\mathrm{erf}(x) = \frac{2}{\sqrt{\pi}} \int_0^x e^{-t^2} dt$$

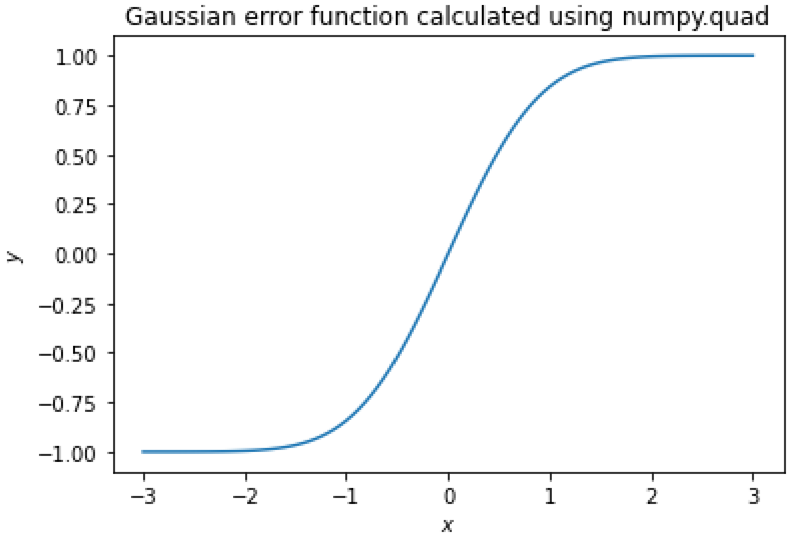
Implement $\mathrm{erf}(x)$ again, this time using numerical integration in Python. Then make a smooth plot of $\mathrm{erf}(x)$ for $-3 \leq x \leq 3$. Your plot should be appropriately formatted.

You might want to make sure your function is working properly by comparing results to the built-in `math.erf` function (available once you `import math` in your code). If you cannot get your erf function to work in the plot then you can use another function, though you will miss some points.

In [3]:
```python
# Define erf function
def erf(x):
    return 2 / np.sqrt(np.pi) * quad(lambda t: np.exp(-1*t**2), 0, x)[0]

# Create x-linespace and populate y-axis
x = np.linspace(-3,3,100)
y = np.zeros(100)
for i in range(100):
    y[i] = erf(x[i])
```

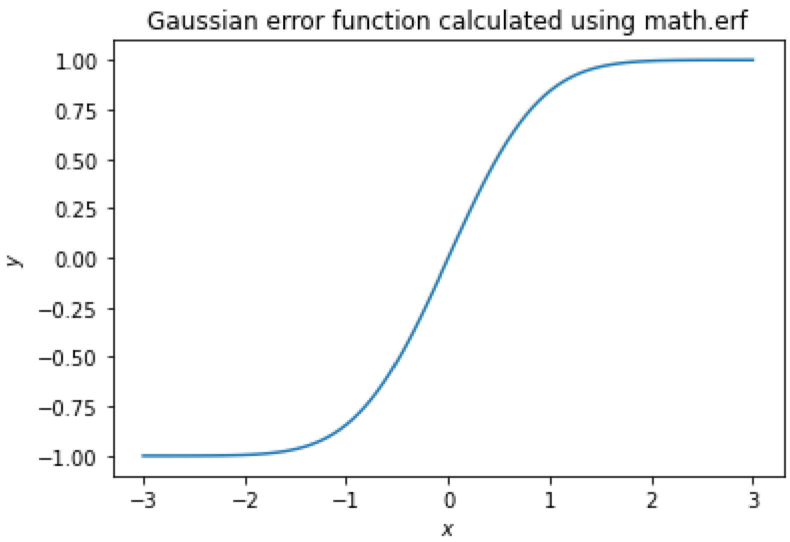
```python
# Plot the x and y axies
plt.title(r"Gaussian error function calculated using numpy.quad")
plt.xlabel(r"$x$")
plt.ylabel(r"$y$")
plt.plot(x, y);
```



In [4]:

```python
# import math and overwrite the y-axis
import math
y = np.vectorize(math.erf)(x)

# Replot the x and y axies
plt.title(r"Gaussian error function calculated using math.erf")
plt.xlabel(r"$x$")
plt.ylabel(r"$y$")
plt.plot(x, y);
```



Problem 4 (14 points)

Make a Python class called `polygon` that:

- Can construct a regular polygon with two required values: number of sides n and length of side L
- Prints a warning if the user tries to construct a polygon object with $n < 3$ or $L \leq 0$
- Has a member function that computes and returns the *area* of the polygon according to the formula

$$A = \frac{L^2 n}{4\tan(\pi/n)}$$

  where $\tan()$ is found in numpy and n and L are the previously stored values
- Has a reasonable amount of documentation

Test your class by making the three polygon objects below and printing out the area of each. Your printed statements should be self-explanatory.

- a square with length 1
- a hexagon with length 0.6
- a hectogon (100 sides!) with length 0.03

In [5]:

```python
class polygon():

    def __init__(self, n, L):          # n = number of equilateral sides. L = length of each side.
        if n < 3 or L <= 0:
            print("Error, invalid polygon")
            return
        self.n = n
        self.L = L
        return

    def area(self):                    # Calculates the area of the polygon using the stored values.
        return self.L**2 * self.n / (4 * np.tan(np.pi/self.n))

    def print_area(self):              # Prints the calculated area.
        print("The area of a/an {N}-sided polygon with side length {L:.2f} is equal to {A:.2f}".format(
            N = self.n,          # number of sides
            L = self.L,          # length of sides
            A = self.area()      # area of polygon
        ))
        return

square = polygon(4, 1)
square.print_area()

hexagon = polygon(6, 0.6)
hexagon.print_area()

hectogon = polygon(100, 0.03)
hectogon.print_area()
```

```
The area of a/an 4-sided polygon with side length 1.00 is equal to 1.00
The area of a/an 6-sided polygon with side length 0.60 is equal to 0.94
The area of a/an 100-sided polygon with side length 0.03 is equal to 0.72
```

Leave 5 minutes for uploading your work

Instructions: Just like for Exam 2, you must submit your exam as a pdf file, but first make sure it is properly displaying the results from your Python code. To this end, do the following

1. Run $\longrightarrow$ Restart Kernel and Run All Cells
2. Covert the ipynb file to html (File $\longrightarrow$ Export Notebook As $\longrightarrow$ HTML)

3. Open the html file in your browser and *print to pdf*. Make sure file name has been changed to your personal name. When you print to pdf, you can shrink the size of the pdf (i.e. less than 100%) to make longer lines of code fit.

```
In [ ]:
```