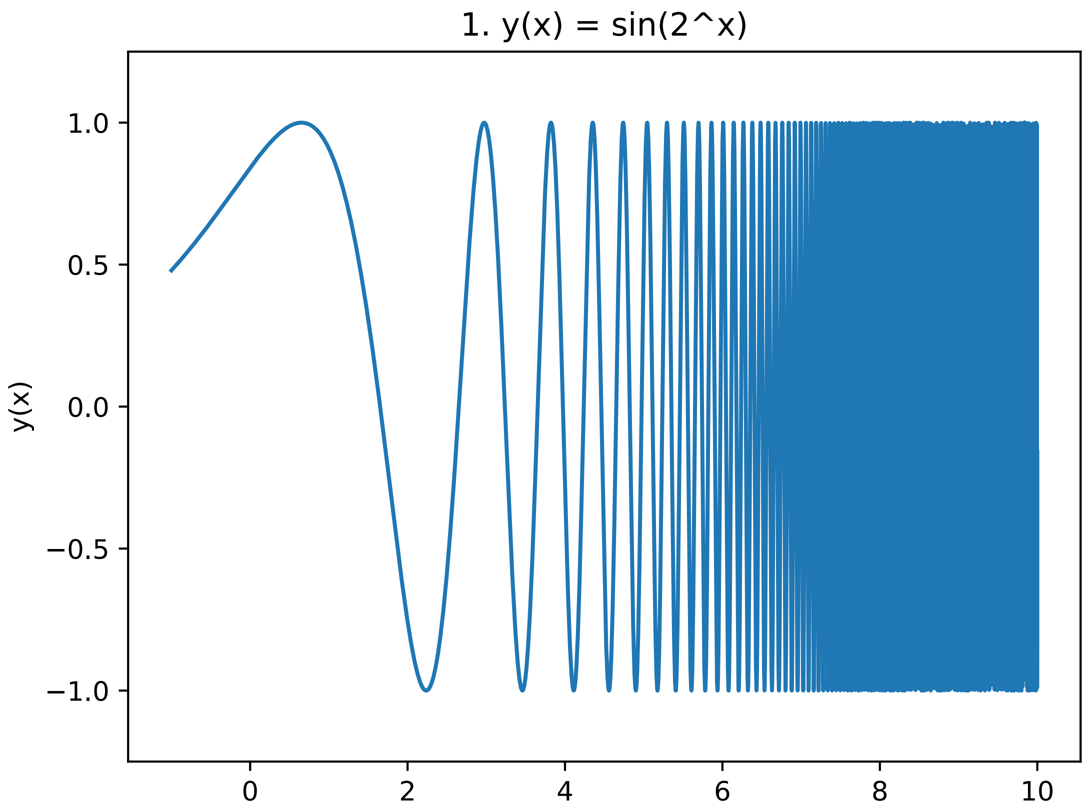
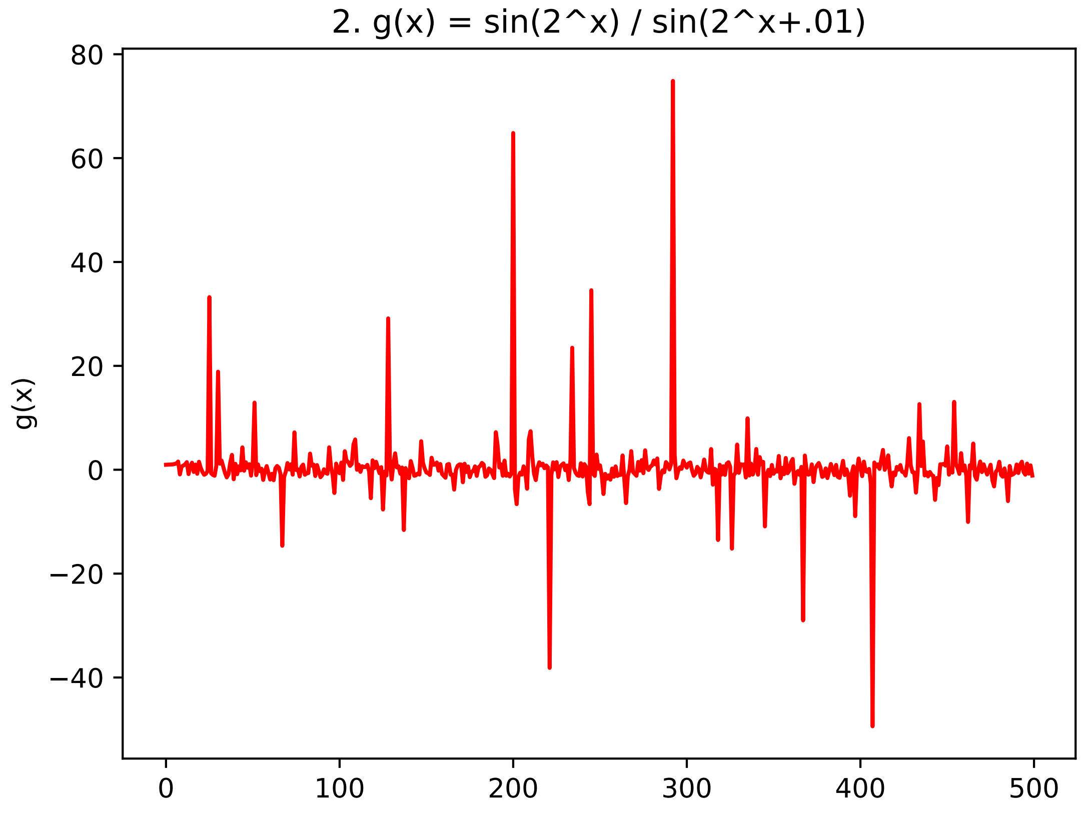
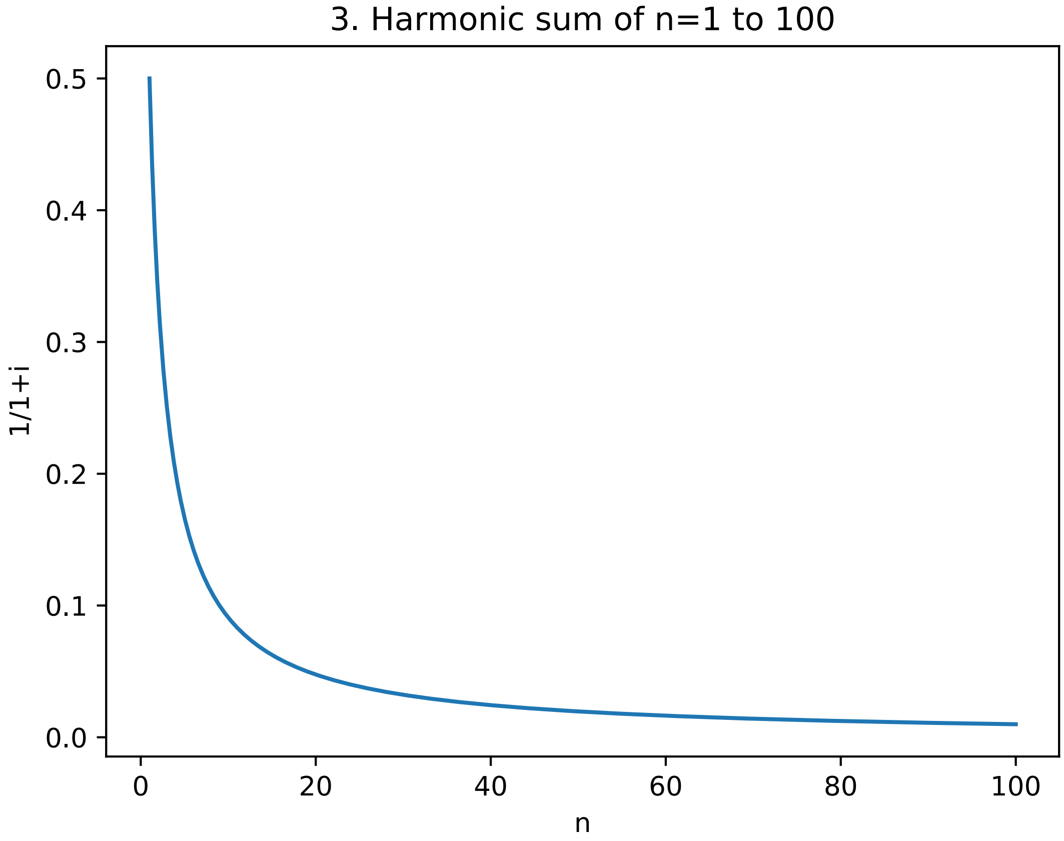
```python
"""written by Joseph Hayes for UC Berkeley CogSci 131, Spring 2020"""

import numpy as np
import matplotlib.pyplot as plt
import math
#
# #problem 1
def xPowSin(x):
    return np.sin(pow(2,x))


x = np.linspace(-1, 10, 10000)
y = xPowSin(x)

plt.title('1. y(x) = sin(2^x)')
plt.xlabel('x')
plt.ylabel('y(x)')
plt.plot(x, y)
plt.ylim(-1.25, 1.25)
# plt.savefig('one_plot.pdf')
plt.show()

#problem 2
x = np.linspace(0, 100, 500)
g = xPowSin(x) / (xPowSin(x + 0.1))

plt.title('2. g(x) = sin(2^x) / sin(2^x+.01)')
plt.xlabel('x')
plt.ylabel('g(x)')

plt.plot(g, 'red')
# plt.savefig('two_plot.pdf')
plt.show()


#problem 3a
"""returns the harmonic of k"""
def harmonic(k):
    return 1 / (k + 1)

"""returns the harmonic series sum up to n"""
def harmonicSum(n):
    total = 0
    for i in n:
        total = total + harmonic(i)

    return total

#problem 3b
x = np.linspace(1, 100, 1000)
```

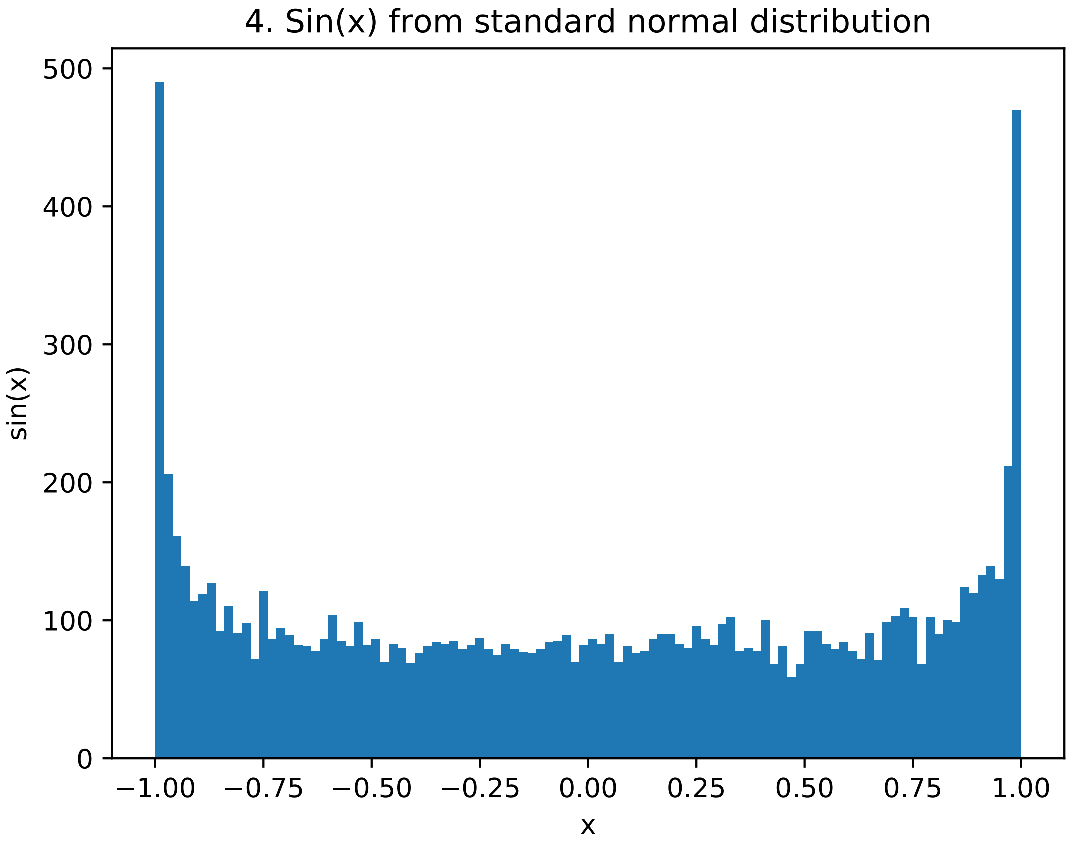
```
y = harmonic(x)
plt.title('3. Harmonic sum of n=1 to 100')
plt.xlabel('n')
plt.ylabel('1/1+i')
plt.plot(x, y)
# plt.savefig('three_b_plot.pdf')
plt.show()


#finished, check scaling

"""plots a histogram of func with sample type of numSamples"""
def histoPlot(func, sampleType, mean=0, stdev=1, numSamples=None,
scale=None):
    x = sampleType(mean, stdev, numSamples)

    if numSamples != None and scale != None:
        plt.hist(func(x), numSamples // scale)


#problem 4
plt.title('4. Sin(x) from standard normal distribution')
plt.xlabel('x')
plt.ylabel('sin(x)')
histoPlot(np.sin, np.random.normal, 0, 1, 10000, 100)
# plt.savefig('four_plot.pdf')
plt.show()


#problem 5
plt.title('5. Exponential of uniform distribution [0, 1.5]')
plt.xlabel('x')
plt.ylabel('e^x')
histoPlot(np.exp, np.random.uniform, 0, 1.5, 10000, 100)
# plt.savefig('five_plot.pdf')


#problem 6a
def getDiff(array): #may need to rewrite
    for i in range(len(array) - 1):
        array[i] = abs(array[i + 1] - array[i])

    array = np.delete(array, 999) #

    return array


def adjHistoPlot(n=1000):
    sample = np.random.normal(0, 1, n)
    sample = np.sort(sample)
    diffArray = getDiff(sample)

    plt.hist(diffArray, n)
    plt.title('6a. Differences Between Normally Distributed Samples')
    plt.xlabel('differences between samples after sorting')
    plt.ylabel('frequency')
    plt.xlim(0, .05)
    # plt.savefig('six_a_plot.pdf')
    plt.show()

    return diffArray

diffArr = adjHistoPlot(1000)
#most samples have a difference of less than .01 between them

#problem 6b Plot the position in the sorted array (first, second,
etc.) vs the difference computed in 6a.
x = np.arange(999)
plt.plot(x, diffArr, color='red')
plt.title("6b. Differences between indices of sorted array")
plt.xlabel('index')
plt.ylabel('difference: arr[i + 1] - arr[i]')
# plt.savefig('six_b_plot.pdf')
plt.show()

#end
```

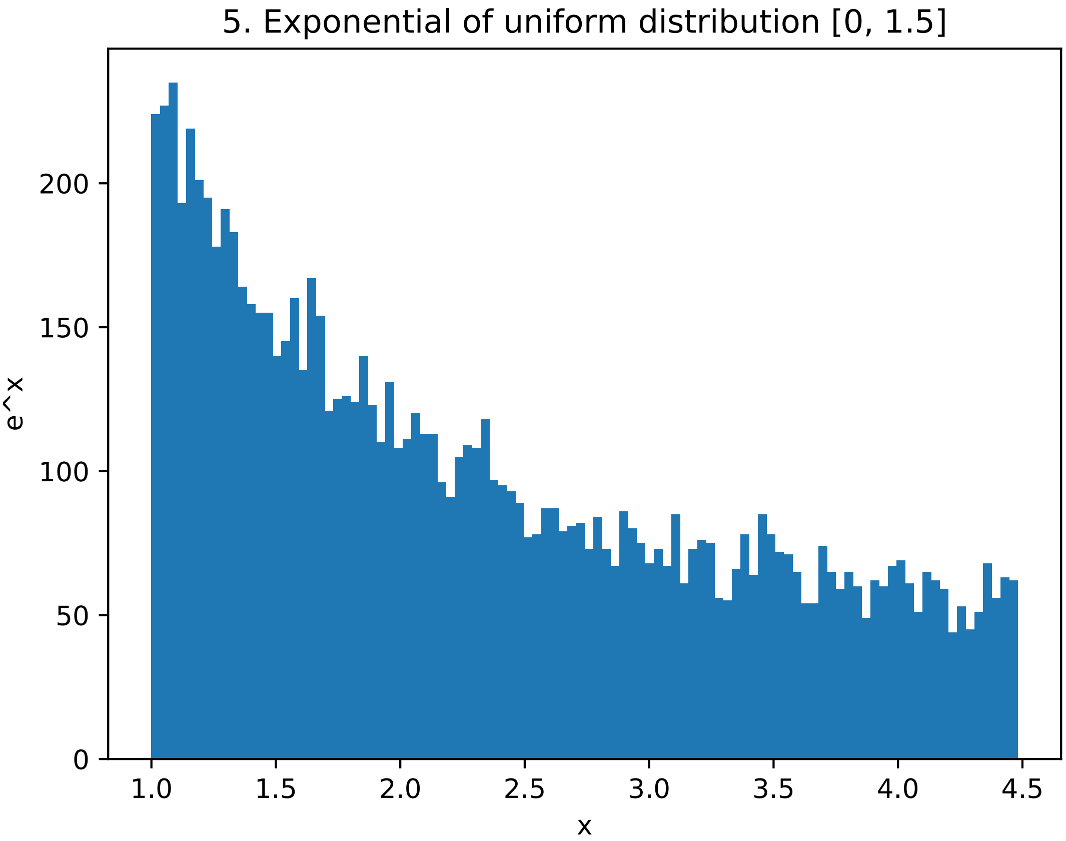

1. y(x) = sin(2^x)
y(x)
x
1.0
0.5
0.0
−0.5
−1.0
0
2
4
6
8
10

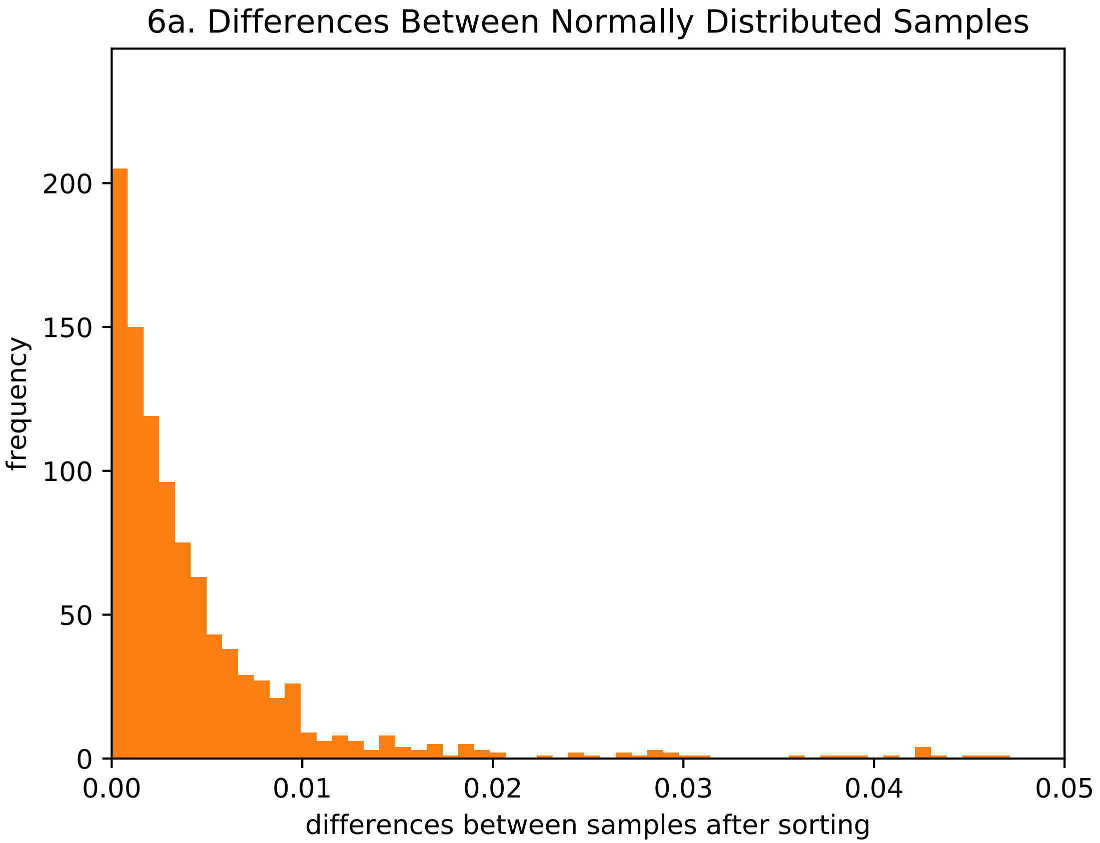


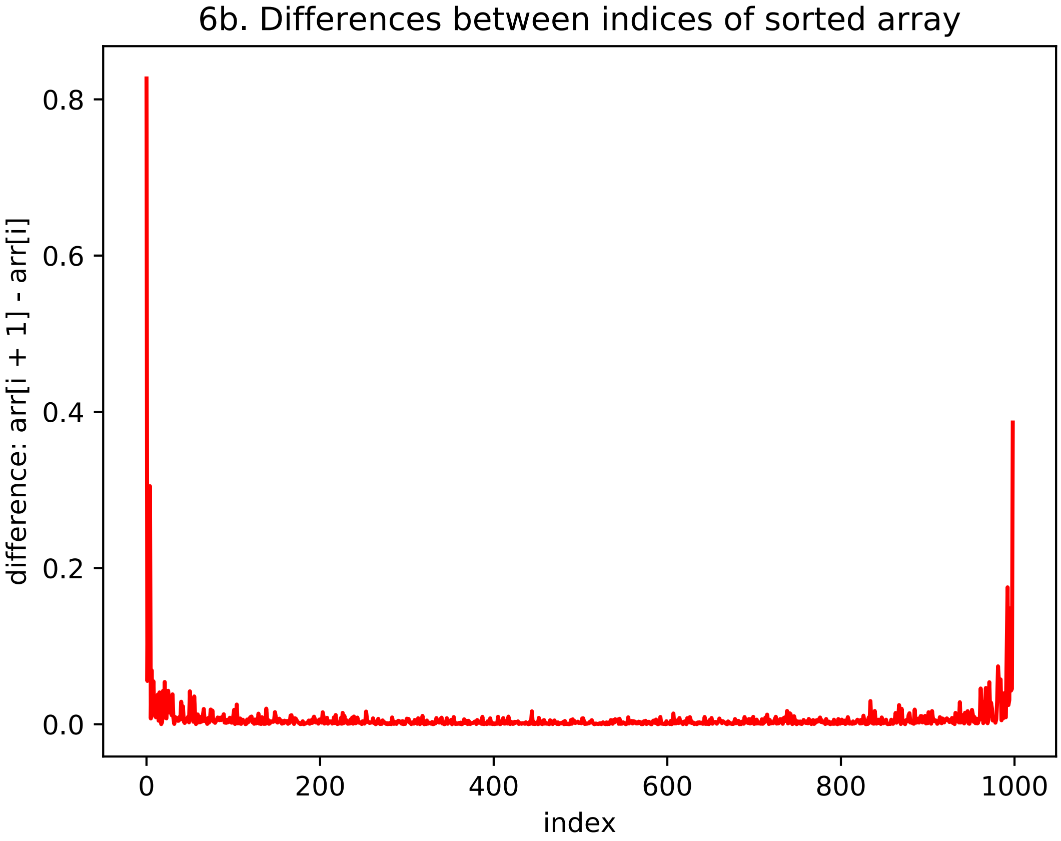
2. g(x) = sin(2^x) / sin(2^x+.01)
80
60
40
20
0
−20
−40
g(x)
0
100
200
300
400
500
x


3. Harmonic sum of n=1 to 100
1/1+i
n
0.0
0.1
0.2
0.3
0.4
0.5
0
20
40
60
80
100


4. Sin(x) from standard normal distribution
sin(x)
0
100
200
300
400
500
−1.00
−0.75
−0.50
−0.25
0.00
0.25
0.50
0.75
1.00
x


5. Exponential of uniform distribution [0, 1.5]
e^x
x
0
50
100
150
200
1.0
1.5
2.0
2.5
3.0
3.5
4.0
4.5


6a. Differences Between Normally Distributed Samples
frequency
differences between samples after sorting
0
50
100
150
200
0.00
0.01
0.02
0.03
0.04
0.05


6b. Differences between indices of sorted array
difference: arr[i + 1] - arr[i]
index
0.0
0.2
0.4
0.6
0.8
0
200
400
600
800
1000