Untitled12

January 23, 2024

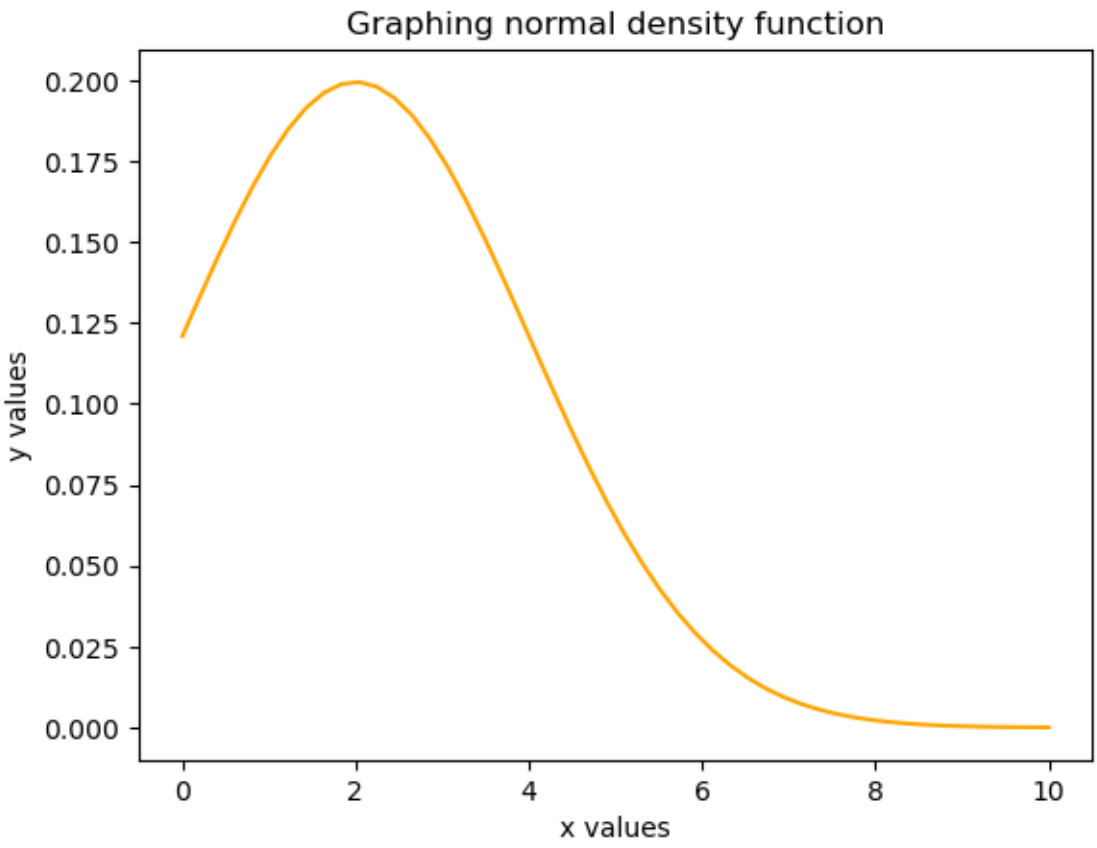
[4]:
```
import numpy as np

def normal_density(mean, variance, x):
      return (1/(np.sqrt(2*np.pi*(variance**2)))*np.exp((-(x-mean)**2)/
 ↪(2*variance**2)))
print (normal_density(2,2,0))
```

```
0.12098536225957168
```

[12]:
```
#Graphing normal density function for mean value of 2 and variance value of 2.␣
 ↪Mean and variance values can be adjusted to change the shape of the graph.
import numpy as np
import matplotlib.pyplot as plt
mean=2
variance=2

def normal_density(mean,variance,x):
     return (1/(np.sqrt(2*np.pi*(variance**2)))*np.exp((-(x-mean)**2)/
 ↪(2*variance**2)))
x=np.linspace (0,10)
plt.plot (x, normal_density(mean,variance,x), color= "orange")
plt.ylabel ("y values")
plt.xlabel ("x values")
plt.title ("Graphing normal density function")
plt.show ()
```



[39]:
```
#Using trapezoid rule in Chapter 7 to write a function that
#integrates the function from before on integral a, b.

#Set bounds of the integral (a,b)
a=-2
b=2

#Using the same mean and variance values from above
mean=2
variance=2

#Redefining the same function as above
def normal_density(mean,variance,x):
    return (1/(np.sqrt(2*np.pi*(variance**2)))*np.exp((-(x-mean)**2)/
 ↪(2*variance**2)))

#Setting up the integration code using trapezoid rule
def integration_by_trapezoid (func,a,b,num_points=5000): #num_points is the␣
 ↪amount of points scattered in the array of x values
```

```
    x_coordinates= np.linspace(a,b,num_points+1) #x coordinates scattered␣
 ↪across a and b, num_points+1 amount of coordinates
    the_integral=np.trapz(func(x_coordinates), x_coordinates) #using trapezoid␣
 ↪rule to return the integral
    return the_integral

def normal_density1 (x): #making normal density a function of x to use in␣
 ↪probability function
    return normal_density(mean,variance,x)

#Coding for probability

func_prob = integration_by_trapezoid(normal_density1, a, b)

print(f'The solution is: {func_prob}')
```

```
The solution is: 0.4772498666120617
```

[40]:
```
#Using code from above, and adjusting parameters to fit part 4:

#Set bounds of the integral. The lower bound is 162 cm and the upper bound is␣
 ↪190 cm.
a=162
b=190

#Adjusting the mean and variance values based on values provided in the question
mean=171
variance= 7.1

#Redefining the same function as above
def normal_density(mean,variance,x):
    return (1/(np.sqrt(2*np.pi*(variance**2)))*np.exp((-(x-mean)**2)/
 ↪(2*variance**2)))

#Setting up the integration code using trapezoid rule
def integration_by_trapezoid (func,a,b,num_points=5000): #num_points is the␣
 ↪amount of points scattered in the array of x values
    x_coordinates= np.linspace(a,b,num_points+1) #x coordinates scattered␣
 ↪across a and b, num_points+1 amount of coordinates
    the_integral=np.trapz(func(x_coordinates), x_coordinates) #using trapezoid␣
 ↪rule to return the integral
    return the_integral

def normal_density1 (x): #making normal density a function of x to use in␣
 ↪probability function
    return normal_density(mean,variance,x)
```

```
#Coding for probability

func_prob = integration_by_trapezoid(normal_density1, a, b)

print(f'The solution is: {func_prob}')
```

```
The solution is: 0.8938058711189701
```

[ ]: