Midterm Exam

DUE DATE and SUBMISSION INSTRUCTIONS:

This exam is due on Gradescope AND Canvas by **5:59PM on Friday October 21**.

- Upload a PDF of this notebook (with all cells run and all LaTex fully rendered) to Gradescope by **5:59PM on Friday October 21**.
- AND upload this Jupyter notebook to Canvas by **5:59PM on Friday October 21**.

Late Submittals: If your exam comes in between 6:00PM and 6:30PM on Friday Oct 21st (regardless of reason/cause etc), then it will be **deducted 50 points**. **No exams will be accepted after 6:30 PM** on Friday October 21.

FORMAT:

- Your solutions to theoretical questions should be done in Markdown and LaTex directly below the associated question.
- Your solutions to computational questions should include any specified Python code and results as well as written commentary on your conclusions.

HERE ARE THE RULES:

1. All work, code and analysis, must be your own.
2. Only work uploaded prior to the deadline will be considered.
3. You MAY use your course notes, posted lecture slides, textbooks, in-class notebooks, and homework solutions as resources. You may also search online for answers to general knowledge questions like the form of a probability distribution function or how to perform a particular operation in Python/Pandas.
4. You may **NOT** copy-paste solutions *from anywhere*.
5. You may **NOT** post to message boards or other online resources asking for help.
6. You may **NOT** collaborate with classmates or anyone else.
7. In short, **your work must be your own**. It really is that simple.
8. This is meant to be like a coding portion of your midterm exam. So, the instructional team will be much less helpful than we typically are with homework. For example, we will not check answers, help debug your code, and so on.
9. If something is left open-ended, it is because we want to see how you approach the kinds of problems you will encounter in the wild, where it will not always be clear what sort of tests/methods should be applied. Feel free to ask clarifying questions though.
10. Please stick to software used/taught in this class: Pandas, NumPy, Python 3.9, etc.

Violation of the above rules will result in an immediate academic sanction (you will receive an F in the course), and a trip to the Honor Code Council.

By submitting this assignment, you agree to abide by the rules given above.

Name: CJ Kennedy

NOTES:

- By the due date, turn in anything you have, there are no extensions, and some credit is better than no credit. **DO NOT attempt your first upload at 5:00 PM** as you may encounter slow internet speeds or outages and again you will miss the upload and hence the entire exam.
- If you have a question for us, post it as a **PRIVATE** message on Piazza. If we decide that the question is appropriate for the entire class, then we will add it to a midterm clarifications thread.
- Do **NOT** load or use any Python packages that are not available in Anaconda 3.9.
- This should go without saying, but... For any question that asks you to calculate something, you **must show all work to receive credit**. Sparse or nonexistent work will receive sparse or nonexistent credit.

Example question #1

On this exam we are looking for HOW you answer questions and not necessarily WHAT you put for an answer.

Tell the grader where to look for your answer and which question the work goes with.

Be thorough, but be concise.

Use full sentences and proper grammar and correct spelling.

Comment your code.

Demonstrate BOTH what you know about the question AND the coding you used to answer the question

Example Question:

What values of x solve the equation: $x^2 - 5x = -6$? Solve analytically and then create a plot supporting your answer.

This correct answer would be worth zero po

$x = 2$ and $x = 3$

This answer would be worth full points :

Solution to Example Question:

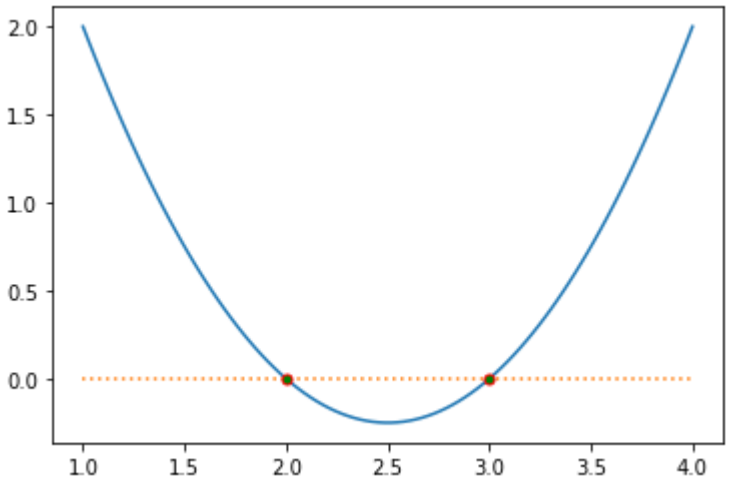
$x^2 - 5x = -6$ can be factored after -6 is added to both sides.

$(x - 2)(x - 3) = 0$. The zero product property reveals that $\boxed{x = 2}$ or $\boxed{x = 3}$.

This equation has two solutions as its graph crosses the x-axis twice as seen below.

In [1]:
```
# AND HERE IS MY COMMENTED CODE TO GO WITH THE EXAMPLE QUESTION #1:

import numpy
import matplotlib.pylab as draw
%matplotlib inline
# matplotlib.pylab is used for plotting. 'inline' ensures the graph opens in Jupyter.

x = numpy.linspace(1,4,100)
# 'linspace' returns evenly spaced numbers over a specified interval
y = x**2-5*x+6
z = 0*x
draw.plot(x,y)
# This is the parabola
draw.plot(x,z, ':')
# This is the x-axis.
draw.plot(2,0, marker="o", markersize=5, markeredgecolor="red", markerfacecolor="greer
draw.plot(3,0, marker="o", markersize=5, markeredgecolor="red", markerfacecolor="greer
# These last two 'draw' commands show the intersection of the parabola and the x-axis.
# The intersections are the sought after solutions. See graph below.
```

Out[1]:
```
[<matplotlib.lines.Line2D at 0x7faafe56b910>]
```



Begin Exam Here

Problem 1

A sample survey of incoming college students (fresh out of high school) has been conducted on campus. Among the massive amounts of data collected we are going to look at just a few items and answer some questions concerning the sample.

The data we want to work with is in the file, "frosh.csv".

Description of the data set This data was collected from a number of incoming freshman attending CU. The data was collected on move-in day as interviewers wandered through the crowds of people and spoke with new students that were willing to fill out a questionairre. The questionairre was administered and completed on a Saturday. The four pieces of data found in frosh.csv here are:

'height' - This is the height of the student in inches. This data is intended to be rational numbers measured to the tenths place. When cleaning this data you might choose to consider the documentation for the Pandas .round function https://pandas.pydata.org/docs/reference/api/pandas.DataFrame.round.html

'age' - This is the age, in years, of the student in question; A18 refers to 'Age 18'. Once cleaned, this data should be integers.

'gender' - This is the gender (sex) of the student. This data should be strings or characters of either 'M' or 'F'.

'lunch_money' - This is the amount of money the student spent for dinner on the previous Friday night. Once cleaned, this data should be rational numbers measured to the hundredths place, thereby representing dollars and cents.

'ID' - This is a 3 or 4 digit number used to identify each student. The datatype should be integer.

Set up your Jupyter notebook (here) with any required imports or loads, and take glance at the actual data.

(2 points) List all your imports below:

In [2]:
```
# SOLUTION
# List the imports that you use here:
import pandas as pd
import numpy as np
# ignore chained assignment warning
pd.options.mode.chained_assignment = None
```

```
import matplotlib.pylab as plt
%matplotlib inline
# binomial coefficient
from scipy.special import binom
# ignore future warnings
import warnings
warnings.simplefilter(action='ignore', category=FutureWarning)
```

(2 points) Load the data to analyze below:

In [3]:
```
# SOLUTION
# To answer questions about the data set, load the data into a dataframe 'dfFrosh'.
# Load your data here:
dfFrosh = pd.read_csv("frosh.csv")
```

(2 points) Look at the data and the datatypes below:

In [4]:
```
# SOLUTION
# Answering questions about data neccesitates looking at the data to understand it.
# Look at your data and the datatypes here:
dfFrosh # print first and last 5 entries
```

Out[4]:

	ID	height	age	gender	lunch_money
0	101	69.377867	A19	F	$15.69
1	102	73.079783	A18	M	$5.23
2	103	66.627820	A19	M	$10.46
3	104	68.466419	A17	F	$20.92
4	105	70.747216	A18	M	$15.69
...	...	...	...	...	...
995	1096	66.570631	A18	F	$20.92
996	1097	65.366322	A19	F	$47.07
997	1098	69.635453	A18	F	$20.92
998	1099	62.765553	A17	F	$36.61
999	1100	72.708883	A18	F	$10.46

1000 rows × 5 columns

Part A

(2 points) How many rows are in the data set?

In [5]:
```
# solution to A here:
rows = len(dfFrosh.index) # len() function
print("There are",rows,"rows in the data set.")
```

```
There are 1000 rows in the data set.
```

Part B

(8 points) Clean the data to match the description of the data above, and take a look at the data again to see/show that it is in fact 'clean'.

In [6]:
```
# SOLUTION:  Cleaning the data
# round height
dfFrosh = pd.read_csv("frosh.csv")
dfFrosh['height']=dfFrosh['height'].round(decimals=1)
# remove 'A' string and convert to int
dfFrosh['age'] = dfFrosh['age'].str.replace('A','')
dfFrosh['age'] = dfFrosh['age'].astype(int)
# remove '$' sign and convert to float
dfFrosh['lunch_money'] = dfFrosh['lunch_money'].str.replace('$','')
dfFrosh['lunch_money'] = dfFrosh['lunch_money'].astype(float)
```

In [7]:
```
# SOLUTION
# Checking to see if the data looks 'clean'
# Look again at the data and the datatypes.
dfFrosh # print first and last 5 entries
```

Out[7]:

	ID	height	age	gender	lunch_money
0	101	69.4	19	F	15.69
1	102	73.1	18	M	5.23
2	103	66.6	19	M	10.46
3	104	68.5	17	F	20.92
4	105	70.7	18	M	15.69
...	...	...	...	...	...
995	1096	66.6	18	F	20.92
996	1097	65.4	19	F	47.07
997	1098	69.6	18	F	20.92
998	1099	62.8	17	F	36.61
999	1100	72.7	18	F	10.46

1000 rows × 5 columns

Part C

(3 points) Delete any rows with missing data (NaN). How many rows were deleted?

In [8]:
```
# solution to Part C
dfFrosh = dfFrosh.dropna() # drop NA rows
row_count = len(dfFrosh.index) # new row count
print("There are now",row_count,"rows in the data set.")
print("Therefore 3 rows were deleted.") # 1000-997 = 3
```

```
There are now 997 rows in the data set.
Therefore 3 rows were deleted.
```

Part D

(3 points) How many 18-year olds are now in this data set?

In [9]:
```
# Solution to Part D here:
eightyo_count = dfFrosh[dfFrosh['age']==18] # array of 18 year olds
print("There are",len(eightyo_count),"18-year olds in the dataset.") # len(array)
```

```
There are 474 18-year olds in the dataset.
```

Part E

(3 points) What is the mean height of these 18-year olds?

In [10]:
```
# Solution to Part E here:
height_mean = dfFrosh.loc[dfFrosh['age'] ==18,'height'].mean() # find mean of only age
print("The mean height of these 18-year olds is",np.round(height_mean,decimals=1),"inc
```

```
The mean height of these 18-year olds is 68.9 inches.
```

Part F

(4 points) On average, who spent more for Friday dinner, 18-year old males or 19-year old females?

In [11]:
```
# Solution to Part F here:
money_18 = dfFrosh.loc[(dfFrosh['gender']=='M') & (dfFrosh['age'] == 18),'lunch_money'
money_19 = dfFrosh.loc[(dfFrosh['gender']=='F') & (dfFrosh['age'] == 19),'lunch_money'
# printing
print("18yo males spent $",np.round(money_18,decimals=2),"while 19yo females spent $",
print("Therefore, 18-year old males spent more for Friday dinner.")
```

```
18yo males spent $ 16.8 while 19yo females spent $ 16.18
Therefore, 18-year old males spent more for Friday dinner.
```

Part G

(5 points)

Create a column called 'height_f' that gives the respondents' height in feet.

Then give a 5-number summary for 'height_f'.

Do **NOT use the .describe() method** when creating your 5-number summary. You **can** use any other built-in methods.

In [12]:
```
# Solution to Part G here:
# create new column with conversion inches to feet
```

```
height_f = np.round(dfFrosh['height']*0.08333333,decimals=2)
dfFrosh['height_f'] = height_f
# adapted from nb02 exercise 2 // hw2
minval = dfFrosh["height_f"].min()
maxval = dfFrosh["height_f"].max()
Q1 = dfFrosh["height_f"].quantile(.25)
Q2 = dfFrosh["height_f"].quantile(.50)
Q3 = dfFrosh["height_f"].quantile(.75)
print("5-Number Summary: {:.2f}    {:.2f}    {:.2f}    {:.2f}    {:.2f}".format(minval
print("(minimum, first, second, and third quantiles, maximum)")
#dfFrosh["height_f"].mode()
```

```
5-Number Summary: 4.97    5.59    5.75    5.92    6.59
(minimum, first, second, and third quantiles, maximum)
```

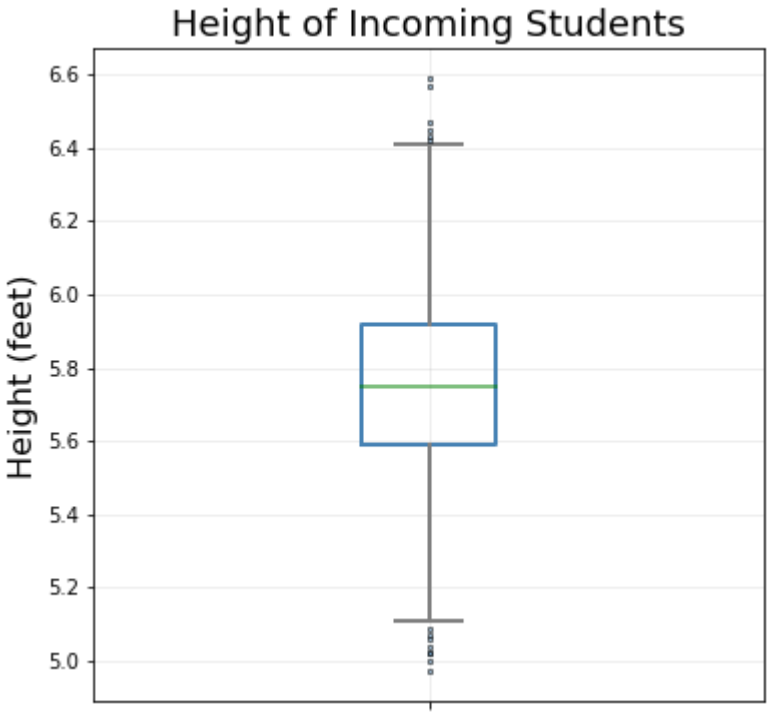
Part H

(8 points) Create a box-and-whisker plot of 'height_f' to provide a visual of the 5-number summary. Label your axes on the plot.

In [13]:

```
# Solution to Part H
# adapted from nb03 // hw2
# Initialize figure
fig, ax = plt.subplots(figsize=(6,6))

# Plot histogram, but this time return dictionary of style parameters for modificatior
bp = dfFrosh.boxplot(column="height_f", ax=ax, widths=[.2], return_type='dict');

# ----------------------------------------
# Set properties of various parts of plot
# ----------------------------------------

# Change properties of boxes
for box in bp['boxes']:
    box.set(color='steelblue', linewidth=2)

# Change properties of whiskers
for whisker in bp['whiskers']:
    whisker.set(color='gray', linewidth=2)

# Change properties of caps
for cap in bp['caps']:
    cap.set(color='gray', linewidth=2)

# Change properties of median
for cap in bp['medians']:
    cap.set(color='green', linewidth=2, alpha=0.5)

# Change properties of fliers (outliers)
for flier in bp['fliers']:
    flier.set(markerfacecolor='steelblue', linewidth=2, marker='s', markersize=2, alph

# Set title and vertical axis label
ax.set_title('Height of Incoming Students', fontsize=18)
ax.set_ylabel("Height (feet)", fontsize=16)

# Set names of plots
plt.xticks([1],[""], rotation=0, fontsize=16)
```

```
# Get rid of automatically generated titles and xlables
plt.suptitle("")
ax.set_xlabel("")

# Make grid-lines lighter
ax.grid(alpha=0.25)
```



Part I

(8 points)

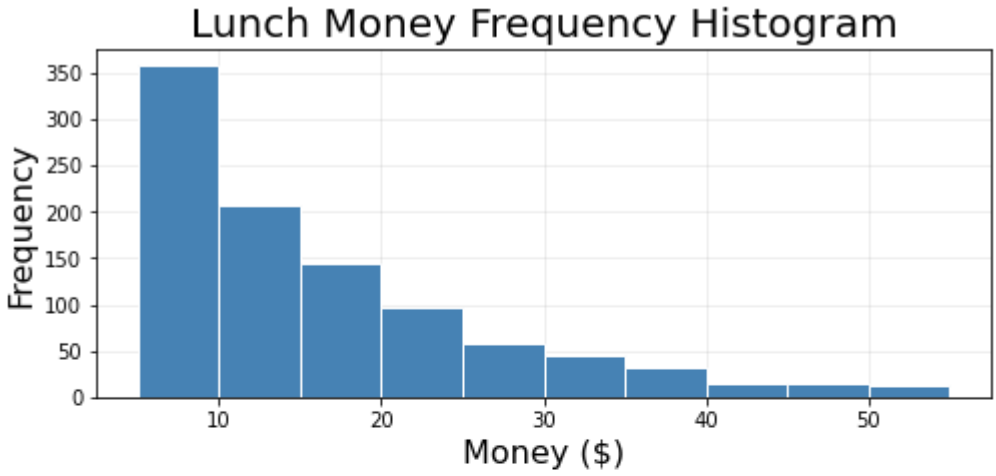
i). Create a **frequency** histogram of 'lunch_money'. Label your axes.

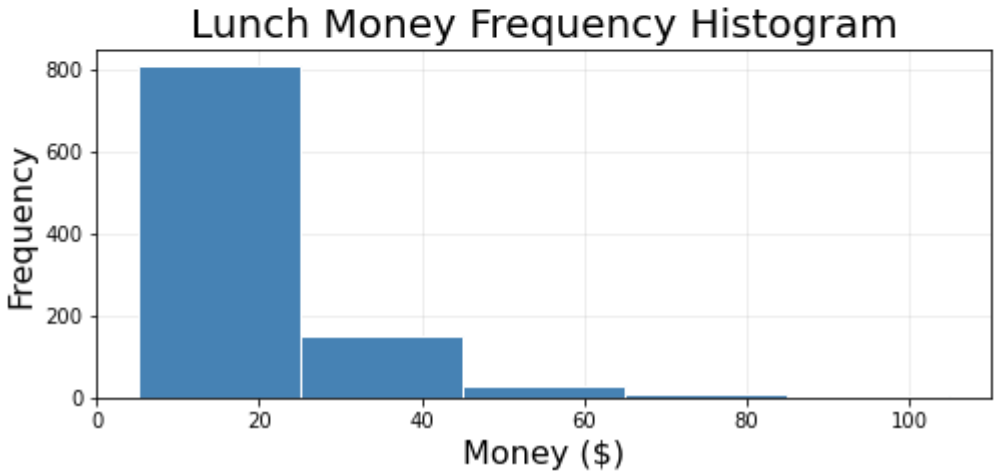
ii). Then ***give a shape descriptive name for this type of distribution.***

In [14]:
```
# Solution to Part I

# i)
# adapted from nb03 exercise 2
# Initialize figure subplots
fig, axes = plt.subplots(nrows=2, ncols=1, figsize=(8,8))
my_bins = range(5,60,5)
# nrows=2, ncols=1 means the histograms will be stacked in 2 rows with 1 column

# ---------------------------------------
# TOP
# ---------------------------------------
dfFrosh.hist(column="lunch_money", ax=axes[0],  bins=my_bins, facecolor="steelblue", e
# ax=axes[0] implies the first of 2 histograms which are indexed as 0 and 1.

# Add titles and labels
axes[0].set_title("Lunch Money Frequency Histogram", fontsize=20)
axes[0].set_xlabel("Money ($)", fontsize=16)
axes[0].set_ylabel("Frequency", fontsize=16)
```

```
# Make grid lighter and set behind data
axes[0].grid(alpha=0.25)
axes[0].set_axisbelow(True)

# ---------------------------------------
# BOTTOM
# ---------------------------------------
my_bins = range(5,110,20)
dfFrosh.hist(column="lunch_money", ax=axes[1], bins=my_bins, facecolor="steelblue", ec
# ax=axes[1] implies the second of 2 histograms which are indexed as 0 and 1.

# Add titles and labels
axes[1].set_title("Lunch Money Frequency Histogram", fontsize=20)
axes[1].set_xlabel("Money ($)", fontsize=16)
axes[1].set_ylabel("Frequency", fontsize=16)

# Make grid lighter and set behind data
axes[1].grid(alpha=0.25)
axes[1].set_axisbelow(True)
# Adjust vertical space so titles/axis labels don't overlap
fig.subplots_adjust(hspace=.5)

# ii)
#print(dfFrosh["lunch_money"].max())
print("Given the range of data ($5.23, $104.6), there are much more points to the left
print("Therefore, the distribution is right skewed")
```

```
Given the range of data ($5.23, $104.6), there are much more points to the left.
Therefore, the distribution is right skewed
```





Part J

After viewing the 'lunch_money' distribution we might conclude that around 80% of students eat a cheap ("C") meal, 18% eat a middle priced ("M") meal, and 2% eat an expensive ("E") meal.

Imagine choosing a student from this list at random to see if they ate a middle priced, "M", meal. You should expect this to be true about 18% of the time.

Suppose you choose a student from the list, record whether or not they at a middle priced, "M", meal or not, and then you randomly chose again, and again, and again, etc.

(10 points)

i). Create a simulation that will randomly choose "C", "M", and "E" according to the probabilites "C" = 80%, "M"=18%, and "E"=2%.
(Note that this simulation will NOT randomly choose from the actual data stored in frosh.csv).

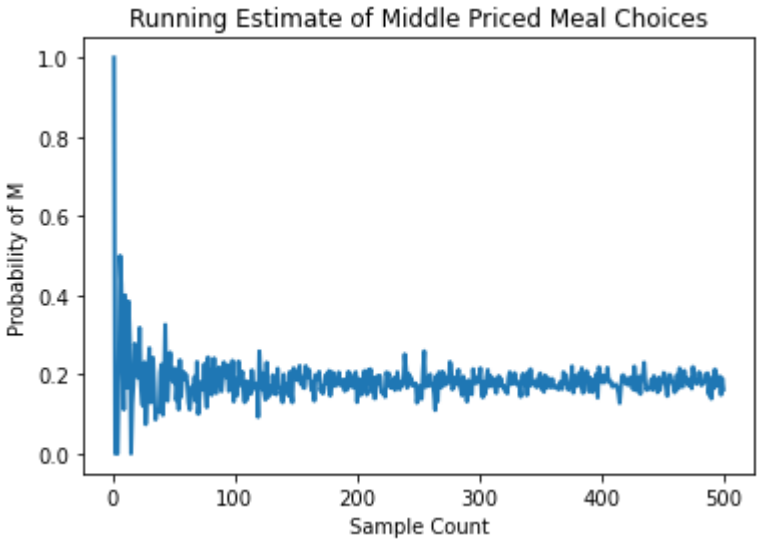
ii). Make a plot of the running estimate of the probability of a student eating a 'middle priced', "M", meal as the number of choices increases:

$\frac{m_1}{1}, \frac{m_2}{2}, \frac{m_3}{3}, \ldots, \frac{m_{500}}{500}$, where m_i is the number of M's discovered in your random choices out of i trials.

Run the simulation for 500 choices, and graph the 500 ratios.

iii). Then comment on if/when the graph flattens out to the expected 18% within these 500 ratios.

In [15]:
```
#Solution to J
# Uncomment the random.seed function below, but do not change the statement or its inp
# Put your code for Part J after the random.seed function


np.random.seed(54321)

# part i)
# adapted from nb05
# np array of meal choices and probability
meal_choices = {'choices' : np.array(['C', 'M', 'E']), 'probs' : np.array([.8, .18, 0.

def sample(choices):
    # randomly choose a meal
    meal = np.random.choice(meal_choices['choices'], p = meal_choices['probs'])
    return meal

def prob(choiceType, meal_choices, num_samples):
    # get a bunch of meal choices
    choice = np.array([sample(meal_choices) for ii in range(num_samples)])
    # compute fraction of wanted meal choice
    return np.sum(choice == choiceType) / num_samples

# part ii)
# run simulation for 500 choices
x = np.linspace(1,500,500)
y = np.zeros(500) # preallocate ratios
for i in range(500): # num trials
    y[i] = prob('M', meal_choices, i+1);

# plot
fig, ax = plt.subplots()
ax.plot(x, y, linewidth=2.0)
# labeling
ax.set_title('Running Estimate of Middle Priced Meal Choices')
ax.set_xlabel('Sample Count')
ax.set_ylabel('Probability of M')
plt.show()
# explanation of graph
print("The graph flattens out past 50-100 samples.")
print("At this point, the probability is less than 0.2.")
print("Thus, the graph hits the expected 18%.")
```

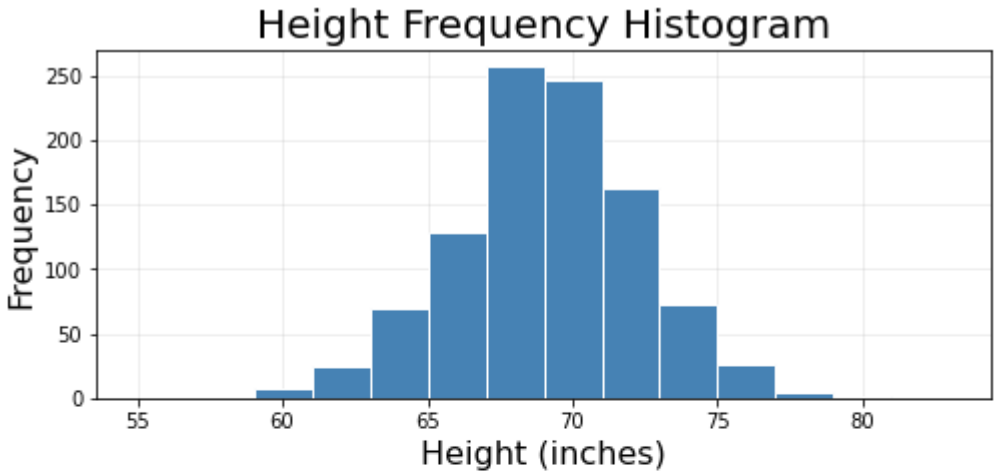


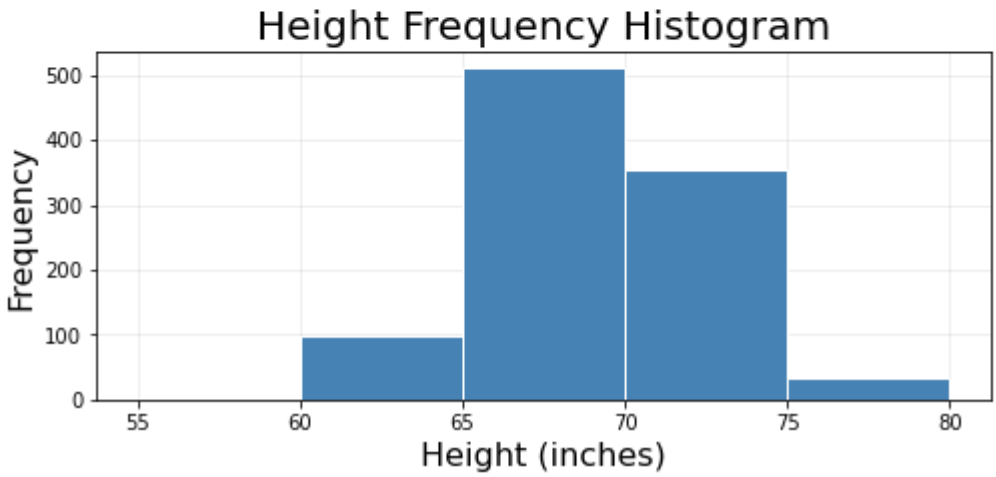
The graph flattens out past 50-100 samples.
At this point, the probability is less than 0.2.
Thus, the graph hits the expected 18%.

Part K

(6 points) Create a **frequency** histogram of 'height'. Label your axes.

In [16]:
```
# Solution to Part K here:
# adapted from nb03 exercise 2
# Initialize figure subplots
fig, axes = plt.subplots(nrows=2, ncols=1, figsize=(8,8))
my_bins = range(55,85,2)

# TOP
dfFrosh.hist(column="height", ax=axes[0],  bins=my_bins, facecolor="steelblue", edgecc

# Add titles and labels
axes[0].set_title("Height Frequency Histogram", fontsize=20)
axes[0].set_xlabel("Height (inches)", fontsize=16)
axes[0].set_ylabel("Frequency", fontsize=16)

# Make grid lighter and set behind data
axes[0].grid(alpha=0.25)
axes[0].set_axisbelow(True)

# BOTTOM
my_bins = range(55,85,5)
dfFrosh.hist(column="height", ax=axes[1], bins=my_bins, facecolor="steelblue", edgecol

# Add titles and labels
axes[1].set_title("Height Frequency Histogram", fontsize=20)
axes[1].set_xlabel("Height (inches)", fontsize=16)
axes[1].set_ylabel("Frequency", fontsize=16)

# Make grid lighter and set behind data
axes[1].grid(alpha=0.25)
axes[1].set_axisbelow(True)

fig.subplots_adjust(hspace=.5)
```





Part L

The height data roughly follows a normal distribution with a mean, μ, and standard deviation, σ, that you can calculate from the data.

(8 points)

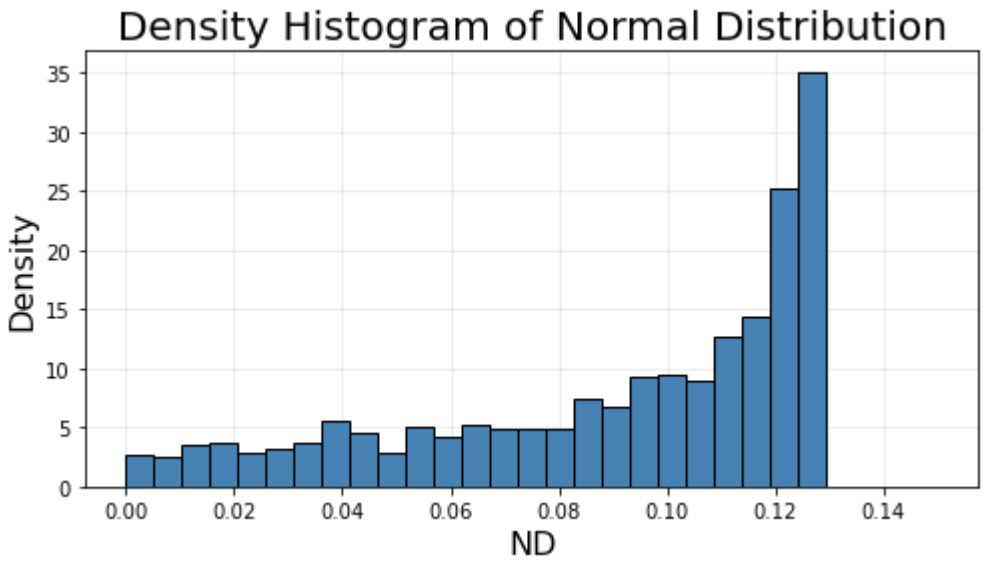
i). Write some code that will create a $N(0,1)$ distribution from this same data. i.e., take the $N(\mu,\sigma^2)$ data currently stored in 'height' and transform it into data from a $N(0,1)$ distribution and store it in another column called 'height_N01'.

ii). Create a **density** histogram of 'height_N01'. Label your axes.

In [17]:
```
# Solution to Part L here:
# i)
def PDF(x,mu,sigma): # normal distribution formula
    P = 1/(sigma*np.sqrt(2*np.pi))*np.exp((-1/2)*((x-mu)/sigma)**2)
    return P
x = dfFrosh['height'] # array of heights
mu = np.mean(x) # calc mean
sigma = np.std(x) # cald standard deviation
P = PDF(x,mu,sigma) # call func
dfFrosh['height_N01'] = P[:] # create new dataframe column
```

```
print("mu =",mu,"and sigma^2 =",sigma**2)

# ii)
# Adapted from nb03
# Initialize figure and ranges for bins
fig, ax = plt.subplots(figsize=(8,4))
my_bins = np.linspace(0,.15,30)
# Plot histogram with custom colors
mycolor = 'steelblue'
dfFrosh.hist(column="height_N01", density=True, ax=ax, bins=my_bins, facecolor=mycolor

# Add a title
ax.set_title("Density Histogram of Normal Distribution", fontsize=20)

# Add axis labels
ax.set_xlabel("ND", fontsize=16)
ax.set_ylabel("Density", fontsize=16)

# Make the grid lines lighter and put them behind data
ax.grid(alpha=0.25)
ax.set_axisbelow(True)
```

```
mu = 69.03691073219655 and sigma^2 = 9.78900871118873
```



Part M

(8 points)

i). What percentage of students in the data are between 5 foot 6 inches and 6 feet tall, i.e. between 5'6''and 6'?

ii). Assuming height is normally distributed with mean and standard deviation calculated from the data, calculate the probability of being between 5 foot 6 inches and 6 feet tall using a normal distribution. Then comment on how this answer compares to your answer in part (i).

In [18]:
```
# Solution for Part M here:
# i)
# find height range
height_count = dfFrosh.loc[(dfFrosh['height'] > 66) & (dfFrosh['height'] < 72),'height
```

```
# count rows
row_count = len(dfFrosh.index)
# calculate percentage
percentage = height_count/row_count*100
# print
print("The percentage of students between 5'6'' and 6' is",np.round(percentage,decimal
```

```
The percentage of students between 5'6'' and 6' is 67.3 %
```

Problem 2

You are at a picnic when you come across a cooler topped with ice so you can't see what drink you might randomly grab from the inside. The cooler happens to be filled with 12 clam juices and 12 cokes.

A **B**

You decide you would like a delicious, refreshing drink on this hot picnic day.

In fact, you decide you would like two drinks.

So, you reach into the cooler and randomly grab the first of two beverages.

The bottles feel remarkably similar, so there is no way to tell the difference until you have the drink in hand.

- If you grab a Coke first, then you will drop that disgusting drink back into the icy water of the cooler and add 2 more warm clam juices to the cooler in order to increase the ratio for the next grab.

- If, however, your first grab is a clam juice, then you will keep it and reach in for a second beverage.

Part A

(10 points) Write a simple simulation with 10,000 trials that estimates the probability that your ***first and second grabs are both Cokes***.

In [19]:
```
# Write your code for Part A here:

x = 12; # initial clam count
y = 12; # initial coke count

def choose_one():
    global count; # global count variable
    global x; # global variable tracking number of clams
    global y; # global variable tracking number of cokes
    global drink_choices; # global variable for np arry of choices
    count = x + y; # calculate total count
    drink_choices = {'choices' : np.array(['clam','coke']), 'probs' : np.array([x/cour
    # randomly choose a drink
    drink = np.random.choice(drink_choices['choices'], p = drink_choices['probs']);
    if drink=='coke': # add 2 clams if coke
        x = x + 2;
        count = count + 2;
    else: # take clam if not coke
        x = x - 1
        count = count - 1;
    return drink

def prob(drink_choices, num_samples):
    # simulate 2 choices a bunch of times
    choice1 = np.array([choose_one() for ii in range(num_samples)]);
    choice2 = np.array([choose_one() for ii in range(num_samples)]);
    # compute fraction of choices both being coke
    return np.sum((choice1 == 'coke') & (choice2 == 'coke')) / num_samples;

# initial np array of drink choices and probability
initial_array = {'choices' : np.array(['clam','coke']), 'probs' : np.array([1/2,1/2])}
fraction = prob(initial_array, 10000);
print("There are",x,"clams,",y,"cokes, and",count,"total drinks after simulation.");
print("The probability of the first and second grabs being Coke is",fraction);
```

```
There are 28 clams, 12 cokes, and 40 total drinks after simulation.
The probability of the first and second grabs being Coke is 0.1086
```

Part B

Your friend Leroy Jenkins is at this same picnic. Leroy tends to run off from the group yelling his name every now and again. The issue is that Leroy can't hold his clam juice. Approximately 75% of the time when Leroy leaves he is actually going to get another clam juice, otherwise he is just off to use the restroom.

You make a bet with your friends that out of the next 10 times that Leroy runs off screaming, he will be going to get another clam juice 7 or fewer of those ten times.

Let X be the random variable that is the number of times Leroy went to get a clam juice when he leaves in his next 10 excursions. So, X can take on the values 0 thru 10. Assume that each trip away from the group is independent from all other trips and the probability of grabbing another clam juice is constant at 75%.

(8 points)

i). Write a function that returns the **Cumulative Distribution Function (CDF)** of X as a Numpy array.

ii). Use the function from part(i) to print out the **Cumulative Distribution Function (CDF)** of X as a table (i.e. input values for X given in one column and decimal outputs of the CDF given in the other column).

iii). What is the probability that you win the bet? Explain how your answer relates to parts(i and/or ii).

In [20]:
```
# Solution to Part B
# i) function for CDF
def CDF():
    p = np.zeros(11) # preallocate PDF
    P = np.zeros(11) # prellocate CDF
    for i in range(0,11): # calc PDF
        p[i] = binom(10,i)*(3/4)**(i)*(1/4)**(10-i)
    P[0] = (1/4)**10 # CDF initial
    for i in range(1,11): # calc CDF
        P[i] = P[i-1]+p[i]
    return P # return CDF array
# ii) print out
array = CDF(); # call func
print("CDF TABLE:\n")
print("X CDF")
for i in range(0,11):
    print(i,np.round(array[i],decimals=6))
# iii) probability of winning
print()
bet_prob = np.round(array[7],decimals=3)
print("The bet is P(X <= 7)=",bet_prob)
print("Therefore, the probability that the bet is won is:",bet_prob*100,"%")
```

```
CDF TABLE:

X CDF
0 1e-06
1 3e-05
2 0.000416
3 0.003506
4 0.019728
5 0.078127
6 0.224125
7 0.474407
8 0.755975
9 0.943686
10 1.0

The bet is P(X <= 7)= 0.474
Therefore, the probability that the bet is won is: 47.4 %
```

In [ ]: