WhiteHat Jr
Live online Coding for Kids

Topic	**Properties of Normal Distribution**
Class Description	**Students learn about the interesting properties which normal distribution displays.**
Class	**C109**
Class time	**45 mins**
Goal	● Find mean of population data from the normal distribution ● Draw inference on the number of data points in the population between one, two and three standard deviations
Resources Required	● Teacher Resources ○ Visual Code Studio ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen ● Student Resources ○ Visual Code Studio ○ Laptop with internet connectivity ○ Earphones with mic ○ Notebook and pen

Class structure	**Warm Up** **Teacher-led Activity** **Student-led Activity** **Wrap up**	**5 mins** **15 min** **15 min** **5 min**

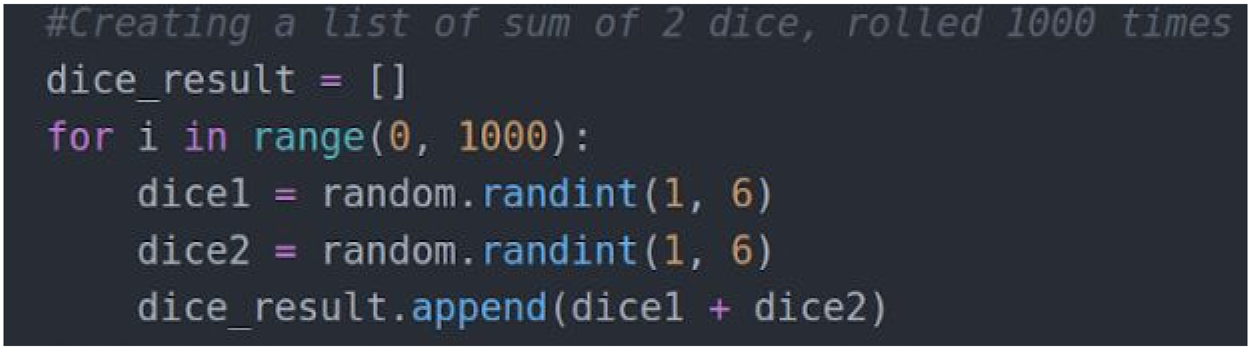
CONTEXT

- **Review normal distribution**

Class Steps	**Teacher Action**	**Student Action**
Step 1: **Warm Up** **(5 mins)**	Hi <Student Name>, Welcome back to the class. We have been learning about how to work with data sets!	ESR: We plotted different kinds of data sets and we observed that most data sets in nature

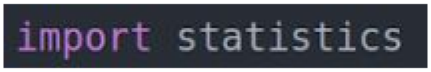
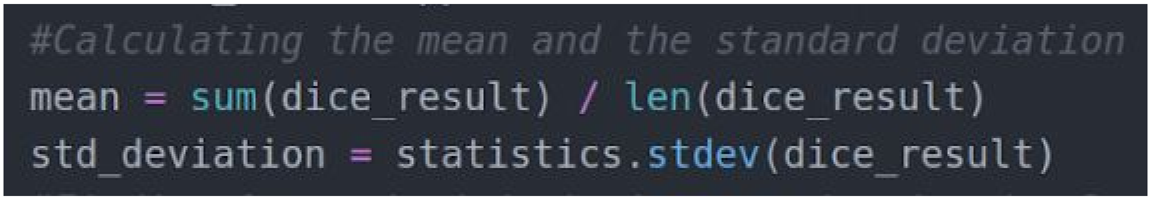
	Can you recall what we did in the last class?	follow a normal distribution - which looks like a bell curve.
	Isn't it amazing that most of the data sets follow such beautiful pattern? What does it make you think about nature and datasets? Allow the student to express their wonder about data sets. Guide them to think how data can be predictable.	ESR: Yes varied
	Today, we are going to learn a little more about the important properties of normal distribution. These properties will be of immense help later when we try to predict data behavior	-
Teacher Initiates Screen Share		
CHALLENGE ● **Programmatically infer the properties of normal distribution**		
Step 2: Teacher-led Activity (15 min)	Let's quickly get the dice data again on which we had worked on in the last class. Do you remember, how we had got the data.	ESR: We had generated two random numbers between 1 to 6 and added them. We repeated this 100 times
	Let's create this dataset again. But this time, let's do it for 1000 times. Can you help me write the code for this? Teacher opens and creates a	Student helps the teacher write code for this.

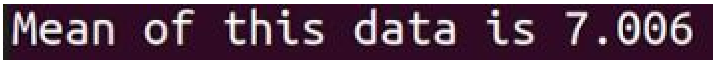
	diceData.py file. 1. Teacher writes code to run a loop 1000 times. 2. In each loop, generate two random numbers between 1 to 6. 3. Add the two numbers and store them in a dice_result list.	

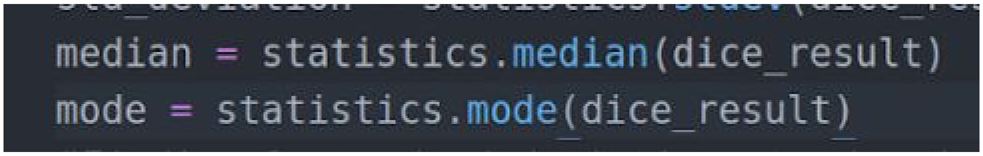


```
#Creating a list of sum of 2 dice, rolled 1000 times
dice_result = []
for i in range(0, 1000):
    dice1 = random.randint(1, 6)
    dice2 = random.randint(1, 6)
    dice_result.append(dice1 + dice2)
```

	Now, let's calculate the mean for this dataset. Can you tell me how to calculate the mean?	ESR: We can add all the numbers in the dice_result list and divide by 1000
	Yes, Mean = Sum of all the data points / number of data points. Let's calculate the mean here. We can use the functions available in statistics package to help us do so. Help in doing this. Teacher writes code to: 1. Import statistics package 2. Use sum() to calculate the sum of all the data points in the dice_result list 3. Use len() to find the number of data points in dice_result 4. Calculate mean	Student helps the teacher write code for calculating mean

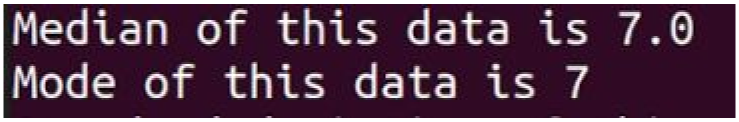

`import statistics`

`#Calculating the mean and the standard deviation`
`mean = sum(dice_result) / len(dice_result)`
`std_deviation = statistics.stdev(dice_result)`		

	Let's run the program to find the mean. Teacher runs the code. What's the mean we got?	ESR: <Mean> observed by the student (Close to 7)
 `Mean of this data is 7.006`		
	Now, let's write the code to calculate the median and mode as well. Do you remember what median and mode are?	ESR: Median is the "middle" value in the list of data. If we list the data in ascending order, the number in the middle (or average of numbers in the middle) is the median. Mode is the number in the data which occurs the most number of times.
	Awesome. Let's use the functions available in statistics package in python to calculate median and mode for the dice data. Teacher writes the code to calculate the median and mode for the data. Teacher uses the statistics reference.	Student helps the teacher in writing the code.



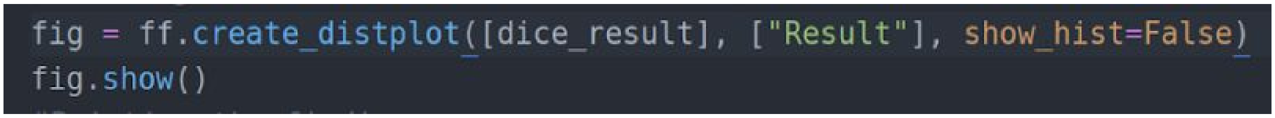
```
median = statistics.median(dice_result)
mode = statistics.mode(dice_result)
```

	Teacher runs the code. What is the median and mode you observe.	ESR: Median and mode are 7.

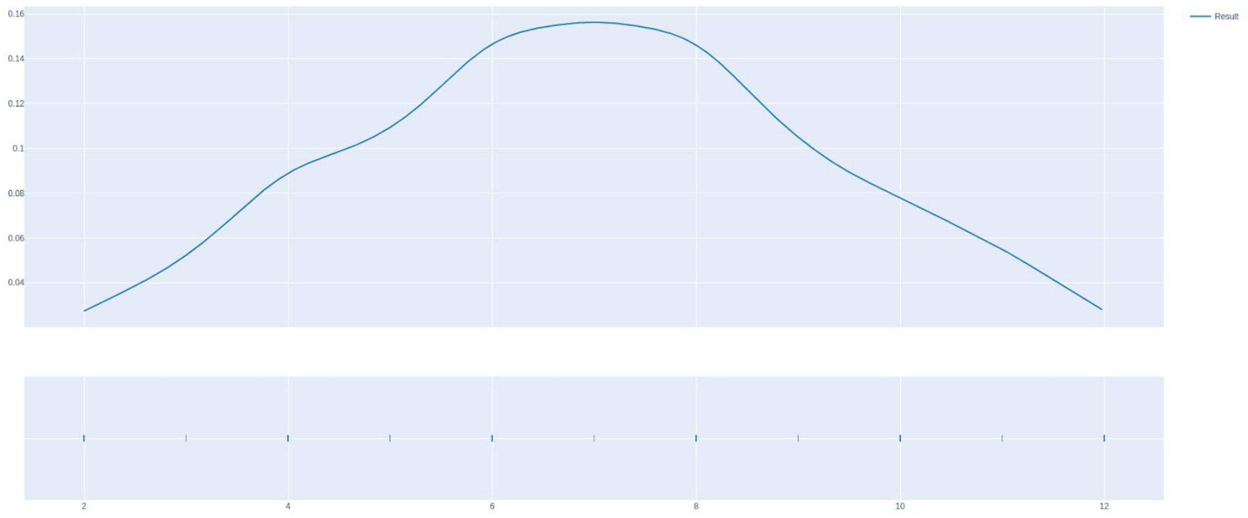


```
Median of this data is 7.0
Mode of this data is 7
```

	In a dataset which follows a normal distribution, the mean, median and mode are all equal! If we plot the normal distribution on a graph, can you imagine where 7 would be.	Allow the student to think about it.
	Let's plot the normal distribution graph for the dice_result and check. Teacher imports the plotly figure factory to plot the distribution. What do you observe? Where is 7?	ESR: 7 is at the maximum point or peak in the normal distribution

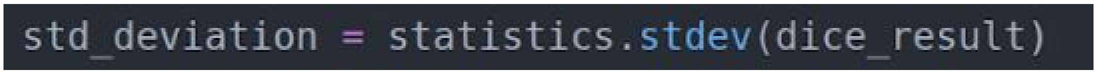
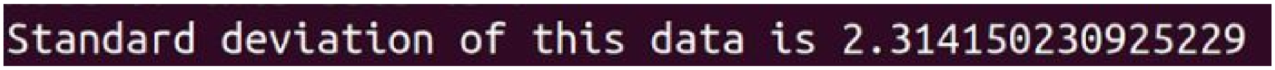


```
fig = ff.create_distplot([dice_result], ["Result"], show_hist=False)
fig.show()
```



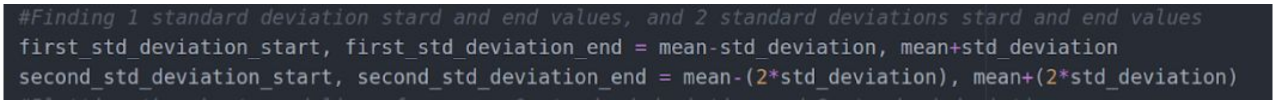
	Yes! Mean, median and mode are all equal in a normal distribution and they are at the peak! This is a property common to all the normal distributions!	Allow the student to think about it.
	Another important thing. So far, in all our distribution plots, we have been thinking y-axis as the number of times a particular data has occurred. Data Scientists would like to think y-axis as the chance (probability) a particular data set has to appear in the data. Higher the number of times a particular data occurs -> higher is its chance of occuring Probability or chance of a particular	ESR: 7! Since it is at the peak of the distribution.

	number occuring in the data wouldn't change if you roll the dice 100 times or 1000 times or 10000 times. Can you look at the distribution and tell which number would have the highest chance of appearing when you roll two dices together and add the numbers?	
	Perfect! You now know which numbers to bet on if you are playing a game with dice. If you observe, 7 (line perpendicular to 7 on x-axis) is almost the line of symmetry. Both sides around the line are similar.	Student observes the symmetrical nature of the normal distribution.
	Normal Distribution has other interesting features too! Let's calculate the standard deviation (sd) for the data set we have. We can look for the population standard deviation function available in the statistics package. Teacher shows the reference link for stdev() to calculate the standard deviation from the data. Teacher writes code to calculate the standard deviation for the data. Teacher runs code to check the output	Student helps the teacher to write code to calculate the standard deviation and observe the output.


`std_deviation = statistics.stdev(dice_result)`

`Standard deviation of this data is 2.314150230925229`		

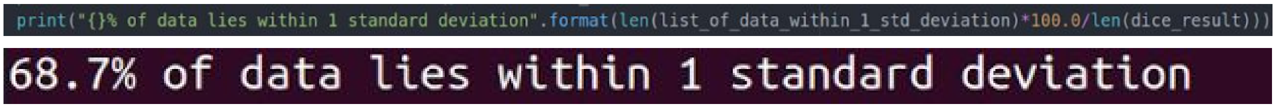
	Now, let's say we want to find what percentage of data lie between one standard deviation from the mean, i.e; what percentage of data lie between mean - sd and mean + sd ; how would you do that?	ESR: We would loop over each data in the dice result, and for each data we will check if the value is between mean - sd and mean + sd. If the data is between the two values, we can increment a count variable. Once, we have the count, we can calculate the percentage of data between these two values.
	Can you write a program for that on your own? You will be surprised to see the results.	
Teacher Stops Screen Share		
	Now it's your turn. Please share your screen with me.	
● **Ask Student to press ESC key to come back to panel** ● **Guide Student to start Screen Share** ● **Teacher gets into Fullscreen**		
ACTIVITY ● **Student writes code to find the number of data points in a normal distribution which lie between one,two and three standard deviations.** ● **Student plots distribution of data set which does not follow a normal pattern.**		

Step 3: Student-Led Activity (15 min)	Guide the student to write the program which finds the count of data points between mean - sd and mean + sd	Student writes the program which loops over all the data set and counts the number of data points which are between mean - sd and mean + sd



```
#Finding 1 standard deviation stard and end values, and 2 standard deviations stard and end values
first_std_deviation_start, first_std_deviation_end = mean-std_deviation, mean+std_deviation
second_std_deviation_start, second_std_deviation_end = mean-(2*std_deviation), mean+(2*std_deviation)
```

	Let's calculate the % of data which lie between mean - sd and mean + sd and print the percentage. What is the percentage? Expected percentage is around 68%	Student calculates and prints the percentage of data points between mean - sd and mean + sd <Expected percentage is 68>

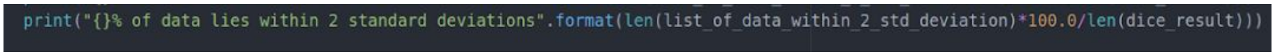


```
print("{}% of data lies within 1 standard deviation".format(len(list_of_data_within_1_std_deviation)*100.0/len(dice_result)))
```

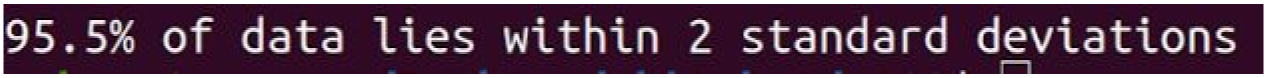
```
68.7% of data lies within 1 standard deviation
```

	In ALL normal distributions, the percentage of data points which lie between mean - sd and mean + sd is always around 68%. Isn't that amazing? This is a very useful information about normal	Student absorbs the information

	distributions which we are going to use in our classes.	
	Now, let's write a program to calculate the % of data which lie between mean - 2sd and mean + 2sd.	Student writes the program to calculate the % of data which lie between mean - 2sd and mean + 2sd

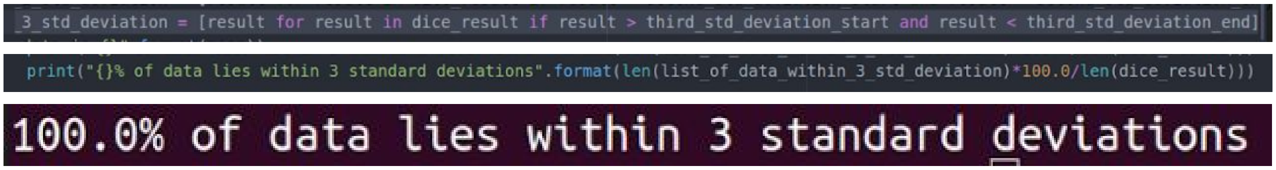


```
print("{}% of data lies within 2 standard deviations".format(len(list_of_data_within_2_std_deviation)*100.0/len(dice_result)))
```



```
95.5% of data lies within 2 standard deviations
```

	What do you get? This is also true for all normal distrsibutions! 95% of data in a normal distribution lie between mean - 2sd and mean + 2sd	ESR: 95%
	Do you want to find the % of data between mean - 3sd and mean + 3sd	Student writes the program to find the % of data between mean - 3sd and mean + 3sd Student observes that 99% of data lie between mean - 3sd and mean + 3sd

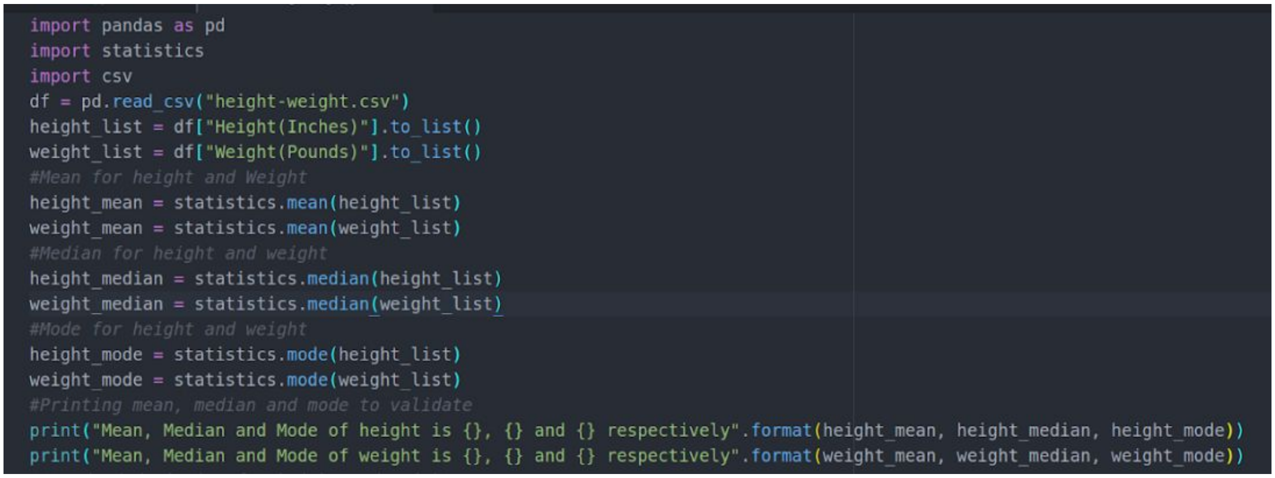


```
3_std_deviation = [result for result in dice_result if result > third_std_deviation_start and result < third_std_deviation_end]
```

```
print("{}% of data lies within 3 standard deviations".format(len(list_of_data_within_3_std_deviation)*100.0/len(dice_result)))
```

```
100.0% of data lies within 3 standard deviations
```

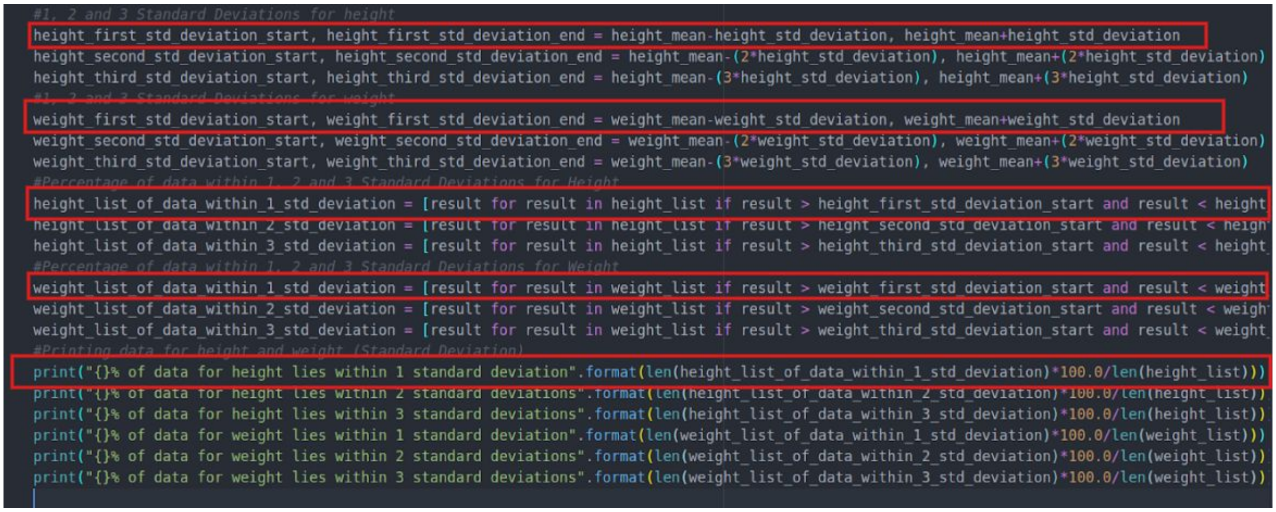
	These findings are true for all normal distributions. Can you summarize what you found? Do you want to verify these findings with another data set?	99% of data lie between mean - 3sd and mean + 3sd 95% of data lie between mean -2sd and mean + 2sd 68% of data lie between mean - sd and mean + sd Yes!
	Let's take the height and weight data you have. Let's verify all the facts we have discovered about normal distributions and check with these data.	ESR: Student downloads the heights and weights data of 18 year olds
	Let's calculate the mean, median and mode for these data and see if they are all nearly equal	Student calculates the mean, median and mode for the heights and weights data and verifies that mean = median = mode in normal distribution



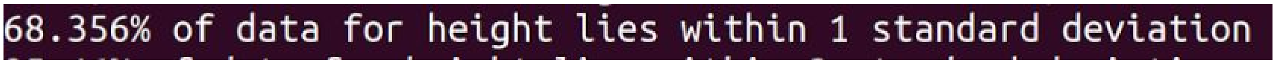
```
import pandas as pd
import statistics
import csv
df = pd.read_csv("height-weight.csv")
height_list = df["Height(Inches)"].to_list()
weight_list = df["Weight(Pounds)"].to_list()
#Mean for height and Weight
height_mean = statistics.mean(height_list)
weight_mean = statistics.mean(weight_list)
#Median for height and weight
height_median = statistics.median(height_list)
weight_median = statistics.median(weight_list)
#Mode for height and weight
height_mode = statistics.mode(height_list)
weight_mode = statistics.mode(weight_list)
#Printing mean, median and mode to validate
print("Mean, Median and Mode of height is {}, {} and {} respectively".format(height_mean, height_median, height_mode))
print("Mean, Median and Mode of weight is {}, {} and {} respectively".format(weight_mean, weight_median, weight_mode))
```

	Let's find out what percentage of data lie between mean - sd and mean + sd	Student writes program to calculate the % of heights and weights which lie between mean - sd and mean + sd and verify it is 68%

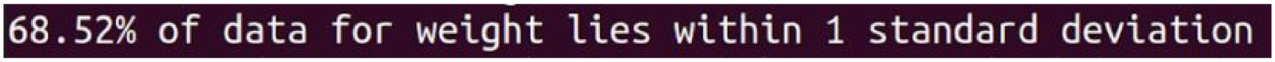
height_list_of_data_within_1_std_deviation = [result for result in height_list if result > height_first_std_deviation_start and result < height_first_std_deviation_end]



```
#1, 2 and 3 Standard Deviations for height
height_first_std_deviation_start, height_first_std_deviation_end = height_mean-height_std_deviation, height_mean+height_std_deviation
height_second_std_deviation_start, height_second_std_deviation_end = height_mean-(2*height_std_deviation), height_mean+(2*height_std_deviation)
height_third_std_deviation_start, height_third_std_deviation_end = height_mean-(3*height_std_deviation), height_mean+(3*height_std_deviation)
#1, 2 and 3 Standard Deviations for weight
weight_first_std_deviation_start, weight_first_std_deviation_end = weight_mean-weight_std_deviation, weight_mean+weight_std_deviation
weight_second_std_deviation_start, weight_second_std_deviation_end = weight_mean-(2*weight_std_deviation), weight_mean+(2*weight_std_deviation)
weight_third_std_deviation_start, weight_third_std_deviation_end = weight_mean-(3*weight_std_deviation), weight_mean+(3*weight_std_deviation)
#Percentage of data within 1, 2 and 3 Standard Deviations for Height
height_list_of_data_within_1_std_deviation = [result for result in height_list if result > height_first_std_deviation_start and result < height
height_list_of_data_within_2_std_deviation = [result for result in height_list if result > height_second_std_deviation_start and result < heigh
height_list_of_data_within_3_std_deviation = [result for result in height_list if result > height_third_std_deviation_start and result < height_
#Percentage of data within 1, 2 and 3 Standard Deviations for Weight
weight_list_of_data_within_1_std_deviation = [result for result in weight_list if result > weight_first_std_deviation_start and result < weight
weight_list_of_data_within_2_std_deviation = [result for result in weight_list if result > weight_second_std_deviation_start and result < weigh
weight_list_of_data_within_3_std_deviation = [result for result in weight_list if result > weight_third_std_deviation_start and result < weight_
#Printing data for height and weight (Standard Deviation)
print("{}% of data for height lies within 1 standard deviation".format(len(height_list_of_data_within_1_std_deviation)*100.0/len(height_list)))
print("{}% of data for height lies within 2 standard deviations".format(len(height_list_of_data_within_2_std_deviation)*100.0/len(height_list))
print("{}% of data for height lies within 3 standard deviations".format(len(height_list_of_data_within_3_std_deviation)*100.0/len(height_list))
print("{}% of data for weight lies within 1 standard deviation".format(len(weight_list_of_data_within_1_std_deviation)*100.0/len(weight_list)))
print("{}% of data for weight lies within 2 standard deviations".format(len(weight_list_of_data_within_2_std_deviation)*100.0/len(weight_list))
print("{}% of data for weight lies within 3 standard deviations".format(len(weight_list_of_data_within_3_std_deviation)*100.0/len(weight_list))
```

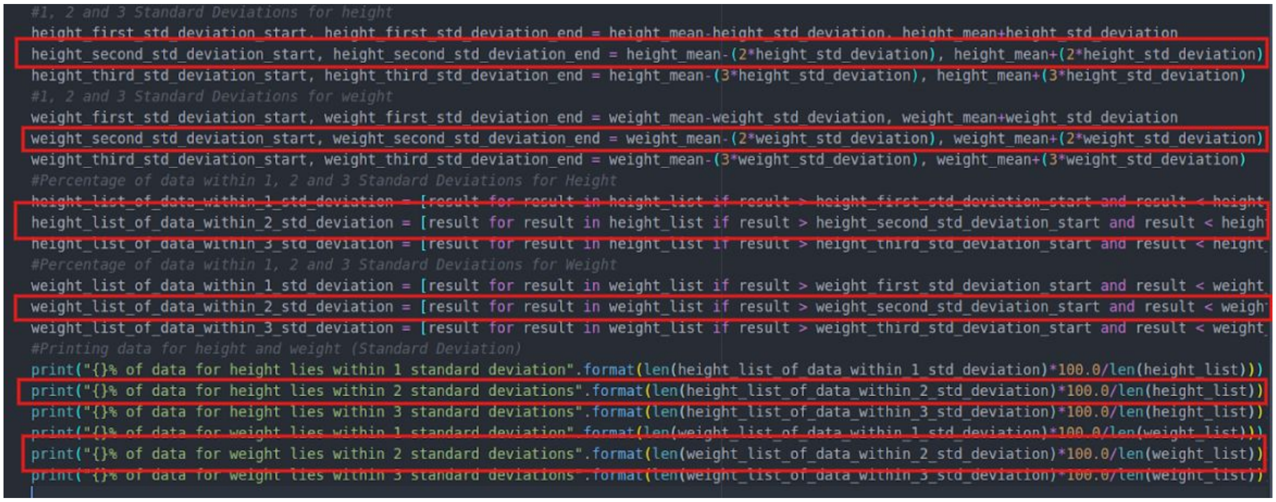


```
68.356% of data for height lies within 1 standard deviation
```

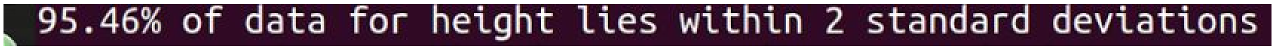


```
68.52% of data for weight lies within 1 standard deviation
```

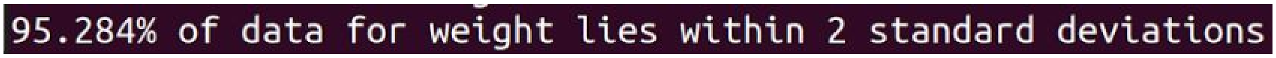
	Let's find out what percentage of data lie between mean - 2sd and mean + 2sd	Student writes program to calculate the % of heights and weights which lie between mean - 2sd and mean + 2sd and verify it is 95%



```python
#1, 2 and 3 Standard Deviations for height
height_first_std_deviation_start, height_first_std_deviation_end = height_mean-height_std_deviation, height_mean+height_std_deviation
height_second_std_deviation_start, height_second_std_deviation_end = height_mean-(2*height_std_deviation), height_mean+(2*height_std_deviation)
height_third_std_deviation_start, height_third_std_deviation_end = height_mean-(3*height_std_deviation), height_mean+(3*height_std_deviation)
#1, 2 and 3 Standard Deviations for weight
weight_first_std_deviation_start, weight_first_std_deviation_end = weight_mean-weight_std_deviation, weight_mean+weight_std_deviation
weight_second_std_deviation_start, weight_second_std_deviation_end = weight_mean-(2*weight_std_deviation), weight_mean+(2*weight_std_deviation)
weight_third_std_deviation_start, weight_third_std_deviation_end = weight_mean-(3*weight_std_deviation), weight_mean+(3*weight_std_deviation)
#Percentage of data within 1, 2 and 3 Standard Deviations for Height
height_list_of_data_within_1_std_deviation = [result for result in height_list if result > height_first_std_deviation_start and result < height
height_list_of_data_within_2_std_deviation = [result for result in height_list if result > height_second_std_deviation_start and result < heigh
height_list_of_data_within_3_std_deviation = [result for result in height_list if result > height_third_std_deviation_start and result < height_
#Percentage of data within 1, 2 and 3 Standard Deviations for Weight
weight_list_of_data_within_1_std_deviation = [result for result in weight_list if result > weight_first_std_deviation_start and result < weight_
weight_list_of_data_within_2_std_deviation = [result for result in weight_list if result > weight_second_std_deviation_start and result < weigh
weight_list_of_data_within_3_std_deviation = [result for result in weight_list if result > weight_third_std_deviation_start and result < weight_
#Printing data for height and weight (Standard Deviation)
print("{}% of data for height lies within 1 standard deviation".format(len(height_list_of_data_within_1_std_deviation)*100.0/len(height_list)))
print("{}% of data for height lies within 2 standard deviations".format(len(height_list_of_data_within_2_std_deviation)*100.0/len(height_list)))
print("{}% of data for height lies within 3 standard deviations".format(len(height_list_of_data_within_3_std_deviation)*100.0/len(height_list)))
print("{}% of data for weight lies within 1 standard deviation".format(len(weight_list_of_data_within_1_std_deviation)*100.0/len(weight_list)))
print("{}% of data for weight lies within 2 standard deviations".format(len(weight_list_of_data_within_2_std_deviation)*100.0/len(weight_list)))
print("{}% of data for weight lies within 3 standard deviations".format(len(weight_list_of_data_within_3_std_deviation)*100.0/len(weight_list)))
```

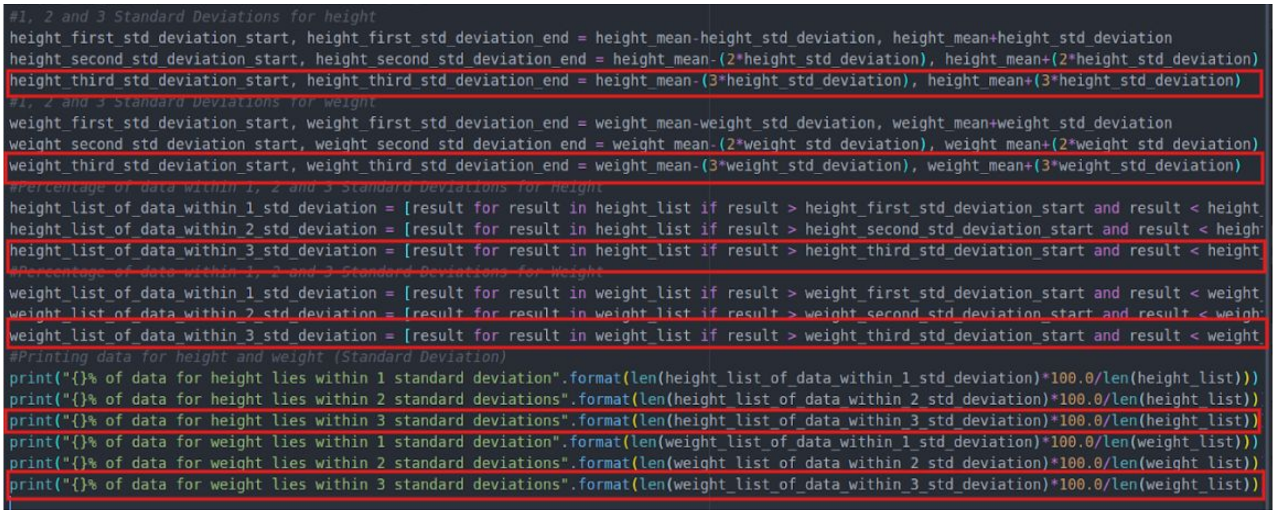


```
95.46% of data for height lies within 2 standard deviations
```



```
95.284% of data for weight lies within 2 standard deviations
```

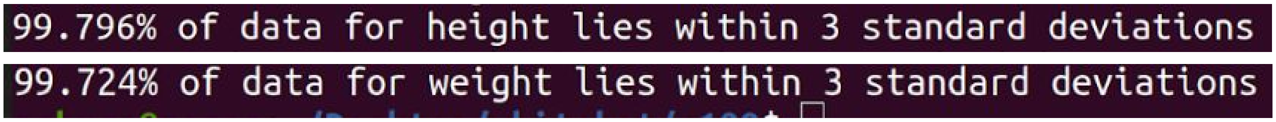
	Let's find out what percentage of data lie between mean - 3sd and mean + 3sd	Student writes program to calculate the % of heights and weights which lie between mean - 3sd and mean + 3sd and verify it is 99%



```
#1, 2 and 3 Standard Deviations for height
height_first_std_deviation_start, height_first_std_deviation_end = height_mean-height_std_deviation, height_mean+height_std_deviation
height_second_std_deviation_start, height_second_std_deviation_end = height_mean-(2*height_std_deviation), height_mean+(2*height_std_deviation)
height_third_std_deviation_start, height_third_std_deviation_end = height_mean-(3*height_std_deviation), height_mean+(3*height_std_deviation)
#1, 2 and 3 Standard Deviations for weight
weight_first_std_deviation_start, weight_first_std_deviation_end = weight_mean-weight_std_deviation, weight_mean+weight_std_deviation
weight_second_std_deviation_start, weight_second_std_deviation_end = weight_mean-(2*weight_std_deviation), weight_mean+(2*weight_std_deviation)
weight_third_std_deviation_start, weight_third_std_deviation_end = weight_mean-(3*weight_std_deviation), weight_mean+(3*weight_std_deviation)
#Percentage of data within 1, 2 and 3 Standard Deviations for Height
height_list_of_data_within_1_std_deviation = [result for result in height_list if result > height_first_std_deviation_start and result < height_
height_list_of_data_within_2_std_deviation = [result for result in height_list if result > height_second_std_deviation_start and result < heigh
height_list_of_data_within_3_std_deviation = [result for result in height_list if result > height_third_std_deviation_start and result < height_
#Percentage of data within 1, 2 and 3 Standard Deviations for Weight
weight_list_of_data_within_1_std_deviation = [result for result in weight_list if result > weight_first_std_deviation_start and result < weight_
weight_list_of_data_within_2_std_deviation = [result for result in weight_list if result > weight_second_std_deviation_start and result < weigh
weight_list_of_data_within_3_std_deviation = [result for result in weight_list if result > weight_third_std_deviation_start and result < weight_
#Printing data for height and weight (Standard Deviation)
print("{}% of data for height lies within 1 standard deviation".format(len(height_list_of_data_within_1_std_deviation)*100.0/len(height_list)))
print("{}% of data for height lies within 2 standard deviations".format(len(height_list_of_data_within_2_std_deviation)*100.0/len(height_list)))
print("{}% of data for height lies within 3 standard deviations".format(len(height_list_of_data_within_3_std_deviation)*100.0/len(height_list)))
print("{}% of data for weight lies within 1 standard deviation".format(len(weight_list_of_data_within_1_std_deviation)*100.0/len(weight_list)))
print("{}% of data for weight lies within 2 standard deviations".format(len(weight_list_of_data_within_2_std_deviation)*100.0/len(weight_list)))
print("{}% of data for weight lies within 3 standard deviations".format(len(weight_list_of_data_within_3_std_deviation)*100.0/len(weight_list)))
```

height_list_of_data_within_3_std_deviation = [result for result in height_list if result > height_third_std_deviation_start and result < height_third_std_deviation_end]

weight_list_of_data_within_3_std_deviation = [result for result in weight_list if result > weight_third_std_deviation_start and result < weight_third_std_deviation_end]



```
99.796% of data for height lies within 3 standard deviations
99.724% of data for weight lies within 3 standard deviations
```

Teacher Guides Student to Stop Screen Share

FEEDBACK

- **Appreciate the student for their efforts**
- **Identify 2 strengths and 1 area of progress for the student**

Step 4: **Wrap-Up** **(5 min)**	Let's summarize what we have learned about normal distributions in today's class	ESR: We have learned that normal distributions can be seen as probability distributions. Mean = Median = Mode in a normal distribution and corresponds to the peak value

		Normal distribution is symmetric around the peak value. 68% of all data lie within one standard deviation of the mean 95% of all the data lie within two standard deviation of the mean 99% of all the data lie within three standard deviation of the mean
	Amazing! Normal distributions are the most interesting pattern in data science. We'll learn to use the learnings you just had about normal distributions in lots of interesting data analysis and machine learning algorithm. But what about data sets which do not follow a normal distribution. For example, I am sharing with you a data set which is taken from sensor reading of a temperature sensor in a room. I want you to plot it and see what its distribution looks like. We will be talking about this in next class!	-
Teacher Clicks	✖ End Class	

Additional Activities	Encourage the student to write reflection notes in their reflection journal using markdown. Use these as guiding questions: ● What happened today? - Describe what happened - Code I wrote ● How did I feel after the class? ● What have I learned about programming and developing games? ● What aspects of the class helped me? What did I find difficult?	The student uses the markdown editor to write her/his reflection in a reflection journal.

Activity	Activity Name	Links
Teacher Activity 1	Solution	https://github.com/whitehatjr/Properties-of-normal-distribution
Student Activity 2	height-weight data	https://raw.githubusercontent.com/whitehatjr/Properties-of-normal-distribution/master/height-weight.csv