2.2.1 Using Comments in Python

In [1]:
```
"""
This is a multiline comment
"""

#This is a single line comment
```

Out[1]: '\nThis is a multiline comment\n'

2.2.3 Importing Packages in Python

In [90]:
```
#Importing packages
import pandas as pd
import numpy as np
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from scipy import stats
import matplotlib.pyplot as plt
import seaborn as sns
```

2.2.4 Getting Data into Python

In [3]:
```
#Importing the bank_marketing_trainind dataset
bank_train = pd.read_csv('C:/School/DSC-530/DataSets/bank_marketing_training')
```

In [5]:
```
#Displaying the data
bank_train
```

Out[5]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	days_since_previ
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	9
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	9
2	41	blue-collar	married	unknown	unknown	no	no	telephone	may	mon	...	1	9
3	25	services	single	high.school	no	yes	no	telephone	may	mon	...	1	9
4	29	blue-collar	single	high.school	no	no	yes	telephone	may	mon	...	1	9
...	...	...	...	...	...	...	...	...	...	...	...	...	
26869	36	admin.	married	university.degree	no	no	no	cellular	nov	fri	...	2	9
26870	37	admin.	married	university.degree	no	yes	no	cellular	nov	fri	...	1	9
26871	29	unemployed	single	basic.4y	no	yes	no	cellular	nov	fri	...	1	
26872	73	retired	married	professional.course	no	yes	no	cellular	nov	fri	...	1	9
26873	46	blue-collar	married	professional.course	no	no	no	cellular	nov	fri	...	1	9

26874 rows × 21 columns

2.2.5 Saving Output in Python

In [6]:
```
#Making a contingency table with no saving
pd.crosstab(bank_train['previous_outcome'], bank_train['response'])
```

Out[6]:

response	no	yes
previous_outcome		
failure	2390	385
nonexistent	21176	2034
success	320	569

In [7]:
```
#Making a contingency table with with saving
crosstab_01 = pd.crosstab(bank_train['previous_outcome'], bank_train['response'])
```

In [8]:
```
#Display the data
crosstab_01
```

Out[8]:

response	no	yes
previous_outcome		
failure	2390	385
nonexistent	21176	2034
success	320	569

2.2.6 Accessing Records and Variables in Python

In [9]:
```
#Retrieving one row
bank_train.loc[0]
```

Out[9]:
```
age                           56
job                    housemaid
marital                  married
education               basic.4y
default                       no
housing                       no
loan                          no
contact                telephone
month                        may
day_of_week                  mon
duration                     261
campaign                       1
days_since_previous          999
previous                       0
previous_outcome     nonexistent
emp.var.rate                 1.1
cons.price.idx            93.994
cons.conf.idx              -36.4
euribor3m                  4.857
nr.employed                 5191
response                      no
Name: 0, dtype: object
```

In [10]:
```
#Extracting three rows
bank_train.loc[[0,2,3]]
```

Out[10]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	days_since_previous	previous
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	0
2	41	blue-collar	married	unknown	unknown	no	no	telephone	may	mon	...	1	999	0
3	25	services	single	high.school	no	yes	no	telephone	may	mon	...	1	999	0

3 rows × 21 columns

In [11]:
```
#Extracting 10 rows using different example
bank_train.loc[0:10]
```

Out[11]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	campaign	days_since_previous	prev
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	1	999	
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	1	999	
2	41	blue-collar	married	unknown	unknown	no	no	telephone	may	mon	...	1	999	
3	25	services	single	high.school	no	yes	no	telephone	may	mon	...	1	999	
4	29	blue-collar	single	high.school	no	no	yes	telephone	may	mon	...	1	999	
5	57	housemaid	divorced	basic.4y	no	yes	no	telephone	may	mon	...	1	999	
6	35	blue-collar	married	basic.6y	no	yes	no	telephone	may	mon	...	1	999	
7	39	management	single	basic.9y	unknown	no	no	telephone	may	mon	...	1	999	
8	30	unemployed	married	high.school	no	no	no	telephone	may	mon	...	1	999	
9	55	retired	single	high.school	no	yes	no	telephone	may	mon	...	1	999	
10	41	technician	single	high.school	no	yes	no	telephone	may	mon	...	1	999	

11 rows × 21 columns

In [12]:
```
#Extracting by column
bank_train['age']
```

Out[12]:
```
0        56
1        57
2        41
3        25
4        29
         ..
26869    36
26870    37
26871    29
26872    73
26873    46
Name: age, Length: 26874, dtype: int64
```

In [13]:
```
#Extracting more column
bank_train[['age', 'job']]
```

Out[13]:

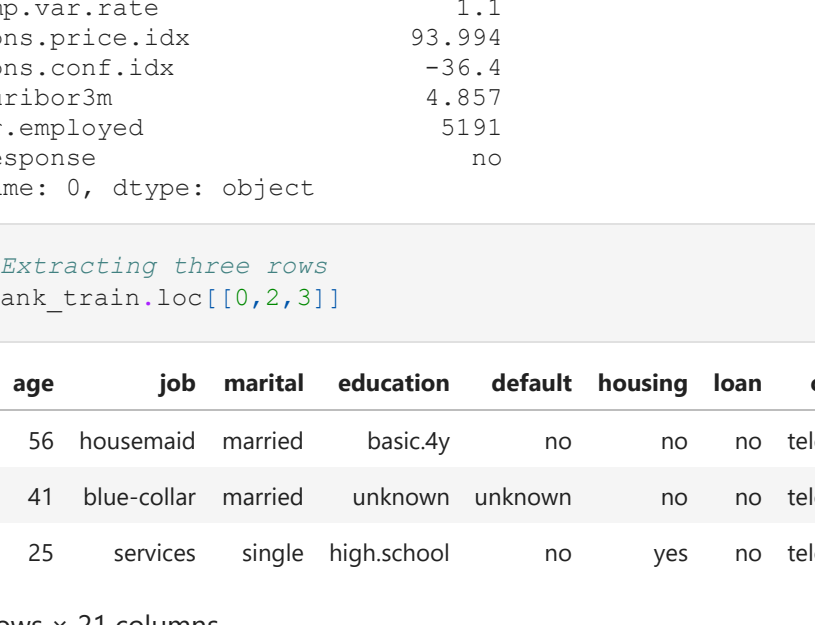
	age	job
0	56	housemaid
1	57	services
2	41	blue-collar
3	25	services
4	29	blue-collar
...	...	...
26869	36	admin.
26870	37	admin.
26871	29	unemployed
26872	73	retired
26873	46	blue-collar

26874 rows × 2 columns

2.2.7 Setting Up Graphics in Python

In [14]:
```
bank_train['age'].plot(kind='hist',title = 'Histogram of Age')
```

Out[14]: <AxesSubplot:title={'center':'Histogram of Age'}, ylabel='Frequency'>



Part 2: Data Preparation in Python

In [26]:
```
#Importing the data set
nutrition = pd.read_csv('C:/School/DSC-530/DataSets/nutrition_subset')
```

In [27]:
```
#Display data
nutrition
```

Out[27]:

	food item	weight_in_grams	saturated_fat	cholesterol
0	GELATIN; DRY 1 ENVELP	7.00	0.0	0
1	SEAWEED; SPIRULINA; DRIED 1 OZ	28.35	0.8	0
2	YEAST; BAKERS; DRY; ACTIVE 1 PKG	7.00	0.0	0
3	PARMESAN CHEESE; GRATED 1 OZ	28.35	5.4	22
4	PARMESAN CHEESE; GRATED 1 CUP	100.00	19.1	79
...	...	...	...	...
956	COFFEE; BREWED 6 FL OZ	180.00	0.0	0
957	TEA; BREWED 8 FL OZ	240.00	0.0	0
958	TEA; INSTANT;PREPRD;UNSWEETEND8 FL OZ	241.00	0.0	0
959	LETTUCE; BUTTERHEAD; RAW;LEAVE1 LEAF	15.00	0.0	0
960	SALT 1 TSP	5.50	0.0	0

961 rows × 4 columns

1. Sort the data set by the saturated fat variable

In [29]:
```
#Adding an index
nutrition['index'] = pd.Series(range(0,961))
```

In [30]:
```
nutrition
```

Out[30]:

	food item	weight_in_grams	saturated_fat	cholesterol	index
0	GELATIN; DRY 1 ENVELP	7.00	0.0	0	0
1	SEAWEED; SPIRULINA; DRIED 1 OZ	28.35	0.8	0	1
2	YEAST; BAKERS; DRY; ACTIVE 1 PKG	7.00	0.0	0	2
3	PARMESAN CHEESE; GRATED 1 OZ	28.35	5.4	22	3
4	PARMESAN CHEESE; GRATED 1 CUP	100.00	19.1	79	4
...	...	...	...	...	...
956	COFFEE; BREWED 6 FL OZ	180.00	0.0	0	956
957	TEA; BREWED 8 FL OZ	240.00	0.0	0	957
958	TEA; INSTANT;PREPRD;UNSWEETEND8 FL OZ	241.00	0.0	0	958
959	LETTUCE; BUTTERHEAD; RAW;LEAVE1 LEAF	15.00	0.0	0	959
960	SALT 1 TSP	5.50	0.0	0	960

961 rows × 5 columns

In [31]:
```
nutrition_sort = nutrition.sort_values(['saturated_fat'], ascending=False)
```

In [32]:
```
nutrition_sort
```

Out[32]:

	food item	weight_in_grams	saturated_fat	cholesterol	index
378	CHEESECAKE 1 CAKE	1110.0	119.9	2053	378
535	ICE CREAM; VANLLA; RICH 16% FT1/2 GAL	1188.0	118.3	703	535
458	YELLOWCAKE W/ CHOCFRSTNG;COMML1 CAKE	1108.0	92.0	609	458
581	CREME PIE 1 PIE	910.0	90.1	46	581
890	LARD 1 CUP	205.0	80.4	195	890
...	...	...	...	...	...
731	TOMATOES; RAW 1 TOMATO	123.0	0.0	0	731
730	RASPBERRIES; RAW 1 CUP	123.0	0.0	0	730
727	TOMATO JUICE; CANNED W/O SALT 1 CUP	244.0	0.0	0	727
726	TOMATO JUICE; CANNED WITH SALT1 CUP	244.0	0.0	0	726
960	SALT 1 TSP	5.5	0.0	0	960

961 rows × 5 columns

In [33]:
```
nutrition_sort.head(5)
```

Out[33]:

	food item	weight_in_grams	saturated_fat	cholesterol	index
378	CHEESECAKE 1 CAKE	1110.0	119.9	2053	378
535	ICE CREAM; VANLLA; RICH 16% FT1/2 GAL	1188.0	118.3	703	535
458	YELLOWCAKE W/ CHOCFRSTNG;COMML1 CAKE	1108.0	92.0	609	458
581	CREME PIE 1 PIE	910.0	90.1	46	581
890	LARD 1 CUP	205.0	80.4	195	890

1. Derive a new variable, saturated_fat_per_gram

In [45]:
```
nutrition['saturated_fat_per_gram'] = nutrition['saturated_fat'].div(nutrition['weight_in_grams'].values)
```

In [49]:
```
nutrition
```

Out[49]:

	food item	weight_in_grams	saturated_fat	cholesterol	index	saturated_fat_per_gram
0	GELATIN; DRY 1 ENVELP	7.00	0.0	0	0	0.000000
1	SEAWEED; SPIRULINA; DRIED 1 OZ	28.35	0.8	0	1	0.028219
2	YEAST; BAKERS; DRY; ACTIVE 1 PKG	7.00	0.0	0	2	0.000000
3	PARMESAN CHEESE; GRATED 1 OZ	28.35	5.4	22	3	0.190476
4	PARMESAN CHEESE; GRATED 1 CUP	100.00	19.1	79	4	0.191000
...	...	...	...	...	...	...
956	COFFEE; BREWED 6 FL OZ	180.00	0.0	0	956	0.000000
957	TEA; BREWED 8 FL OZ	240.00	0.0	0	957	0.000000
958	TEA; INSTANT;PREPRD;UNSWEETEND8 FL OZ	241.00	0.0	0	958	0.000000
959	LETTUCE; BUTTERHEAD; RAW;LEAVE1 LEAF	15.00	0.0	0	959	0.000000
960	SALT 1 TSP	5.50	0.0	0	960	0.000000

961 rows × 6 columns

In [50]:
```
nutrition_gram_sort = nutrition.sort_values(['saturated_fat_per_gram'], ascending=False)
```

In [51]:
```
nutrition_gram_sort.head(5)
```

Out[51]:

	food item	weight_in_grams	saturated_fat	cholesterol	index	saturated_fat_per_gram
908	BUTTER; SALTED 1 TBSP	14.0	7.1	31	908	0.507143
909	BUTTER; UNSALTED 1 TBSP	14.0	7.1	31	909	0.507143
710	BUTTER; UNSALTED 1/2 CUP	113.0	57.1	247	710	0.505310
709	BUTTER; SALTED 1/2 CUP	113.0	57.1	247	709	0.505310
913	BUTTER; UNSALTED 1 PAT	5.0	2.5	11	913	0.500000

Derive a new variable, cholesterol_per_gram

In [52]:
```
nutrition['cholesterol_per_gram'] = nutrition['cholesterol'].div(nutrition['weight_in_grams'].values)
```

In [54]:
```
nutrition_chol_gram = nutrition.sort_values(['cholesterol_per_gram'], ascending=False)
```

In [55]:
```
nutrition_chol_gram.head(5)
```

Out[55]:

	food item	weight_in_grams	saturated_fat	cholesterol	index	saturated_fat_per_gram	cholesterol_per_gram
119	EGGS; RAW; YOLK 1 YOLK	17.0	1.6	213	119	0.094118	12.529412
58	CHICKEN LIVER; COOKED 1 LIVER	20.0	0.4	126	58	0.020000	6.300000
45	BEEF LIVER; FRIED 3 OZ	85.0	2.5	410	45	0.029412	4.823529
167	EGGS; COOKED; FRIED 1 EGG	46.0	1.9	211	167	0.041304	4.586957
186	EGGS; COOKED; HARD-COOKED 1 EGG	50.0	1.6	213	186	0.032000	4.260000

1. Standardize the feld saturated_fat_per_gram

In [59]:
```
nutrition['saturated_fat_per_gram_z']= stats.zscore(nutrition['saturated_fat_per_gram'])
```

In [60]:
```
nutrition
```

Out[60]:

	food item	weight_in_grams	saturated_fat	cholesterol	index	saturated_fat_per_gram	cholesterol_per_gram	saturated
0	GELATIN; DRY 1 ENVELP	7.00	0.0	0	0	0.000000	0.000000	
1	SEAWEED; SPIRULINA; DRIED 1 OZ	28.35	0.8	0	1	0.028219	0.000000	
2	YEAST; BAKERS; DRY; ACTIVE 1 PKG	7.00	0.0	0	2	0.000000	0.000000	
3	PARMESAN CHEESE; GRATED 1 OZ	28.35	5.4	22	3	0.190476	0.776014	
4	PARMESAN CHEESE; GRATED 1 CUP	100.00	19.1	79	4	0.191000	0.790000	
...	...	...	...	...	...	...	...	
956	COFFEE; BREWED 6 FL OZ	180.00	0.0	0	956	0.000000	0.000000	
957	TEA; BREWED 8 FL OZ	240.00	0.0	0	957	0.000000	0.000000	
958	TEA; INSTANT;PREPRD;UNSWEETEND8 FL OZ	241.00	0.0	0	958	0.000000	0.000000	
959	LETTUCE; BUTTERHEAD; RAW;LEAVE1 LEAF	15.00	0.0	0	959	0.000000	0.000000	
960	SALT 1 TSP	5.50	0.0	0	960	0.000000	0.000000	

961 rows × 8 columns

In [67]:
```
saturated_fat_per_gram_outliers = nutrition.query('saturated_fat_per_gram_z > 3 | saturated_fat_per_gram_z < -
```

In [64]:
```
saturated_fat_per_gram_outliers
```

Out[64]:

	food item	weight_in_grams	saturated_fat	cholesterol	index	saturated_fat_per_gram	cholesterol_per_gram	saturated_fat_per_gra
3	PARMESAN CHEESE; GRATED 1 OZ	28.35	5.4	22	3	0.190476	0.776014	2.319
4	PARMESAN CHEESE; GRATED 1 CUP	100.00	19.1	79	4	0.191000	0.790000	2.327
5	PARMESAN CHEESE; GRATED 1 TBSP	5.00	1.0	4	5	0.200000	0.800000	2.463
18	PORK; CURED; BACON; REGUL;CKED3 SLICE	19.00	3.3	16	18	0.173684	0.842105	2.065
36	SWISS CHEESE 1 OZ	28.35	5.0	26	36	0.176367	0.917108	2.106
63	CHEDDDAR CHEESE; SHREDDED 1 CUP	113.00	23.8	119	63	0.210619	1.053097	2.624
72	CHEDDAR CHEESE 1 OZ	28.35	6.0	30	72	0.211640	1.058201	2.639
74	MUENSTER CHEESE 1 OZ	28.35	5.4	27	74	0.190476	0.952381	2.319
85	CHEDDAR CHEESE 1 CU IN	17.00	3.6	18	85	0.211765	1.058824	2.641
101	PASTERZD PROCES CHEESE;AMERICN1 OZ	28.35	5.6	27	101	0.197531	0.952381	2.426
102	BLUE CHEESE 1 OZ	28.35	5.3	21	102	0.186949	0.740741	2.266
210	CHOCOLATE; BITTER OT BAKING 1 OZ	28.35	9.0	0	210	0.317460	0.000000	4.240
328	MILK CHOCOLATE CANDY; PLAIN 1 OZ	28.35	5.4	6	328	0.190476	0.211640	2.319
329	CREAM CHEESE 1 OZ	28.35	6.2	31	329	0.218695	1.093474	2.746
421	SEMISWEET CHOCOLATE 1 CUP	170.00	36.2	0	421	0.212941	0.000000	2.659
448	COCONUT; RAW; SHREDDED 1 CUP	80.00	23.8	0	448	0.297500	0.000000	3.938
469	SWEET (DARK) CHOCOLATE 1 OZ	28.35	5.9	0	469	0.208113	0.000000	2.586
492	COCONUT; DRIED; SWEETND;SHREDD1 CUP	93.00	29.3	0	492	0.315054	0.000000	4.204
576	COCONUT; RAW; PIECE 1 PIECE	45.00	13.4	0	576	0.297778	0.000000	3.942
587	WHIPPING CREAM; UNWHIPED;HEAVY1 CUP	238.00	54.8	326	587	0.230252	1.369748	2.921
588	WHIPPING CREAM; UNWHIPED;LIGHT1 CUP	239.00	46.2	265	588	0.193305	1.108787	2.362
655	IMITATN WHIPD TOPING;PRESSRZD 1 CUP	70.00	13.2	0	655	0.188571	0.000000	2.290
663	IMITATION WHIPPED TOPPING;FRZN1 CUP	75.00	16.3	0	663	0.217333	0.000000	2.725
709	BUTTER; SALTED 1/2 CUP	113.00	57.1	247	709	0.505310	2.185841	7.082
710	BUTTER; UNSALTED 1/2 CUP	113.00	57.1	247	710	0.505310	2.185841	7.082
890	LARD 1 CUP	205.00	80.4	195	890	0.392195	0.951220	5.371
893	PEANUT OIL 1 TBSP	14.00	2.4	0	893	0.171429	0.000000	2.031
896	SOYBEAN-COTTONSEED OIL; HYDRGN1 TBSP	14.00	2.5	0	896	0.178571	0.000000	2.139
898	FATS; COOKING/VEGETBL SHORTENG1 TBSP	13.00	3.3	0	898	0.253846	0.000000	3.278
899	LARD 1 TBSP	13.00	5.1	12	899	0.392308	0.923077	5.373
905	SOYBEAN-COTTONSEED OIL; HYDRGN1 CUP	218.00	39.2	0	905	0.179817	0.000000	2.158
907	FATS; COOKING/VEGETBL SHORTENG1 CUP	205.00	51.3	0	907	0.250244	0.000000	3.223
908	BUTTER; SALTED 1 TBSP	14.00	7.1	31	908	0.507143	2.214286	7.110
909	BUTTER; UNSALTED 1 TBSP	14.00	7.1	31	909	0.507143	2.214286	7.110
912	BUTTER; SALTED 1 PAT	5.00	2.5	11	912	0.500000	2.200000	7.002
913	BUTTER; UNSALTED 1 PAT	5.00	2.5	11	913	0.500000	2.200000	7.002
920	IMITATION CREAMERS; POWDERED 1 TSP	2.00	0.7	0	920	0.350000	0.000000	4.732
926	IMITATION WHIPPED TOPPING;FRZN1 TBSP	4.00	0.9	0	926	0.225000	0.000000	2.841
928	WHIPPING CREAM; UNWHIPED;HEAVY1 TBSP	15.00	3.5	21	928	0.233333	1.400000	2.967
930	WHIPPING CREAM; UNWHIPED;LIGHT1 TBSP	15.00	2.9	17	930	0.193333	1.133333	2.362
936	IMITATN WHIPD TOPING;PRESSRZD 1 TBSP	4.00	0.8	0	936	0.200000	0.000000	2.463

In [65]:
```
saturated_fat_per_gram_outliers_sort = saturated_fat_per_gram_outliers.sort_values(['saturated_fat_per_gram'],
```

In [71]:
```
saturated_fat_per_gram_outliers_sort
```

Out[71]:

	food item	weight_in_grams	saturated_fat	cholesterol	index	saturated_fat_per_gram	cholesterol_per_gram	saturated_fat_per_gra
908	BUTTER; SALTED 1 TBSP	14.00	7.1	31	908	0.507143	2.214286	7.110
909	BUTTER; UNSALTED 1 TBSP	14.00	7.1	31	909	0.507143	2.214286	7.110
710	BUTTER; UNSALTED 1/2 CUP	113.00	57.1	247	710	0.505310	2.185841	7.082
709	BUTTER; SALTED 1/2 CUP	113.00	57.1	247	709	0.505310	2.185841	7.082
912	BUTTER; SALTED 1 PAT	5.00	2.5	11	912	0.500000	2.200000	7.002
913	BUTTER; UNSALTED 1 PAT	5.00	2.5	11	913	0.500000	2.200000	7.002
899	LARD 1 TBSP	13.00	5.1	12	899	0.392308	0.923077	5.373
890	LARD 1 CUP	205.00	80.4	195	890	0.392195	0.951220	5.371
920	IMITATION CREAMERS; POWDERED 1 TSP	2.00	0.7	0	920	0.350000	0.000000	4.732
210	CHOCOLATE; BITTER OT BAKING 1 OZ	28.35	9.0	0	210	0.317460	0.000000	4.240
492	COCONUT; DRIED; SWEETND;SHREDD1 CUP	93.00	29.3	0	492	0.315054	0.000000	4.204
576	COCONUT; RAW; PIECE 1 PIECE	45.00	13.4	0	576	0.297778	0.000000	3.942
448	COCONUT; RAW; SHREDDED 1 CUP	80.00	23.8	0	448	0.297500	0.000000	3.938
898	FATS; COOKING/VEGETBL SHORTENG1 TBSP	13.00	3.3	0	898	0.253846	0.000000	3.278
907	FATS; COOKING/VEGETBL SHORTENG1 CUP	205.00	51.3	0	907	0.250244	0.000000	3.223
928	WHIPPING CREAM; UNWHIPED;HEAVY1 TBSP	15.00	3.5	21	928	0.233333	1.400000	2.967
587	WHIPPING CREAM; UNWHIPED;HEAVY1 CUP	238.00	54.8	326	587	0.230252	1.369748	2.921
926	IMITATION WHIPPED TOPPING;FRZN1 TBSP	4.00	0.9	0	926	0.225000	0.000000	2.841
329	CREAM CHEESE 1 OZ	28.35	6.2	31	329	0.218695	1.093474	2.746
663	IMITATION WHIPPED TOPPING;FRZN1 CUP	75.00	16.3	0	663	0.217333	0.000000	2.725
421	SEMISWEET CHOCOLATE 1 CUP	170.00	36.2	0	421	0.212941	0.000000	2.659
85	CHEDDAR CHEESE 1 CU IN	17.00	3.6	18	85	0.211765	1.058824	2.641
72	CHEDDAR CHEESE 1 OZ	28.35	6.0	30	72	0.211640	1.058201	2.639
63	CHEDDDAR CHEESE; SHREDDED 1 CUP	113.00	23.8	119	63	0.210619	1.053097	2.624
469	SWEET (DARK) CHOCOLATE 1 OZ	28.35	5.9	0	469	0.208113	0.000000	2.586
5	PARMESAN CHEESE; GRATED 1 TBSP	5.00	1.0	4	5	0.200000	0.800000	2.463
936	IMITATN WHIPD TOPING;PRESSRZD 1 TBSP	4.00	0.8	0	936	0.200000	0.000000	2.463
101	PASTERZD PROCES CHEESE;AMERICN1 OZ	28.35	5.6	27	101	0.197531	0.952381	2.426
930	WHIPPING CREAM; UNWHIPED;LIGHT1 TBSP	15.00	2.9	17	930	0.193333	1.133333	2.362
588	WHIPPING CREAM; UNWHIPED;LIGHT1 CUP	239.00	46.2	265	588	0.193305	1.108787	2.362
4	PARMESAN CHEESE; GRATED 1 CUP	100.00	19.1	79	4	0.191000	0.790000	2.327
328	MILK CHOCOLATE CANDY; PLAIN 1 OZ	28.35	5.4	6	328	0.190476	0.211640	2.319
74	MUENSTER CHEESE 1 OZ	28.35	5.4	27	74	0.190476	0.952381	2.319
3	PARMESAN CHEESE; GRATED 1 OZ	28.35	5.4	22	3	0.190476	0.776014	2.319
655	IMITATN WHIPD TOPING;PRESSRZD 1 CUP	70.00	13.2	0	655	0.188571	0.000000	2.290
102	BLUE CHEESE 1 OZ	28.35	5.3	21	102	0.186949	0.740741	2.266
905	SOYBEAN-COTTONSEED OIL; HYDRGN1 CUP	218.00	39.2	0	905	0.179817	0.000000	2.158
896	SOYBEAN-COTTONSEED OIL; HYDRGN1 TBSP	14.00	2.5	0	896	0.178571	0.000000	2.139
36	SWISS CHEESE 1 OZ	28.35	5.0	26	36	0.176367	0.917108	2.106
18	PORK; CURED; BACON; REGUL;CKED3 SLICE	19.00	3.3	16	18	0.173684	0.842105	2.065
893	PEANUT OIL 1 TBSP	14.00	2.4	0	893	0.171429	0.000000	2.031

1. Standardize the feld cholesterol_per_gram

In [73]:
```
nutrition['nutrition_chol_gram_z']= stats.zscore(nutrition['cholesterol_per_gram'])
```

In [74]:
```
nutrition
```

Out[74]:

	food item	weight_in_grams	saturated_fat	cholesterol	index	saturated_fat_per_gram	cholesterol_per_gram	saturated
0	GELATIN; DRY 1 ENVELP	7.00	0.0	0	0	0.000000	0.000000	
1	SEAWEED; SPIRULINA; DRIED 1 OZ	28.35	0.8	0	1	0.028219	0.000000	
2	YEAST; BAKERS; DRY; ACTIVE 1 PKG	7.00	0.0	0	2	0.000000	0.000000	
3	PARMESAN CHEESE; GRATED 1 OZ	28.35	5.4	22	3	0.190476	0.776014	
4	PARMESAN CHEESE; GRATED 1 CUP	100.00	19.1	79	4	0.191000	0.790000	
...	...	...	...	...	...	...	...	
956	COFFEE; BREWED 6 FL OZ	180.00	0.0	0	956	0.000000	0.000000	
957	TEA; BREWED 8 FL OZ	240.00	0.0	0	957	0.000000	0.000000	
958	TEA; INSTANT;PREPRD;UNSWEETEND8 FL OZ	241.00	0.0	0	958	0.000000	0.000000	
959	LETTUCE; BUTTERHEAD; RAW;LEAVE1 LEAF	15.00	0.0	0	959	0.000000	0.000000	
960	SALT 1 TSP	5.50	0.0	0	960	0.000000	0.000000	

961 rows × 9 columns

In [75]:
```
nutrition_chol_gram_outliers = nutrition.query('nutrition_chol_gram_z > 3 | nutrition_chol_gram_z < -3')
```

In [76]:
```
nutrition_chol_gram_outliers
```

Out[76]:

	food item	weight_in_grams	saturated_fat	cholesterol	index	saturated_fat_per_gram	cholesterol_per_gram	saturated_fat_per_g
45	BEEF LIVER; FRIED 3 OZ	85.0	2.5	410	45	0.029412	4.823529	-0.1
58	CHICKEN LIVER; COOKED 1 LIVER	20.0	0.4	126	58	0.020000	6.300000	-0.2
119	EGGS; RAW; YOLK 1 YOLK	17.0	1.6	213	119	0.094118	12.529412	0.8
167	EGGS; COOKED; FRIED 1 EGG	46.0	1.9	211	167	0.041304	4.586957	0.0
184	EGGS; RAW; WHOLE 1 EGG	50.0	1.6	213	184	0.032000	4.260000	-0.0
185	EGGS; COOKED; POACHED 1 EGG	50.0	1.5	212	185	0.030000	4.240000	-0.1
186	EGGS; COOKED; HARD-COOKED 1 EGG	50.0	1.6	213	186	0.032000	4.260000	-0.0
189	EGGS; COOKED; SCRAMBLED/OMELET1 EGG	61.0	2.2	215	189	0.036066	3.524590	-0.0

In [77]:
```
nutrition_chol_gram_outliers_sort = nutrition_chol_gram_outliers.sort_values(['cholesterol_per_gram'], ascendir
```

In [78]:
```
nutrition_chol_gram_outliers_sort
```

Out[78]:

	food item	weight_in_grams	saturated_fat	cholesterol	index	saturated_fat_per_gram	cholesterol_per_gram	saturated_fat_per_g
119	EGGS; RAW; YOLK 1 YOLK	17.0	1.6	213	119	0.094118	12.529412	0.8
58	CHICKEN LIVER; COOKED 1 LIVER	20.0	0.4	126	58	0.020000	6.300000	-0.2
45	BEEF LIVER; FRIED 3 OZ	85.0	2.5	410	45	0.029412	4.823529	-0.1
167	EGGS; COOKED; FRIED 1 EGG	46.0	1.9	211	167	0.041304	4.586957	0.0
184	EGGS; RAW; WHOLE 1 EGG	50.0	1.6	213	184	0.032000	4.260000	-0.0
186	EGGS; COOKED; HARD-COOKED 1 EGG	50.0	1.6	213	186	0.032000	4.260000	-0.0
185	EGGS; COOKED; POACHED 1 EGG	50.0	1.5	212	185	0.030000	4.240000	-0.1
189	EGGS; COOKED; SCRAMBLED/OMELET1 EGG	61.0	2.2	215	189	0.036066	3.524590	-0.0

Part 3: Data Visualization in Python

In [85]:
```
#Adding an index
bank_train['index'] = pd.Series(range(0,26874))
```

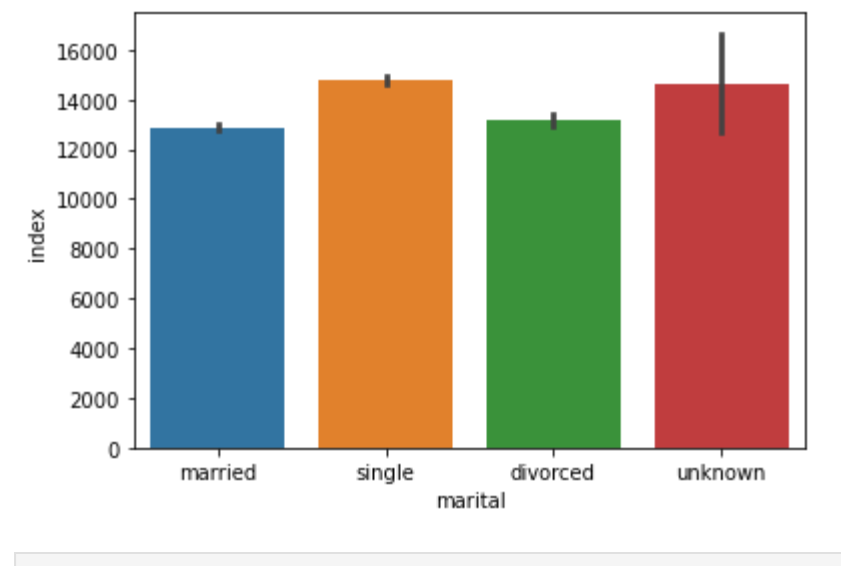
In [86]:
```
bank_train
```

Out[86]:

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	days_since_previous	previo
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	999	
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	999	
2	41	blue-collar	married	unknown	unknown	no	no	telephone	may	mon	...	999	
3	25	services	single	high.school	no	yes	no	telephone	may	mon	...	999	
4	29	blue-collar	single	high.school	no	no	yes	telephone	may	mon	...	999	
...	...	...	...	...	...	...	...	...	...	...	...	...	
26869	36	admin.	married	university.degree	no	no	no	cellular	nov	fri	...	999	
26870	37	admin.	married	university.degree	no	yes	no	cellular	nov	fri	...	999	
26871	29	unemployed	single	basic.4y	no	yes	no	cellular	nov	fri	...	9	
26872	73	retired	married	professional.course	no	yes	no	cellular	nov	fri	...	999	
26873	46	blue-collar	married	professional.course	no	no	no	cellular	nov	fri	...	999	

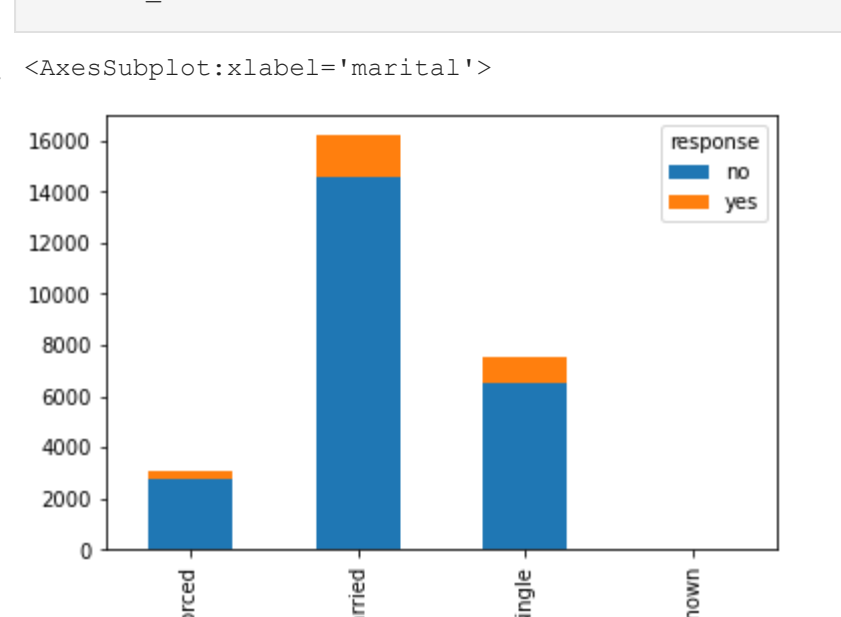
26874 rows × 22 columns

In [91]:
```
# 1 Bar graph of marital
sns.barplot(x=bank_train['marital'], y=bank_train['index'])
plt.show()
```



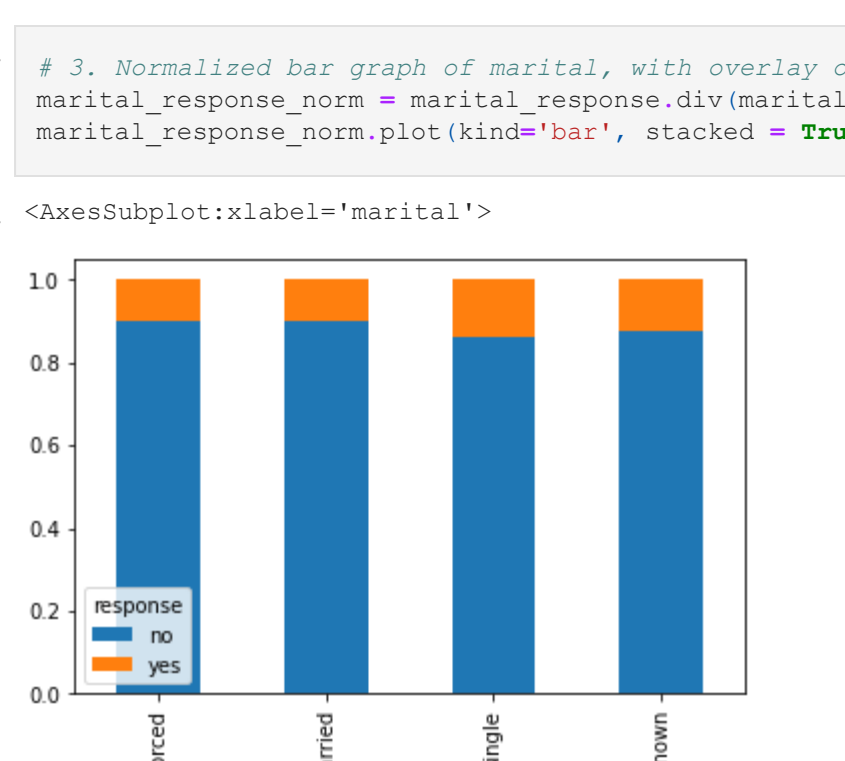
In [100…
```
# 2 Bar graph of marital, with overlay of response.
marital_response = pd.crosstab(bank_train['marital'],bank_train['response'])
marital_response.plot(kind='bar', stacked = True)
```

Out[100… <AxesSubplot:xlabel='marital'>



In [102…
```
# 3. Normalized bar graph of marital, with overlay of response.
marital_response_norm = marital_response.div(marital_response.sum(1), axis = 0)
marital_response_norm.plot(kind='bar', stacked = True)
```

Out[102… <AxesSubplot:xlabel='marital'>



In [108…
```
#  Build a contingency table column percentage
marital_response_cont = pd.crosstab(bank_train['response'], bank_train['marital'])
```

In [109…
```
marital_response_cont
```

Out[109…

marital	divorced	married	single	unknown
response				
no	2743	14579	6514	50
yes	312	1608	1061	7

In [110…
```
round(marital_response_cont.div(marital_response_cont.sum(0), axis=1)*100, 1)
```

Out[110…

marital	divorced	married	single	unknown
response				
no	89.8	90.1	86.0	87.7
yes	10.2	9.9	14.0	12.3

In [111…
```
marital_response_cont_row = pd.crosstab(bank_train['marital'], bank_train['response'])
```

In [112…
```
marital_response_cont_row
```

Out[112…

response	no	yes
marital		
divorced	2743	312
married	14579	1608
single	6514	1061
unknown	50	7

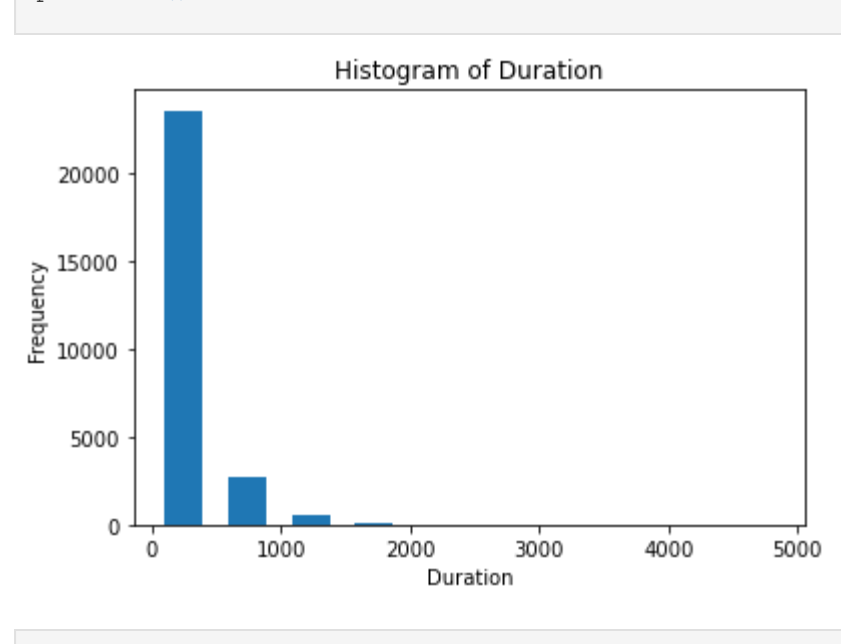
In [114…
```
round(marital_response_cont_row.div(marital_response_cont_row.sum(0), axis=1)*100, 1)
```

Out[114…

response	no	yes
marital		
divorced	11.5	10.4
married	61.0	53.8
single	27.3	35.5
unknown	0.2	0.2

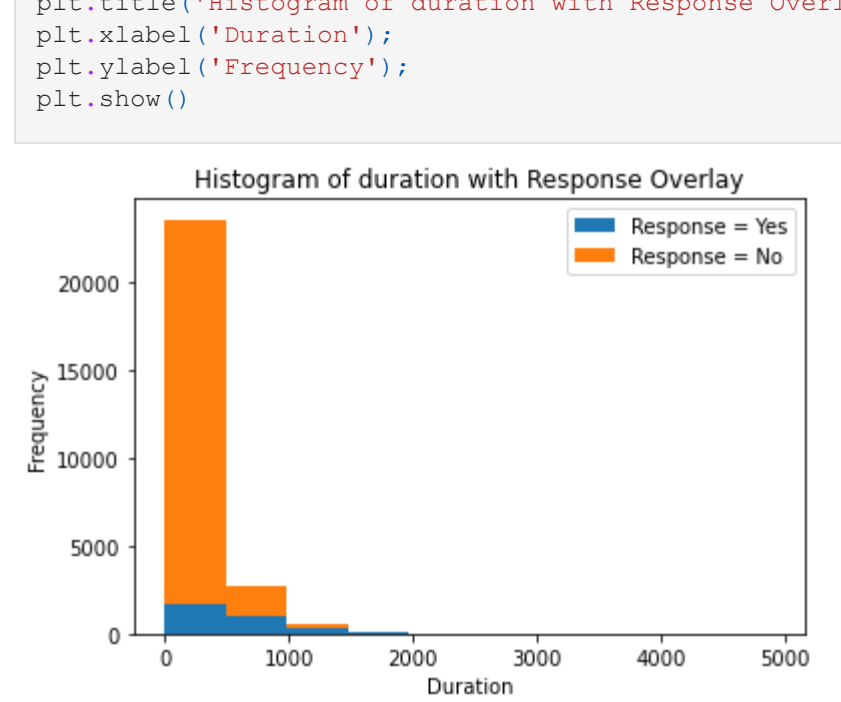
Produce the following graphs. What are the strengths and weaknesses of each graph?

In [121…
```
#Histogram Plot
duration = bank_train['duration']
plt.hist([duration] , rwidth=0.6)
plt.title('Histogram of Duration')
plt.xlabel('Duration');
plt.ylabel('Frequency');
plt.show()
```

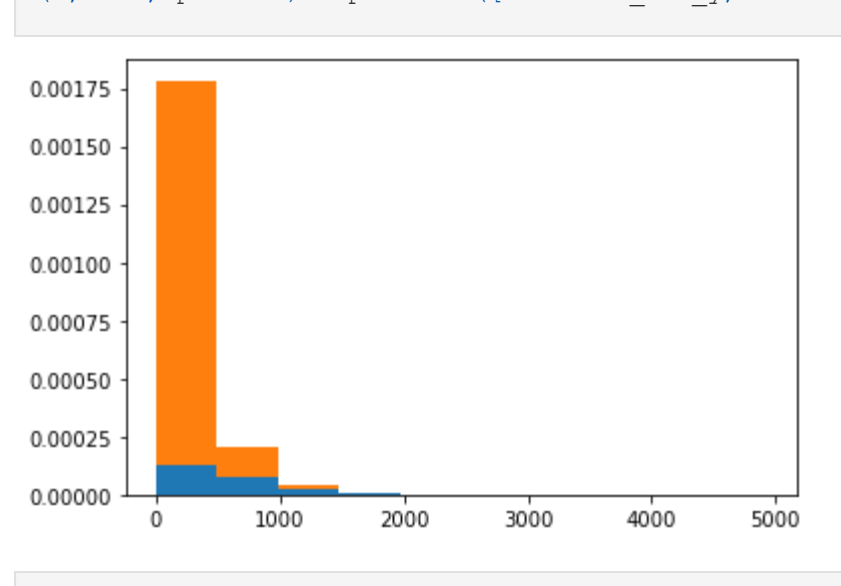


In [122…
```
#Histogram of duration, with overlay of response
duration_res_y = bank_train[bank_train.response == 'yes']['duration']
duration_res_n = bank_train[bank_train.response == 'no']['duration']
```

In [123…
```
plt.hist([duration_res_y, duration_res_n], bins = 10, stacked = True)
plt.legend(['Response = Yes', 'Response = No'])
plt.title('Histogram of duration with Response Overlay')
plt.xlabel('Duration');
plt.ylabel('Frequency');
plt.show()
```



In [153…
```
#Normalized histogram of duration, with overlay of response.
(n,bins, patches) = plt.hist([duration_res_y, duration_res_n], bins=10, stacked=True, density=True)
```

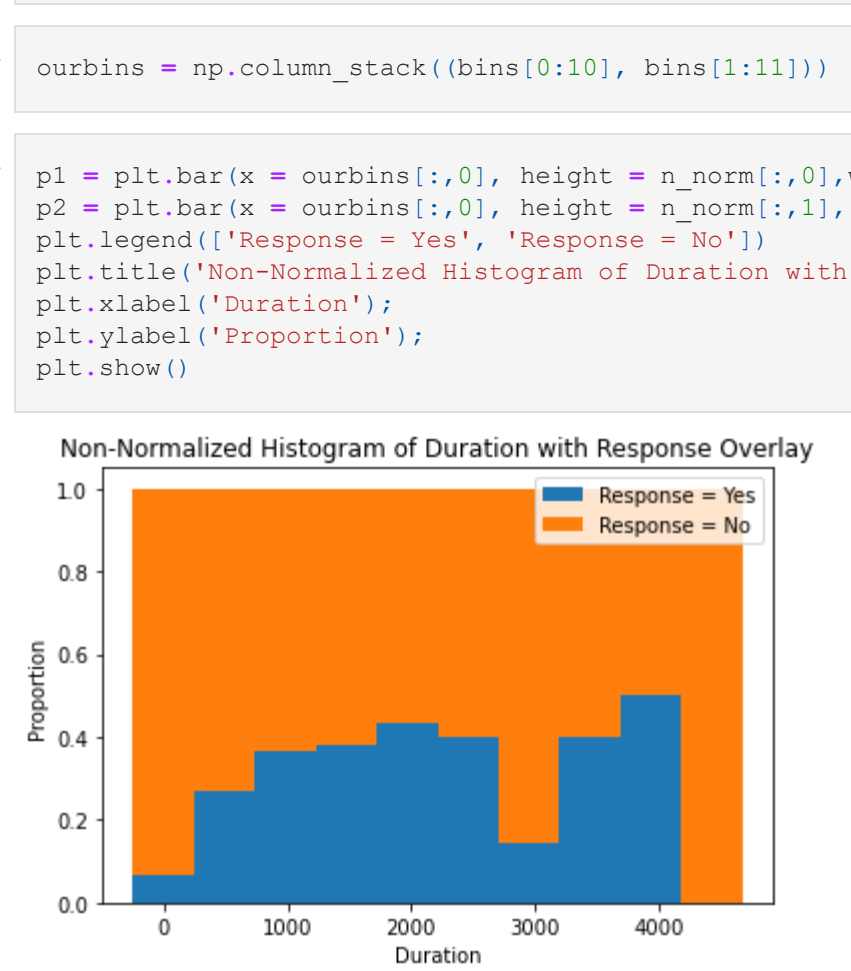


In [154…
```
n_table = np.column_stack((n[0], n[1]))
```

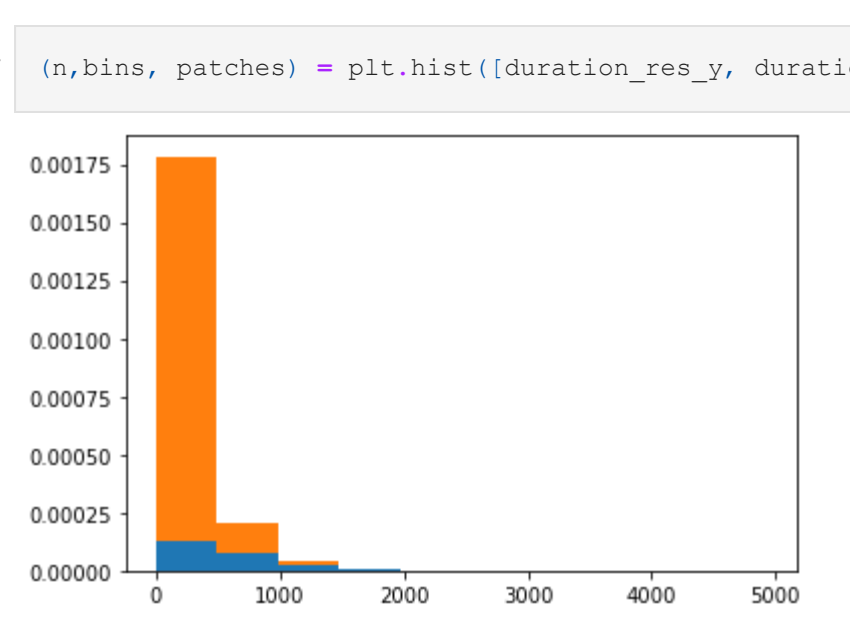
In [155…
```
n_norm = n_table /n_table.sum(axis=1)[:, None]
```

In [156…
```
ourbins = np.column_stack((bins[0:10], bins[1:11]))
```

In [158…
```
p1 = plt.bar(x = ourbins[:,0], height = n_norm[:,0],width = ourbins[:, 1] - ourbins[:, 0])
p2 = plt.bar(x = ourbins[:,0], height = n_norm[:,1], width = ourbins[:, 1] - ourbins[:,0],bottom = n_norm[:,0])
plt.legend(['Response = Yes', 'Response = No'])
plt.title('Non-Normalized Histogram of Duration with Response Overlay')
plt.xlabel('Duration');
plt.ylabel('Proportion');
plt.show()
```



In [127…
```
(n,bins, patches) = plt.hist([duration_res_y, duration_res_n], bins=10, stacked=True, density=True)
```



In [133…
```
#Creating a cutt of point duration_binned
bank_train['duration_binned'] = pd.cut(x=bank_train['duration'], bins=[0,100,200,400], labels=['Under 100', '10
```

In [134…
```
bank_train
```

Out[134…

	age	job	marital	education	default	housing	loan	contact	month	day_of_week	...	previous	previous_outcome
0	56	housemaid	married	basic.4y	no	no	no	telephone	may	mon	...	0	nonexistent
1	57	services	married	high.school	unknown	no	no	telephone	may	mon	...	0	nonexistent
2	41	blue-collar	married	unknown	unknown	no	no	telephone	may	mon	...	0	nonexistent
3	25	services	single	high.school	no	yes	no	telephone	may	mon	...	0	nonexistent
4	29	blue-collar	single	high.school	no	no	yes	telephone	may	mon	...	0	nonexistent
...	...	...	...	...	...	...	...	...	...	...	...	...	...
26869	36	admin.	married	university.degree	no	no	no	cellular	nov	fri	...	0	nonexistent
26870	37	admin.	married	university.degree	no	yes	no	cellular	nov	fri	...	0	nonexistent
26871	29	unemployed	single	basic.4y	no	yes	no	cellular	nov	fri	...	1	success
26872	73	retired	married	professional.course	no	yes	no	cellular	nov	fri	...	0	nonexistent
26873	46	blue-collar	married	professional.course	no	no	no	cellular	nov	fri	...	0	nonexistent

26874 rows × 23 columns

In [135…
```
# Contingency table of duration_binned with response
duration_binned_cont = pd.crosstab(bank_train['response'], bank_train['duration_binned'])
```

In [136…
```
duration_binned_cont
```

Out[136…

duration_binned	Under 100	100 to 200	Over 200
response			
no	6360	7945	6517
yes	46	406	913

In [137…
```
# count of column percentage
round(duration_binned_cont.div(duration_binned_cont.sum(0), axis=1)*100, 1)
```

Out[137…

duration_binned	Under 100	100 to 200	Over 200
response			
no	99.3	95.1	87.7
yes	0.7	4.9	12.3

In [139…
```
# Non-normalized bar graph of duration_binned, with response overlay.
duration_binned_gra = pd.crosstab(bank_train['duration_binned'], bank_train['response'])
duration_binned_gra.plot(kind='bar', stacked = True, title = 'Bar Graph of Duration (Binned) with Response Over
```

Out[139… <AxesSubplot:title={'center':'Bar Graph of Duration (Binned) with Response Overlay'}, xlabel='duration_binned'>