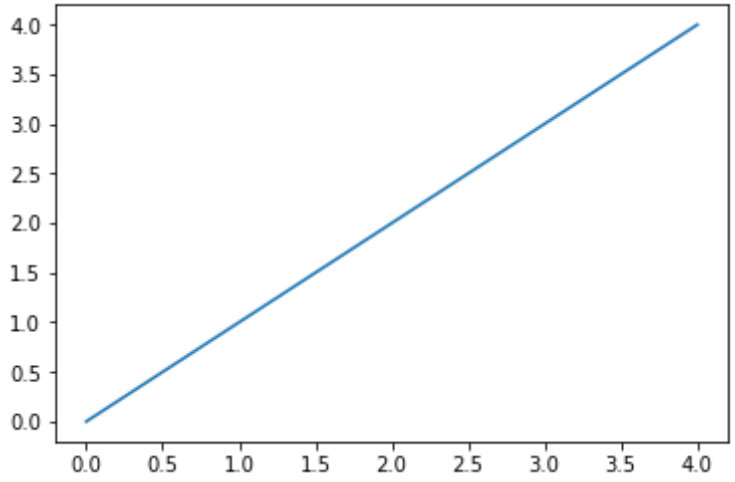
In [6]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [7]:
```
%matplotlib inline
plt.plot(range(5))
```

Out[7]:
```
[<matplotlib.lines.Line2D at 0x11da126d8>]
```



In [10]:
```
df=pd.read_csv("/Users/sunepha/Desktop/train.csv") #reading the dataset
 in a dataframe using pandas
```

In [18]:
```
df.head(10) #Printing first 10 rows of dataset
```

Out[18]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
0	LP001002	Male	No	0	Graduate	No	5849
1	LP001003	Male	Yes	1	Graduate	No	4583

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
2	LP001005	Male	Yes	0	Graduate	Yes	3000
3	LP001006	Male	Yes	0	Not Graduate	No	2583
4	LP001008	Male	No	0	Graduate	No	6000
5	LP001011	Male	Yes	2	Graduate	Yes	5417
6	LP001013	Male	Yes	0	Not Graduate	No	2333
7	LP001014	Male	Yes	3+	Graduate	No	3036
8	LP001018	Male	Yes	2	Graduate	No	4006
9	LP001020	Male	Yes	1	Graduate	No	12841

In [20]:
```
df.describe() #get summary of numerical variables to understand populat
ion distribution
```

Out[20]:

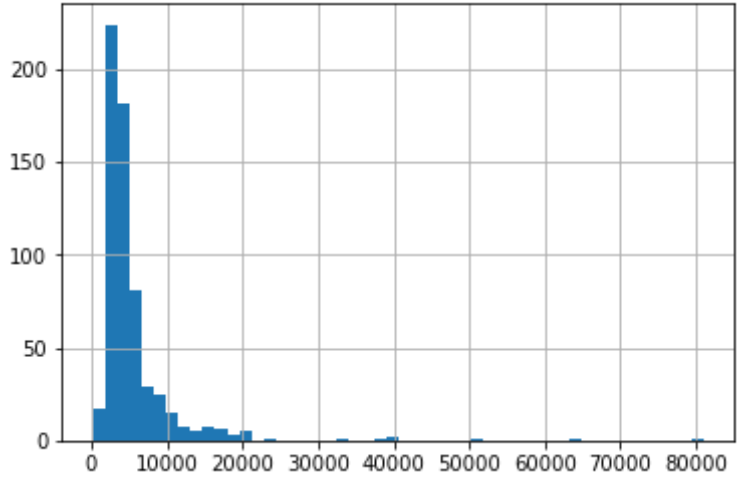
	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_H
count	614.000000	614.000000	592.000000	600.00000	564.0000
mean	5403.459283	1621.245798	146.412162	342.00000	0.842199
std	6109.041673	2926.248369	85.587325	65.12041	0.364878
min	150.000000	0.000000	9.000000	12.00000	0.000000
25%	2877.500000	0.000000	100.000000	360.00000	1.000000
50%	3812.500000	1188.500000	128.000000	360.00000	1.000000
75%	5795.000000	2297.250000	168.000000	360.00000	1.000000
max	81000.000000	41667.000000	700.000000	480.00000	1.000000

In [21]:
```
df['Property_Area'].value_counts()
```

Out[21]:
```
Semiurban    233
Urban        202
Rural        179
Name: Property_Area, dtype: int64
```

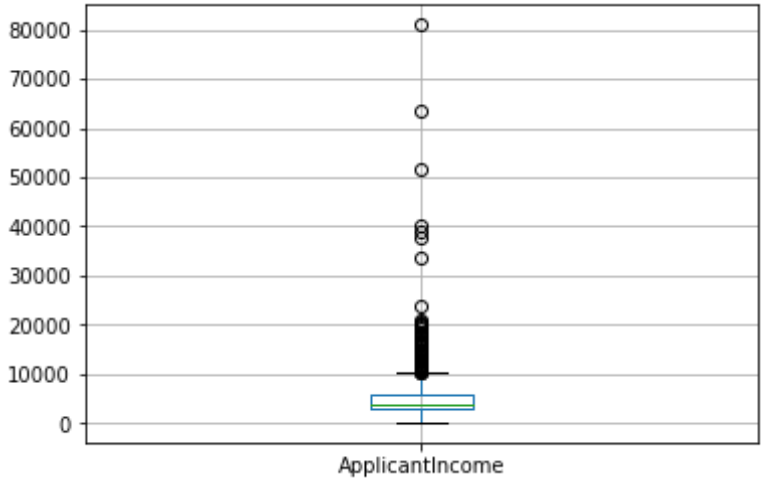
In [23]:
```
df['ApplicantIncome'].hist(bins=50)
```

Out[23]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x114de2da0>
```



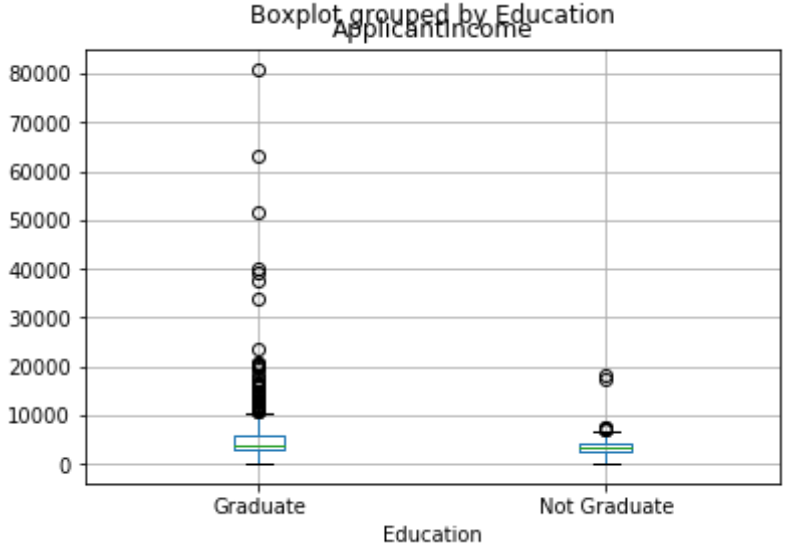
In [24]:
```
df.boxplot(column='ApplicantIncome')
```

Out[24]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11504bf28>
```



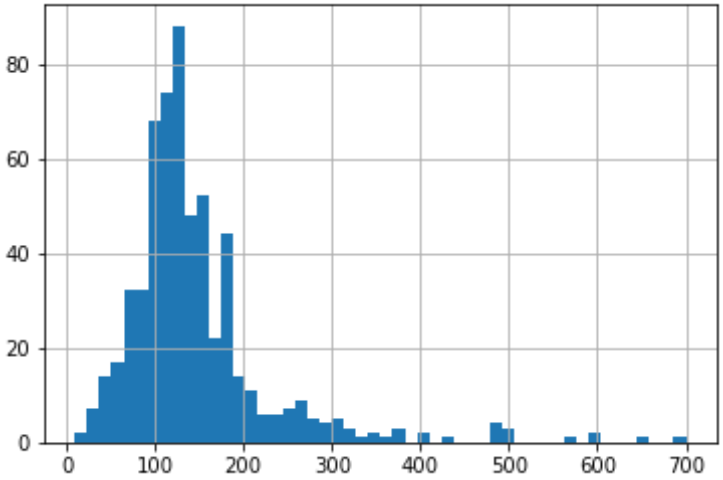
In [25]:
```
df.boxplot(column='ApplicantIncome', by='Education')
```

Out[25]: <matplotlib.axes._subplots.AxesSubplot at 0x115795b70>



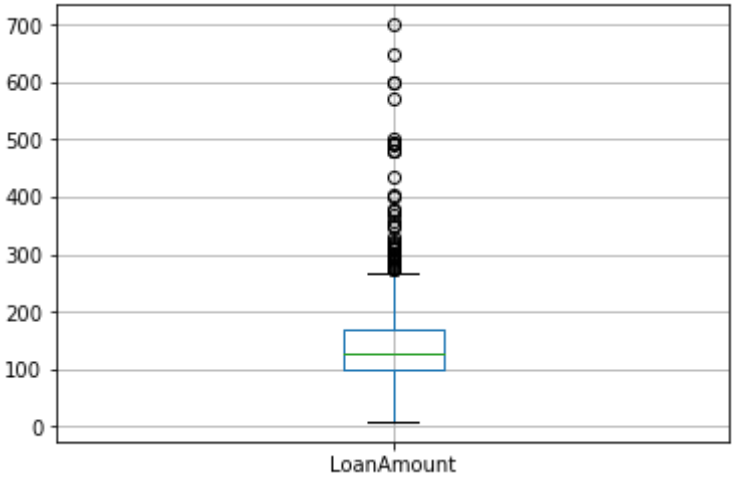
In [26]:
```
df['LoanAmount'].hist(bins=50)
```

Out[26]: <matplotlib.axes._subplots.AxesSubplot at 0x1157bf668>



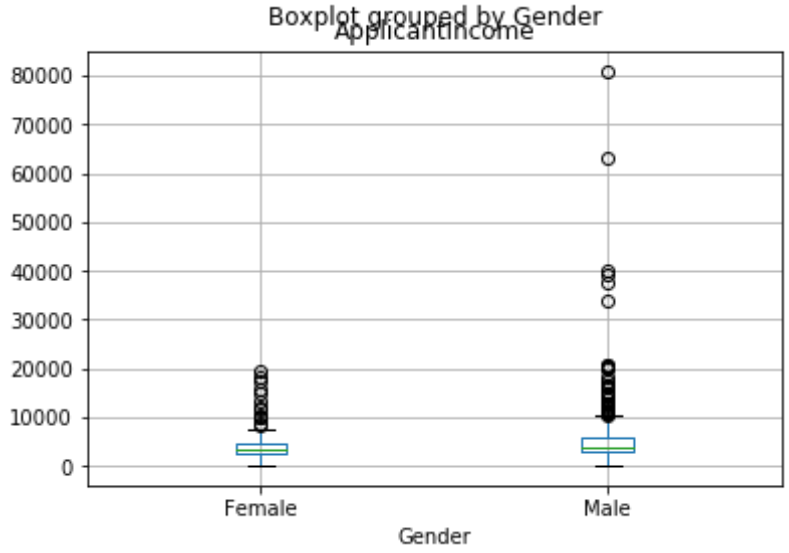
In [27]:
```
df.boxplot(column='LoanAmount')
```

Out[27]: <matplotlib.axes._subplots.AxesSubplot at 0x115a8bba8>



In [28]:
```
df.boxplot(column='ApplicantIncome', by = 'Gender')
```

Out[28]: <matplotlib.axes._subplots.AxesSubplot at 0x11587bcc0>

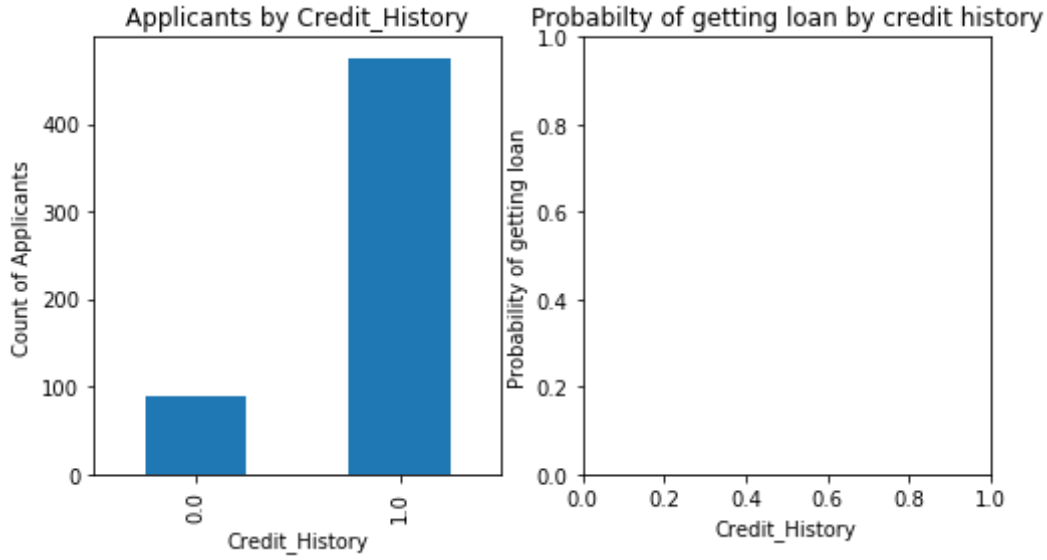


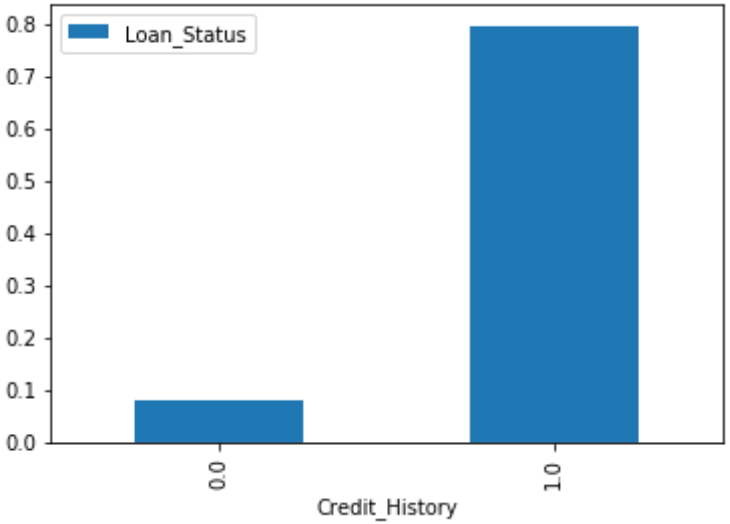
In [33]:
```
temp1=df['Credit_History'].value_counts(ascending=True)
temp2=df.pivot_table(values='Loan_Status', index=['Credit_History'], ag
gfunc=lambda x: x.map({'Y':1, 'N':0}).mean())
print('Frequency Table for Credit History:')
print(temp1)

print('\nProbability of getting loan for each Credit History class:')
print(temp2)
```

```
Frequency Table for Credit History:
0.0     89
1.0    475
Name: Credit_History, dtype: int64

Probability of getting loan for each Credit History class:
                Loan_Status
Credit_History
0.0                0.078652
1.0                0.795789
```

In [34]:
```
fig=plt.figure(figsize=(8,4))
ax1=fig.add_subplot(121)
ax1.set_xlabel('Credit_History')
ax1.set_ylabel('Count of Applicants')
ax1.set_title("Applicants by Credit_History")
temp1.plot(kind='bar')

ax2=fig.add_subplot(122)
temp2.plot(kind='bar')
ax2.set_xlabel('Credit_History')
ax2.set_ylabel('Probability of getting loan')
ax2.set_title("Probabilty of getting loan by credit history")
```

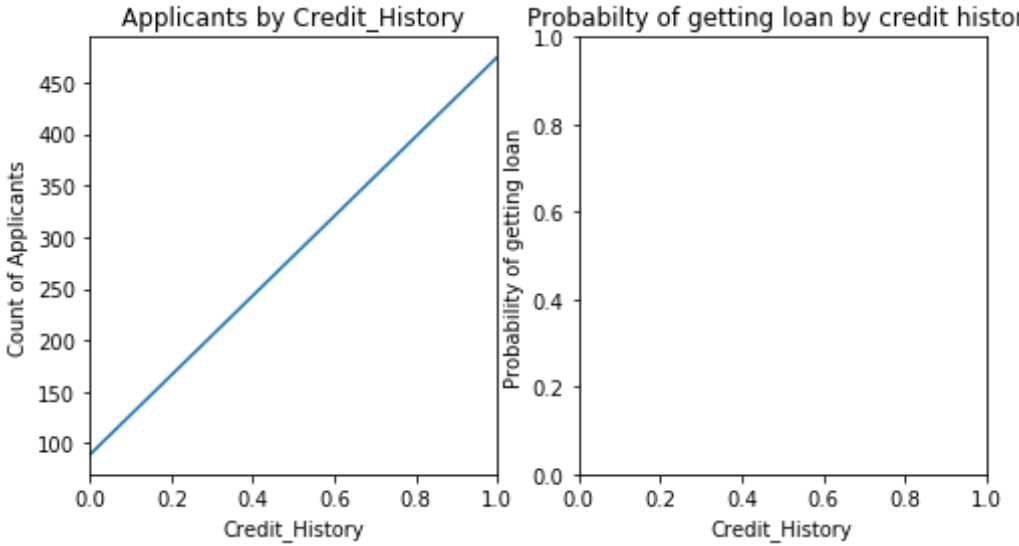
Out[34]:
```
<matplotlib.text.Text at 0x115ed6160>
```





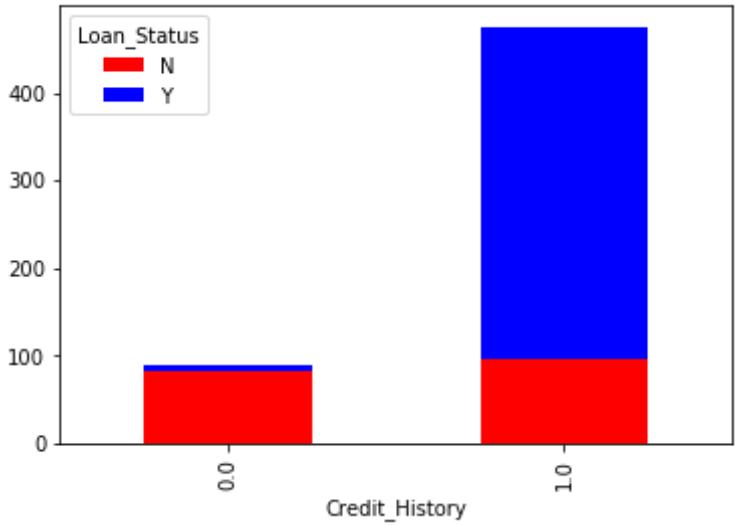
In [36]:
```
fig=plt.figure(figsize=(8,4))
ax1=fig.add_subplot(121)
ax1.set_xlabel('Credit_History')
ax1.set_ylabel('Count of Applicants')
ax1.set_title("Applicants by Credit_History")
temp1.plot(kind='line')

ax2=fig.add_subplot(122)
temp2.plot(kind='line')
ax2.set_xlabel('Credit_History')
ax2.set_ylabel('Probability of getting loan')
ax2.set_title("Probabilty of getting loan by credit history")
```

Out[36]:
```
<matplotlib.text.Text at 0x11630a5c0>
```



In [42]:
```
temp3=pd.crosstab(df['Credit_History'], df['Loan_Status'])
temp3.plot(kind='bar', stacked=True, color=['red', 'blue'], grid=False)
```

Out[42]: <matplotlib.axes._subplots.AxesSubplot at 0x115fade10>



In [43]:
```
df.apply(lambda x: sum(x.isnull()), axis=0)
```

Out[43]:
```
Loan_ID              0
Gender              13
Married              3
Dependents          15
Education            0
Self_Employed       32
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount          22
Loan_Amount_Term    14
Credit_History      50
Property_Area        0
Loan_Status          0
dtype: int64
```

In [29]:
```
df['LoanAmount'].fillna(df['LoanAmount'].mean(), inplace=True)
```

In [14]:
```
df['Self_Employed'].value_counts()
```

Out[14]:
```
No     500
```

```
Yes     82
Name: Self_Employed, dtype: int64
```

In [28]:
```
df['Self_Employed'].fillna('No', inplace=True)
```

In [17]:
```
df.apply(lambda x: sum(x.isnull()), axis=0)
```

Out[17]:
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             0
Self_Employed         0
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount            0
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
LoanAmount_log       22
TotalIncome           0
TotalIncome_log       0
dtype: int64
```

In [31]:
```
df['LoanAmount_log']=np.log(df['LoanAmount'])
```

In [24]:
```
df['Credit_History'].fillna('1', inplace=True)
```

In [25]:
```
temp3=df['Gender'].value_counts(ascending=True)
temp4=df['Married'].value_counts(ascending=True)
temp5=df['Dependents'].value_counts(ascending=True)
temp6=df['Loan_Amount_Term'].value_counts(ascending=True)
print(temp3, temp4, temp5, temp6)
```

```
Female    112
Male      489
Name: Gender, dtype: int64 No     213
```

```
Yes    398
Name: Married, dtype: int64 3+     51
2     101
1     102
0     345
Name: Dependents, dtype: int64 12.0       1
60.0       2
36.0       2
120.0      3
240.0      4
84.0       4
300.0     13
480.0     15
180.0     44
360.0    512
Name: Loan_Amount_Term, dtype: int64
```

In [26]:
```
df['Gender'].fillna('Male', inplace=True)
df['Married'].fillna('Yes', inplace=True)
df['Dependents'].fillna('0', inplace=True)
df['Loan_Amount_Term'].fillna('360', inplace=True)
```

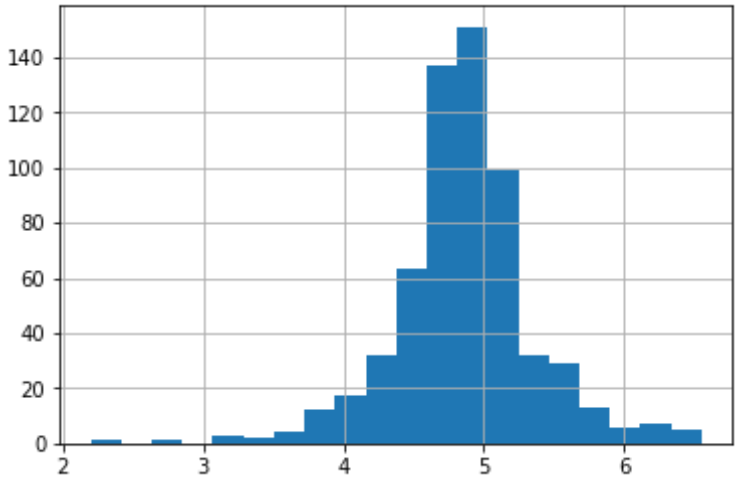
In [32]:
```
df.apply(lambda x: sum(x.isnull()), axis=0)
```

Out[32]:
```
Loan_ID              0
Gender               0
Married              0
Dependents           0
Education            0
Self_Employed        0
ApplicantIncome      0
CoapplicantIncome    0
LoanAmount           0
Loan_Amount_Term     0
Credit_History       0
Property_Area        0
Loan_Status          0
TotalIncome          0
TotalIncome_log      0
```

```
LoanAmount_log       0
dtype: int64
```

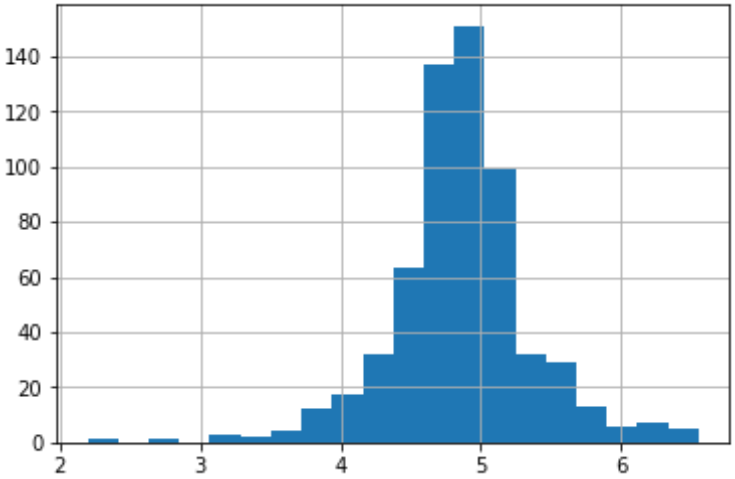
In [33]:
```
df['LoanAmount_log'].hist(bins=20)
```

Out[33]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11e056160>
```



In [34]:
```
df['TotalIncome']=df['ApplicantIncome']+df['CoapplicantIncome']
df['TotalIncome_log']=np.log(df['TotalIncome'])
df['LoanAmount_log'].hist(bins=20)
```

Out[34]:
```
<matplotlib.axes._subplots.AxesSubplot at 0x11e208ef0>
```



In [37]:
```
from sklearn.preprocessing import LabelEncoder
var_mod =
['Gender','Married','Dependents','Education','Self_Employed','Property_
Area','Loan_Status']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i])
df.dtypes
```

Out[37]:
```
Loan_ID              object
Gender                int64
Married               int64
Dependents            int64
Education             int64
Self_Employed         int64
ApplicantIncome       int64
CoapplicantIncome   float64
LoanAmount          float64
Loan_Amount_Term     object
Credit_History       object
Property_Area         int64
Loan_Status           int64
TotalIncome         float64
TotalIncome_log     float64
```

```
LoanAmount_log       float64
dtype: object
```

In [38]:
```
df.head(10)
```

Out[38]:

	Loan_ID	Gender	Married	Dependents	Education	Self_Employed	ApplicantIncome
0	LP001002	1	0	0	0	0	5849
1	LP001003	1	1	1	0	0	4583
2	LP001005	1	1	0	0	1	3000
3	LP001006	1	1	0	1	0	2583
4	LP001008	1	0	0	0	0	6000
5	LP001011	1	1	2	0	1	5417
6	LP001013	1	1	0	1	0	2333
7	LP001014	1	1	3	0	0	3036
8	LP001018	1	1	2	0	0	4006
9	LP001020	1	1	1	0	0	12841

In [39]:
```
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import KFold   #For K-fold cross validati
on
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn import metrics
```

In [41]:
```
#Generic function for making a classification model and accessing perfo
rmance:
def classification_model(model, df, predictors, outcome):
  #Fit the model:
```

```
  model.fit(df[predictors],df[outcome])

  #Make predictions on training set:
  predictions = model.predict(df[predictors])

  #Print accuracy
  accuracy = metrics.accuracy_score(predictions,df[outcome])
  print("Accuracy : %s" % "{0:.3%}".format(accuracy))

  #Perform k-fold cross-validation with 5 folds
  kf = KFold(df.shape[0], n_folds=5)
  error = []
  for train, test in kf:
    # Filter training data
    train_predictors = (df[predictors].iloc[train,:])

    # The target we're using to train the algorithm.
    train_target = df[outcome].iloc[train]

    # Training the algorithm using the predictors and target.
    model.fit(train_predictors, train_target)

    #Record error from each cross-validation run
    error.append(model.score(df[predictors].iloc[test,:], df[outcome].i
loc[test]))

  print("Cross-Validation Score : %s" % "
{0:.3%}".format(np.mean(error)))

  #Fit the model again so that it can be refered outside the function:
  model.fit(df[predictors],df[outcome])
```

In [42]:
```
outcome_var = 'Loan_Status'
model = LogisticRegression()
predictor_var = ['Credit_History']
classification_model(model, df,predictor_var,outcome_var)
```

```
Accuracy : 80.945%
Cross-Validation Score : 80.946%
```

In [43]:
```
#We can try different combination of variables:
predictor_var =
['Credit_History','Education','Married','Self_Employed','Property_Area']

classification_model(model, df,predictor_var,outcome_var)
```

```
Accuracy : 80.945%
Cross-Validation Score : 80.946%
```

In [44]:
```
model = DecisionTreeClassifier()
predictor_var = ['Credit_History','Gender','Married','Education']
classification_model(model, df,predictor_var,outcome_var)
```

```
Accuracy : 80.945%
Cross-Validation Score : 80.946%
```

In [45]:
```
predictor_var = ['Credit_History','Loan_Amount_Term','LoanAmount_log']
classification_model(model, df,predictor_var,outcome_var)
```

```
Accuracy : 89.414%
Cross-Validation Score : 68.722%
```

In [46]:
```
model=RandomForestClassifier(n_estimators=100)
predictor_var=['Gender', 'Married', 'Dependents', 'Education', 'Self_Em
ployed', 'Loan_Amount_Term', 'Credit_History', 'Property_Area', 'LoanAm
ount_log', 'TotalIncome_log']
classification_model(model, df, predictor_var, outcome_var)
```

```
Accuracy : 100.000%
Cross-Validation Score : 78.343%
```

In [47]:
```
featimp=pd.Series(model.feature_importances_, index=predictor_var).sort
_values(ascending=False)
```

In [48]:
```
print(featimp)
```

```
Credit_History      0.269784
```

```
TotalIncome_log     0.260923
LoanAmount_log      0.234196
Dependents          0.050322
Property_Area       0.046534
Loan_Amount_Term    0.043569
Married             0.025055
Education           0.024671
Self_Employed       0.022589
Gender              0.022357
dtype: float64
```

In [49]:
```
model = RandomForestClassifier(n_estimators=25, min_samples_split=25, max_depth=7, max_features=1)
predictor_var = ['TotalIncome_log','LoanAmount_log','Credit_History','Dependents','Property_Area']
classification_model(model, df,predictor_var,outcome_var)
```

```
Accuracy : 82.899%
Cross-Validation Score : 80.949%
```

In [ ]:
```

```