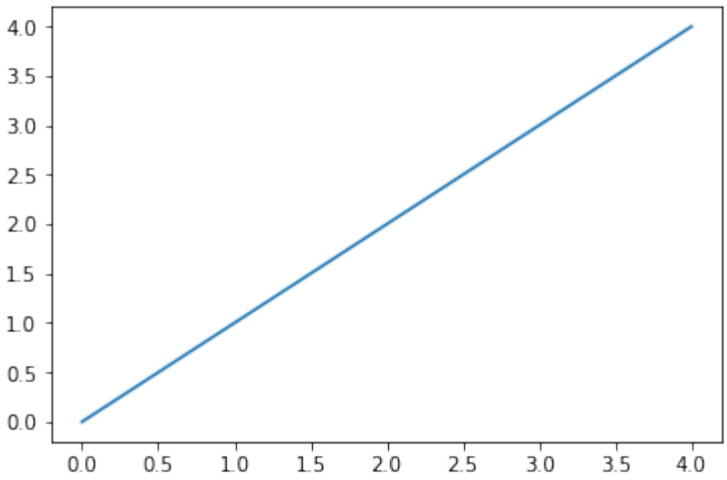
In [1]:
```
import pandas as pd
import numpy as np
import pylab as py
import matplotlib as ml
%matplotlib inline
py.plot(range(5))
```

Out[1]:
```
[<matplotlib.lines.Line2D at 0x8d0a438>]
```




In [3]:
```
df = pd.read_csv('/train.csv')
df.head(20)
```

Out[3]:

| | Loan_ID | Gender | Married | Dependents | Education | Self_Employed | ApplicantIncome | Co |
|---|---|---|---|---|---|---|---|---|
| **0** | LP001002 | Male | No | 0.0 | Graduate | No | 5849 | 0.0 |
| **1** | LP001003 | Male | Yes | 1.0 | Graduate | No | 4583 | 150 |
| **2** | LP001005 | Male | Yes | 0.0 | Graduate | Yes | 3000 | 0.0 |
| **3** | LP001006 | Male | Yes | 0.0 | Not Graduate | No | 2583 | 235 |
| **4** | LP001008 | Male | No | 0.0 | Graduate | No | 6000 | 0.0 |
| **5** | LP001011 | Male | Yes | 2.0 | Graduate | Yes | 5417 | 419 |
| **6** | LP001013 | Male | Yes | 0.0 | Not Graduate | No | 2333 | 151 |
| **7** | LP001014 | Male | Yes | 3.0 | Graduate | No | 3036 | 250 |
| **8** | LP001018 | Male | Yes | 2.0 | Graduate | No | 4006 | 152 |
| **9** | LP001020 | Male | Yes | 1.0 | Graduate | No | 12841 | 109 |
| **10** | LP001024 | Male | Yes | 2.0 | Graduate | No | 3200 | 700 |
| **11** | LP001027 | Male | Yes | 2.0 | Graduate | NaN | 2500 | 184 |
| **12** | LP001028 | Male | Yes | 2.0 | Graduate | No | 3073 | 810 |

| | | | | | | | | |
|---|---|---|---|---|---|---|---|---|
| **13** | LP001029 | Male | No | 0.0 | Graduate | No | 1853 | 284 |
| **14** | LP001030 | Male | Yes | 2.0 | Graduate | No | 1299 | 108 |
| **15** | LP001032 | Male | No | 0.0 | Graduate | No | 4950 | 0.0 |
| **16** | LP001034 | Male | No | 1.0 | Not Graduate | No | 3596 | 0.0 |
| **17** | LP001036 | Female | No | 0.0 | Graduate | No | 3510 | 0.0 |
| **18** | LP001038 | Male | Yes | 0.0 | Not Graduate | No | 4887 | 0.0 |
| **19** | LP001041 | Male | Yes | 0.0 | Graduate | NaN | 2600 | 350 |

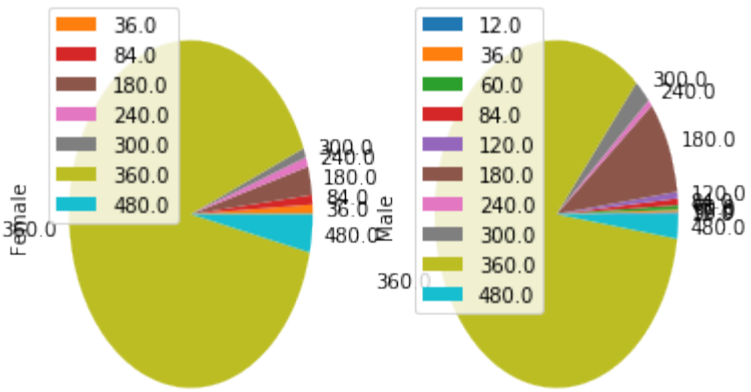
In [4]:
```
df.describe()
```

Out[4]:

| | Dependents | ApplicantIncome | CoapplicantIncome | LoanAmount | Loan_Amount_Term |
|---|---|---|---|---|---|
| **count** | 609.000000 | 624.000000 | 624.000000 | 601.000000 | 610.000000 |
| **mean** | 0.766831 | 5400.620192 | 1611.131282 | 146.911814 | 341.704918 |
| **std** | 1.015416 | 6070.897352 | 2908.374317 | 86.445959 | 65.279328 |
| **min** | 0.000000 | 150.000000 | 0.000000 | 9.000000 | 12.000000 |
| **25%** | 0.000000 | 2887.250000 | 0.000000 | 100.000000 | 360.000000 |
| **50%** | 0.000000 | 3813.500000 | 1149.000000 | 128.000000 | 360.000000 |
| **75%** | 2.000000 | 5803.750000 | 2287.750000 | 168.000000 | 360.000000 |
| **max** | 3.000000 | 81000.000000 | 41667.000000 | 700.000000 | 480.000000 |

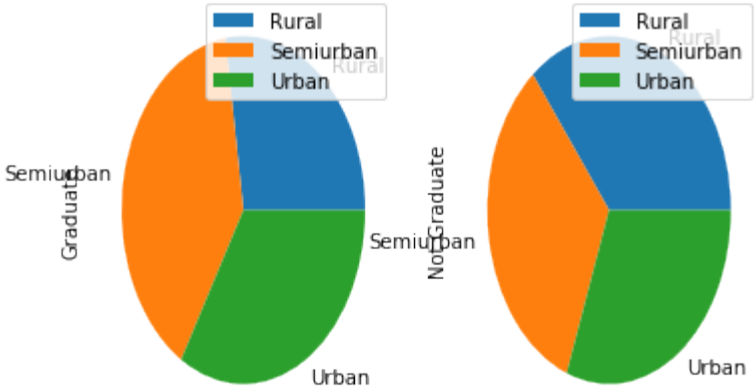
In [5]:
```
temp1 = pd.crosstab(df['Loan_Amount_Term'], df['Gender'])
temp1.plot(kind='pie', subplots=True, grid=False)
```

Out[5]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x0000000009144828>
,
       <matplotlib.axes._subplots.AxesSubplot object at 0x00000000094DF860>
],
      dtype=object)
```



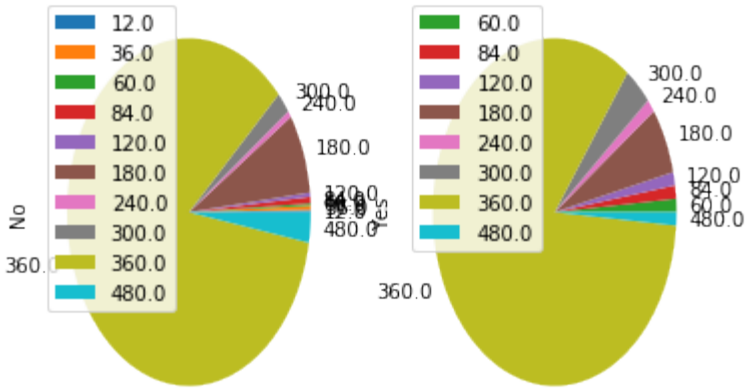
In [6]:
```
temp2 = pd.crosstab(df['Property_Area'], df['Education'])
temp2.plot(kind='pie', subplots=True, grid=False)
```

Out[6]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x00000000094C75F8>
,
       <matplotlib.axes._subplots.AxesSubplot object at 0x000000000957F128>
],
      dtype=object)
```




In [7]:
```
temp2 = pd.crosstab(df['Loan_Amount_Term'], df['Self_Employed'])
temp2.plot(kind='pie', subplots=True, grid=False)
```

Out[7]:
```
array([<matplotlib.axes._subplots.AxesSubplot object at 0x0000000009558630>
,
       <matplotlib.axes._subplots.AxesSubplot object at 0x000000000965B780>
],
      dtype=object)
```

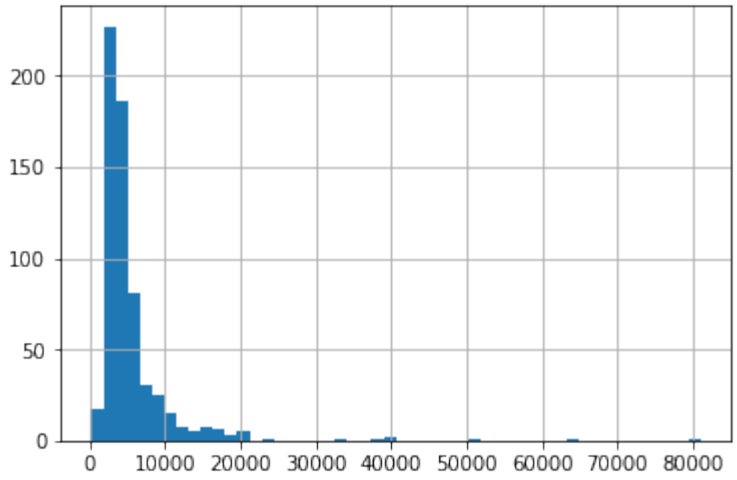



In [8]:
```
df['Property_Area'].value_counts()
```

Out[8]:
```
Semiurban    236
Urban        205
Rural        183
Name: Property_Area, dtype: int64
```

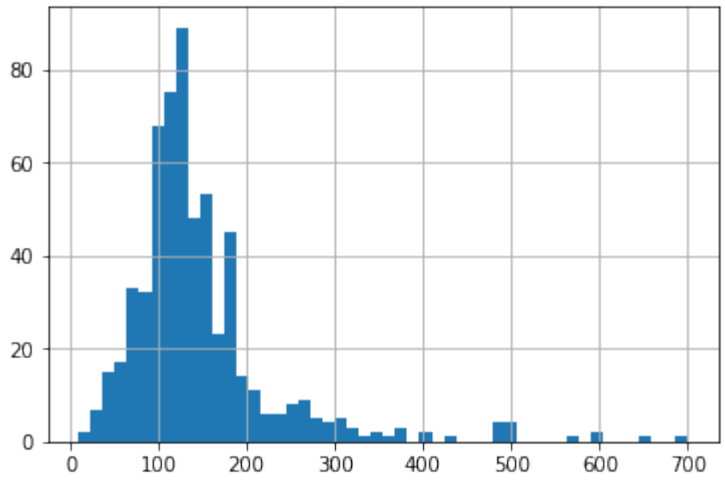
In [83]:
```
df['ApplicantIncome'].hist(bins=50)
```

Out[83]: <matplotlib.axes._subplots.AxesSubplot at 0xf10bf98>



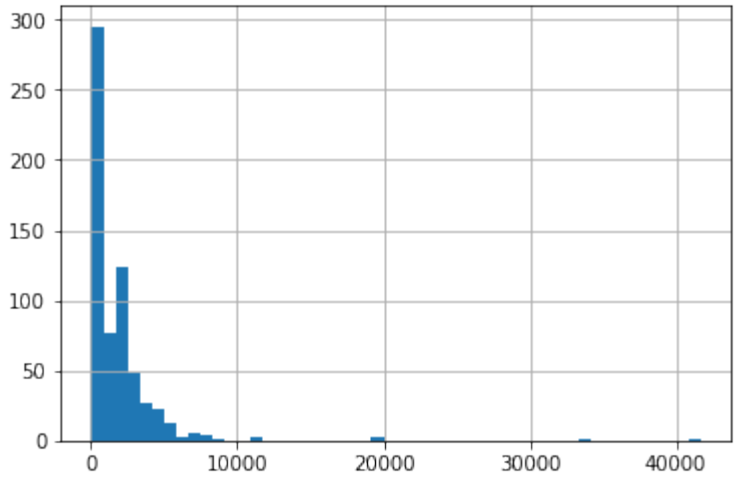

In [17]:
```
df['LoanAmount'].hist(bins=50)
```

Out[17]: <matplotlib.axes._subplots.AxesSubplot at 0xac1f4a8>



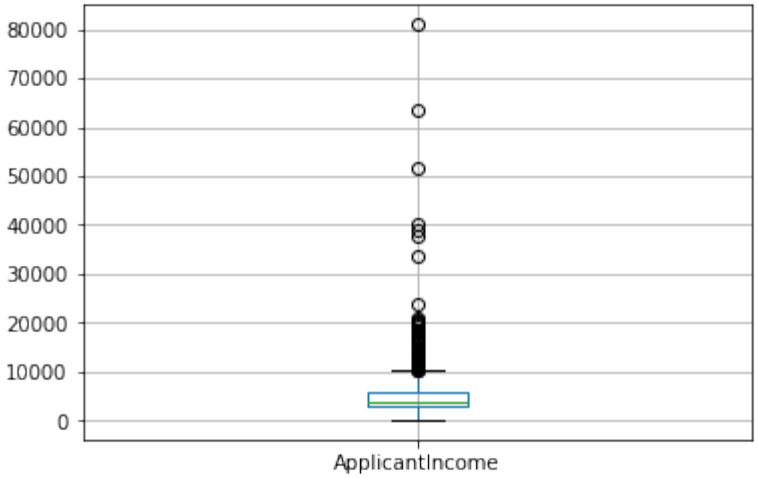

In [18]:
```
df['CoapplicantIncome'].hist(bins=50)
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0xac8f400>



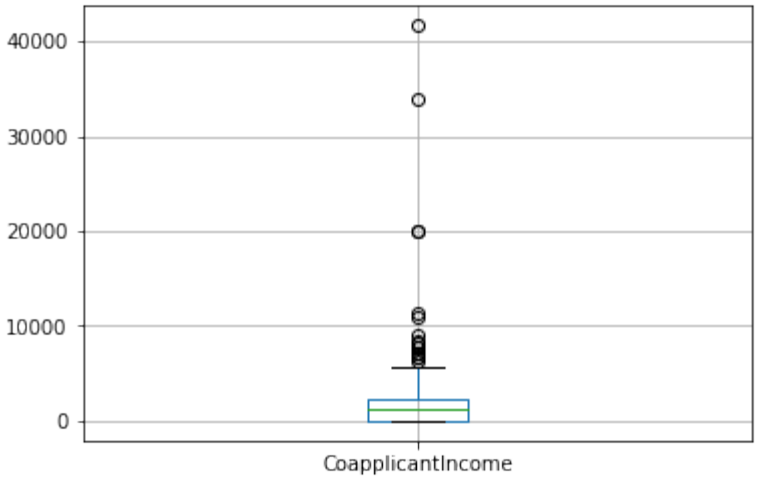
In [84]:
```
df.boxplot(column='ApplicantIncome')
```

Out[84]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xf0f6f60>
```



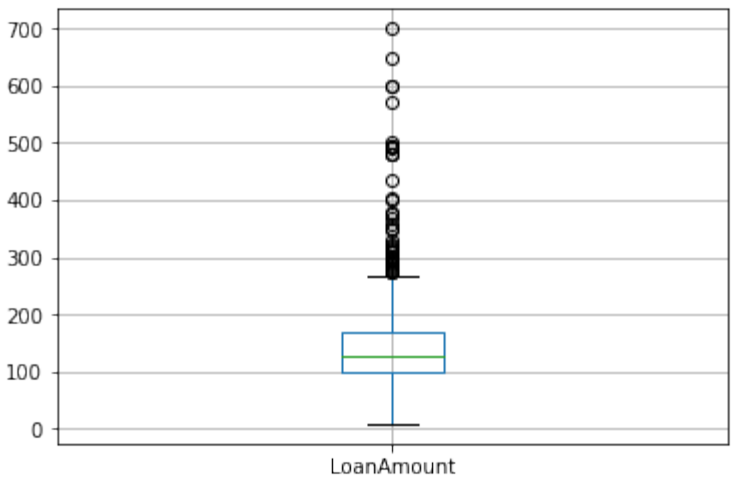

In [16]:
```
df.boxplot(column='CoapplicantIncome')
```

Out[16]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xab0d588>
```




In [13]:
```
df.boxplot(column='LoanAmount')
```

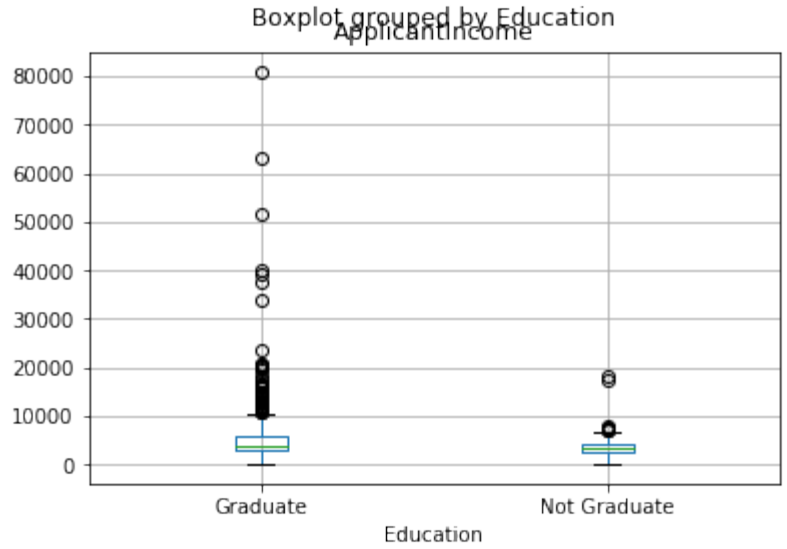
Out[13]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xa9e41d0>
```



In [85]:
```
df.boxplot(column='ApplicantIncome', by = 'Education')
```

```
C:\ProgramData\Anaconda3\lib\site-packages\numpy\core\fromnumeric.py:57: Fu
tureWarning: reshape is deprecated and will raise in a subsequent release.
Please use .values.reshape(...) instead
  return getattr(obj, method)(*args, **kwds)
```

Out[85]:
```
<matplotlib.axes._subplots.AxesSubplot at 0xf220f98>
```



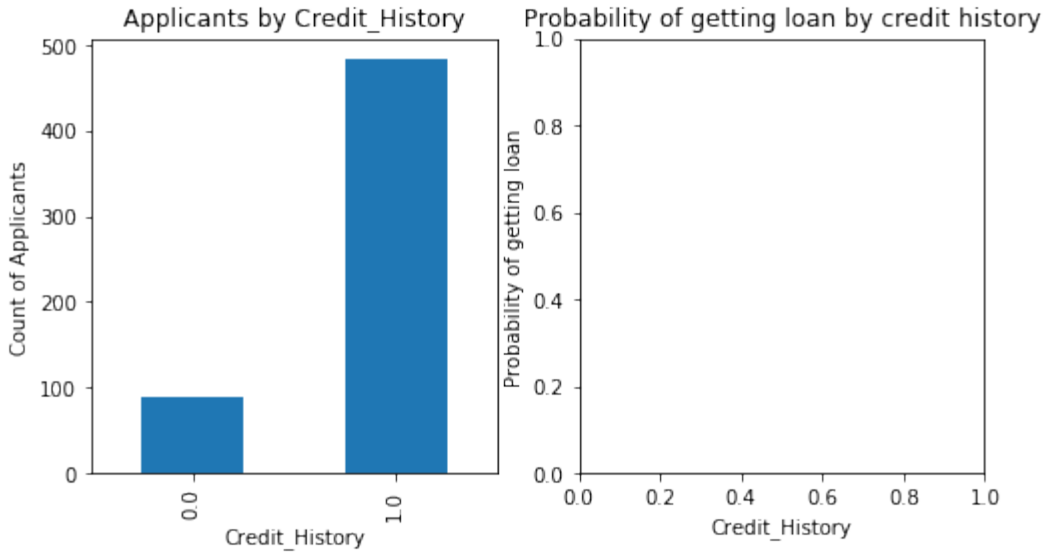

In [88]:
```
temp1 = df['Credit_History'].value_counts(ascending=True)
temp2 = df.pivot_table(values='Loan_Status',index=['Credit_History'],aggfu
nc=lambda x: x.map({'Y':1,'N':0}).mean())
print ('Frequency Table for Credit History:')
print (temp1)

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

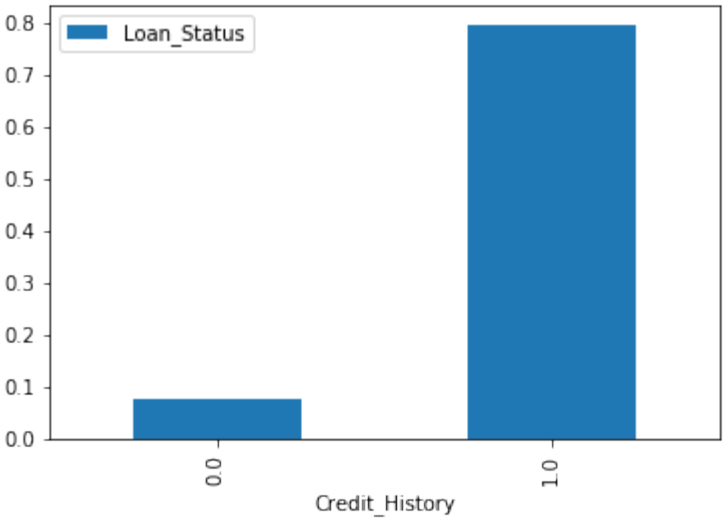
```
Frequency Table for Credit History:
0.0     90
1.0    484
Name: Credit_History, dtype: int64

Probility of getting loan for each Credit History class:
                Loan_Status
Credit_History
0.0                0.077778
1.0                0.797521
```

In [89]:
```
import matplotlib.pyplot as plt
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("Probability of getting loan by credit history")
```

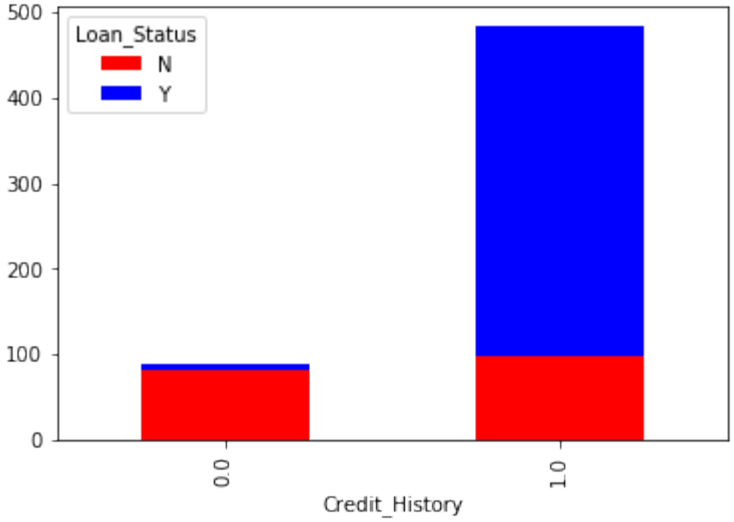
Out[89]: Text(0.5,1,'Probability of getting loan by credit history')



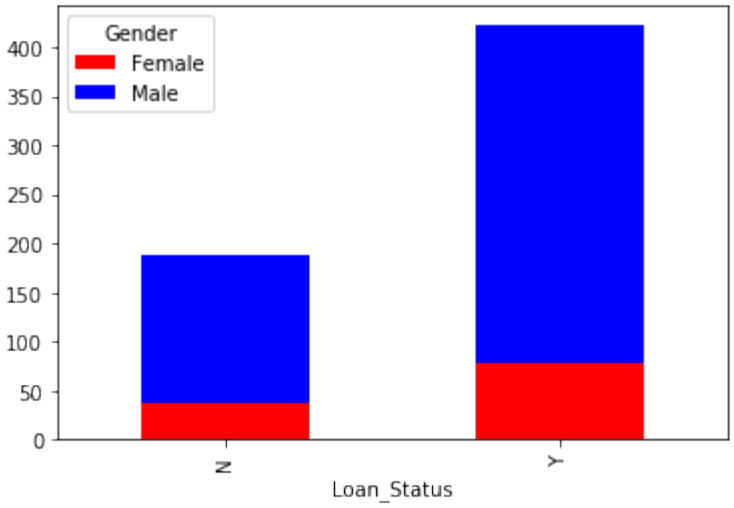

In [90]:
```
temp3 = pd.crosstab(df['Credit_History'], df['Loan_Status'])
temp3.plot(kind='bar', stacked=True, color=['red','blue'], grid=False)
```

Out[90]: <matplotlib.axes._subplots.AxesSubplot at 0xc1c40b8>




In [18]:
```
temp3 = pd.crosstab(df['Loan_Status'], df['Gender'])
temp3.plot(kind='bar', stacked=True, color=['red','blue'], grid=False)
```

Out[18]: <matplotlib.axes._subplots.AxesSubplot at 0x10be6eb8>




In [91]:
```
df.apply(lambda x: sum(x.isnull()),axis=0)
```

Out[91]:
```
Loan_ID               0
Gender               13
Married               3
Dependents           15
Education             1
Self_Employed        32
ApplicantIncome       0
CoapplicantIncome     0
LoanAmount           23
Loan_Amount_Term     14
Credit_History       50
Property_Area         0
Loan_Status           0
dtype: int64
```

In [92]:
```
df['Self_Employed'].fillna('No',inplace=True)
```

In [93]:
```
df['Gender'].fillna('Male',inplace=True)
```

In [94]:
```
df['Married'].fillna('No',inplace=True)
```

In [95]:
```
df['Dependents'].fillna(4,inplace=True)
```

In [96]:
```
df['Education'].fillna('Graduate',inplace=True)
```

In [98]:
```
df['LoanAmount'].fillna(df['LoanAmount'].mean(),inplace=True)
```

In [99]:
```
df['Loan_Amount_Term'].fillna(360.0,inplace=True)
```

In [100]:
```
df['Credit_History'].fillna(1.0,inplace=True)
```

In [101]:
```
df.apply(lambda x: sum(x.isnull()),axis=0)
```

Out[101]:
```
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
dtype: int64
```

In [102]:
```
df.dtypes
```

Out[102]:
```
Loan_ID               object
Gender                object
Married               object
Dependents           float64
Education             object
Self_Employed         object
ApplicantIncome        int64
CoapplicantIncome    float64
LoanAmount           float64
Loan_Amount_Term     float64
Credit_History       float64
Property_Area         object
Loan_Status           object
dtype: object
```

In [103]:
```
from sklearn.preprocessing import LabelEncoder
var_mod = ['Gender', 'Married']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i])
```

```
df.dtypes
```

Out[103]:
```
Loan_ID              object
Gender                int64
Married               int64
Dependents          float64
Education            object
Self_Employed        object
ApplicantIncome       int64
CoapplicantIncome   float64
LoanAmount          float64
Loan_Amount_Term    float64
Credit_History      float64
Property_Area        object
Loan_Status          object
dtype: object
```

In [104]:
```
from sklearn.preprocessing import LabelEncoder
var_mod = ['Loan_ID']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i])
df.dtypes
```

Out[104]:
```
Loan_ID               int64
Gender                int64
Married               int64
Dependents          float64
Education            object
Self_Employed        object
ApplicantIncome       int64
CoapplicantIncome   float64
LoanAmount          float64
Loan_Amount_Term    float64
Credit_History      float64
Property_Area        object
Loan_Status          object
dtype: object
```

In [105]:
```
from sklearn.preprocessing import LabelEncoder
var_mod = ['Gender', 'Married', 'Dependents']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i])
df.dtypes
```

Out[105]:
```
Loan_ID               int64
Gender                int64
Married               int64
Dependents            int64
Education            object
Self_Employed        object
ApplicantIncome       int64
CoapplicantIncome   float64
LoanAmount          float64
Loan_Amount_Term    float64
Credit_History      float64
```

```
Property_Area         object
Loan_Status           object
dtype: object
```

In [113]:
```
from sklearn.preprocessing import LabelEncoder
var_mod = ['Education', 'Self_Employed', 'CoapplicantIncome', 'LoanAmount'
, 'Loan_Amount_Term','Credit_History','Property_Area','Loan_Status']
le = LabelEncoder()
for i in var_mod:
    df[i] = le.fit_transform(df[i])
df.dtypes
```

Out[113]:
```
Loan_ID              int64
Gender               int64
Married              int64
Dependents           int64
Education            int64
Self_Employed        int64
ApplicantIncome      int64
CoapplicantIncome    int64
LoanAmount           int64
Loan_Amount_Term     int64
Credit_History       int64
Property_Area        int64
Loan_Status          int64
dtype: object
```

In [119]:
```
#Import models from scikit learn module:
from sklearn.linear_model import LogisticRegression
from sklearn.cross_validation import KFold   #For K-fold cross validation
from sklearn.ensemble import RandomForestClassifier
from sklearn.tree import DecisionTreeClassifier, export_graphviz
from sklearn import metrics

#Generic function for making a classification model and accessing performan
ce:
def classification_model(model, data, predictors, outcome):
  #Fit the model:
  model.fit(data[predictors],data[outcome])

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

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

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

    # The target we're using to train the algorithm.
    train_target = data[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(data[predictors].iloc[test,:], data[outcome].
iloc[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(data[predictors],data[outcome])
```

In [116]:
```
outcome_var = 'Loan_Status'
model = LogisticRegression()
predictor_var = ['Credit_History']
classification_model(model, df,predictor_var,outcome_var)
```

```
Accuracy : 81.090%
Cross-Validation Score : 81.090%
```

In [118]:
```
outcome_var = 'Loan_Status'
model = LogisticRegression()
predictor_var = ['Credit_History','Dependents','LoanAmount']
classification_model(model, df,predictor_var,outcome_var)
```

```
Accuracy : 81.090%
Cross-Validation Score : 81.090%
```

In [120]:
```
outcome_var = 'Loan_Status'
model = LogisticRegression()
predictor_var = ['Credit_History','Dependents','LoanAmount','Loan_Amount_T
erm']
classification_model(model, df,predictor_var,outcome_var)
```

```
Accuracy : 81.090%
Cross-Validation Score : 81.090%
```

In [121]:
```
model = DecisionTreeClassifier()
predictor_var = ['Credit_History','Gender','Married','Education']
classification_model(model, df,predictor_var,outcome_var)
```

```
Accuracy : 81.090%
Cross-Validation Score : 81.090%
```

In [122]:
```
model = DecisionTreeClassifier()
predictor_var = ['Credit_History','Gender','Married','Education','LoanAmou
nt','Loan_Amount_Term']
classification_model(model, df,predictor_var,outcome_var)
```

```
Accuracy : 95.513%
Cross-Validation Score : 70.192%
```

In [125]:
```
model = DecisionTreeClassifier()
predictor_var = ['Loan_Status','CoapplicantIncome','Married','Education','
Loan_Amount_Term']
classification_model(model, df,predictor_var,outcome_var)
```

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