IMPORTING MODULES AND LOADING DATASETS

In [1]:
```
import pandas as pd
import numpy as np
import seaborn as sns
from matplotlib import pyplot as plt
import matplotlib
%matplotlib inline
import warnings
warnings.filterwarnings('ignore')
```

In [2]:
```
df = pd.read_csv(r"C:\Users\Windows\Downloads\archive (1)\train_u6lujuX_CVtuZ9i.c
df.head()
```

Out[2]:

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

In [3]:
```
df.describe()
```

Out[3]:

	ApplicantIncome	CoapplicantIncome	LoanAmount	Loan_Amount_Term	Credit_History
count	614.000000	614.000000	592.000000	600.00000	564.000000
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 [4]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 614 entries, 0 to 613
Data columns (total 13 columns):
 #   Column             Non-Null Count  Dtype  
---  ------             --------------  -----  
 0   Loan_ID            614 non-null    object 
 1   Gender             601 non-null    object 
 2   Married            611 non-null    object 
 3   Dependents         599 non-null    object 
 4   Education          614 non-null    object 
 5   Self_Employed      582 non-null    object 
 6   ApplicantIncome    614 non-null    int64  
 7   CoapplicantIncome  614 non-null    float64
 8   LoanAmount         592 non-null    float64
 9   Loan_Amount_Term   600 non-null    float64
 10  Credit_History     564 non-null    float64
 11  Property_Area      614 non-null    object 
 12  Loan_Status        614 non-null    object 
dtypes: float64(4), int64(1), object(8)
memory usage: 62.5+ KB
```

PREPROCESSING THE DATASET

In [5]:
```
# find the null values
df.isnull().sum()
```

Out[5]:
```
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 [6]:
```
# fill the missing values for numerical terms - mean
df['LoanAmount'] = df['LoanAmount'].fillna(df['LoanAmount'].mean())
df['Loan_Amount_Term'] = df['Loan_Amount_Term'].fillna(df['Loan_Amount_Term'].mea
df['Credit_History'] = df['Credit_History'].fillna(df['Credit_History'].mean())
```

In [7]:
```
# fill the missing values for categorical terms - mode
df['Gender'] = df["Gender"].fillna(df['Gender'].mode()[0])
df['Married'] = df["Married"].fillna(df['Married'].mode()[0])
df['Dependents'] = df["Dependents"].fillna(df['Dependents'].mode()[0])
df['Self_Employed'] = df["Self_Employed"].fillna(df['Self_Employed'].mode()[0])
```

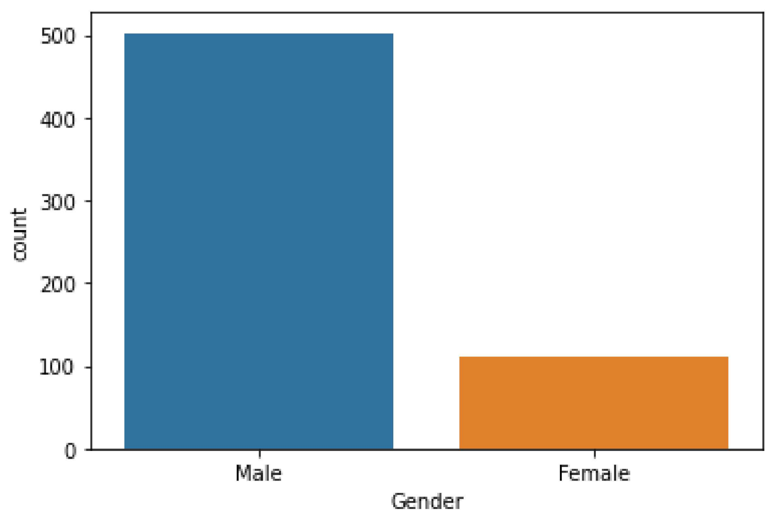
In [8]:
```
df.isnull().sum()
```

Out[8]:
```
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
```

EXPLORATORY DATA ANALYSIS

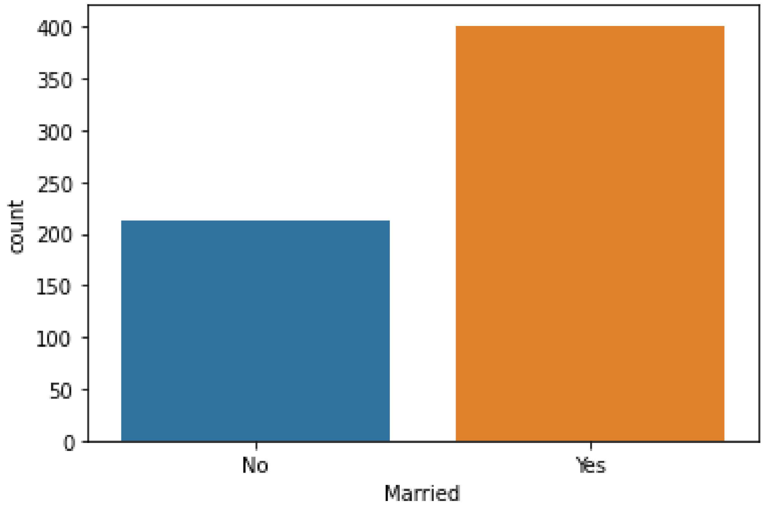
In [9]:
```
# categorical attributes visualization
sns.countplot(df['Gender'])
```

Out[9]:
```
<AxesSubplot:xlabel='Gender', ylabel='count'>
```



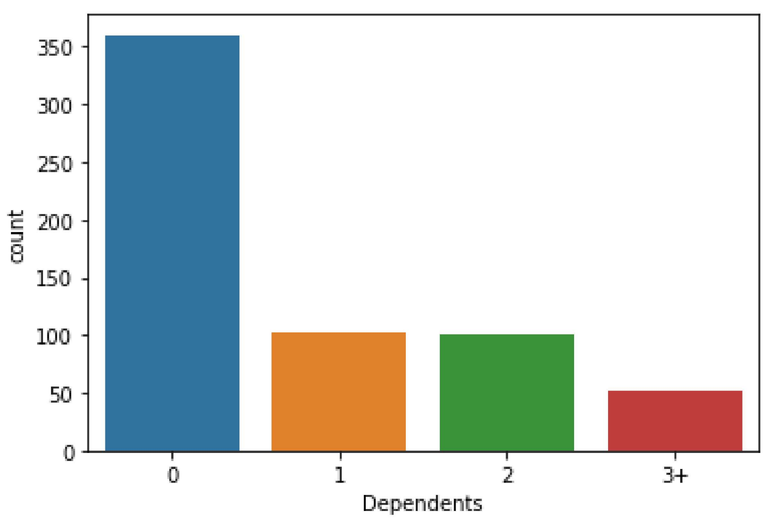
In [10]:
```
sns.countplot(df['Married'])
```

Out[10]: <AxesSubplot:xlabel='Married', ylabel='count'>



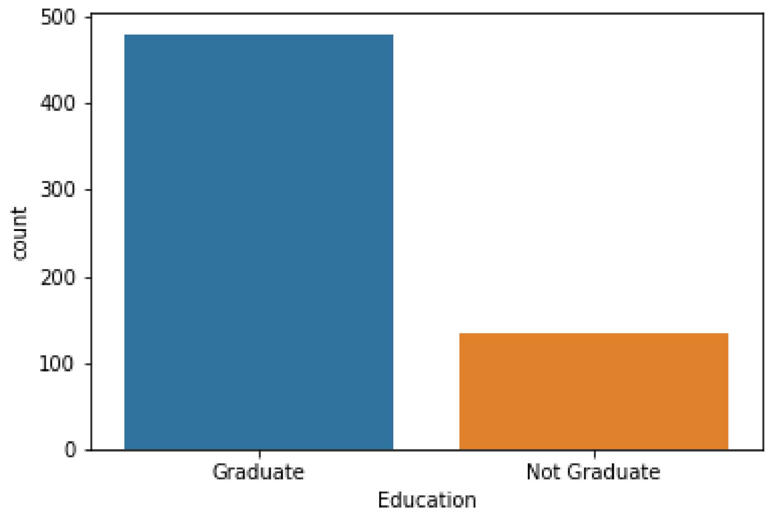
In [11]:
```
sns.countplot(df['Dependents'])
```

Out[11]: <AxesSubplot:xlabel='Dependents', ylabel='count'>



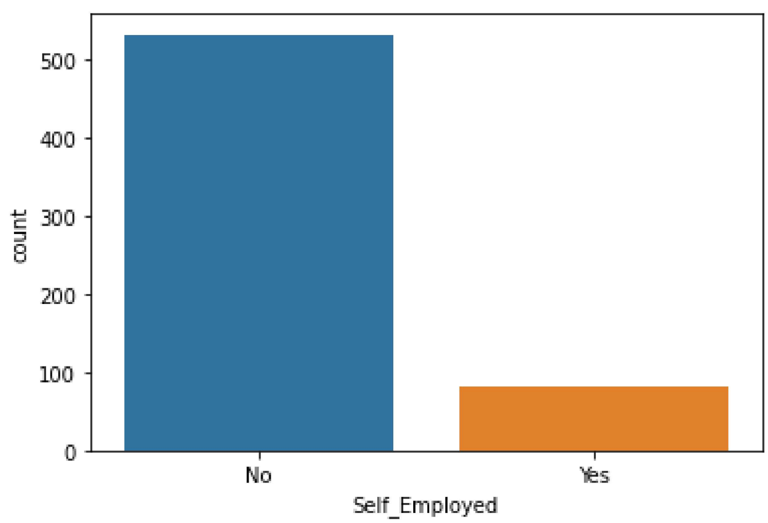
In [12]:
```
sns.countplot(df['Education'])
```

Out[12]: <AxesSubplot:xlabel='Education', ylabel='count'>



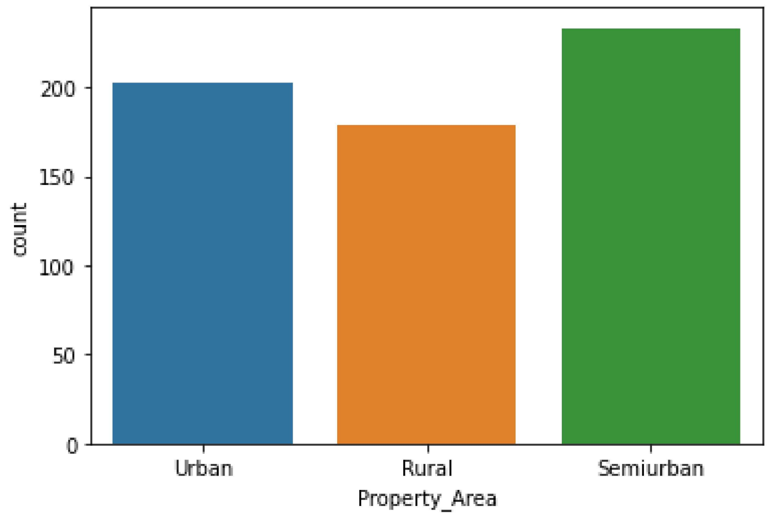
In [13]:
```
sns.countplot(df['Self_Employed'])
```

Out[13]: <AxesSubplot:xlabel='Self_Employed', ylabel='count'>



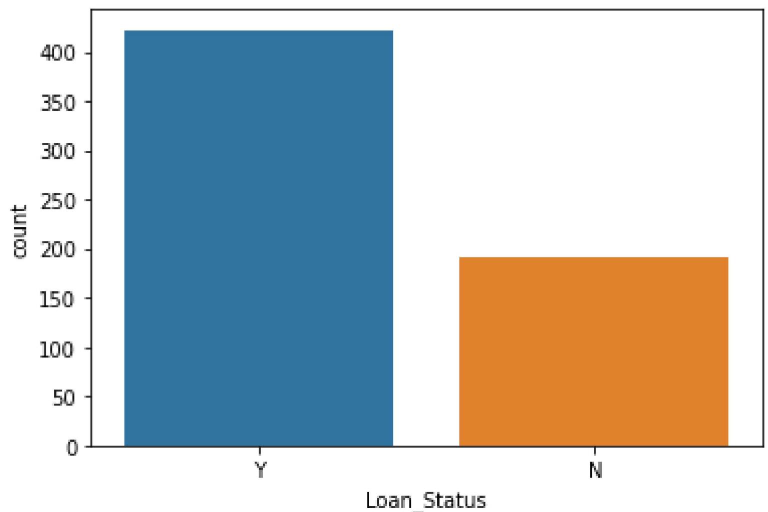
In [14]:
```
sns.countplot(df['Property_Area'])
```

Out[14]: <AxesSubplot:xlabel='Property_Area', ylabel='count'>



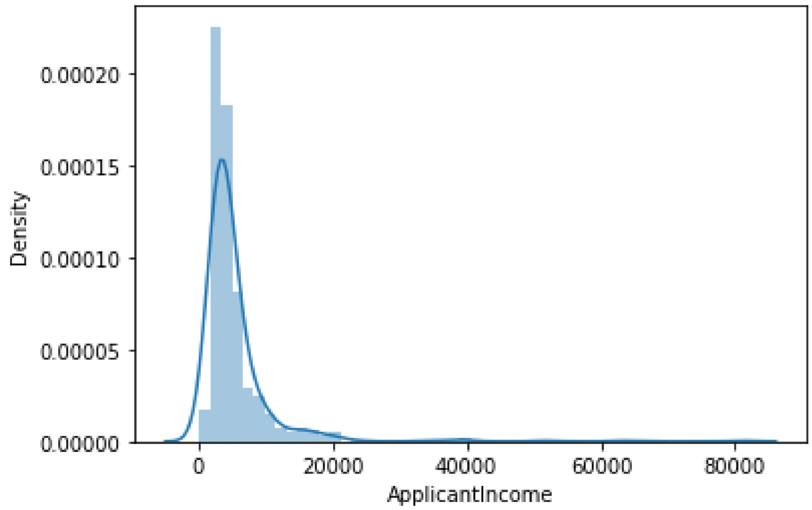
In [15]:
```
sns.countplot(df['Loan_Status'])
```

Out[15]: <AxesSubplot:xlabel='Loan_Status', ylabel='count'>



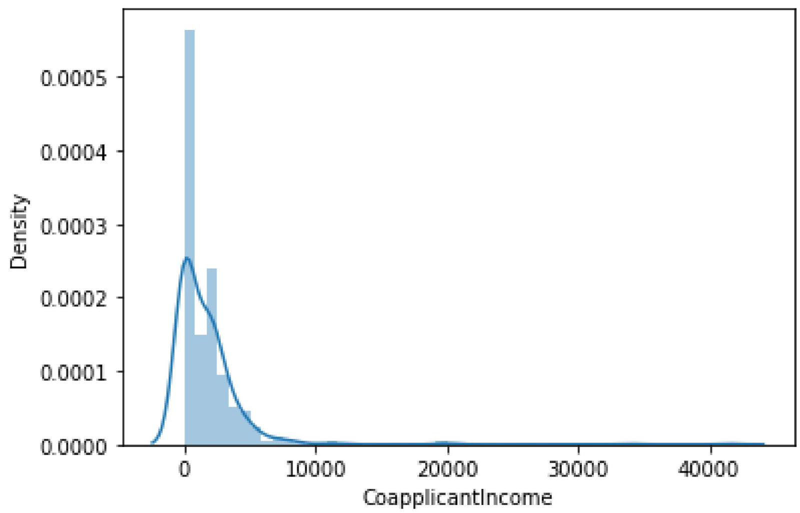
In [16]:
```
# numerical attributes visualization
sns.distplot(df["ApplicantIncome"])
```

Out[16]: <AxesSubplot:xlabel='ApplicantIncome', ylabel='Density'>



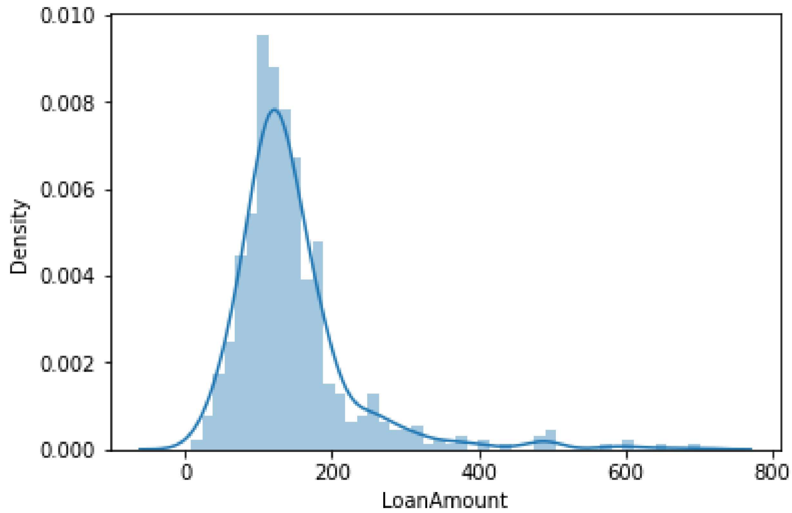
In [17]:
```
sns.distplot(df["CoapplicantIncome"])
```

Out[17]: <AxesSubplot:xlabel='CoapplicantIncome', ylabel='Density'>



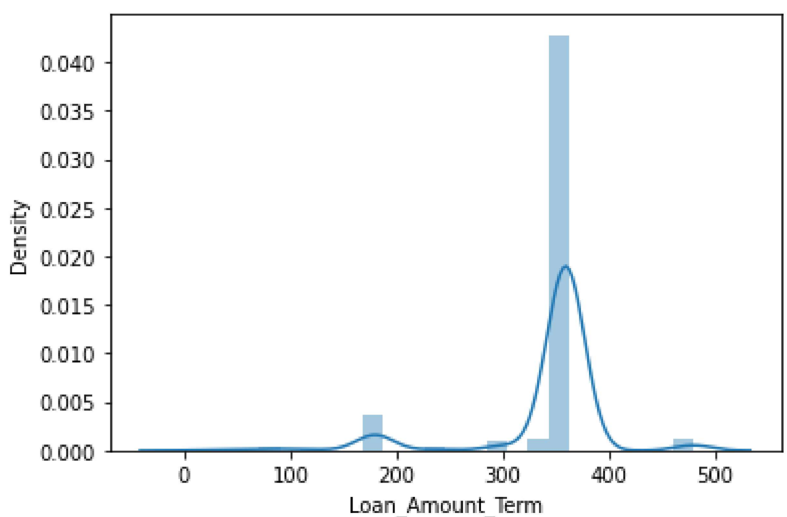
In [18]:
```
sns.distplot(df["LoanAmount"])
```

Out[18]: <AxesSubplot:xlabel='LoanAmount', ylabel='Density'>



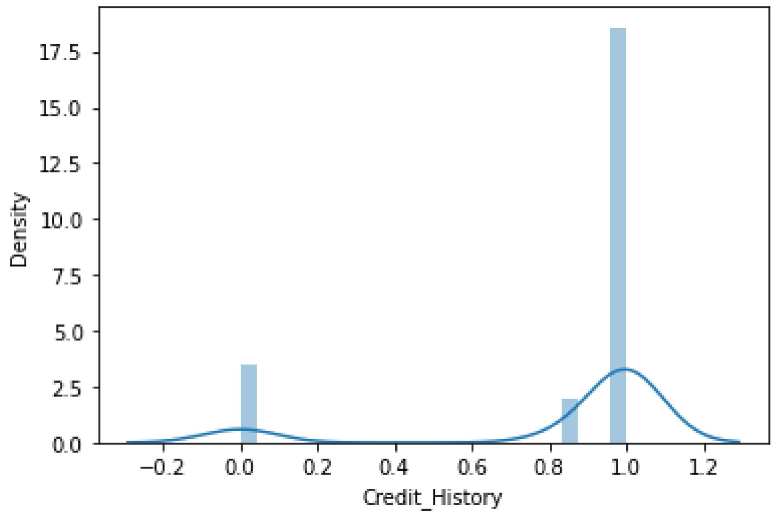
In [19]:
```
sns.distplot(df['Loan_Amount_Term'])
```

Out[19]: <AxesSubplot:xlabel='Loan_Amount_Term', ylabel='Density'>



In [20]:
```
sns.distplot(df['Credit_History'])
```

Out[20]: <AxesSubplot:xlabel='Credit_History', ylabel='Density'>



CREATION OF NEW ATTRIBUTES

In [21]:
```
# total income
df['Total_Income'] = df['ApplicantIncome'] + df['CoapplicantIncome']
df.head()
```

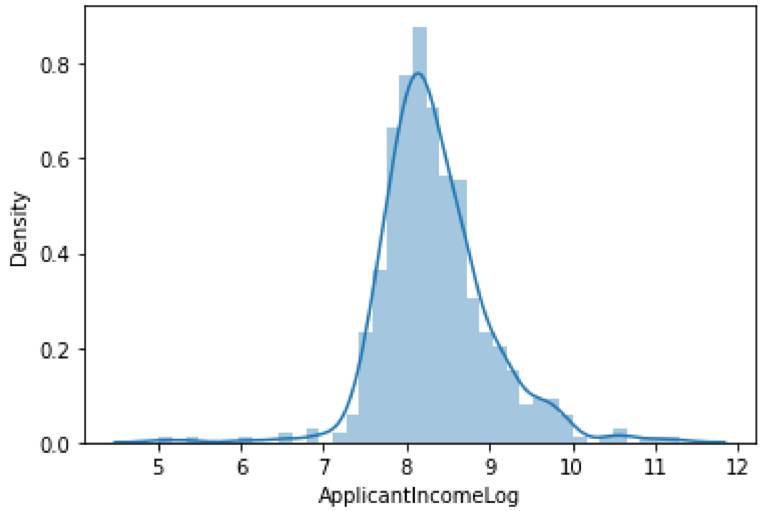
Out[21]:

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

LOG TRANSFORMATION

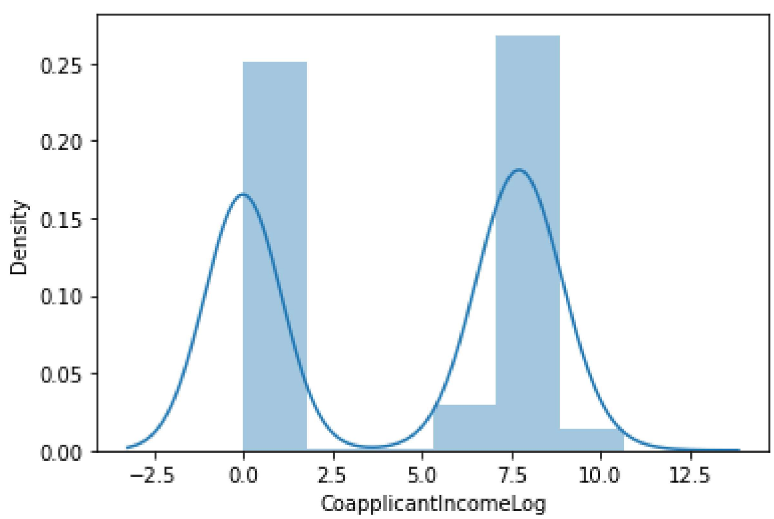
In [22]:
```
# apply log transformation to the attribute
df['ApplicantIncomeLog'] = np.log(df['ApplicantIncome']+1)
sns.distplot(df["ApplicantIncomeLog"])
```

Out[22]: <AxesSubplot:xlabel='ApplicantIncomeLog', ylabel='Density'>



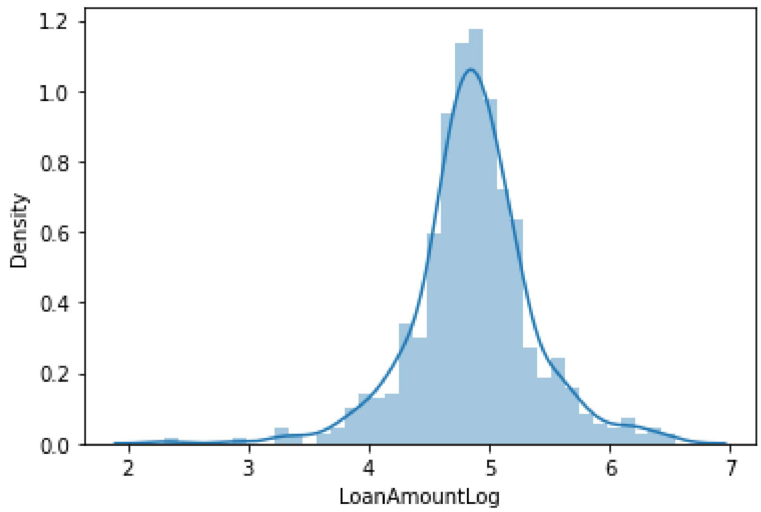
In [23]:
```
df['CoapplicantIncomeLog'] = np.log(df['CoapplicantIncome']+1)
sns.distplot(df["CoapplicantIncomeLog"])
```

Out[23]: <AxesSubplot:xlabel='CoapplicantIncomeLog', ylabel='Density'>



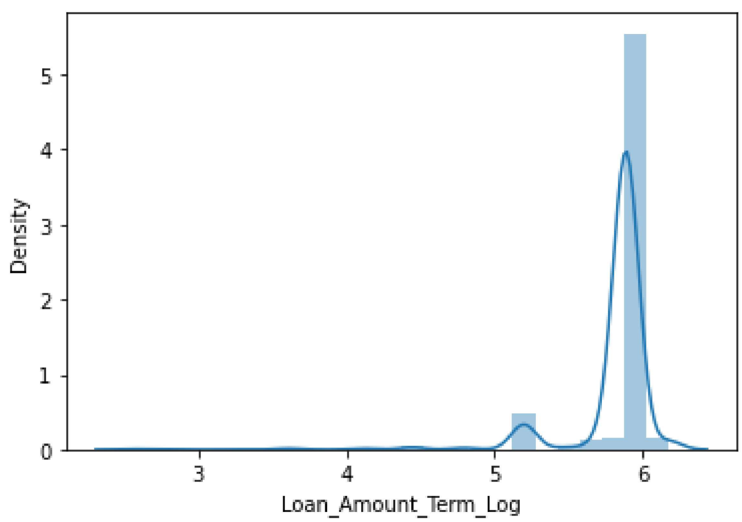
In [24]:
```
df['LoanAmountLog'] = np.log(df['LoanAmount']+1)
sns.distplot(df["LoanAmountLog"])
```

Out[24]: <AxesSubplot:xlabel='LoanAmountLog', ylabel='Density'>



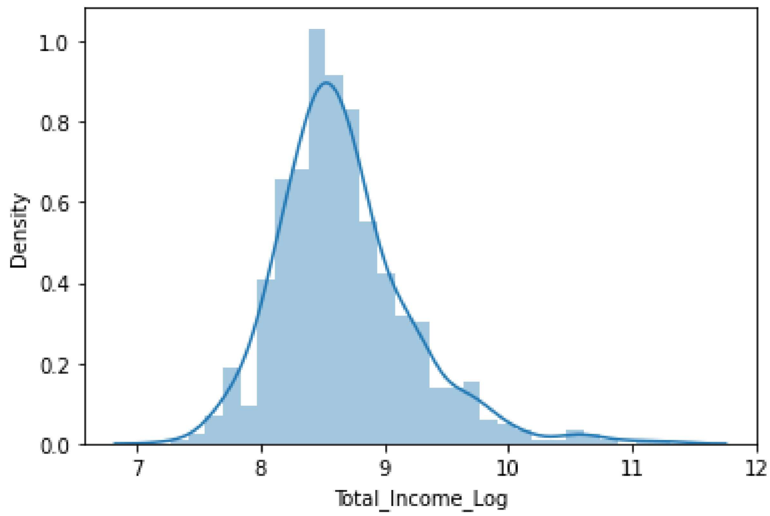
In [25]:
```
df['Loan_Amount_Term_Log'] = np.log(df['Loan_Amount_Term']+1)
sns.distplot(df["Loan_Amount_Term_Log"])
```

Out[25]: <AxesSubplot:xlabel='Loan_Amount_Term_Log', ylabel='Density'>



In [26]:
```
df['Total_Income_Log'] = np.log(df['Total_Income']+1)
sns.distplot(df["Total_Income_Log"])
```

Out[26]: <AxesSubplot:xlabel='Total_Income_Log', ylabel='Density'>



COORELATION MATRIX

In [27]:
```
corr = df.corr()
plt.figure(figsize=(15,10))
sns.heatmap(corr, annot = True, cmap="BuPu")
```

Out[27]: <AxesSubplot:>

In [28]:
```
df.head()
```

Out[28]:

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

In [29]:
```
# drop unnecessary columns
cols = ['ApplicantIncome', 'CoapplicantIncome', "LoanAmount", "Loan_Amount_Term",
df = df.drop(columns=cols, axis=1)
df.head()
```

Out[29]:

	Gender	Married	Dependents	Education	Self_Employed	Credit_History	Property_Area	Loan_S
0	Male	No	0	Graduate	No	1.0	Urban	
1	Male	Yes	1	Graduate	No	1.0	Rural	
2	Male	Yes	0	Graduate	Yes	1.0	Urban	
3	Male	Yes	0	Not Graduate	No	1.0	Urban	
4	Male	No	0	Graduate	No	1.0	Urban	

LABEL ENCODING

In [30]:
```
from sklearn.preprocessing import LabelEncoder
cols = ['Gender',"Married","Education",'Self_Employed',"Property_Area","Loan_Stat
le = LabelEncoder()
for col in cols:
    df[col] = le.fit_transform(df[col])
```

In [31]:
```
df.head()
```

Out[31]:

	Gender	Married	Dependents	Education	Self_Employed	Credit_History	Property_Area	Loan_S
0	1	0	0	0	0	1.0	2	
1	1	1	1	0	0	1.0	0	
2	1	1	0	0	1	1.0	2	
3	1	1	0	1	0	1.0	2	
4	1	0	0	0	0	1.0	2	

TRAIN-TEST SPLIT

In [32]:
```
# specify input and output attributes
X = df.drop(columns=['Loan_Status'], axis=1)
y = df['Loan_Status']
```

In [33]:
```
from sklearn.model_selection import train_test_split
x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.25, random_
```

MODEL TRAINING

In [36]:
```
# classify function
from sklearn.model_selection import cross_val_score
def classify(model, x, y):
    x_train, x_test, y_train, y_test = train_test_split(X, y, test_size=0.25, rar
    model.fit(x_train, y_train)
    print("Accuracy is", model.score(x_test, y_test)*100)
    # cross validation - it is used for better validation of model
    # eg: cv-5, train-4, test-1
    score = cross_val_score(model, x, y, cv=5)
    print("Cross validation is",np.mean(score)*100)
```

In [37]:
```
from sklearn.linear_model import LogisticRegression
model = LogisticRegression()
classify(model, X, y)
```

```
Accuracy is 77.27272727272727
Cross validation is 80.9462881514061
```

In [38]:
```
from sklearn.tree import DecisionTreeClassifier
model = DecisionTreeClassifier()
classify(model, X, y)
```

```
Accuracy is 71.42857142857143
Cross validation is 71.8286018925763
```

In [39]:
```
from sklearn.ensemble import RandomForestClassifier,ExtraTreesClassifier
model = RandomForestClassifier()
classify(model, X, y)
```

```
Accuracy is 79.22077922077922
Cross validation is 78.17672930827668
```

In [40]:
```
model = ExtraTreesClassifier()
classify(model, X, y)
```

```
Accuracy is 74.67532467532467
Cross validation is 76.22417699586832
```