In [1]:

```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
%matplotlib inline
import sklearn.metrics
```

In [2]:

```
df=pd.read_csv("C:\\heart disease dataset (1).csv")
```

In [3]:

```
df
```

Out[3]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentH
0	1	39	4.0	0	0.0	0.0	0	
1	0	46	2.0	0	0.0	0.0	0	
2	1	48	1.0	1	20.0	0.0	0	
3	0	61	3.0	1	30.0	0.0	0	
4	0	46	3.0	1	23.0	0.0	0	
...	...	...	...	...	...	...	...	
4233	1	50	1.0	1	1.0	0.0	0	
4234	1	51	3.0	1	43.0	0.0	0	
4235	0	48	2.0	1	20.0	NaN	0	
4236	0	44	1.0	1	15.0	0.0	0	
4237	0	52	2.0	0	0.0	0.0	0	

4238 rows × 16 columns

In [4]:

```
df.head()
```

Out[4]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentHyp
0	1	39	4.0	0	0.0	0.0	0	0
1	0	46	2.0	0	0.0	0.0	0	0
2	1	48	1.0	1	20.0	0.0	0	0
3	0	61	3.0	1	30.0	0.0	0	1
4	0	46	3.0	1	23.0	0.0	0	0

In [5]:

```
df.tail()
```

Out[5]:

	male	age	education	currentSmoker	cigsPerDay	BPMeds	prevalentStroke	prevalentH
4233	1	50	1.0	1	1.0	0.0	0	
4234	1	51	3.0	1	43.0	0.0	0	
4235	0	48	2.0	1	20.0	NaN	0	
4236	0	44	1.0	1	15.0	0.0	0	
4237	0	52	2.0	0	0.0	0.0	0	

In [6]:

```
df.isnull
```

Out[6]:

```
<bound method DataFrame.isnull of       male  age  education  currentSmoke
r  cigsPerDay  BPMeds  \
0        1   39        4.0             0         0.0     0.0
1        0   46        2.0             0         0.0     0.0
2        1   48        1.0             1        20.0     0.0
3        0   61        3.0             1        30.0     0.0
4        0   46        3.0             1        23.0     0.0
...    ...  ...        ...           ...         ...     ...
4233     1   50        1.0             1         1.0     0.0
4234     1   51        3.0             1        43.0     0.0
4235     0   48        2.0             1        20.0     NaN
4236     0   44        1.0             1        15.0     0.0
4237     0   52        2.0             0         0.0     0.0

      prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BM
I  \
0                   0             0         0    195.0  106.0   70.0  26.9
7
1                   0             0         0    250.0  121.0   81.0  28.7
3
2                   0             0         0    245.0  127.5   80.0  25.3
4
3                   0             1         0    225.0  150.0   95.0  28.5
8
4                   0             0         0    285.0  130.0   84.0  23.1
0
...               ...           ...       ...      ...    ...    ...
...
4233                0             1         0    313.0  179.0   92.0  25.9
7
4234                0             0         0    207.0  126.5   80.0  19.7
1
4235                0             0         0    248.0  131.0   72.0  22.0
0
4236                0             0         0    210.0  126.5   87.0  19.1
6
4237                0             0         0    269.0  133.5   83.0  21.4
7

      heartRate  glucose  TenYearCHD
0          80.0     77.0           0
1          95.0     76.0           0
2          75.0     70.0           0
3          65.0    103.0           1
4          85.0     85.0           0
...         ...      ...         ...
4233       66.0     86.0           1
4234       65.0     68.0           0
4235       84.0     86.0           0
4236       86.0      NaN           0
4237       80.0    107.0           0

[4238 rows x 16 columns]>
```

In [8]:

```
df.shape
```

Out[8]:

```
(4238, 16)
```

In [9]:

```
df.describe
```

Out[9]:

```
<bound method NDFrame.describe of       male  age  education  currentSmoke
r  cigsPerDay  BPMeds  \
0        1   39        4.0             0         0.0     0.0
1        0   46        2.0             0         0.0     0.0
2        1   48        1.0             1        20.0     0.0
3        0   61        3.0             1        30.0     0.0
4        0   46        3.0             1        23.0     0.0
...    ...  ...        ...           ...         ...     ...
4233     1   50        1.0             1         1.0     0.0
4234     1   51        3.0             1        43.0     0.0
4235     0   48        2.0             1        20.0     NaN
4236     0   44        1.0             1        15.0     0.0
4237     0   52        2.0             0         0.0     0.0

      prevalentStroke  prevalentHyp  diabetes  totChol  sysBP  diaBP    BM
I  \
0                   0             0         0    195.0  106.0   70.0  26.9
7
1                   0             0         0    250.0  121.0   81.0  28.7
3
2                   0             0         0    245.0  127.5   80.0  25.3
4
3                   0             1         0    225.0  150.0   95.0  28.5
8
4                   0             0         0    285.0  130.0   84.0  23.1
0
...               ...           ...       ...      ...    ...    ...
...
4233                0             1         0    313.0  179.0   92.0  25.9
7
4234                0             0         0    207.0  126.5   80.0  19.7
1
4235                0             0         0    248.0  131.0   72.0  22.0
0
4236                0             0         0    210.0  126.5   87.0  19.1
6
4237                0             0         0    269.0  133.5   83.0  21.4
7

      heartRate  glucose  TenYearCHD
0          80.0     77.0           0
1          95.0     76.0           0
2          75.0     70.0           0
3          65.0    103.0           1
4          85.0     85.0           0
...         ...      ...         ...
4233       66.0     86.0           1
4234       65.0     68.0           0
4235       84.0     86.0           0
4236       86.0      NaN           0
4237       80.0    107.0           0

[4238 rows x 16 columns]>
```

In [10]:

```
x = df.iloc [:, :-1].values
```

In [11]:

```
x
```

Out[11]:

```
array([[  1.  ,  39.  ,   4.  , ...,  26.97,  80.  ,  77.  ],
       [  0.  ,  46.  ,   2.  , ...,  28.73,  95.  ,  76.  ],
       [  1.  ,  48.  ,   1.  , ...,  25.34,  75.  ,  70.  ],
       ...,
       [  0.  ,  48.  ,   2.  , ...,  22.  ,  84.  ,  86.  ],
       [  0.  ,  44.  ,   1.  , ...,  19.16,  86.  ,    nan],
       [  0.  ,  52.  ,   2.  , ...,  21.47,  80.  , 107.  ]])
```

In [12]:

```
y = df.iloc [:, :-1].values
```

In [13]:

```
y
```

Out[13]:

```
array([[  1.  ,  39.  ,   4.  , ...,  26.97,  80.  ,  77.  ],
       [  0.  ,  46.  ,   2.  , ...,  28.73,  95.  ,  76.  ],
       [  1.  ,  48.  ,   1.  , ...,  25.34,  75.  ,  70.  ],
       ...,
       [  0.  ,  48.  ,   2.  , ...,  22.  ,  84.  ,  86.  ],
       [  0.  ,  44.  ,   1.  , ...,  19.16,  86.  ,    nan],
       [  0.  ,  52.  ,   2.  , ...,  21.47,  80.  , 107.  ]])
```

In [16]:

```
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
```

In [28]:

```
x = np.arange(10).reshape(-1, 1)
x
```

Out[28]:

```
array([[0],
       [1],
       [2],
       [3],
       [4],
       [5],
       [6],
       [7],
       [8],
       [9]])
```

In [31]:

```
y = np.array([0,0,0,0,1,1,1,1,1,1])
y
```

Out[31]:

```
array([0, 0, 0, 0, 1, 1, 1, 1, 1, 1])
```

In [32]:

```
model = LogisticRegression(solver = 'liblinear', random_state = 0).fit(x,y)
```

In [33]:

```
model
```

Out[33]:

```
LogisticRegression(random_state=0, solver='liblinear')
```

In [34]:

```
model.intercept_
```

Out[34]:

```
array([-1.04608067])
```

In [35]:

```
model.predict_proba(x)
```

Out[35]:

```
array([[0.74002157, 0.25997843],
       [0.62975524, 0.37024476],
       [0.5040632 , 0.4959368 ],
       [0.37785549, 0.62214451],
       [0.26628093, 0.73371907],
       [0.17821501, 0.82178499],
       [0.11472079, 0.88527921],
       [0.07186982, 0.92813018],
       [0.04422513, 0.95577487],
       [0.02690569, 0.97309431]])
```

In [37]:

```
model.coef_
```

Out[37]:

```
array([[0.51491375]])
```

In [38]:

```
model.predict(x)
```

Out[38]:

```
array([0, 0, 0, 1, 1, 1, 1, 1, 1, 1])
```

In [39]:

```
model.score(x,y)
```

Out[39]:

```
0.9
```

In [40]:

```
confusion_matrix(y, model.predict(x))
```

Out[40]:

```
array([[3, 1],
       [0, 6]], dtype=int64)
```

In [46]:

```
# Improve the model
# regularization strength C equal to 10.0, instead of the default value of 1.0
model = LogisticRegression(solver = 'liblinear', C=10.0, random_state = 0)
model.fit(x,y)
```

Out[46]:

```
LogisticRegression(C=10.0, random_state=0, solver='liblinear')
```

In [47]:

```
print('Inercept: ', model.intercept_)
print('Coefficient: ', model.coef_)
print('Prediction Probability: ', model.predict_proba(x))
print("X: ", model.predict(x))
```

```
Inercept:  [-3.51335372]
Coefficient:  [[1.12066084]]
Prediction Probability:  [[0.97106534 0.02893466]
 [0.9162684  0.0837316 ]
 [0.7810904  0.2189096 ]
 [0.53777071 0.46222929]
 [0.27502212 0.72497788]
 [0.11007743 0.88992257]
 [0.03876835 0.96123165]
 [0.01298011 0.98701989]
 [0.0042697  0.9957303 ]
 [0.00139621 0.99860379]]
X:  [0 0 0 0 1 1 1 1 1 1]
```

In [48]:

```
#Log Reg with scikit-learn
import numpy as np
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report, confusion_matrix
```

In [49]:

```
x = np.arange(10).reshape(-1,1)
y = np.array([0,1,0,0,1,1,1,1,1,1])
print(x)
print(y)
```

```
[[0]
 [1]
 [2]
 [3]
 [4]
 [5]
 [6]
 [7]
 [8]
 [9]]
[0 1 0 0 1 1 1 1 1 1]
```

In [50]:

```
model = LogisticRegression(solver = 'liblinear', C=10.0, random_state = 0)
model.fit(x,y)
```

Out[50]:

```
LogisticRegression(C=10.0, random_state=0, solver='liblinear')
```

In [51]:

```
p_pred = model.predict_proba(x)
p_pred
```

Out[51]:

```
array([[0.81999686, 0.18000314],
       [0.69272057, 0.30727943],
       [0.52732579, 0.47267421],
       [0.35570732, 0.64429268],
       [0.21458576, 0.78541424],
       [0.11910229, 0.88089771],
       [0.06271329, 0.93728671],
       [0.03205032, 0.96794968],
       [0.0161218 , 0.9838782 ],
       [0.00804372, 0.99195628]])
```

In [52]:

```
y_pred = model.predict(x)
y_pred
```

Out[52]:

```
array([0, 0, 0, 1, 1, 1, 1, 1, 1, 1])
```

In [53]:

```
conf_m = confusion_matrix(y, y_pred)
conf_m
```

Out[53]:

```
array([[2, 1],
       [1, 6]], dtype=int64)
```

In [54]:

```
report = classification_report(y, y_pred)
report
```

Out[54]:

```
'              precision    recall  f1-score   support\n\n           0
0.67      0.67      0.67         3\n           1       0.86      0.86
0.86         7\n\n    accuracy                           0.80        10\n
macro avg       0.76      0.76      0.76        10\nweighted avg       0.8
0      0.80      0.80        10\n'
```

In [58]:

```
print('x:', x, sep='\n')
print('y:', y, sep='\n', end='\n\n')
print('intercept:', model.intercept_)
print('coef:', model.coef_, end='\n\n')
print('p_pred:', p_pred, sep='\n', end='\n\n')
print('y_pred:', y_pred, end='\n\n')

print('conf_m:', conf_m, sep='\n', end='\n\n')
print('report:', report, sep='\n')
```

```
x:
[[0]
 [1]
 [2]
 [3]
 [4]
 [5]
 [6]
 [7]
 [8]
 [9]]
y:
[0 1 0 0 1 1 1 1 1 1]

intercept: [-1.51632619]
coef: [[0.703457]]

p_pred:
[[0.81999686 0.18000314]
 [0.69272057 0.30727943]
 [0.52732579 0.47267421]
 [0.35570732 0.64429268]
 [0.21458576 0.78541424]
 [0.11910229 0.88089771]
 [0.06271329 0.93728671]
 [0.03205032 0.96794968]
 [0.0161218  0.9838782 ]
 [0.00804372 0.99195628]]

y_pred: [0 0 0 1 1 1 1 1 1 1]

conf_m:
[[2 1]
 [1 6]]

report:
              precision    recall  f1-score   support

           0       0.67      0.67      0.67         3
           1       0.86      0.86      0.86         7

    accuracy                           0.80        10
   macro avg       0.76      0.76      0.76        10
weighted avg       0.80      0.80      0.80        10
```

In [60]:

```
# Logistic Regression with StatsModels:
import numpy as np
import statsmodels.api as sm
```

In [68]:

```
x = np.arange(10).reshape(-1, 1)
y = np.array([0, 1, 0, 0, 1, 1, 1, 1, 1, 1])
x = sm.add_constant(x)
model = sm.Logit(y, x)
```

In [62]:

```
x
```

Out[62]:

```
array([[1., 0.],
       [1., 1.],
       [1., 2.],
       [1., 3.],
       [1., 4.],
       [1., 5.],
       [1., 6.],
       [1., 7.],
       [1., 8.],
       [1., 9.]])
```

In [63]:

```
y
```

Out[63]:

```
array([0, 1, 0, 0, 1, 1, 1, 1, 1, 1])
```

In [69]:

```
result = model.fit(method='newton')
```

```
Optimization terminated successfully.
         Current function value: 0.350471
         Iterations 7
```

In [70]:

```
result.params # Obtain values of b0 and b1
```

Out[70]:

```
array([-1.972805  ,  0.82240094])
```

In [71]:

```
result.predict()
```

Out[71]:

```
array([0.12208792, 0.24041529, 0.41872657, 0.62114189, 0.78864861,
       0.89465521, 0.95080891, 0.97777369, 0.99011108, 0.99563083])
```

In [72]:

```
(result.predict(x) >= 0.5).astype(int)
```

Out[72]:

```
array([0, 0, 0, 1, 1, 1, 1, 1, 1, 1])
```

In [73]:

```
result.pred_table()
```

Out[73]:

```
array([[2., 1.],
       [1., 6.]])
```

In [74]:

```
result.summary()
```

Out[74]:

Logit Regression Results

Dep. Variable:	y	**No. Observations:**	10
Model:	Logit	**Df Residuals:**	8
Method:	MLE	**Df Model:**	1
Date:	Wed, 18 Jan 2023	**Pseudo R-squ.:**	0.4263
Time:	17:26:13	**Log-Likelihood:**	-3.5047
converged:	True	**LL-Null:**	-6.1086
Covariance Type:	nonrobust	**LLR p-value:**	0.02248

	coef	std err	z	P>|z|	[0.025	0.975]
const	-1.9728	1.737	-1.136	0.256	-5.377	1.431
x1	0.8224	0.528	1.557	0.119	-0.213	1.858

In [75]:

```
result.summary2()
```

Out[75]:

Model:	Logit	Pseudo R-squared:	0.426
Dependent Variable:	y	AIC:	11.0094
Date:	2023-01-18 17:26	BIC:	11.6146
No. Observations:	10	Log-Likelihood:	-3.5047
Df Model:	1	LL-Null:	-6.1086
Df Residuals:	8	LLR p-value:	0.022485
Converged:	1.0000	Scale:	1.0000
No. Iterations:	7.0000		

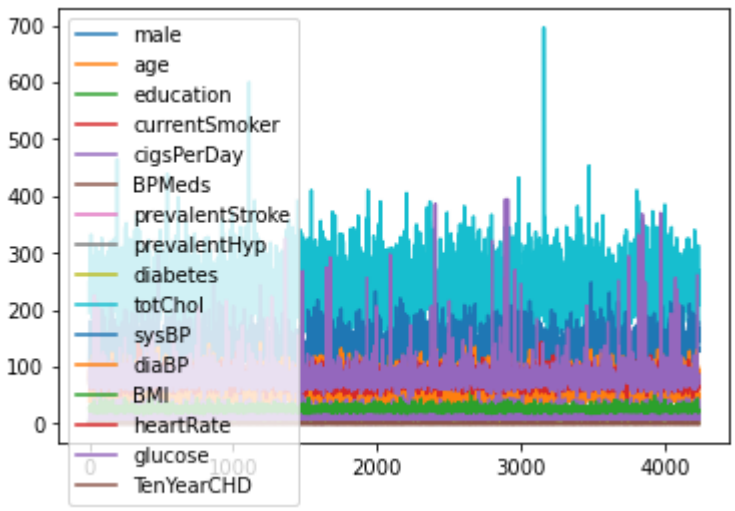
	Coef.	Std.Err.	z	P>|z|	[0.025	0.975]
const	-1.9728	1.7366	-1.1360	0.2560	-5.3765	1.4309
x1	0.8224	0.5281	1.5572	0.1194	-0.2127	1.8575

In [78]:

```
df.plot()
```

Out[78]:

```
<AxesSubplot:>
```

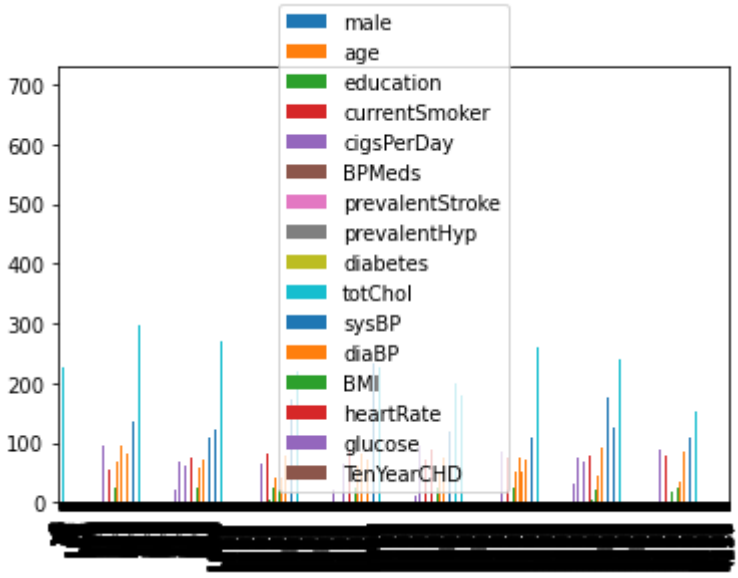


In [79]:

```
df.plot.bar()
```

Out[79]:

<AxesSubplot:>

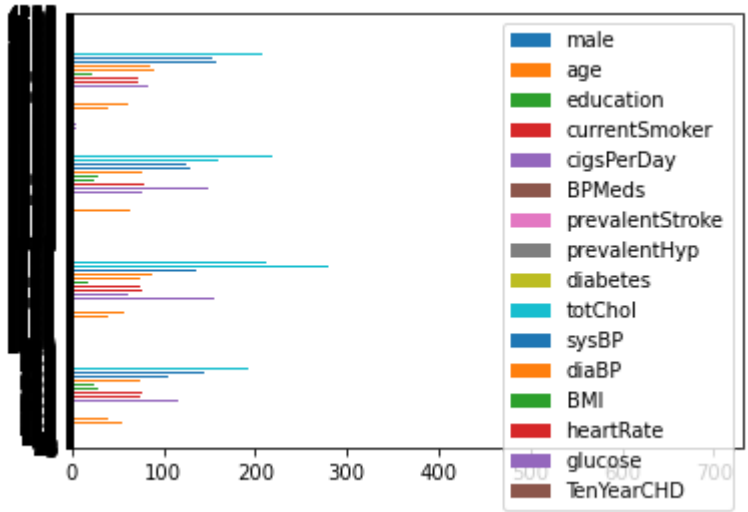


In [80]:

```
df.plot.barh ()
```

Out[80]:

<AxesSubplot:>

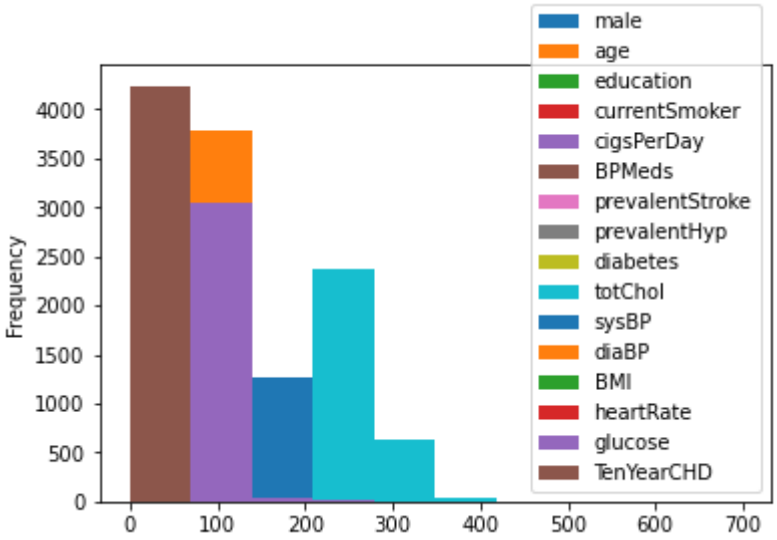


In [81]:

```
df.plot.hist()
```

Out[81]:

```
<AxesSubplot:ylabel='Frequency'>
```

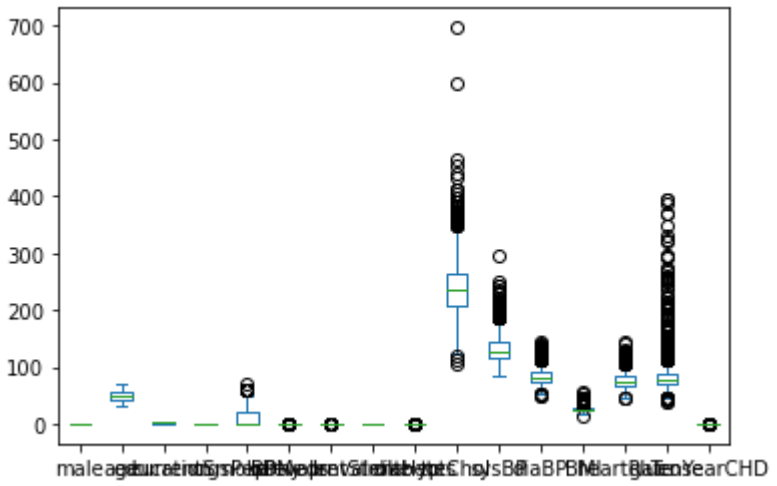


In [82]:

```
df.plot.box()
```

Out[82]:

```
<AxesSubplot:>
```



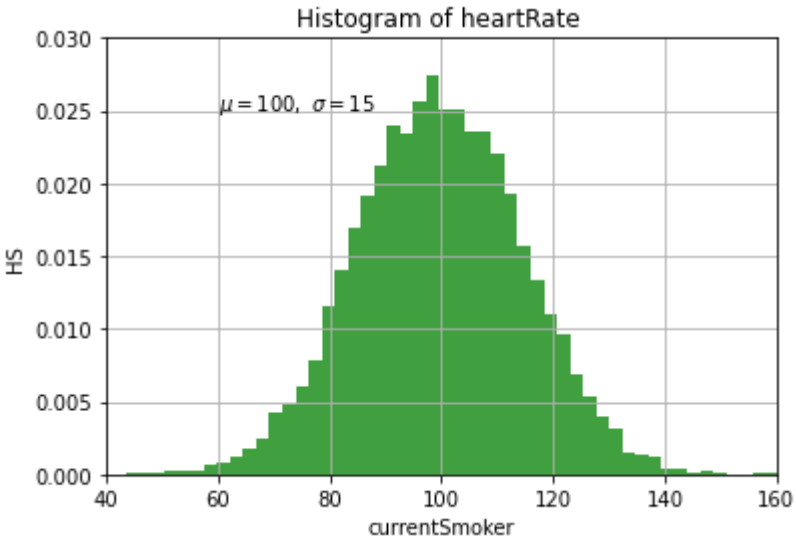
In [85]:

```
# Fixing random state for reproducibility
np.random.seed(19680801)

mu, sigma = 100, 15
x = mu + sigma * np.random.randn(10000)

# the histogram of the data
n, bins, patches = plt.hist(x, 50, density=True, facecolor='g', alpha=0.75)


plt.xlabel('currentSmoker')
plt.ylabel('HS')
plt.title('Histogram of heartRate ')
plt.text(60, .025, r'$\mu=100,\ \sigma=15$')
plt.xlim(40, 160)
plt.ylim(0, 0.03)
plt.grid(True)
plt.show()
```



In [ ]: