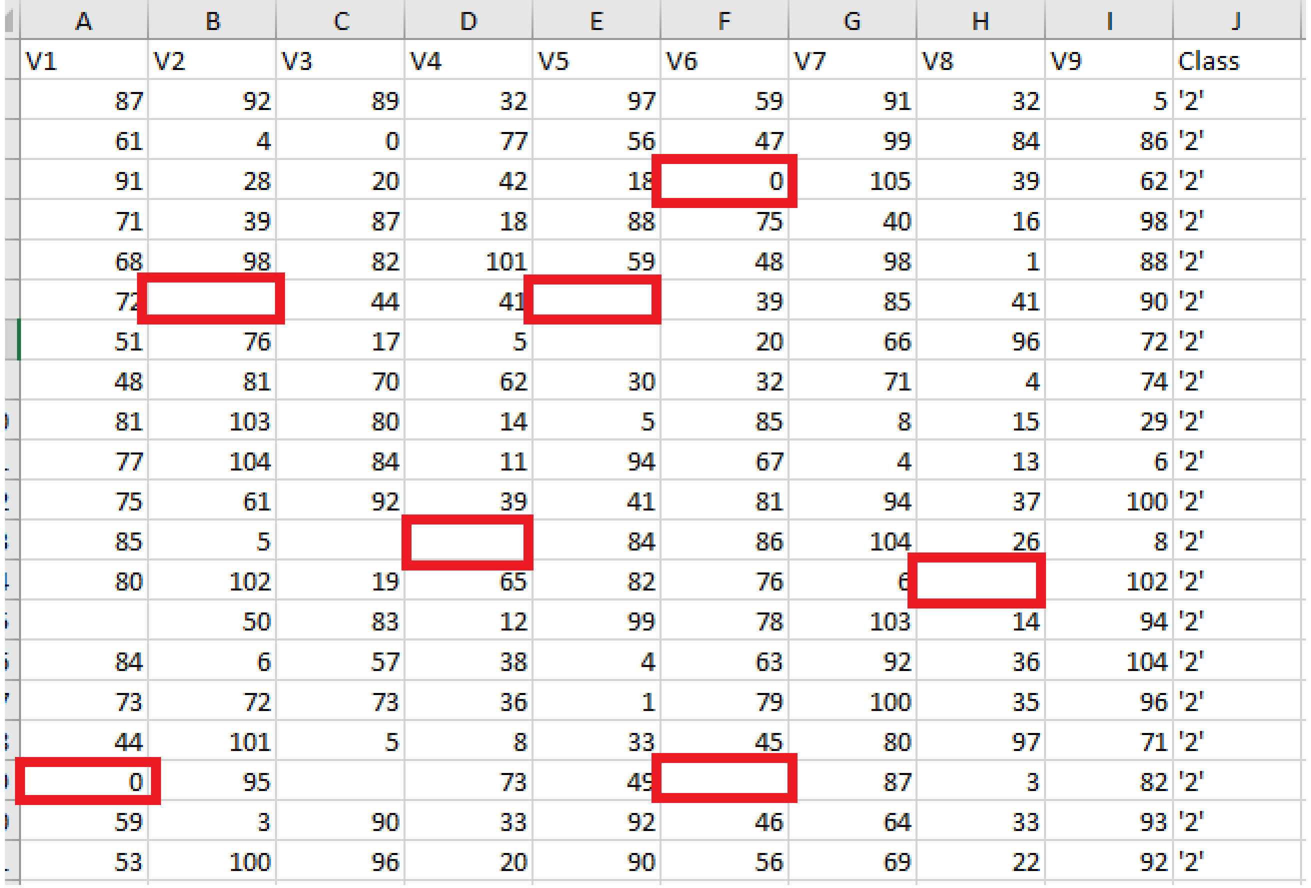
Section2: Sklearn

SCIKIT LEARN

Data Cleaning

- Data cleaning is the process of fixing or removing incorrect, corrupted, incorrectly formatted, duplicate, or incomplete data within a dataset. When combining multiple data sources, there are many opportunities for data to be duplicated or mislabeled.

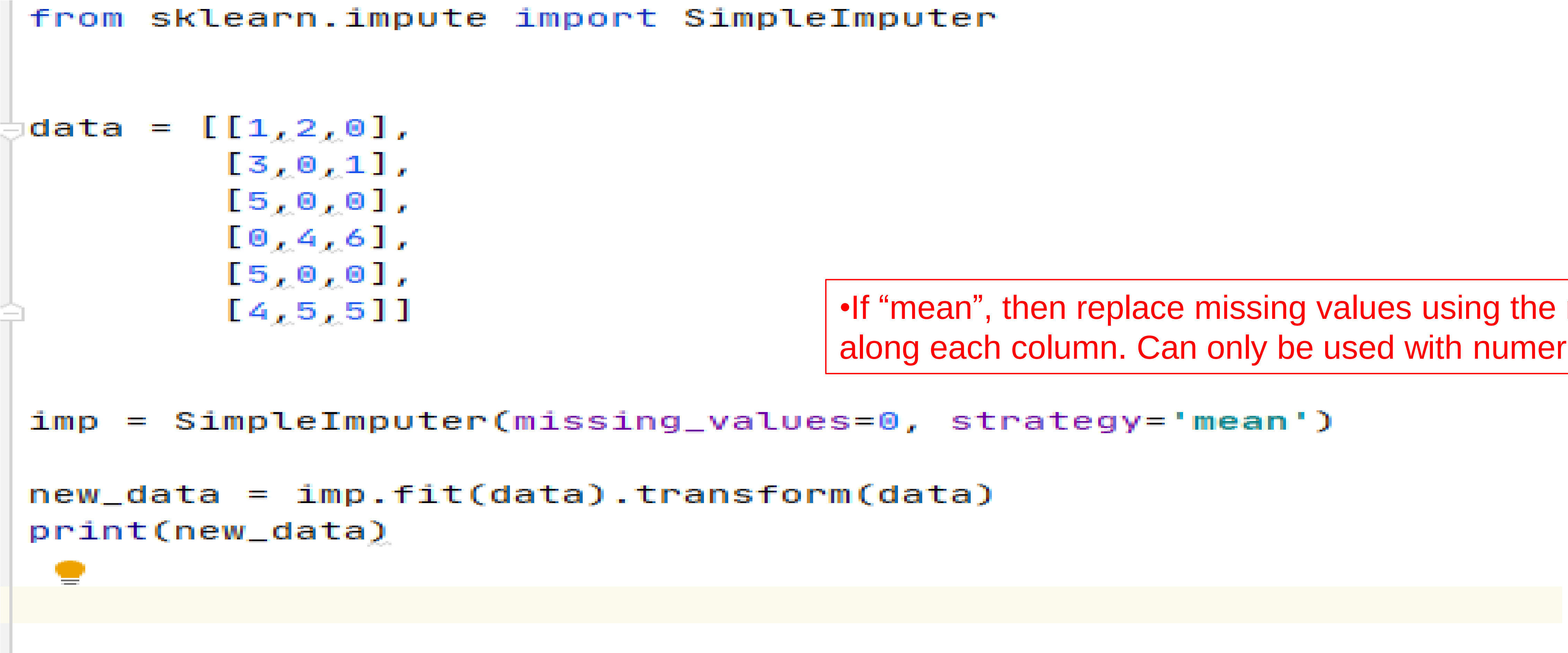


A	B	C	D	E	F	G	H	I	J
V1	V2	V3	V4	V5	V6	V7	V8	V9	Class
87	92	89	32	97	59	91	32	5	'2'
61	4	0	77	56	47	99	84	86	'2'
91	28	20	42	18	0	105	39	62	'2'
71	39	87	18	88	75	40	16	98	'2'
68	98	82	101	59	48	98	1	88	'2'
72		44	41		39	85	41	90	'2'
51	76	17	5		20	66	96	72	'2'
48	81	70	62	30	32	71	4	74	'2'
81	103	80	14	5	85	8	15	29	'2'
77	104	84	11	94	67	4	13	6	'2'
75	61	92	39	41	81	94	37	100	'2'
85	5			84	86	104	26	8	'2'
80	102	19	65	82	76	6		102	'2'
	50	83	12	99	78	103	14	94	'2'
84	6	57	38	4	63	92	36	104	'2'
73	72	73	36	1	79	100	35	96	'2'
44	101	5	8	33	45	80	97	71	'2'
0	95		73	49		87	3	82	'2'
59	3	90	33	92	46	64	33	93	'2'
53	100	96	20	90	56	69	22	92	'2'

Data cleaning: strategies

1. mean
2. median
3. most_frequent
4. constant

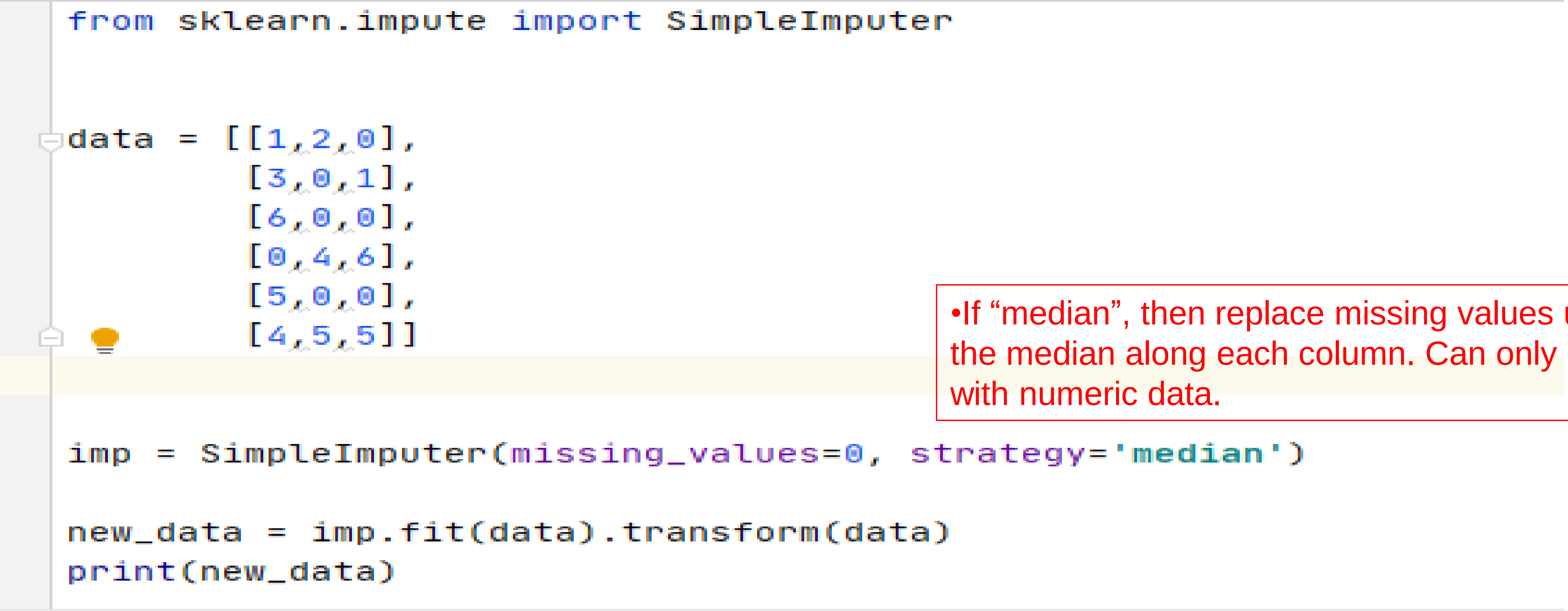
Imputation strategy: mean



```
from sklearn.impute import SimpleImputer


data = [[1,2,0],
        [3,0,1],
        [5,0,0],
        [0,4,6],
        [5,0,0],
        [4,5,5]]


imp = SimpleImputer(missing_values=0, strategy='mean')

new_data = imp.fit(data).transform(data)
print(new_data)
```

- If “mean”, then replace missing values using the mean along each column. Can only be used with numeric data.

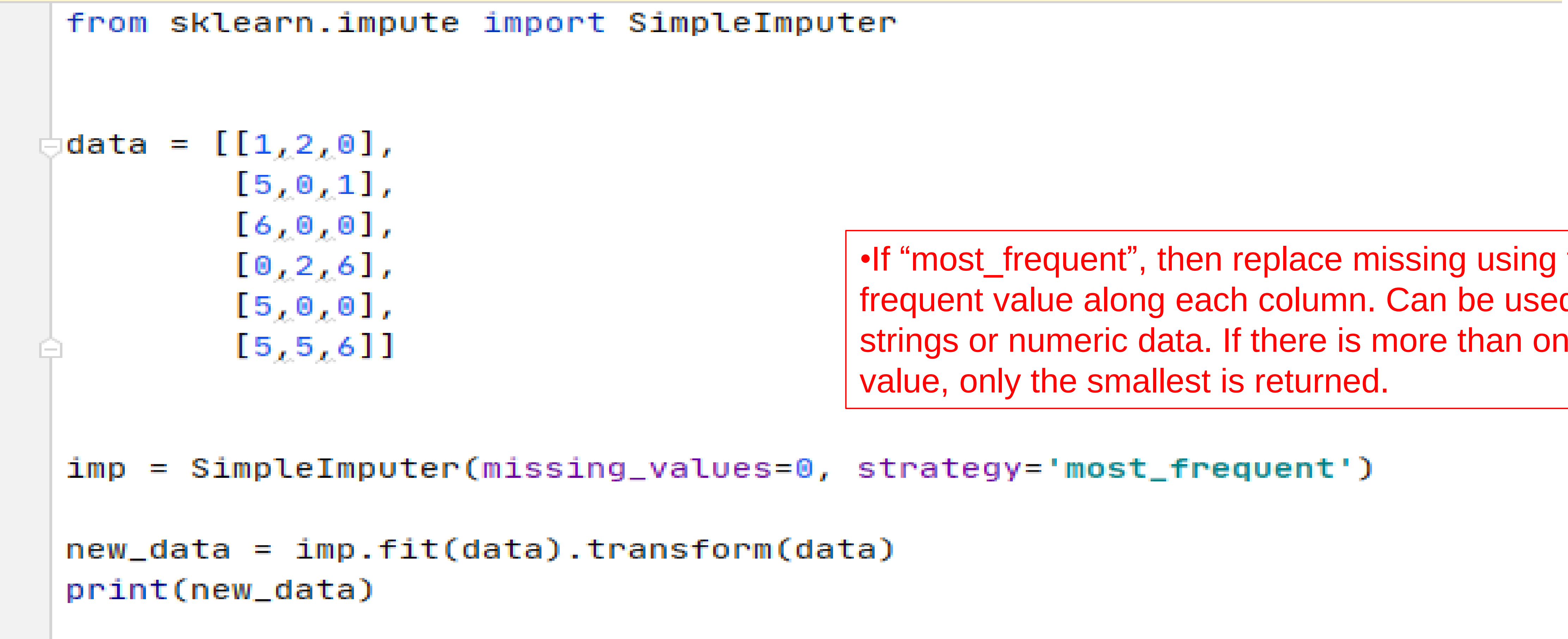
Imputation strategy: median



```
from sklearn.impute import SimpleImputer


data = [[1,2,0],
        [3,0,1],
        [6,0,0],
        [0,4,6],
        [5,0,0],
        [4,5,5]]


imp = SimpleImputer(missing_values=0, strategy='median')

new_data = imp.fit(data).transform(data)
print(new_data)
```

•If “median”, then replace missing values using the median along each column. Can only be used with numeric data.

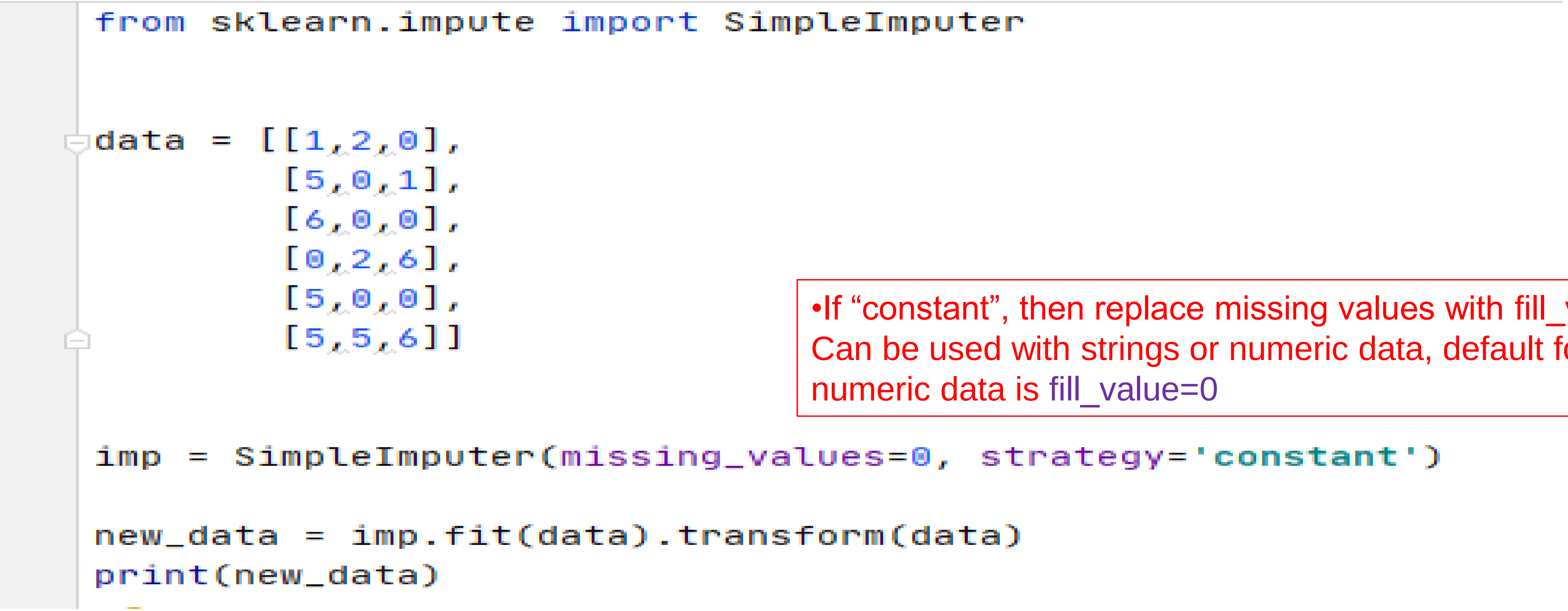
Imputation strategy: most_frequent



```
from sklearn.impute import SimpleImputer


data = [[1,2,0],
        [5,0,1],
        [6,0,0],
        [0,2,6],
        [5,0,0],
        [5,5,6]]


imp = SimpleImputer(missing_values=0, strategy='most_frequent')

new_data = imp.fit(data).transform(data)
print(new_data)
```

- If “most_frequent”, then replace missing using the most frequent value along each column. Can be used with strings or numeric data. If there is more than one such value, only the smallest is returned.

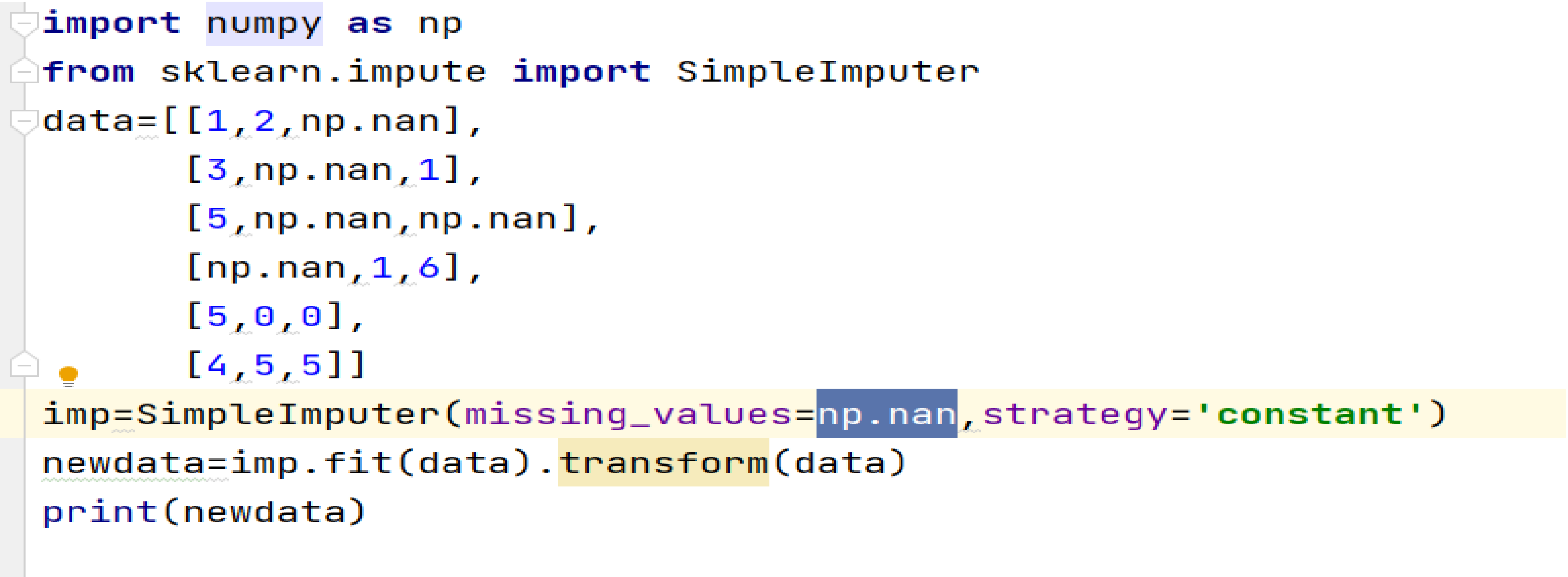
Imputation strategy: constant



```
from sklearn.impute import SimpleImputer


data = [[1,2,0],
        [5,0,1],
        [6,0,0],
        [0,2,6],
        [5,0,0],
        [5,5,6]]


imp = SimpleImputer(missing_values=0, strategy='constant')

new_data = imp.fit(data).transform(data)
print(new_data)
```

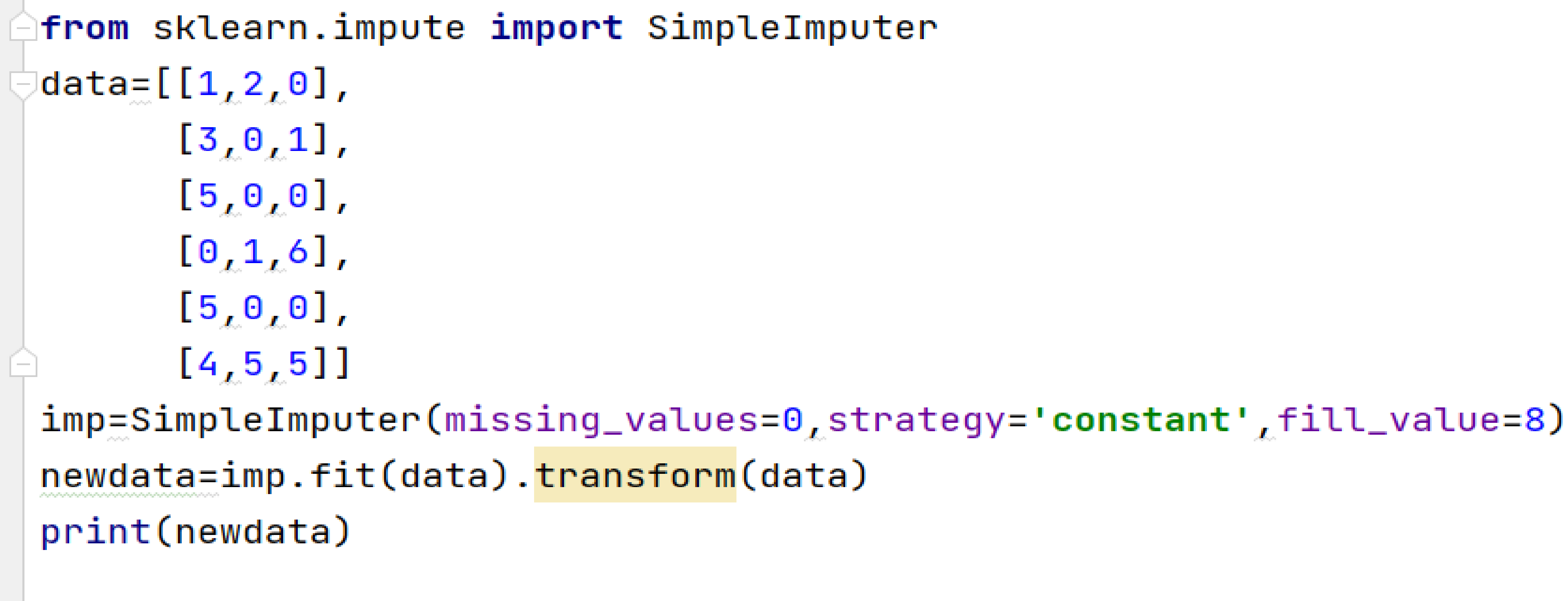
•If “constant”, then replace missing values with fill_value. Can be used with strings or numeric data, default for numeric data is fill_value=0

Imputation strategy: constant



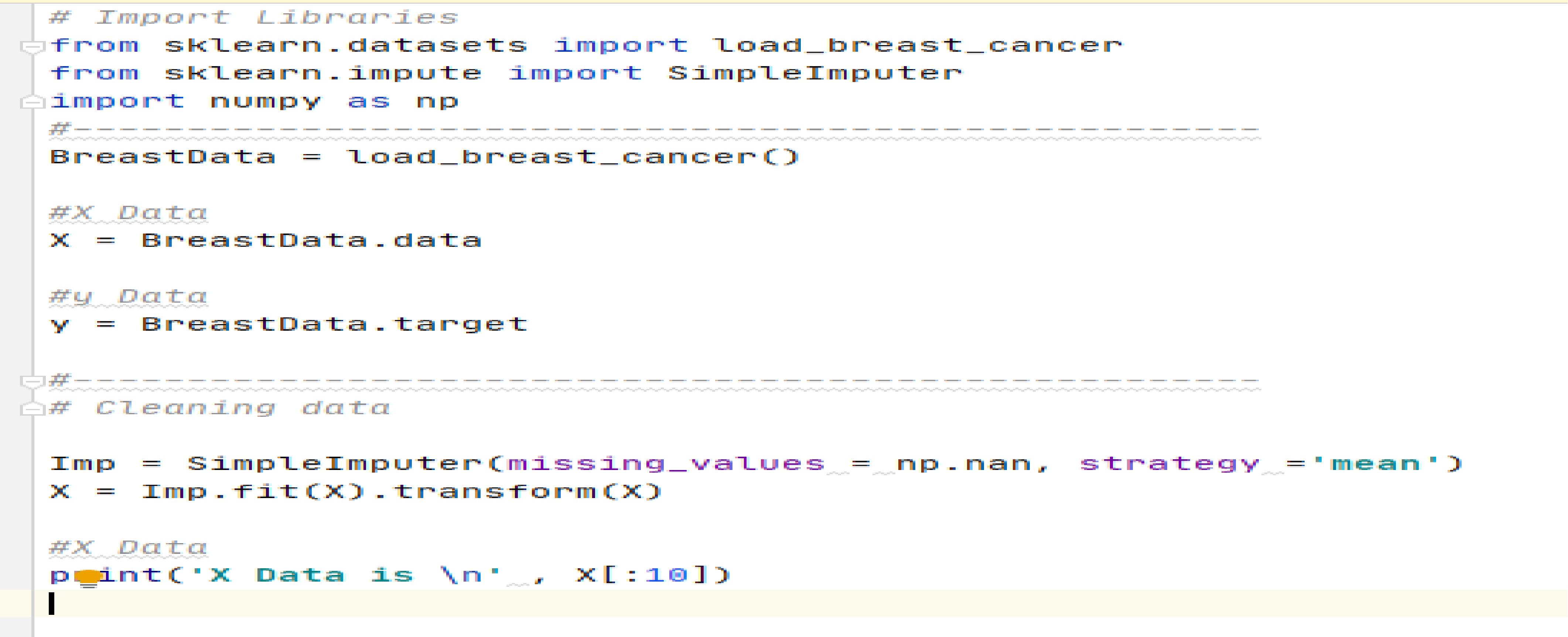
```
import numpy as np
from sklearn.impute import SimpleImputer
data=[[1,2,np.nan],
      [3,np.nan,1],
      [5,np.nan,np.nan],
      [np.nan,1,6],
      [5,0,0],
      [4,5,5]]
imp=SimpleImputer(missing_values=np.nan,strategy='constant')
newdata=imp.fit(data).transform(data)
print(newdata)
```

Imputation strategy: constant



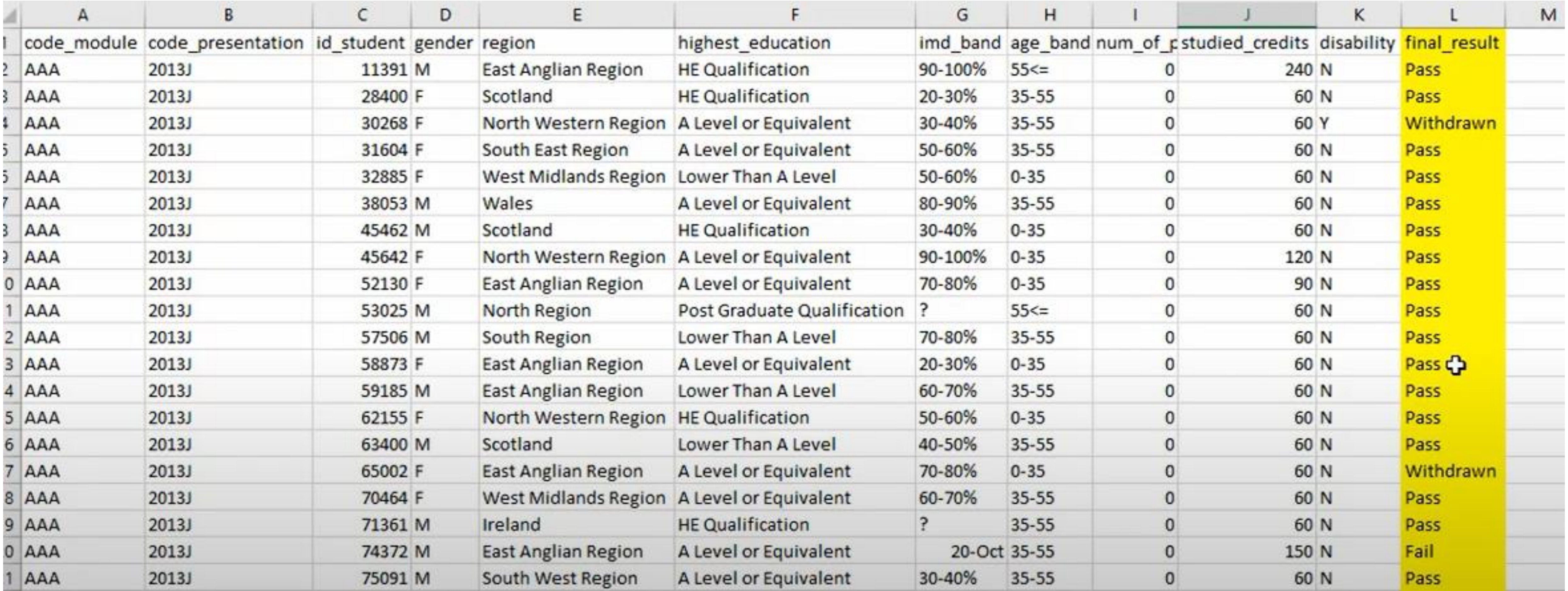
```python
from sklearn.impute import SimpleImputer
data=[[1,2,0],
      [3,0,1],
      [5,0,0],
      [0,1,6],
      [5,0,0],
      [4,5,5]]
imp=SimpleImputer(missing_values=0,strategy='constant',fill_value=8)
newdata=imp.fit(data).transform(data)
print(newdata)
```

Data Cleaning: load_breast_cancer



```
# Import Libraries
from sklearn.datasets import load_breast_cancer
from sklearn.impute import SimpleImputer
import numpy as np
#----------------------------------------------------
BreastData = load_breast_cancer()

#X Data
X = BreastData.data

#y Data
y = BreastData.target

#----------------------------------------------------
# Cleaning data

Imp = SimpleImputer(missing_values = np.nan, strategy ='mean')
X = Imp.fit(X).transform(X)

#X Data
print('X Data is \n' , X[:10])
```

Feature Selection

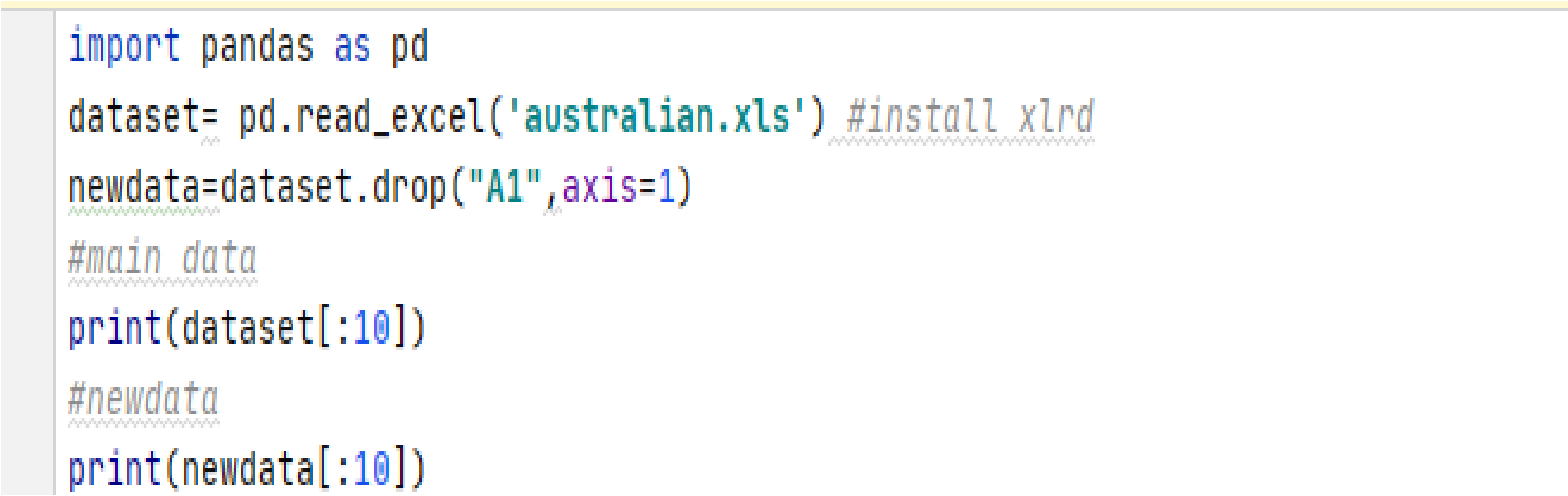


code_module	code_presentation	id_student	gender	region	highest_education	imd_band	age_band	num_of_p	studied_credits	disability	final_result
AAA	2013J	11391	M	East Anglian Region	HE Qualification	90-100%	55<=	0	240	N	Pass
AAA	2013J	28400	F	Scotland	HE Qualification	20-30%	35-55	0	60	N	Pass
AAA	2013J	30268	F	North Western Region	A Level or Equivalent	30-40%	35-55	0	60	Y	Withdrawn
AAA	2013J	31604	F	South East Region	A Level or Equivalent	50-60%	35-55	0	60	N	Pass
AAA	2013J	32885	F	West Midlands Region	Lower Than A Level	50-60%	0-35	0	60	N	Pass
AAA	2013J	38053	M	Wales	A Level or Equivalent	80-90%	35-55	0	60	N	Pass
AAA	2013J	45462	M	Scotland	HE Qualification	30-40%	0-35	0	60	N	Pass
AAA	2013J	45642	F	North Western Region	A Level or Equivalent	90-100%	0-35	0	120	N	Pass
AAA	2013J	52130	F	East Anglian Region	A Level or Equivalent	70-80%	0-35	0	90	N	Pass
AAA	2013J	53025	M	North Region	Post Graduate Qualification	?	55<=	0	60	N	Pass
AAA	2013J	57506	M	South Region	Lower Than A Level	70-80%	35-55	0	60	N	Pass
AAA	2013J	58873	F	East Anglian Region	A Level or Equivalent	20-30%	0-35	0	60	N	Pass
AAA	2013J	59185	M	East Anglian Region	Lower Than A Level	60-70%	35-55	0	60	N	Pass
AAA	2013J	62155	F	North Western Region	HE Qualification	50-60%	0-35	0	60	N	Pass
AAA	2013J	63400	M	Scotland	Lower Than A Level	40-50%	35-55	0	60	N	Pass
AAA	2013J	65002	F	East Anglian Region	A Level or Equivalent	70-80%	0-35	0	60	N	Withdrawn
AAA	2013J	70464	F	West Midlands Region	A Level or Equivalent	60-70%	35-55	0	60	N	Pass
AAA	2013J	71361	M	Ireland	HE Qualification	?	35-55	0	60	N	Pass
AAA	2013J	74372	M	East Anglian Region	A Level or Equivalent	20-Oct	35-55	0	150	N	Fail
AAA	2013J	75091	M	South West Region	A Level or Equivalent	30-40%	35-55	0	60	N	Pass

Feature Selection

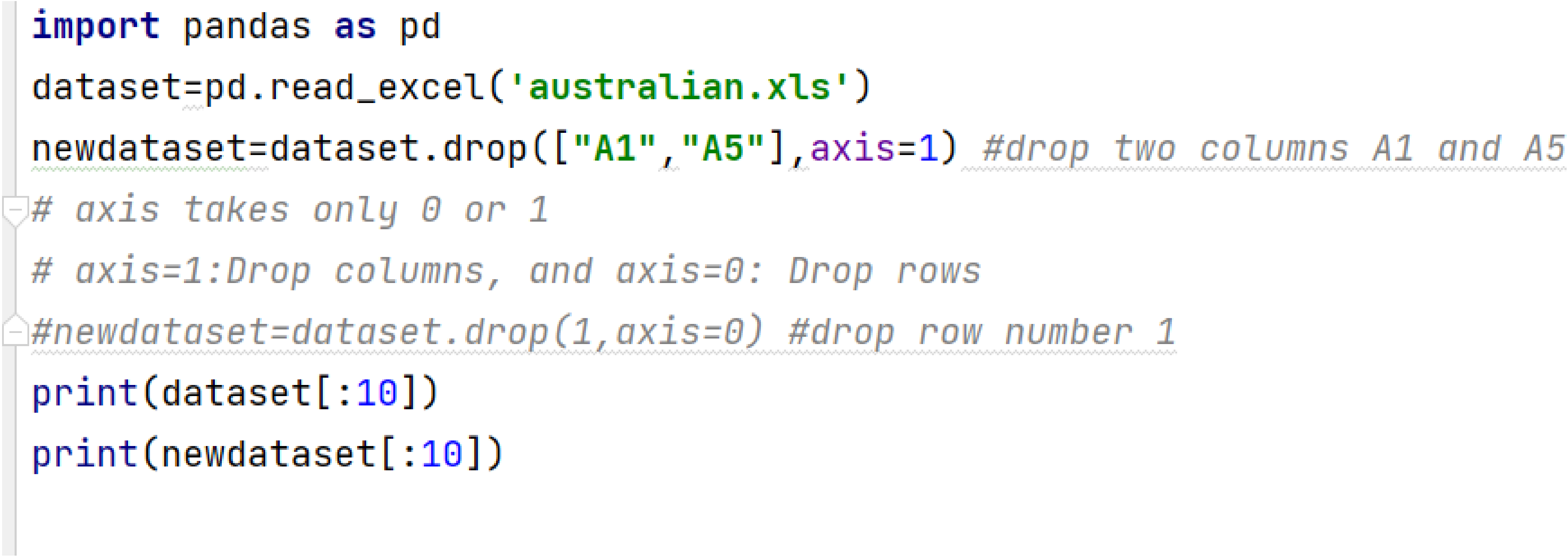
- Feature selection is the process of reducing the number of input variables when developing a predictive model. It is desirable to reduce the number of input variables to both reduce the computational cost of modeling and, in some cases, to improve the performance of the model

Feature Selection



```
import pandas as pd
dataset= pd.read_excel('australian.xls') #install xlrd
newdata=dataset.drop("A1",axis=1)
#main data
print(dataset[:10])
#newdata
print(newdata[:10])
```

Feature Selection



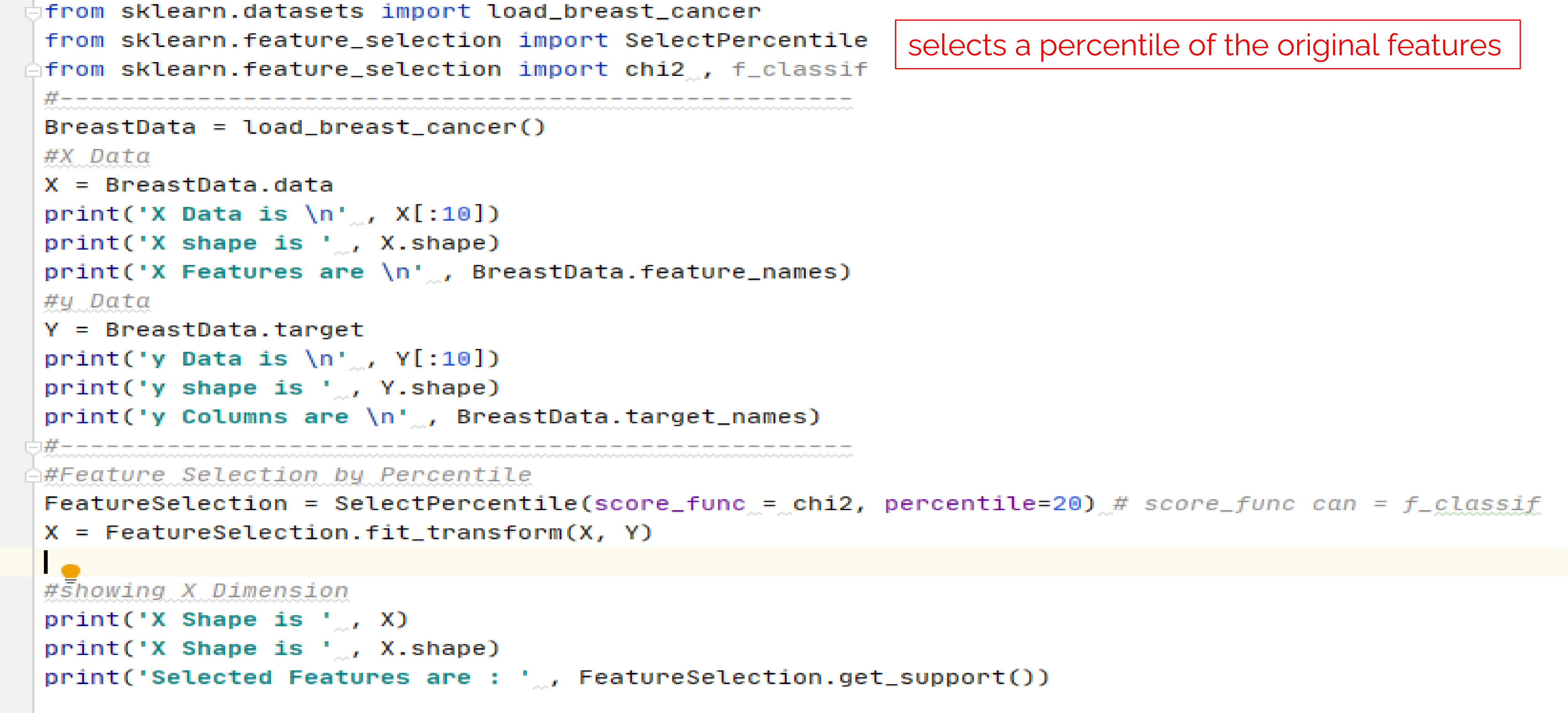
```
import pandas as pd
dataset=pd.read_excel('australian.xls')
newdataset=dataset.drop(["A1","A5"],axis=1) #drop two columns A1 and A5
# axis takes only 0 or 1
# axis=1:Drop columns, and axis=0: Drop rows
#newdataset=dataset.drop(1,axis=0) #drop row number 1
print(dataset[:10])
print(newdataset[:10])
```

Feature Selection strategies

1) feature_selection.SelectPercentile
2) feature_selection.GenericUnivariateSelect
3) feature_selection.SelectKBest
4) feature_selection.SelectFromModel

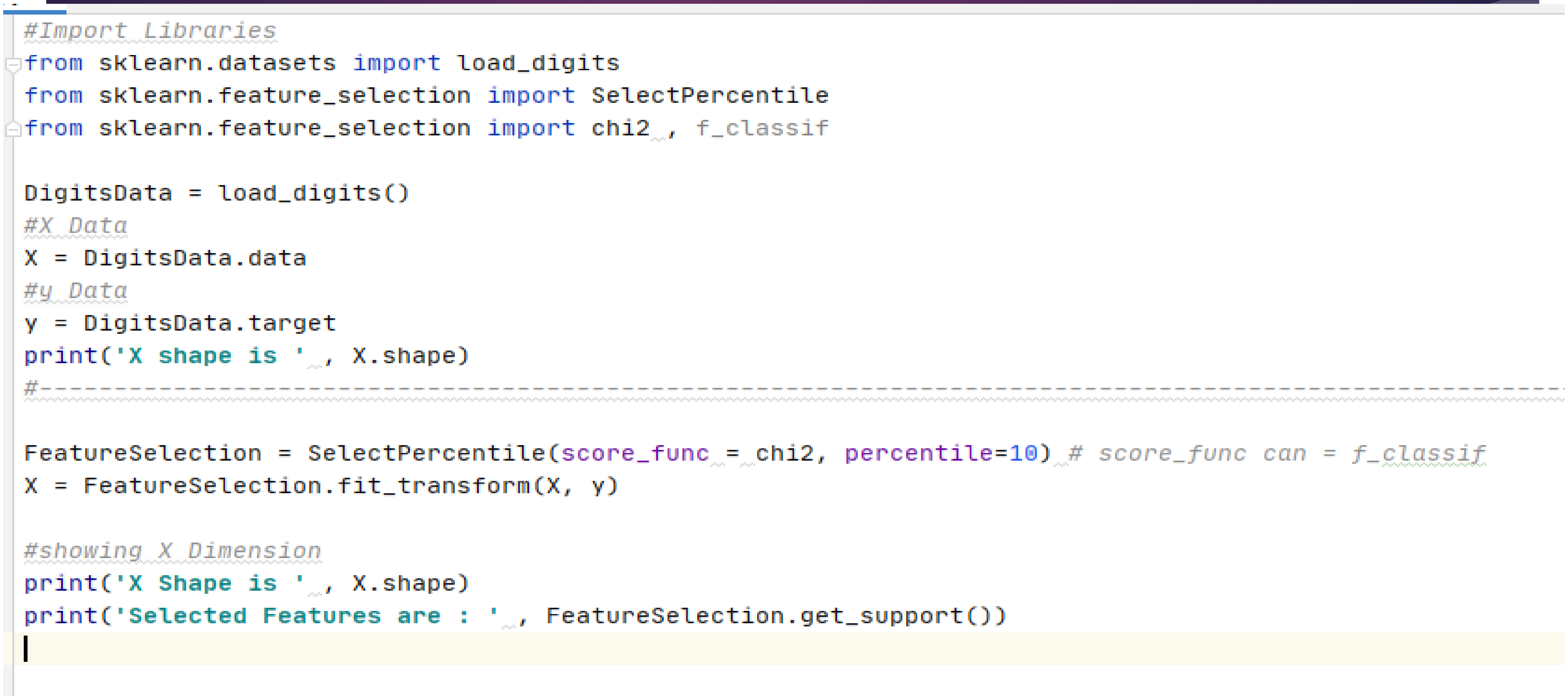
SelectPercentile: load_breast_cancer

selects a percentile of the original features



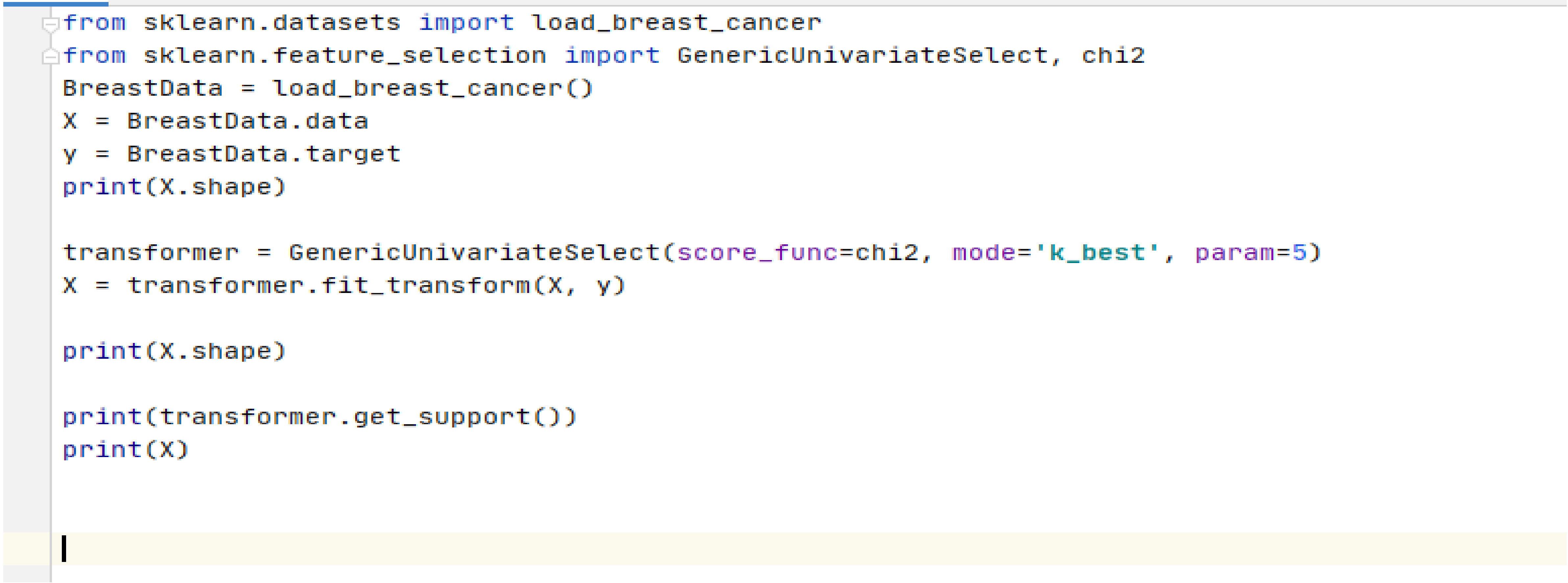
```python
from sklearn.datasets import load_breast_cancer
from sklearn.feature_selection import SelectPercentile
from sklearn.feature_selection import chi2 , f_classif
#----------------------------------------------------
BreastData = load_breast_cancer()
#X Data
X = BreastData.data
print('X Data is \n' , X[:10])
print('X shape is ' , X.shape)
print('X Features are \n' , BreastData.feature_names)
#y Data
Y = BreastData.target
print('y Data is \n' , Y[:10])
print('y shape is ' , Y.shape)
print('y Columns are \n' , BreastData.target_names)
#----------------------------------------------------
#Feature Selection by Percentile
FeatureSelection = SelectPercentile(score_func = chi2, percentile=20) # score_func can = f_classif
X = FeatureSelection.fit_transform(X, Y)

#showing X Dimension
print('X Shape is ' , X)
print('X Shape is ' , X.shape)
print('Selected Features are : ' , FeatureSelection.get_support())
```

SelectPercentile: load_digits



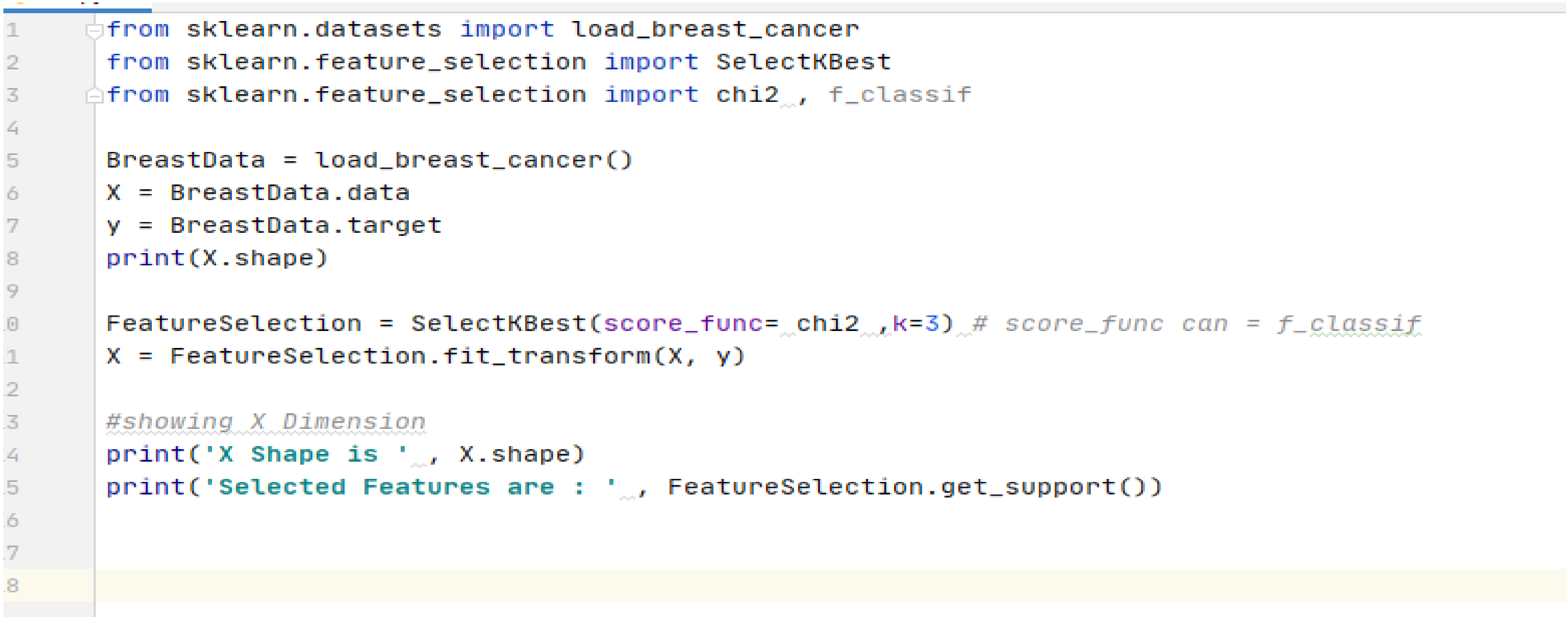
```
#Import Libraries
from sklearn.datasets import load_digits
from sklearn.feature_selection import SelectPercentile
from sklearn.feature_selection import chi2 , f_classif

DigitsData = load_digits()
#X Data
X = DigitsData.data
#y Data
y = DigitsData.target
print('X shape is ' , X.shape)
#-----------------------------------------------------------------------------------------------------

FeatureSelection = SelectPercentile(score_func = chi2, percentile=10) # score_func can = f_classif
X = FeatureSelection.fit_transform(X, y)

#showing X Dimension
print('X Shape is ' , X.shape)
print('Selected Features are : ' , FeatureSelection.get_support())
```

GenericUnivariateSelect : load_breast_cancer



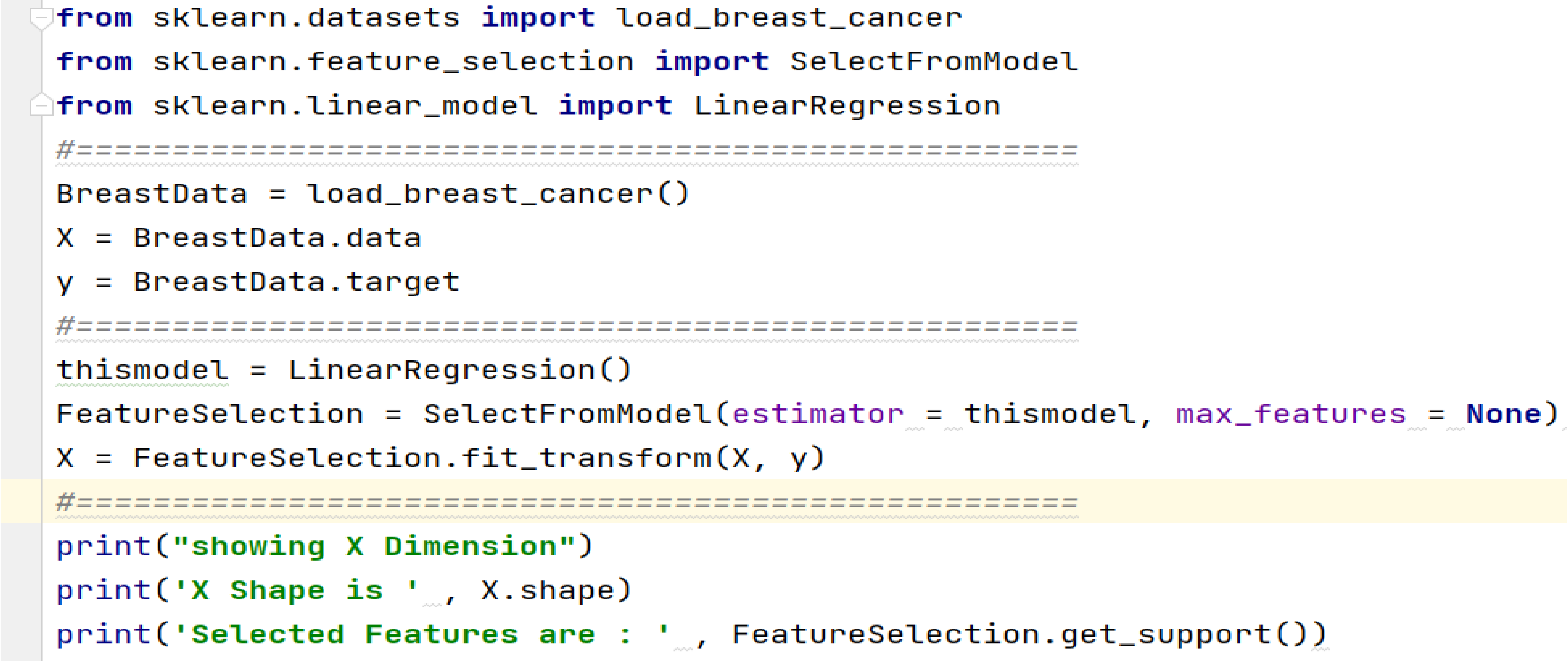
```
from sklearn.datasets import load_breast_cancer
from sklearn.feature_selection import GenericUnivariateSelect, chi2
BreastData = load_breast_cancer()
X = BreastData.data
y = BreastData.target
print(X.shape)

transformer = GenericUnivariateSelect(score_func=chi2, mode='k_best', param=5)
X = transformer.fit_transform(X, y)

print(X.shape)

print(transformer.get_support())
print(X)
```

SelectKBest: load_breast_cancer



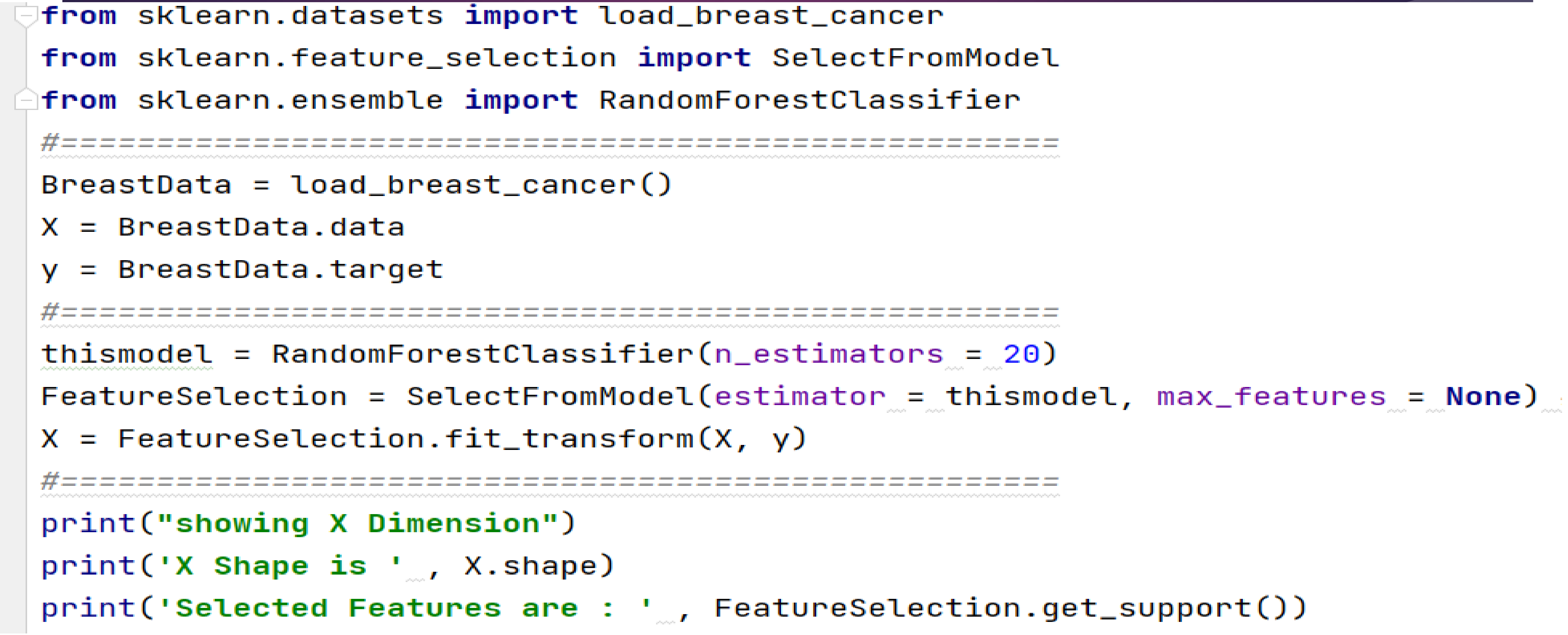
```
from sklearn.datasets import load_breast_cancer
from sklearn.feature_selection import SelectKBest
from sklearn.feature_selection import chi2 , f_classif

BreastData = load_breast_cancer()
X = BreastData.data
y = BreastData.target
print(X.shape)

FeatureSelection = SelectKBest(score_func= chi2 ,k=3) # score_func can = f_classif
X = FeatureSelection.fit_transform(X, y)

#showing X Dimension
print('X Shape is ' , X.shape)
print('Selected Features are : ' , FeatureSelection.get_support())
```

SelectFromModel: load_breast_cancer Linear Regression



```
from sklearn.datasets import load_breast_cancer
from sklearn.feature_selection import SelectFromModel
from sklearn.linear_model import LinearRegression
#====================================================
BreastData = load_breast_cancer()
X = BreastData.data
y = BreastData.target
#====================================================
thismodel = LinearRegression()
FeatureSelection = SelectFromModel(estimator = thismodel, max_features = None)
X = FeatureSelection.fit_transform(X, y)
#====================================================
print("showing X Dimension")
print('X Shape is ' , X.shape)
print('Selected Features are : ' , FeatureSelection.get_support())
```

SelectFromModel: load_breast_cancer Random Forest



```
from sklearn.datasets import load_breast_cancer
from sklearn.feature_selection import SelectFromModel
from sklearn.ensemble import RandomForestClassifier
#====================================================
BreastData = load_breast_cancer()
X = BreastData.data
y = BreastData.target
#====================================================
thismodel = RandomForestClassifier(n_estimators = 20)
FeatureSelection = SelectFromModel(estimator = thismodel, max_features = None)
X = FeatureSelection.fit_transform(X, y)
#====================================================
print("showing X Dimension")
print('X Shape is ' , X.shape)
print('Selected Features are : ' , FeatureSelection.get_support())
```