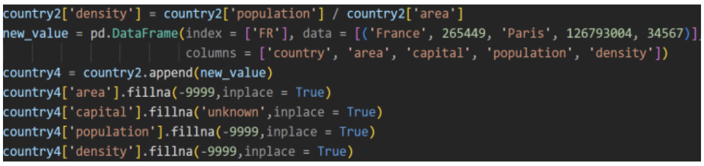
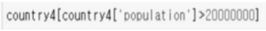
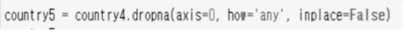


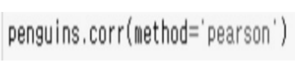
```
country2['density'] = country2['population'] / country2['area']
new_value = pd.DataFrame(index = ['FR'], data = [('France', 265449, 'Paris', 126793004, 34567)],
                         columns = ['country', 'area', 'capital', 'population', 'density'])
country4 = country2.append(new_value)
country4['area'].fillna(-9999,inplace = True)
country4['capital'].fillna('unknown',inplace = True)
country4['population'].fillna(-9999,inplace = True)
country4['density'].fillna(-9999,inplace = True)
```



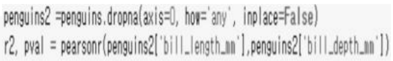
```
country4[country4['population']>20000000]
```



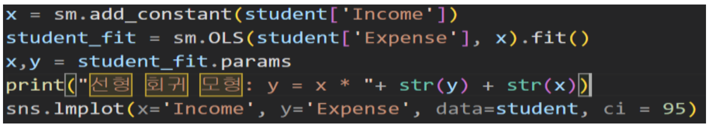
```
country5 = country4.dropna(axis=0, how='any', inplace=False)
```



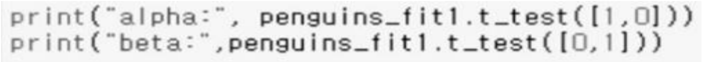
```
penguins.corr(method='pearson')
```



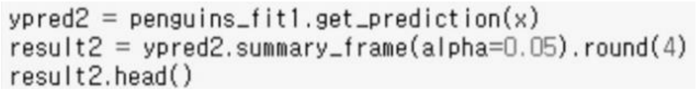
```
penguins2 =penguins.dropna(axis=0, how='any', inplace=False)
r2, pval = pearsonr(penguins2['bill_length_mm'],penguins2['bill_depth_mm'])
```



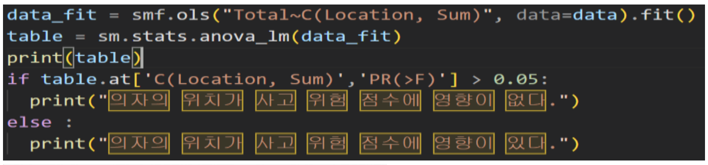
```
x = sm.add_constant(student['Income'])
student_fit = sm.OLS(student['Expense'], x).fit()
x,y = student_fit.params
print("선형 회귀 모형: y = x * "+ str(y) + str(x))
sns.lmplot(x='Income', y='Expense', data=student, ci = 95)
```



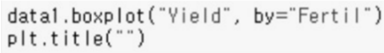
```
print("alpha:", penguins_fit1.t_test([1,0]))
print("beta:",penguins_fit1.t_test([0,1]))
```



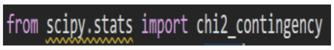
```
ypred2 = penguins_fit1.get_prediction(x)
result2 = ypred2.summary_frame(alpha=0.05).round(4)
result2.head()
```



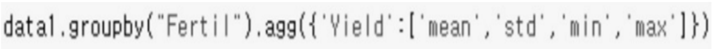
```
data_fit = smf.ols("Total~C(Location, Sum)", data=data).fit()
table = sm.stats.anova_lm(data_fit)
print(table)
if table.at['C(Location, Sum)','PR(>F)'] > 0.05:
  print("의자의 위치가 사고 위험 점수에 영향이 없다.")
else :
  print("의자의 위치가 사고 위험 점수에 영향이 있다.")
```



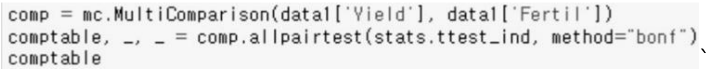
```
data1.boxplot("Yield", by="Fertil")
plt.title("")
```



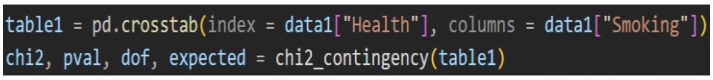
```
from scipy.stats import chi2_contingency
```



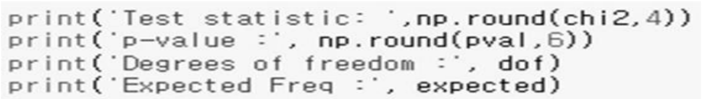
```
data1.groupby("Fertil").agg({'Yield':['mean','std','min','max']})
```



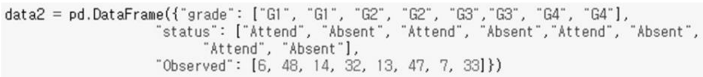
```
comp = mc.MultiComparison(data1['Yield'], data1['Fertil'])
comptable, _, _ = comp.allpairtest(stats.ttest_ind, method="bonf")
comptable
```



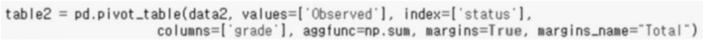
```
table1 = pd.crosstab(index = data1["Health"], columns = data1["Smoking"])
chi2, pval, dof, expected = chi2_contingency(table1)
```



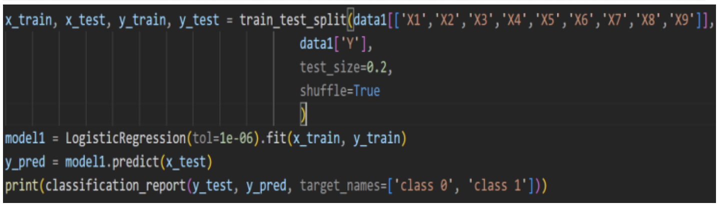
```
print('Test statistic: ',np.round(chi2,4))
print('p-value :', np.round(pval,6))
print('Degrees of freedom :', dof)
print('Expected Freq :', expected)
```



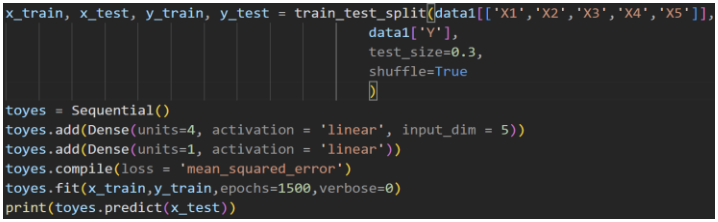
```
data2 = pd.DataFrame({"grade": ["G1", "G1", "G2", "G2", "G3","G3", "G4", "G4"],
                      "status": ["Attend", "Absent", "Attend", "Absent","Attend", "Absent",
                                 "Attend", "Absent"],
                      "Observed": [6, 48, 14, 32, 13, 47, 7, 33]})
```



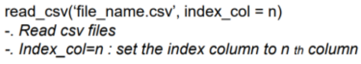
```
table2 = pd.pivot_table(data2, values=['Observed'], index=['status'],
                        columns=['grade'], aggfunc=np.sum, margins=True, margins_name="Total")
```



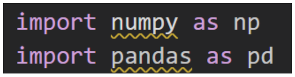
```
x_train, x_test, y_train, y_test = train_test_split(data1[['X1','X2','X3','X4','X5','X6','X7','X8','X9']],
                                                    data1['Y'],
                                                    test_size=0.2,
                                                    shuffle=True
                                                    )
model1 = LogisticRegression(tol=1e-06).fit(x_train, y_train)
y_pred = model1.predict(x_test)
print(classification_report(y_test, y_pred, target_names=['class 0', 'class 1']))
```



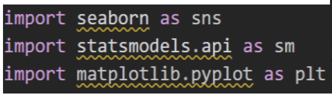
```
x_train, x_test, y_train, y_test = train_test_split(data1[['X1','X2','X3','X4','X5']],
                                                    data1['Y'],
                                                    test_size=0.3,
                                                    shuffle=True
                                                    )
toyes = Sequential()
toyes.add(Dense(units=4, activation = 'linear', input_dim = 5))
toyes.add(Dense(units=1, activation = 'linear'))
toyes.compile(loss = 'mean_squared_error')
toyes.fit(x_train,y_train,epochs=1500,verbose=0)
print(toyes.predict(x_test))
```



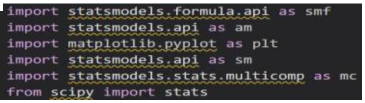
read_csv('file_name.csv', index_col = n)
-. *Read csv files*
-. *Index_col=n : set the index column to n th column*



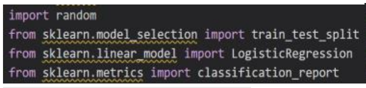
```
import numpy as np
import pandas as pd
```



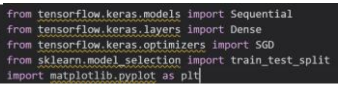
```
import seaborn as sns
import statsmodels.api as sm
import matplotlib.pyplot as plt
```



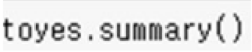
```
import statsmodels.formula.api as smf
import statsmodels.api as am
import matplotlib.pyplot as plt
import statsmodels.api as sm
import statsmodels.stats.multicomp as mc
from scipy import stats
```



```
import random
from sklearn.model_selection import train_test_split
from sklearn.linear_model import LogisticRegression
from sklearn.metrics import classification_report
```



```
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Dense
from tensorflow.keras.optimizers import SGD
from sklearn.model_selection import train_test_split
import matplotlib.pyplot as plt
```



```
toyes.summary()
```