FINAL TEST
HEART DISEASE UCI
DATA SCIENCE CLUB

Disusun oleh:

FABYAN KINDARYA
2110191024
2 D4 IT A

PROGRAM STUDI D-IV TEKNIK INFORMATIKA
POLITEKNIK ELEKTRONIKA NEGERI SURABAYA

1. Mencari Dataset

Informasi atribut :

1. age
2. sex
3. chest pain type (4 type)
4. resting blood pressure
5. serum cholestoral in mg/dl
6. fasting blood sugar > 120 mg/dl
7. resting electrocardiographic results (values 0,1,2)
8. maximum heart rate achieved
9. exercise induced angina
10. oldpeak = ST depression induced by exercise relative to rest
11. the slope of the peak exercise ST segment
12. number of major vessels (0-3) colored by flourosopy
13. thal: 3 = normal; 6 = fixed defect; 7 = reversable defect

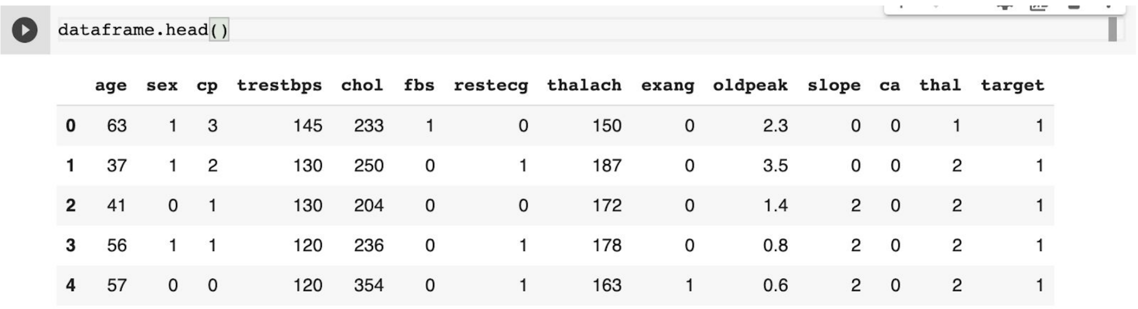
Pembuat :

1. Hungarian Institute of Cardiology. Budapest: Andras Janosi, M.D.
2. University Hospital, Zurich, Switzerland: William Steinbrunn, M.D.
3. University Hospital, Basel, Switzerland: Matthias Pfisterer, M.D.
4. V.A. Medical Center, Long Beach and Cleveland Clinic Foundation: Robert Detrano, M.D., Ph.D.

Dataset : https://www.kaggle.com/ronitf/heart-disease-uci

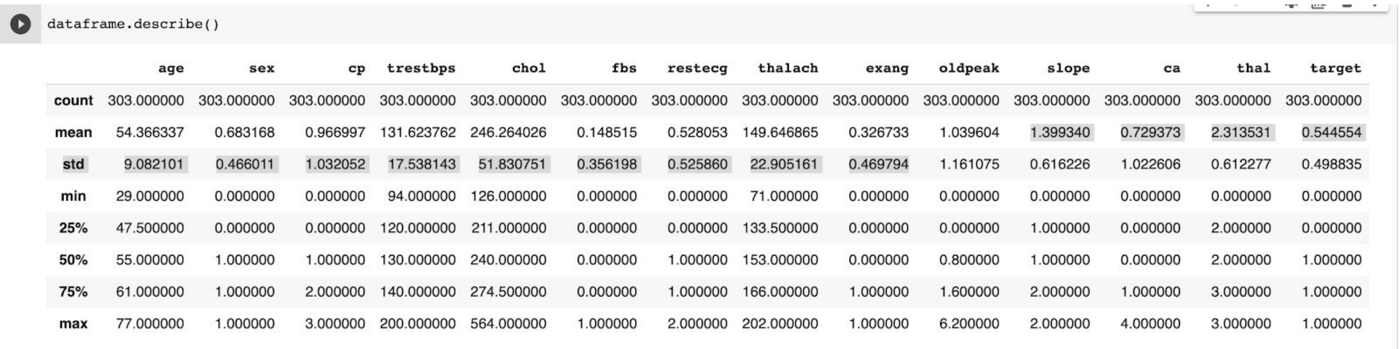
2. Preprocessing

```
import os
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
dataframe = pd.read_csv("heart.csv")
```



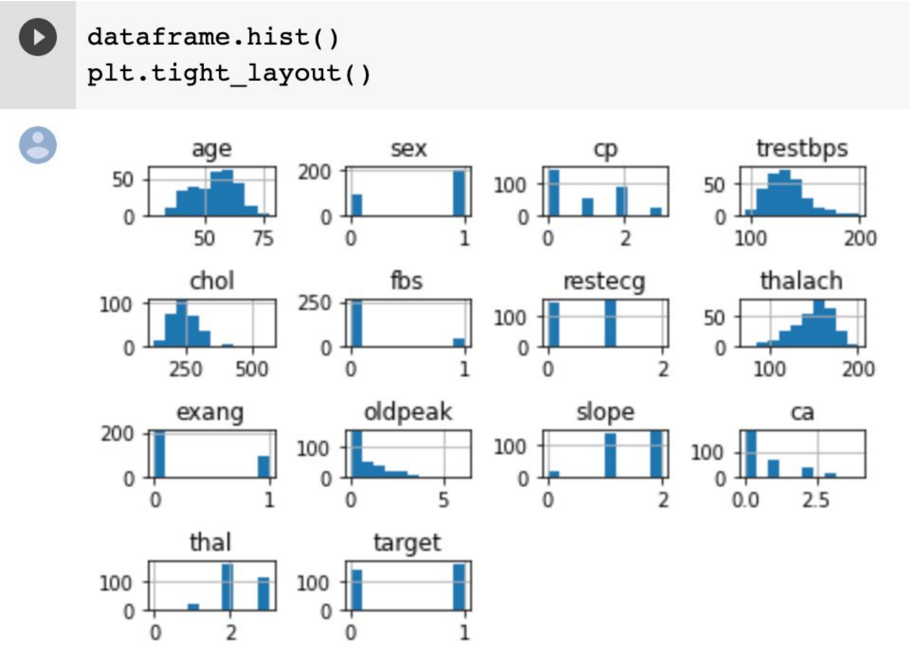
```
dataframe.head()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2	1
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2	1
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2	1
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2	1



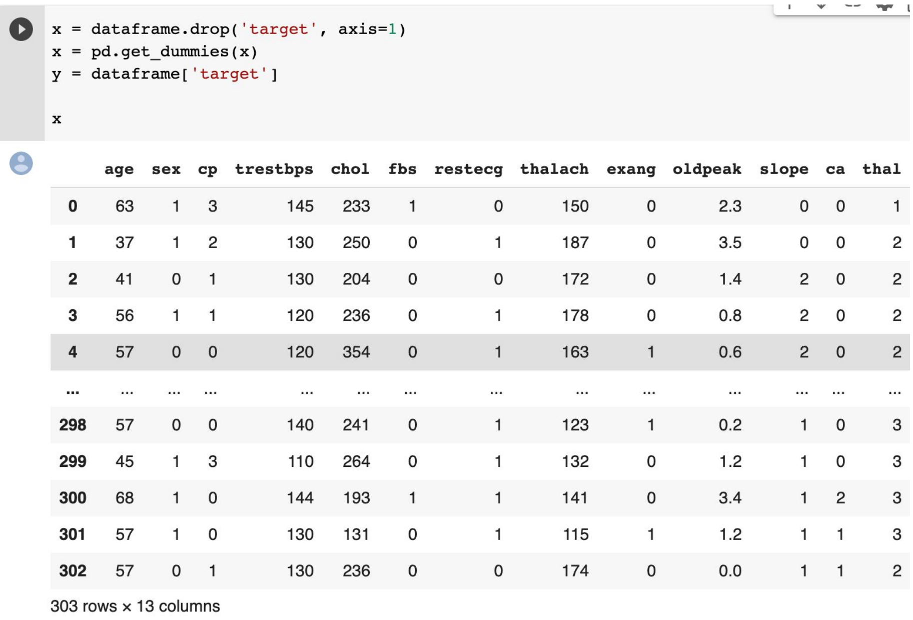
```
dataframe.describe()
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal	target
count	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000	303.000000
mean	54.366337	0.683168	0.966997	131.623762	246.264026	0.148515	0.528053	149.646865	0.326733	1.039604	1.399340	0.729373	2.313531	0.544554
std	9.082101	0.466011	1.032052	17.538143	51.830751	0.356198	0.525860	22.905161	0.469794	1.161075	0.616226	1.022606	0.612277	0.498835
min	29.000000	0.000000	0.000000	94.000000	126.000000	0.000000	0.000000	71.000000	0.000000	0.000000	0.000000	0.000000	0.000000	0.000000
25%	47.500000	0.000000	0.000000	120.000000	211.000000	0.000000	0.000000	133.500000	0.000000	0.000000	1.000000	0.000000	2.000000	0.000000
50%	55.000000	1.000000	1.000000	130.000000	240.000000	0.000000	1.000000	153.000000	0.000000	0.800000	1.000000	0.000000	2.000000	1.000000
75%	61.000000	1.000000	2.000000	140.000000	274.500000	0.000000	1.000000	166.000000	1.000000	1.600000	2.000000	1.000000	3.000000	1.000000
max	77.000000	1.000000	3.000000	200.000000	564.000000	1.000000	2.000000	202.000000	1.000000	6.200000	2.000000	4.000000	3.000000	1.000000



```
dataframe.hist()
plt.tight_layout()
```

```
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import StandardScaler
```



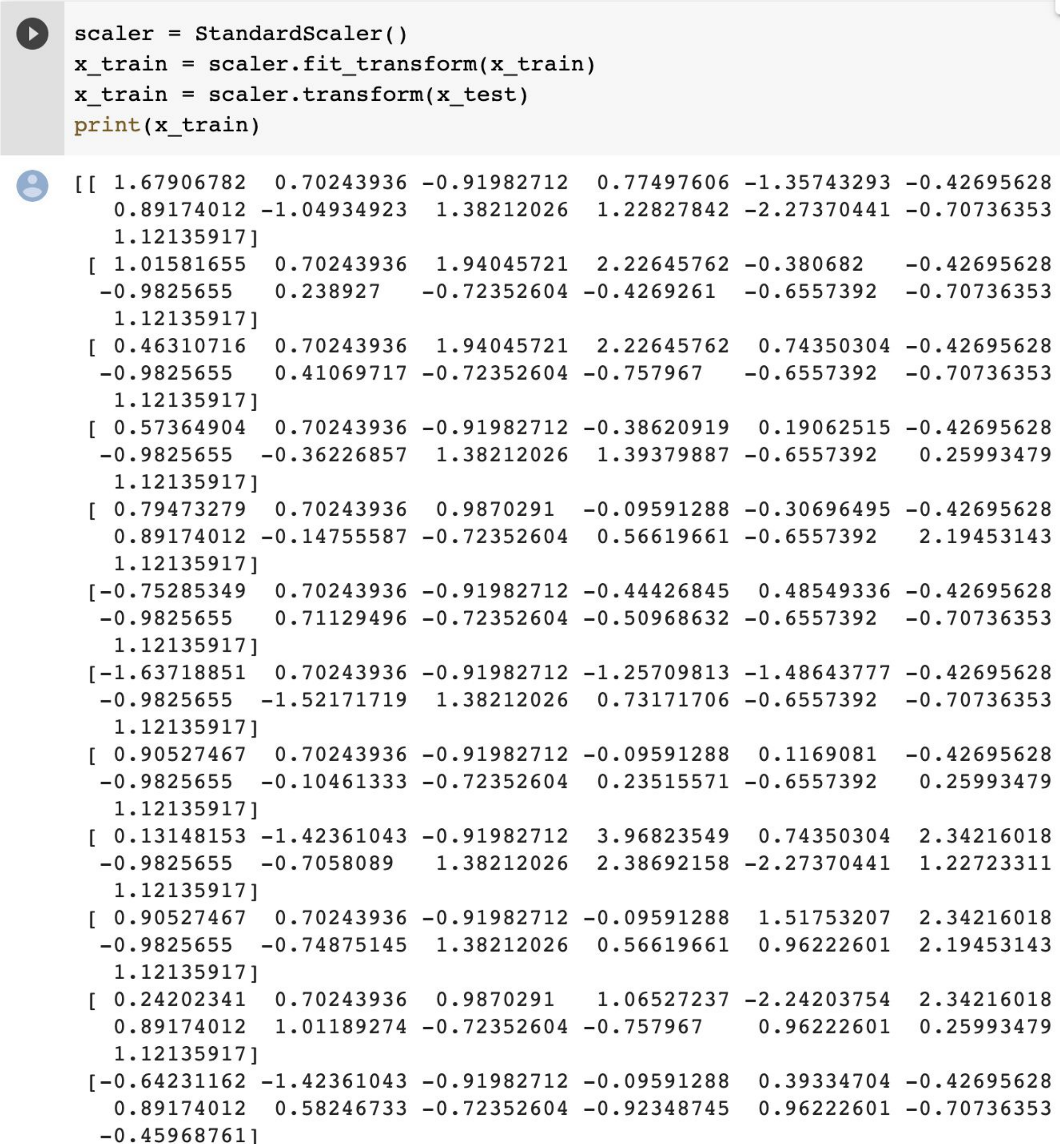
```
x = dataframe.drop('target', axis=1)
x = pd.get_dummies(x)
y = dataframe['target']

x
```

	age	sex	cp	trestbps	chol	fbs	restecg	thalach	exang	oldpeak	slope	ca	thal
0	63	1	3	145	233	1	0	150	0	2.3	0	0	1
1	37	1	2	130	250	0	1	187	0	3.5	0	0	2
2	41	0	1	130	204	0	0	172	0	1.4	2	0	2
3	56	1	1	120	236	0	1	178	0	0.8	2	0	2
4	57	0	0	120	354	0	1	163	1	0.6	2	0	2
...	...	...	...	...	...	...	...	...	...	...	...	...	...
298	57	0	0	140	241	0	1	123	1	0.2	1	0	3
299	45	1	3	110	264	0	1	132	0	1.2	1	0	3
300	68	1	0	144	193	1	1	141	0	3.4	1	2	3
301	57	1	0	130	131	0	1	115	1	1.2	1	1	3
302	57	0	1	130	236	0	0	174	0	0.0	1	1	2

303 rows × 13 columns

```
x_train, x_test, y_train, y_test = train_test_split(x, y, random_state=0)
```

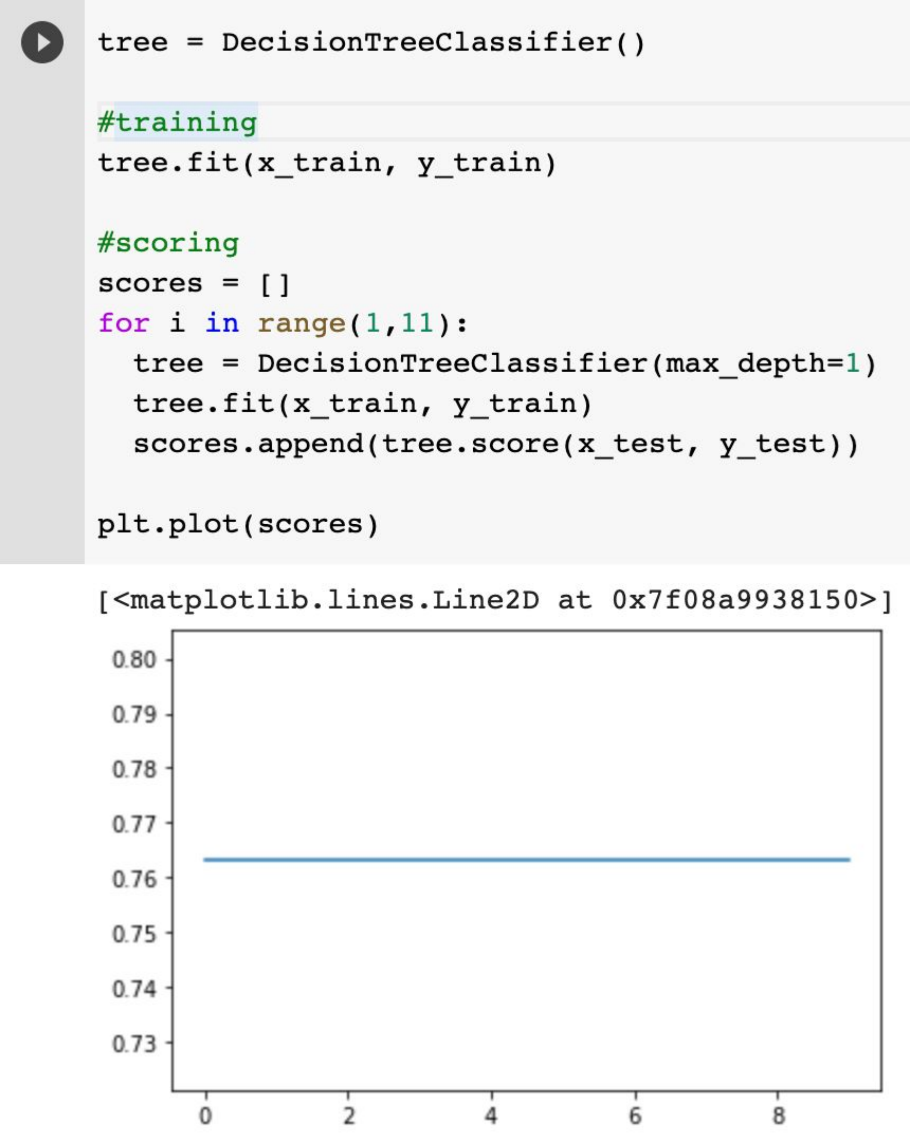


```
scaler = StandardScaler()
x_train = scaler.fit_transform(x_train)
x_train = scaler.transform(x_test)
print(x_train)
```

```
[[ 1.67906782  0.70243936 -0.91982712  0.77497606 -1.35743293 -0.42695628
   0.89174012 -1.04934923  1.38212026  1.22827842 -2.27370441 -0.70736353
   1.12135917]
 [ 1.01581655  0.70243936  1.94045721  2.22645762 -0.380682   -0.42695628
  -0.9825655   0.238927   -0.72352604 -0.4269261  -0.6557392  -0.70736353
   1.12135917]
 [ 0.46310716  0.70243936  1.94045721  2.22645762  0.74350304 -0.42695628
  -0.9825655   0.41069717 -0.72352604 -0.757967   -0.6557392  -0.70736353
   1.12135917]
 [ 0.57364904  0.70243936 -0.91982712 -0.38620919  0.19062515 -0.42695628
  -0.9825655  -0.36226857  1.38212026  1.39379887 -0.6557392   0.25993479
   1.12135917]
 [ 0.79473279  0.70243936  0.9870291  -0.09591288 -0.30696495 -0.42695628
   0.89174012 -0.14755587 -0.72352604  0.56619661 -0.6557392   2.19453143
   1.12135917]
 [-0.75285349  0.70243936 -0.91982712 -0.44426845  0.48549336 -0.42695628
  -0.9825655   0.71129496 -0.72352604 -0.50968632 -0.6557392  -0.70736353
   1.12135917]
 [-1.63718851  0.70243936 -0.91982712 -1.25709813 -1.48643777 -0.42695628
  -0.9825655  -1.52171719  1.38212026  0.73171706 -0.6557392  -0.70736353
   1.12135917]
 [ 0.90527467  0.70243936 -0.91982712 -0.09591288  0.1169081  -0.42695628
  -0.9825655  -0.10461333 -0.72352604  0.23515571 -0.6557392   0.25993479
   1.12135917]
 [ 0.13148153 -1.42361043 -0.91982712  3.96823549  0.74350304  2.34216018
  -0.9825655  -0.7058089   1.38212026  2.38692158 -2.27370441  1.22723311
   1.12135917]
 [ 0.90527467  0.70243936 -0.91982712 -0.09591288  1.51753207  2.34216018
  -0.9825655  -0.74875145  1.38212026  0.56619661  0.96222601  2.19453143
   1.12135917]
 [ 0.24202341  0.70243936  0.9870291   1.06527237 -2.24203754  2.34216018
   0.89174012  1.01189274 -0.72352604 -0.757967    0.96222601  0.25993479
   1.12135917]
 [-0.64231162 -1.42361043 -0.91982712 -0.09591288  0.39334704 -0.42695628
   0.89174012  0.58246733 -0.72352604 -0.92348745  0.96222601 -0.70736353
  -0.45968761]
```

3. Prediktif Modelling

```
from sklearn.tree import DecisionTreeClassifier
from sklearn.tree import DecisionTreeRegressor
from sklearn.ensemble import RandomForestClassifier
from sklearn.ensemble import RandomForestRegressor
from sklearn.metrics import f1_score, roc_auc_score
```



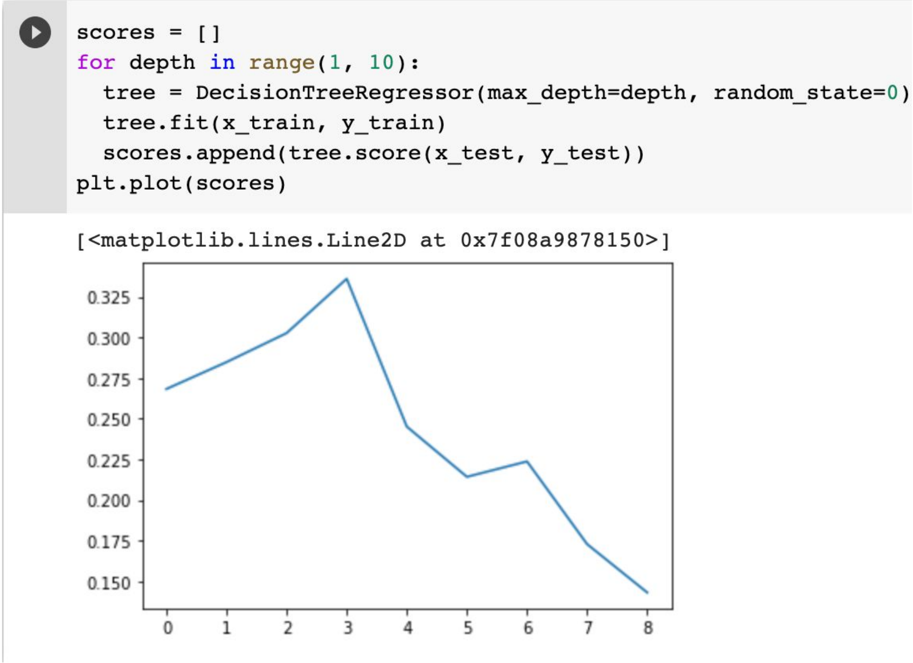
```
tree = DecisionTreeClassifier()

#training
tree.fit(x_train, y_train)

#scoring
scores = []
for i in range(1,11):
  tree = DecisionTreeClassifier(max_depth=1)
  tree.fit(x_train, y_train)
  scores.append(tree.score(x_test, y_test))

plt.plot(scores)
```

```
[<matplotlib.lines.Line2D at 0x7f08a9938150>]
```



```
scores = []
for depth in range(1, 10):
  tree = DecisionTreeRegressor(max_depth=depth, random_state=0)
  tree.fit(x_train, y_train)
  scores.append(tree.score(x_test, y_test))
plt.plot(scores)
```

```
[<matplotlib.lines.Line2D at 0x7f08a9878150>]
```