In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pylab as plt
import seaborn as sns
from scipy import stats as st


n = pd.read_csv("notes.csv")
n.head()
```

Out[1]:

	is_genuine	diagonal	height_left	height_right	margin_low	margin_up	length
0	True	171.81	104.86	104.95	4.52	2.89	112.83
1	True	171.67	103.74	103.70	4.01	2.87	113.29
2	True	171.83	103.76	103.76	4.40	2.88	113.84
3	True	171.80	103.78	103.65	3.73	3.12	113.63
4	True	172.05	103.70	103.75	5.04	2.27	113.55

In [2]:
```
n.dtypes
```

Out[2]:
```
is_genuine        bool
diagonal       float64
height_left    float64
height_right   float64
margin_low     float64
margin_up      float64
length         float64
dtype: object
```

In [3]:
```
n.index
```

Out[3]:
```
RangeIndex(start=0, stop=170, step=1)
```

In [4]:
```
n.isna().any()
```

Out[4]:
```
is_genuine      False
diagonal        False
height_left     False
height_right    False
margin_low      False
margin_up       False
length          False
dtype: bool
```

In [5]:
```
n.isnull().any()
```

Out[5]:
```
is_genuine      False
diagonal        False
height_left     False
height_right    False
margin_low      False
margin_up       False
length          False
dtype: bool
```

In [6]:
```
n.duplicated().any()
```

Out[6]:
```
False
```

pas de données manquants ni dupliqué(sauf pour is_genuine car il y a True et False)

In [7]:
```
#J'ajoute une colonne pour la largeur et la superficie
#(a**2 + b**2 = c**2, b**2 = c**2 - a**2 ou largeur = R2(diagonal**2 - longeur**2))
n["large"] = round(np.sqrt((n["diagonal"]**2) - n["length"]**2), 2) #colonne largeur
n["A"] = round(n["length"] * n["large"], 2) #colonne superficie
n.head()
```

Out[7]:

	is_genuine	diagonal	height_left	height_right	margin_low	margin_up	length	large	A
0	True	171.81	104.86	104.95	4.52	2.89	112.83	129.57	14619.38
1	True	171.67	103.74	103.70	4.01	2.87	113.29	128.98	14612.14
2	True	171.83	103.76	103.76	4.40	2.88	113.84	128.71	14652.35
3	True	171.80	103.78	103.65	3.73	3.12	113.63	128.85	14641.23
4	True	172.05	103.70	103.75	5.04	2.27	113.55	129.26	14677.47

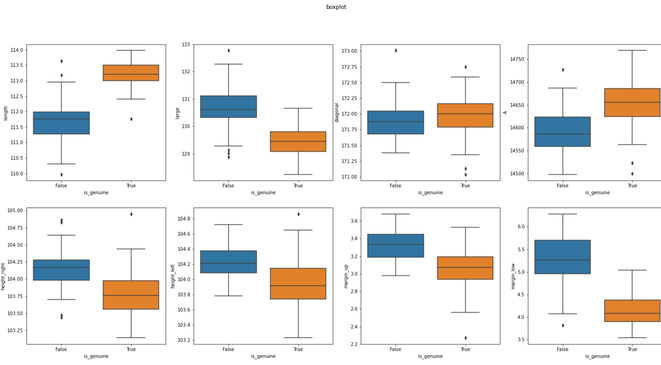
In [8]:
```
n.describe(include = "all") #affichage de la dispersion
```

Out[8]:

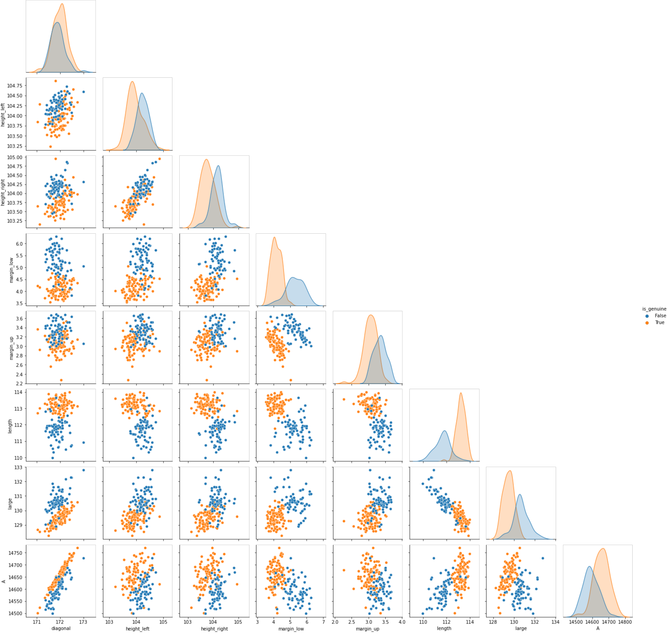
	is_genuine	diagonal	height_left	height_right	margin_low	margin_up	length	large	A
count	170	170.000000	170.000000	170.000000	170.000000	170.000000	170.000000	170.000000	170.000000
unique	2	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
top	True	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
freq	100	NaN	NaN	NaN	NaN	NaN	NaN	NaN	NaN
mean	NaN	171.940588	104.066353	103.928118	4.612118	3.170412	112.570412	129.961235	14629.087412
std	NaN	0.305768	0.298185	0.330980	0.702103	0.236361	0.924448	0.864020	56.986367
min	NaN	171.040000	103.230000	103.140000	3.540000	2.270000	109.970000	128.250000	14498.440000
25%	NaN	171.730000	103.842500	103.690000	4.050000	3.012500	111.855000	129.290000	14588.145000
50%	NaN	171.945000	104.055000	103.950000	4.450000	3.170000	112.845000	129.845000	14631.570000
75%	NaN	172.137500	104.287500	104.170000	5.127500	3.330000	113.287500	130.540000	14671.255000
max	NaN	173.010000	104.860000	104.950000	6.280000	3.680000	113.980000	132.780000	14768.950000

Analyse des données uni et bivaries

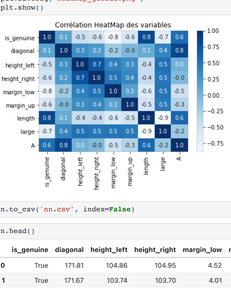
In [9]:
```
fig, axes = plt.subplots(2, 4, figsize=(25, 12))

fig.suptitle('boxplot')


sns.boxplot(ax=axes[0, 0], data=n, x="is_genuine", y='length')
sns.boxplot(ax=axes[0, 1], data=n, x="is_genuine", y='large')
sns.boxplot(ax=axes[0, 2], data=n, x="is_genuine", y='diagonal')
sns.boxplot(ax=axes[0, 3], data=n, x="is_genuine", y='A')
sns.boxplot(ax=axes[1, 0], data=n, x="is_genuine", y='height_right')
sns.boxplot(ax=axes[1, 1], data=n, x="is_genuine", y='height_left')
sns.boxplot(ax=axes[1, 2], data=n, x="is_genuine", y="margin_up")
sns.boxplot(ax=axes[1, 3], data=n, x="is_genuine", y="margin_low")
plt.savefig("boxplot.png")
```



In [10]:
```
sns.pairplot(n, hue='is_genuine', corner=True)
plt.savefig("paisplot.png")
plt.show()
```



In [11]:
```
#Nous pouvons rendre le tableau de corr plus visible avec une heatmap, ajoutant des couleurs qui indiquent
#le degrée de corr.
#Méthode .corr() avec par défaut la formule de la corrélation linéaire de Pearson
sns.heatmap(n.corr(), annot=True, fmt='.1f', cmap='Blues')#(annot=marquage du #, fmt=nr apres virgule, cmap=colormap)
plt.title('Corrélation HeatMap des variables')
plt.savefig('heatmap_global.png')
plt.show()
```



In [12]:
```
n.to_csv('nn.csv', index=False)
```

In [13]:
```
n.head()
```

Out[13]:

	is_genuine	diagonal	height_left	height_right	margin_low	margin_up	length	large	A
0	True	171.81	104.86	104.95	4.52	2.89	112.83	129.57	14619.38
1	True	171.67	103.74	103.70	4.01	2.87	113.29	128.98	14612.14
2	True	171.83	103.76	103.76	4.40	2.88	113.84	128.71	14652.35
3	True	171.80	103.78	103.65	3.73	3.12	113.63	128.85	14641.23
4	True	172.05	103.70	103.75	5.04	2.27	113.55	129.26	14677.47

In [ ]:
```

```