作品二、Python 迴圈、矩陣與外部檔案的練習

410978040 統計三 黃冠翔

本作品藉由繪圖和製作卡方表練習(1)迴圈與矩陣的運算、(2)繪製散佈圖、(3)線性代數的矩陣運算處理重複性計算、(4)利用 Pandas 與 Scipy 套件，處裡資料、(5)將矩陣儲存到 EXCEL 檔案

1.繪製下圖（線條顏色、符號與數量都可以由程式輕易變更）

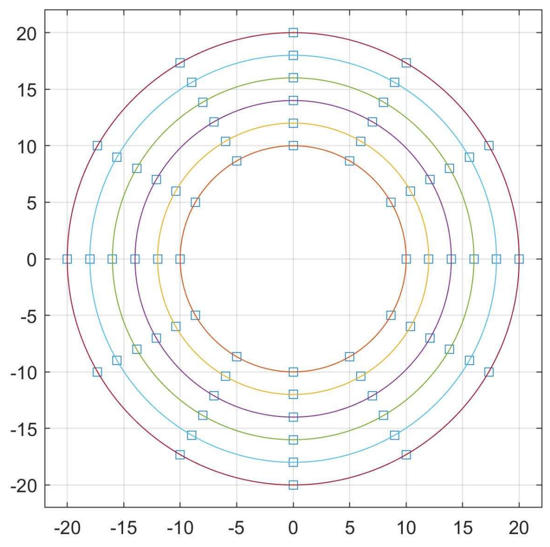
繪製6圈的同心圓，半徑分別為10、12、14、16、18和20，並在各圈的圓上均勻畫上12個方框。

注意事項:
將數量的決定放在 code 第一條，譬如，n = 6

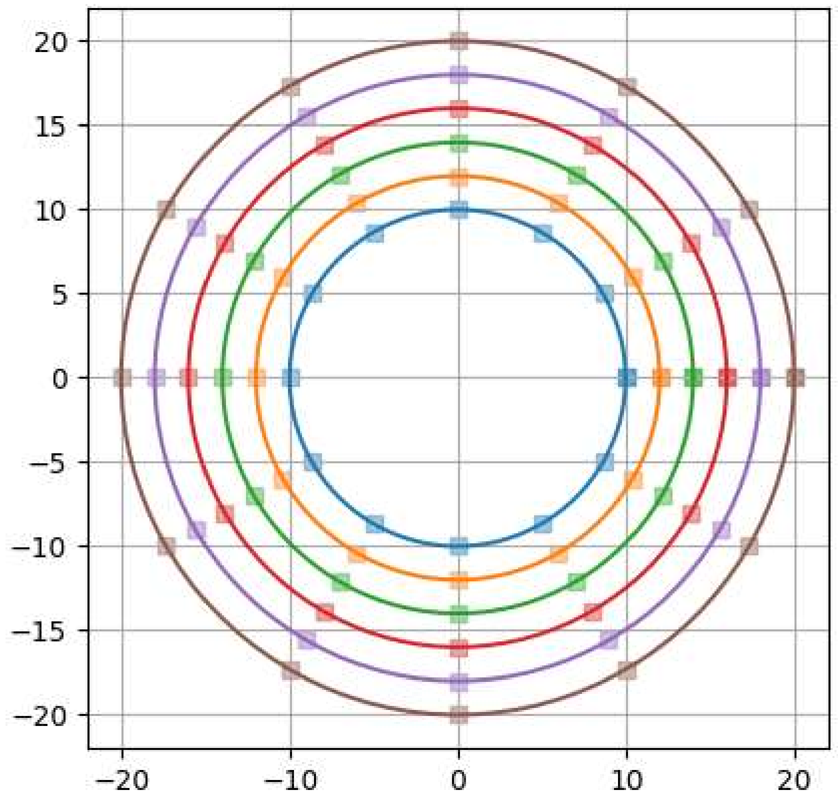
將符號的決定放在 code 第二條，譬如，marker = ‘s’

不論畫同心圓或圓上方塊符號，可以採用迴圈方式或非迴圈的矩陣計算方式。最好兩者都試試看（從自己最有把握的方法先做），才知道 python 的長處（可以先不論顏色）。

上述方式可以迅速變更設定，看到結果。



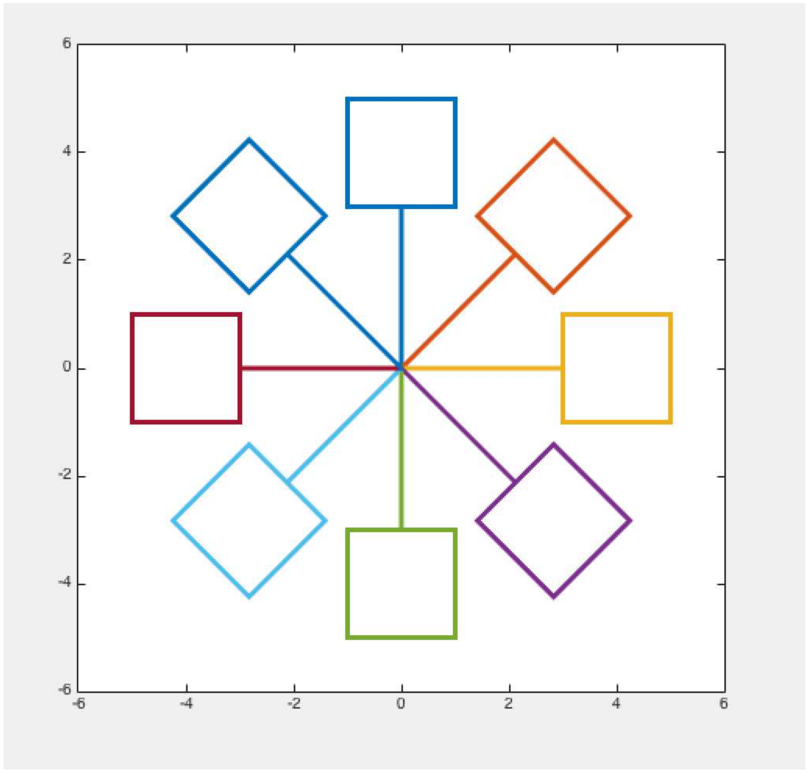
In [ ]:
```
from cmath import sqrt
import numpy as np
import matplotlib.pyplot as plt
import turtle

figure, ax = plt.subplots()

r=np.linspace(10, 20, 6)
theta=np.linspace(0, 2*np.pi, 100)
theta1=np.linspace(0, 2*np.pi, 13)
for i in r:
    x=i*np.cos(theta)
    y=i*np.sin(theta)
    plt.plot(x, y)

for i in r:
    x=i*np.cos(theta1)
    y=i*np.sin(theta1)
    plt.scatter(x, y, marker='s', alpha=0.4)

ax.set_aspect(1)
ax.set_xlim([-22, 22])
ax.set_ylim([-22, 22])
ax.grid(True)
```

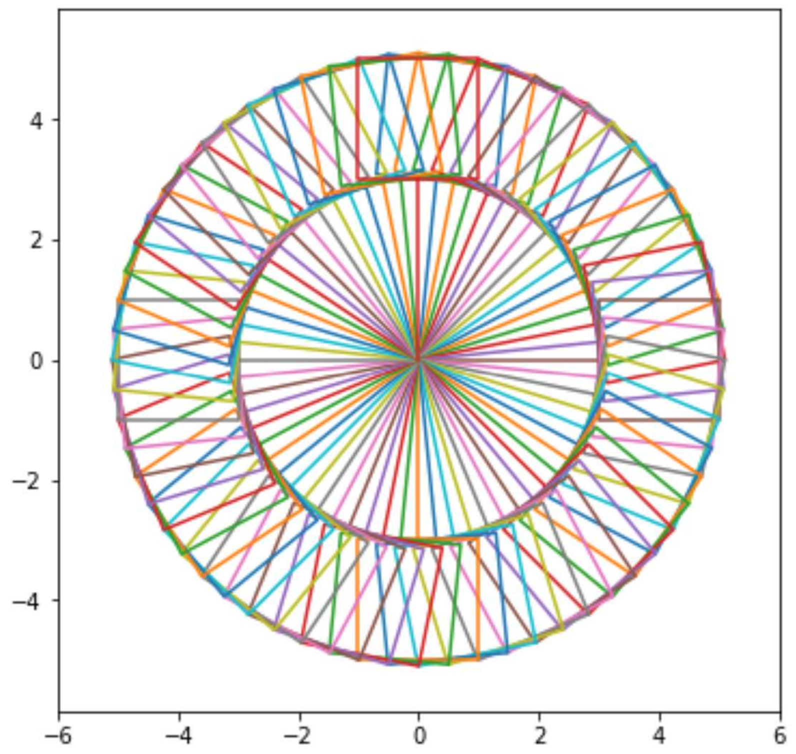


2. 繪製下圖（線條顏色與數量都可以由程式輕易決定）

注意事項:

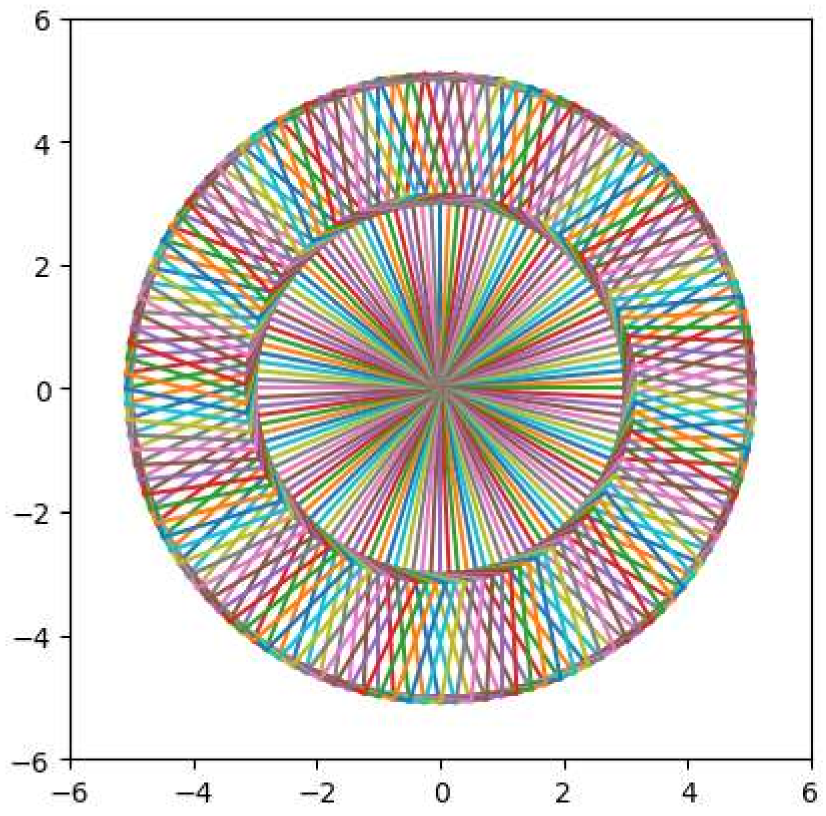
將方框數量的決定放在 code 第一條，譬如，n = 8，改變 n 值，便能看到結果的改變，譬如，n = 128 得到右圖。


6
4
2
0
-2
-4
-6
-6
-4
-2
0
2
4
6



In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt

figure, ax = plt.subplots()

x=[0, 0, -1, -1, 1, 1, 0]
y=[0, 3, 3, 5, 5, 3, 3]
ax.set_xlim([-6, 6])
ax.set_ylim([-6, 6])
S=np.array([[0, 0, -1, -1, 1, 1, 0], [0, 3, 3, 5, 5, 3, 3]])

#n=8
n=128

for i in range(n):
    A=np.array([[np.cos(i*np.pi/(n/2)), np.sin(i*np.pi/(n/2))], [-np.sin(i*np.pi/(n/2)), np.cos(i*np.pi/(n/2))]]).dot(S
    B=A[0, ]
    C=A[1, ]
    plt.plot(B, C)
```

```
ax.set_aspect(1)
```

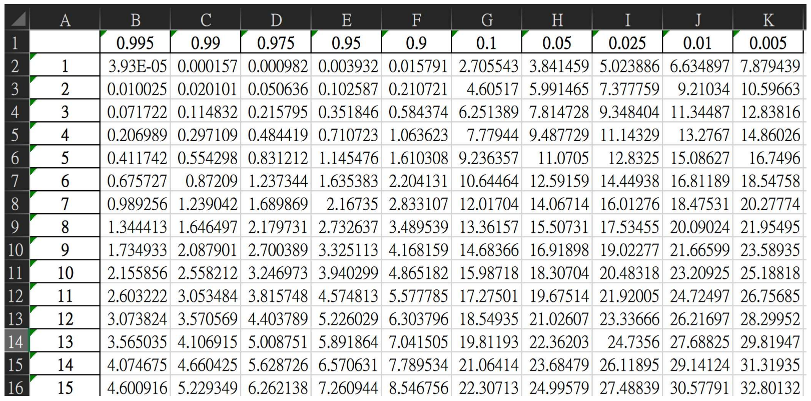


3.計算如下右圖的卡方右尾面積與自由度對照表，並輸出到 **EXCEL** 檔，檔名為：**Chi2Table.xlsx**，含欄與列的名稱，如下左圖。

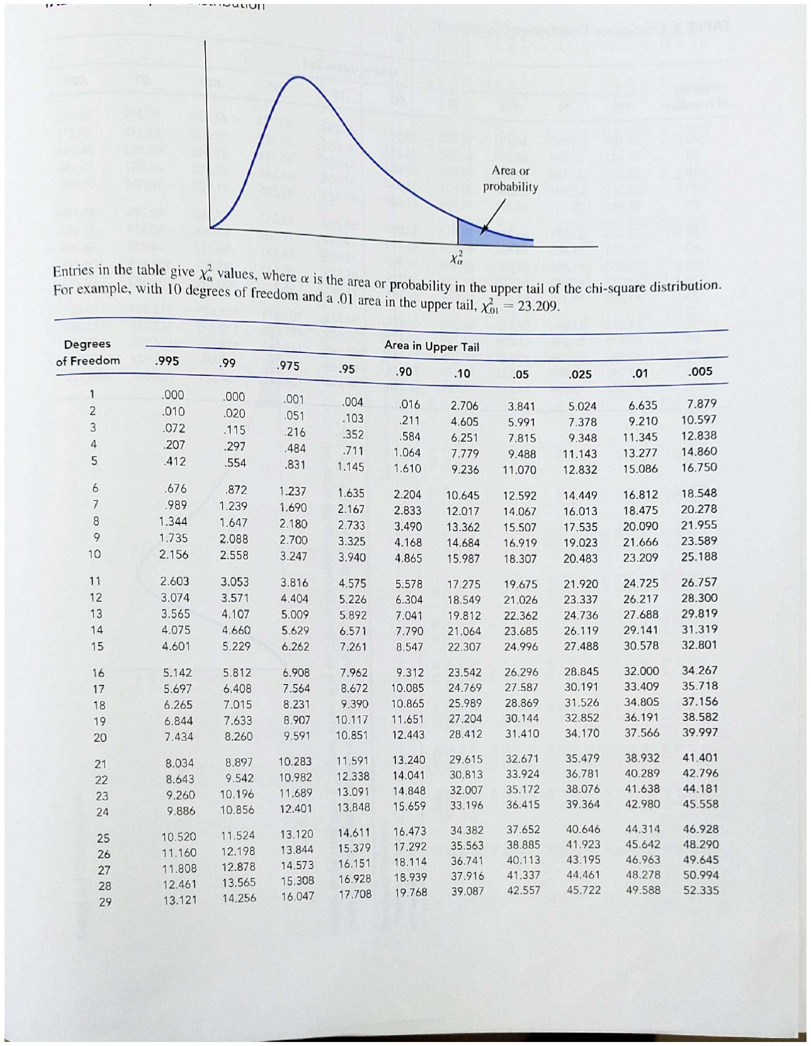
注意事項:

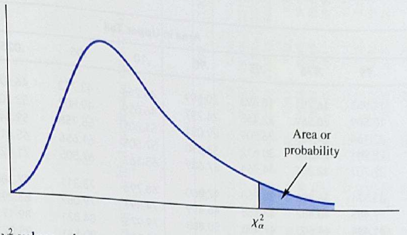
提示的套件與程式碼如下方 code 區所示。

利用 pandas 將矩陣內容儲存到 EXCEL 檔，製作方式請參考 pandas 手冊。



	A	B	C	D	E	F	G	H	I	J	K
1		0.995	0.99	0.975	0.95	0.9	0.1	0.05	0.025	0.01	0.005
2	1	3.93E-05	0.000157	0.000982	0.003932	0.015791	2.705543	3.841459	5.023886	6.634897	7.879439
3	2	0.010025	0.020101	0.050636	0.102587	0.210721	4.60517	5.991465	7.377759	9.21034	10.59663
4	3	0.071722	0.114832	0.215795	0.351846	0.584374	6.251389	7.814728	9.348404	11.34487	12.83816
5	4	0.206989	0.297109	0.484419	0.710723	1.063623	7.77944	9.487729	11.14329	13.2767	14.86026
6	5	0.411742	0.554298	0.831212	1.145476	1.610308	9.236357	11.0705	12.8325	15.08627	16.7496
7	6	0.675727	0.87209	1.237344	1.635383	2.204131	10.64464	12.59159	14.44938	16.81189	18.54758
8	7	0.989256	1.239042	1.689869	2.16735	2.833107	12.01704	14.06714	16.01276	18.47531	20.27774
9	8	1.344413	1.646497	2.179731	2.732637	3.489539	13.36157	15.50731	17.53455	20.09024	21.95495
10	9	1.734933	2.087901	2.700389	3.325113	4.168159	14.68366	16.91898	19.02277	21.66599	23.58935
11	10	2.155856	2.558212	3.246973	3.940299	4.865182	15.98718	18.30704	20.48318	23.20925	25.18818
12	11	2.603222	3.053484	3.815748	4.574813	5.577785	17.27501	19.67514	21.92005	24.72497	26.75685
13	12	3.073824	3.570569	4.403789	5.226029	6.303796	18.54935	21.02607	23.33666	26.21697	28.29952
14	13	3.565035	4.106915	5.008751	5.891864	7.041505	19.81193	22.36203	24.7356	27.68825	29.81947
15	14	4.074675	4.660425	5.628726	6.570631	7.789534	21.06414	23.68479	26.11895	29.14124	31.31935
16	15	4.600916	5.229349	6.262138	7.260944	8.546756	22.30713	24.99579	27.48839	30.57791	32.80132





Entries in the table give χ^2_α values, where α is the area or probability in the upper tail of the chi-square distribution. For example, with 10 degrees of freedom and a .01 area in the upper tail, $\chi^2_{.01} = 23.209$.

Degrees	Area in Upper Tail									
of Freedom	.995	.99	.975	.95	.90	.10	.05	.025	.01	.005
1	.000	.000	.001	.004	.016	2.706	3.841	5.024	6.635	7.879
2	.010	.020	.051	.103	.211	4.605	5.991	7.378	9.210	10.597
3	.072	.115	.216	.352	.584	6.251	7.815	9.348	11.345	12.838
4	.207	.297	.484	.711	1.064	7.779	9.488	11.143	13.277	14.860
5	.412	.554	.831	1.145	1.610	9.236	11.070	12.832	15.086	16.750
6	.676	.872	1.237	1.635	2.204	10.645	12.592	14.449	16.812	18.548
7	.989	1.239	1.690	2.167	2.833	12.017	14.067	16.013	18.475	20.278
8	1.344	1.647	2.180	2.733	3.490	13.362	15.507	17.535	20.090	21.955
9	1.735	2.088	2.700	3.325	4.168	14.684	16.919	19.023	21.666	23.589
10	2.156	2.558	3.247	3.940	4.865	15.987	18.307	20.483	23.209	25.188
11	2.603	3.053	3.816	4.575	5.578	17.275	19.675	21.920	24.725	26.757
12	3.074	3.571	4.404	5.226	6.304	18.549	21.026	23.337	26.217	28.300
13	3.565	4.107	5.009	5.892	7.041	19.812	22.362	24.736	27.688	29.819
14	4.075	4.660	5.629	6.571	7.790	21.064	23.685	26.119	29.141	31.319
15	4.601	5.229	6.262	7.261	8.547	22.307	24.996	27.488	30.578	32.801
16	5.142	5.812	6.908	7.962	9.312	23.542	26.296	28.845	32.000	34.267
17	5.697	6.408	7.564	8.672	10.085	24.769	27.587	30.191	33.409	35.718
18	6.265	7.015	8.231	9.390	10.865	25.989	28.869	31.526	34.805	37.156
19	6.844	7.633	8.907	10.117	11.651	27.204	30.144	32.852	36.191	38.582
20	7.434	8.260	9.591	10.851	12.443	28.412	31.410	34.170	37.566	39.997
21	8.034	8.897	10.283	11.591	13.240	29.615	32.671	35.479	38.932	41.401
22	8.643	9.542	10.982	12.338	14.041	30.813	33.924	36.781	40.289	42.796
23	9.260	10.196	11.689	13.091	14.848	32.007	35.172	38.076	41.638	44.181
24	9.886	10.856	12.401	13.848	15.659	33.196	36.415	39.364	42.980	45.558
25	10.520	11.524	13.120	14.611	16.473	34.382	37.652	40.646	44.314	46.928
26	11.160	12.198	13.844	15.379	17.292	35.563	38.885	41.923	45.642	48.290
27	11.808	12.878	14.573	16.151	18.114	36.741	40.113	43.195	46.963	49.645
28	12.461	13.565	15.308	16.928	18.939	37.916	41.337	44.461	48.278	50.994
29	13.121	14.256	16.047	17.708	19.768	39.087	42.557	45.722	49.588	52.335

In [ ]:

```
from scipy.stats import chi2
import pandas as pd
import numpy as np
import math

#F = 0.995 # cumulative to 1
#df = 1
#x = chi2.ppf(F, df)  # inverse of CDF
#print(x)

df=np.linspace(1, 29, 29)
F=np.array([[0.995], [0.99], [0.975], [0.95], [0.9], [0.1], [0.05], [0.025], [0.01], [0.005]])
x = chi2.ppf(1-F, df)
A = pd.DataFrame(x)
B=A.T
B.columns=["0.995", "0.99", "0.975", "0.95", "0.9", "0.1", "0.05", "0.025", "0.01", "0.005"]
B.index=[np.linspace(1, 29, 29)]
B

file_name = 'Chi2Table.xlsx'
B.to_excel(file_name)

#for i in F:
 #   for j in df:
  #      x=chi2.ppf(1-i, j)
   #     print(x)
```