# 第二次作品：Python 函數繪製的觀念與技巧

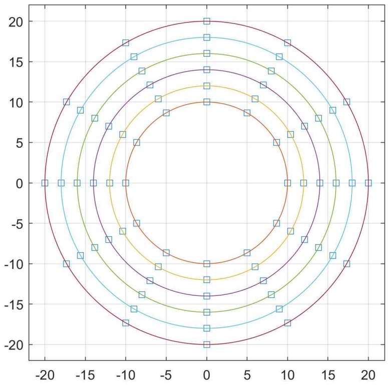
學號：411073088

姓名：陳敬翰

---

作品目標：利用 Python 函數繪製的觀念與技巧，繪製出一個簡單的圖形，說明程式碼的功能與用途。

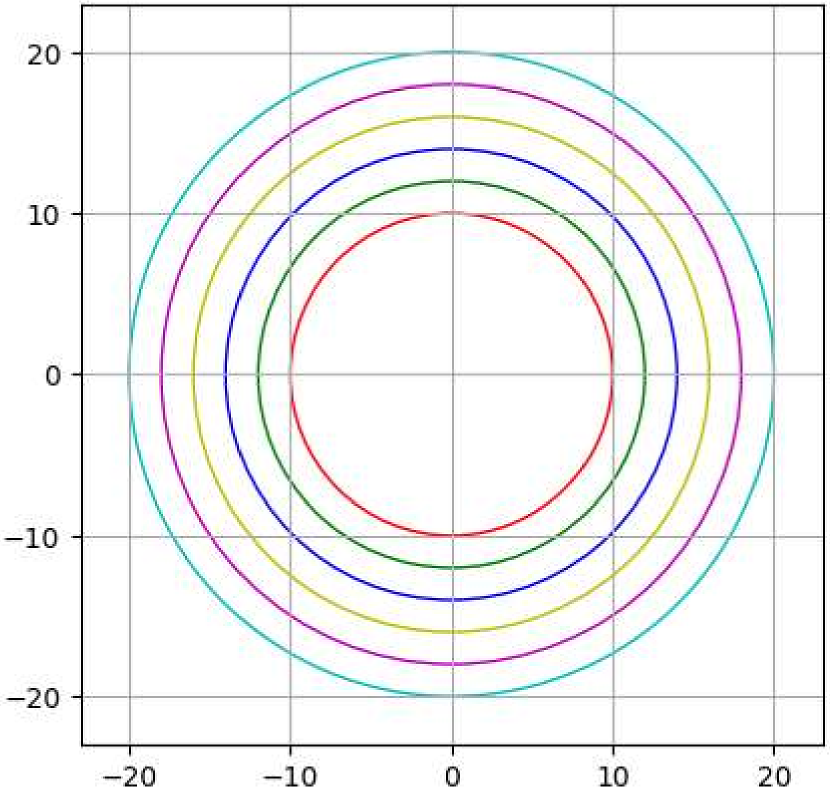
1. 繪製函數



從半徑10到20做6個圓

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

# 設定圖片大小
plt.figure(figsize=(20,2))

fig, ax = plt.subplots()

# 圓形
radius = np.linspace(10, 20, 6)

color = ['r', 'g', 'b', 'y', 'm', 'c']

for i in range(6):
    circle = patches.Circle((0, 0), radius=radius[i], fill=False, color=color[i])
    ax.add_patch(circle)

plt.gca().set_aspect('equal', adjustable='box')

ax.set_xlim(-23, 23)
ax.set_ylim(-23, 23)

plt.grid(True)

plt.show()
```

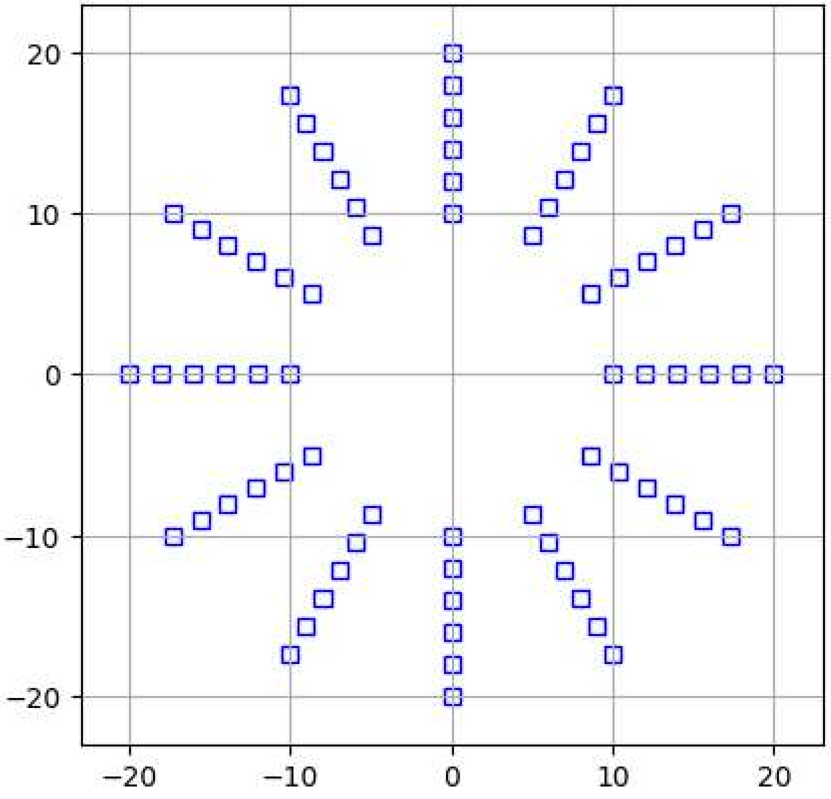
```
<Figure size 2000x200 with 0 Axes>
```



根據座標做正方形

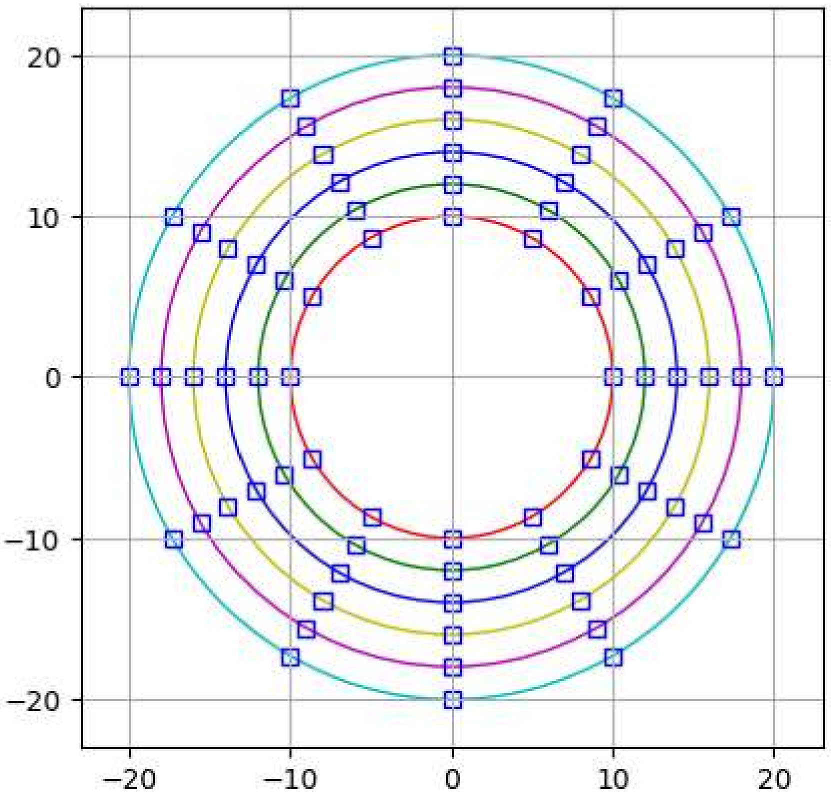
In [ ]:

```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.patches as patches


fig, ax = plt.subplots()

radius = np.linspace(10, 20, 6)

plt.gca().set_aspect('equal', adjustable='box')

# 正方形
square_x = np.array([1, 3**0.5/2, 1/2, 0, -1/2, -3**0.5/2, -1, -3**0.5/2, -1/2, 0,
square_y = np.array([0, 1/2, 3**0.5/2, 1, 3**0.5/2, 1/2, 0, -1/2, -3**0.5/2, -1, -

for i in radius:
    for j in range(len(square_x)):
        square = patches.Rectangle((square_x[j]*i-0.5, square_y[j]*i-0.5), 1, 1, f
        ax.add_patch(square)

ax.set_xlim(-23, 23)
ax.set_ylim(-23, 23)

plt.grid(True)

plt.show()
```



In [ ]:

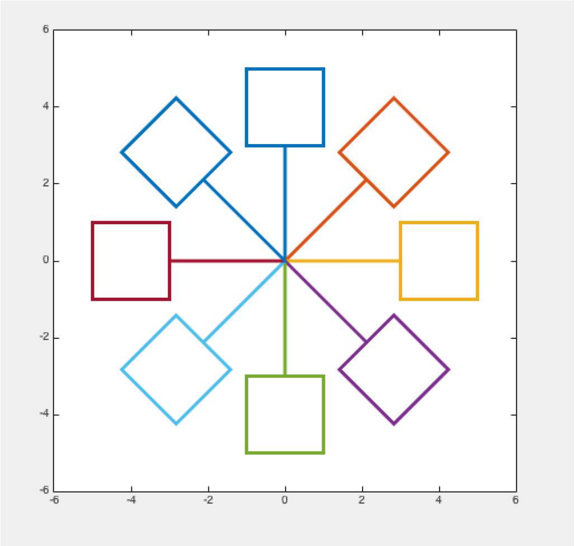
```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.patches as patches


fig, ax = plt.subplots()

# 圓形
n = 6
radius = np.linspace(10, 20, n)
color = ['r', 'g', 'b', 'y', 'm', 'c']

for i in range(n):
    circle = patches.Circle((0, 0), radius=radius[i], fill=False, color=color[i])
    ax.add_patch(circle)

plt.gca().set_aspect('equal', adjustable='box')

# 正方形
square_x = np.array([1, 3**0.5/2, 1/2, 0, -1/2, -3**0.5/2, -1, -3**0.5/2, -1/2, 0,
square_y = np.array([0, 1/2, 3**0.5/2, 1, 3**0.5/2, 1/2, 0, -1/2, -3**0.5/2, -1, -

for i in radius:
    for j in range(len(square_x)):
        square = patches.Rectangle((square_x[j]*i-0.5, square_y[j]*i-0.5), 1, 1, f
        ax.add_patch(square)

ax.set_xlim(-23, 23)
ax.set_ylim(-23, 23)

plt.grid(True)

plt.show()
```



**注意事項與討論：**

- 本作品的函數繪製，皆以 Python 的 `matplotlib.pyplot` 模組為主，並以 `numpy` 模組的 `linspace` 函數產生 $x$ 值的陣列，以及 `array` 函數產生 $y$ 值的陣列。

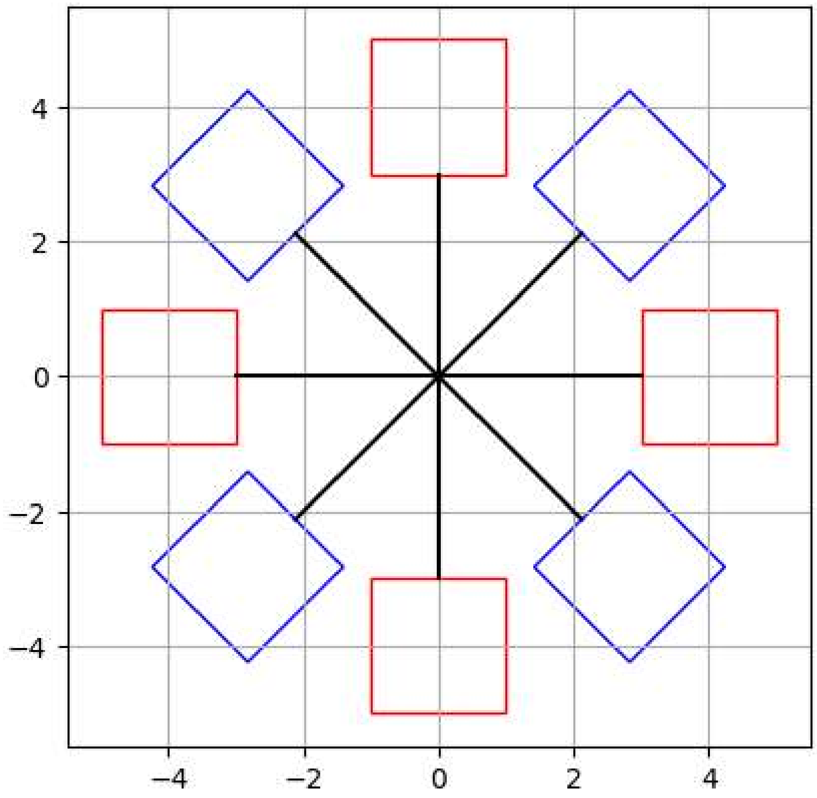
## 2. 繪製函數



暴力解

In [ ]:

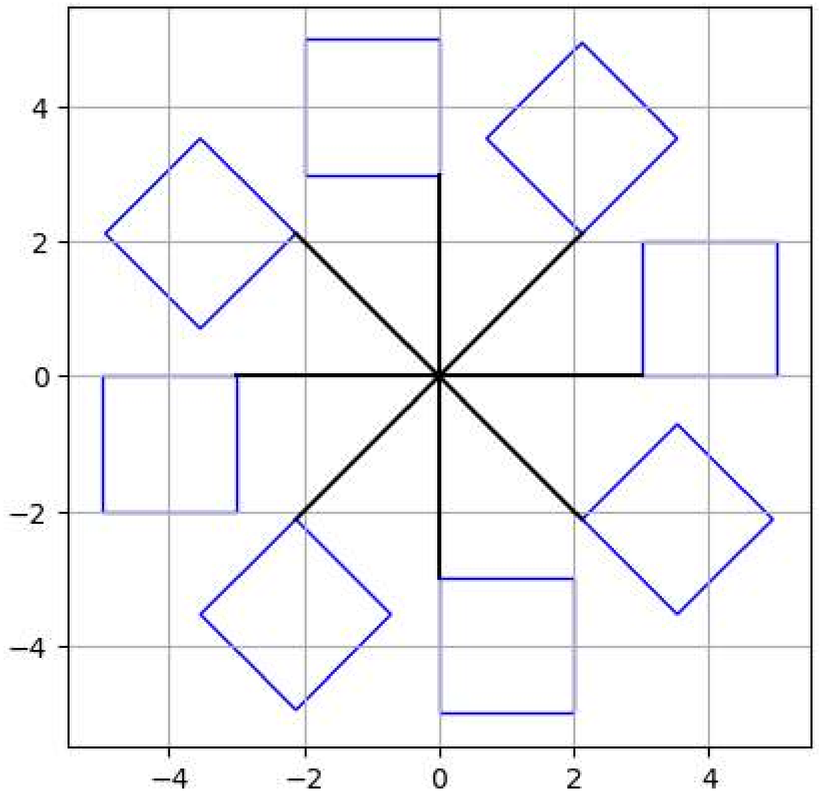
```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.patches as patches


fig, ax = plt.subplots()

plt.plot([0,0],[3,-3],color='k')
plt.plot([3,-3],[0,0],color='k')
plt.plot([-3/2**0.5,3/2**0.5],[3/2**0.5,-3/2**0.5],color='k')
plt.plot([-3/2**0.5,3/2**0.5],[-3/2**0.5,3/2**0.5],color='k')


square = patches.Rectangle((-1, 3), 2, 2, fill=False, color='r')
ax.add_patch(square)

square = patches.Rectangle((-1, -5), 2, 2, fill=False, color='r')
ax.add_patch(square)

square = patches.Rectangle((3, -1), 2, 2, fill=False, color='r')
ax.add_patch(square)

square = patches.Rectangle((-5, -1), 2, 2, fill=False, color='r')
ax.add_patch(square)

square = patches.Rectangle((4/(2**0.5), 2/(2**0.5)), 2, 2, fill=False, color='b', 
ax.add_patch(square)

square = patches.Rectangle((4/(2**0.5), -6/(2**0.5)), 2, 2, fill=False, color='b',
ax.add_patch(square)

square = patches.Rectangle((-4/(2**0.5), 2/(2**0.5)), 2, 2, fill=False, color='b',
ax.add_patch(square)

square = patches.Rectangle((-4/(2**0.5), -6/(2**0.5)), 2, 2, fill=False, color='b'
ax.add_patch(square)


plt.gca().set_aspect('equal', adjustable='box')

plt.grid(True)
plt.show()
```



讓n變成變數，但正方形沒有對齊

In [ ]:

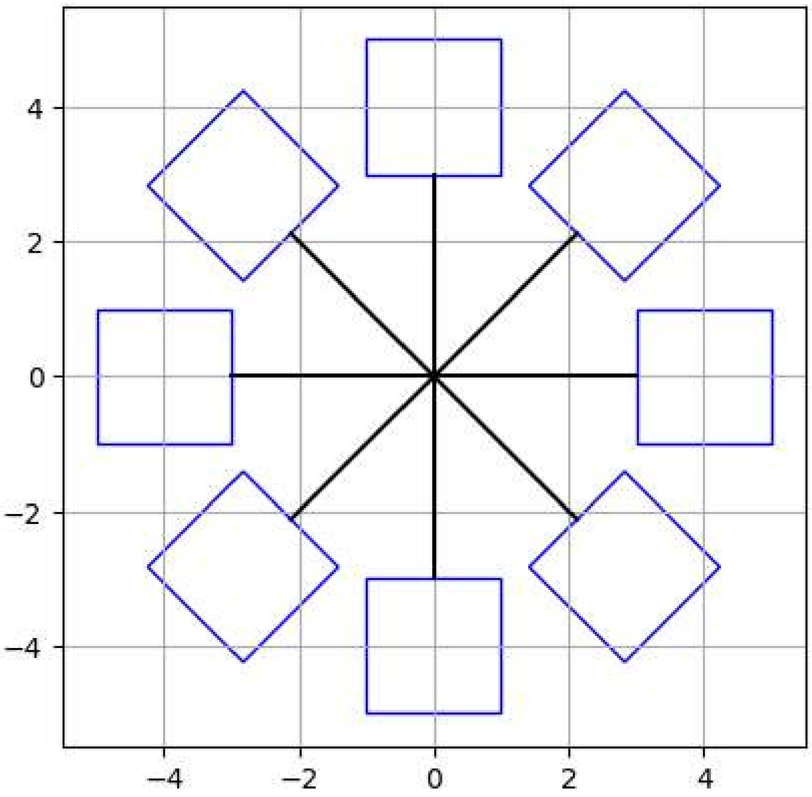
```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.patches as patches

fig, ax = plt.subplots()
n = 8
angle = 360 / n
radius = 3



for i in range(n):
    plt.plot([0, radius*np.cos(np.deg2rad(angle*i))], [0, radius*np.sin(np.deg2rad(
    ori_x = radius*np.cos(np.deg2rad(angle*i))
    ori_y = radius*np.sin(np.deg2rad(angle*i))
    square = patches.Rectangle((ori_x,ori_y), 2, 2, fill=False, color='b', angle=ar
    ax.add_patch(square)

plt.gca().set_aspect('equal', adjustable='box')
plt.grid(True)
plt.show()
```



調整正方形的xy座標

In [ ]:

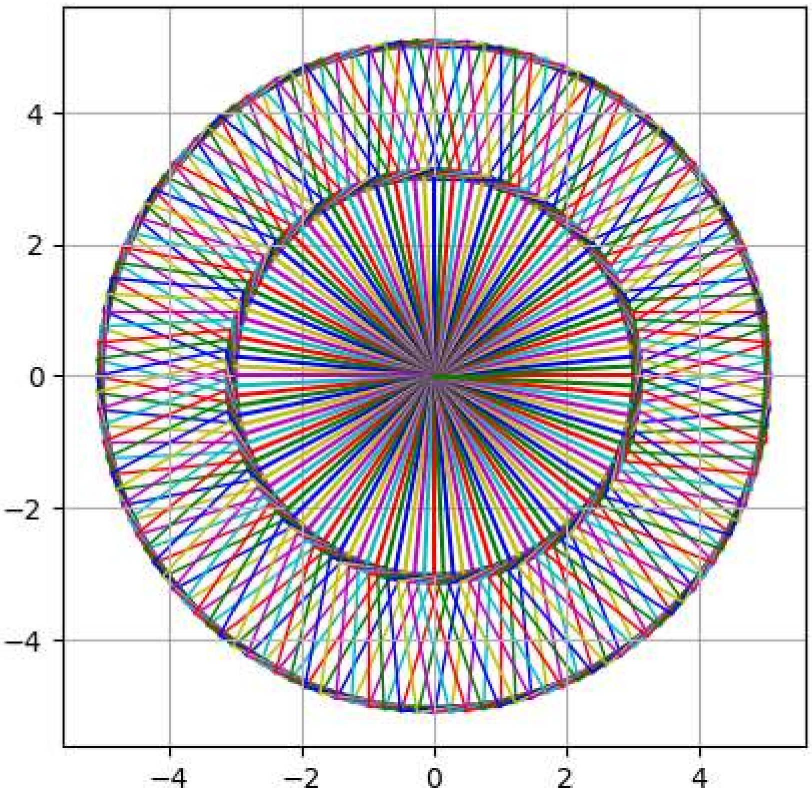
```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.patches as patches

fig, ax = plt.subplots()
n = 8
angle = 360 / n
radius = 3



for i in range(n):
    plt.plot([0, radius*np.cos(np.deg2rad(angle*i))], [0, radius*np.sin(np.deg2rad(
    ori_x = radius*np.cos(np.deg2rad(angle*i))
    ori_y = radius*np.sin(np.deg2rad(angle*i))
    x = radius*np.cos(np.deg2rad(angle*i)) + (2 / 2 * np.sin(np.deg2rad(angle*i)))
    y = radius*np.sin(np.deg2rad(angle*i)) - (2 / 2 * np.cos(np.deg2rad(angle*i)))
    square = patches.Rectangle((x,y), 2, 2, fill=False, color='b', angle=angle*i)
    ax.add_patch(square)

plt.gca().set_aspect('equal', adjustable='box')
plt.grid(True)
plt.show()
```



n = 128

In [ ]:

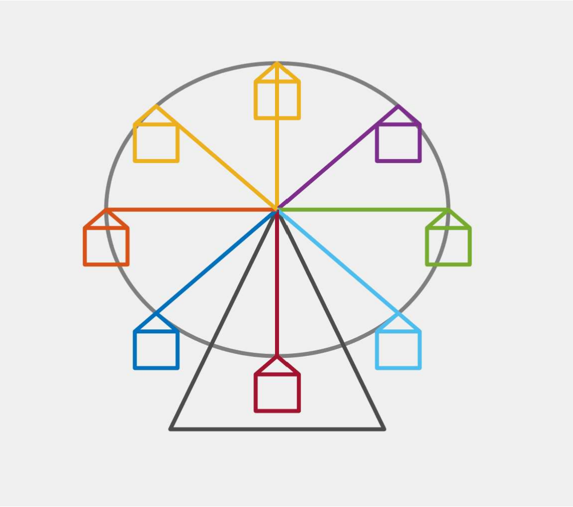
```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.patches as patches

fig, ax = plt.subplots()
n = 128
angle = 360 / n
radius = 3
color = ['r', 'g', 'b', 'y', 'm', 'c']


for i in range(n):
    plt.plot([0, radius*np.cos(np.deg2rad(angle*i))], [0, radius*np.sin(np.deg2rad(
    ori_x = radius*np.cos(np.deg2rad(angle*i))
    ori_y = radius*np.sin(np.deg2rad(angle*i))
    x = radius*np.cos(np.deg2rad(angle*i)) + (2 / 2 * np.sin(np.deg2rad(angle*i)))
    y = radius*np.sin(np.deg2rad(angle*i)) - (2 / 2 * np.cos(np.deg2rad(angle*i)))
    square = patches.Rectangle((x,y), 2, 2, fill=False, color=color[i%6], angle=ang
    ax.add_patch(square)

plt.gca().set_aspect('equal', adjustable='box')
plt.grid(True)
plt.show()
```



**注意事項與討論：**

- 本作品的函數繪製，皆以 Python 的 `matplotlib.pyplot` 模組為主，並以 `numpy` 模組的 `linspace` 函數產生 $x$ 值的陣列，以及 `array` 函數產生 $y$ 值的陣列。

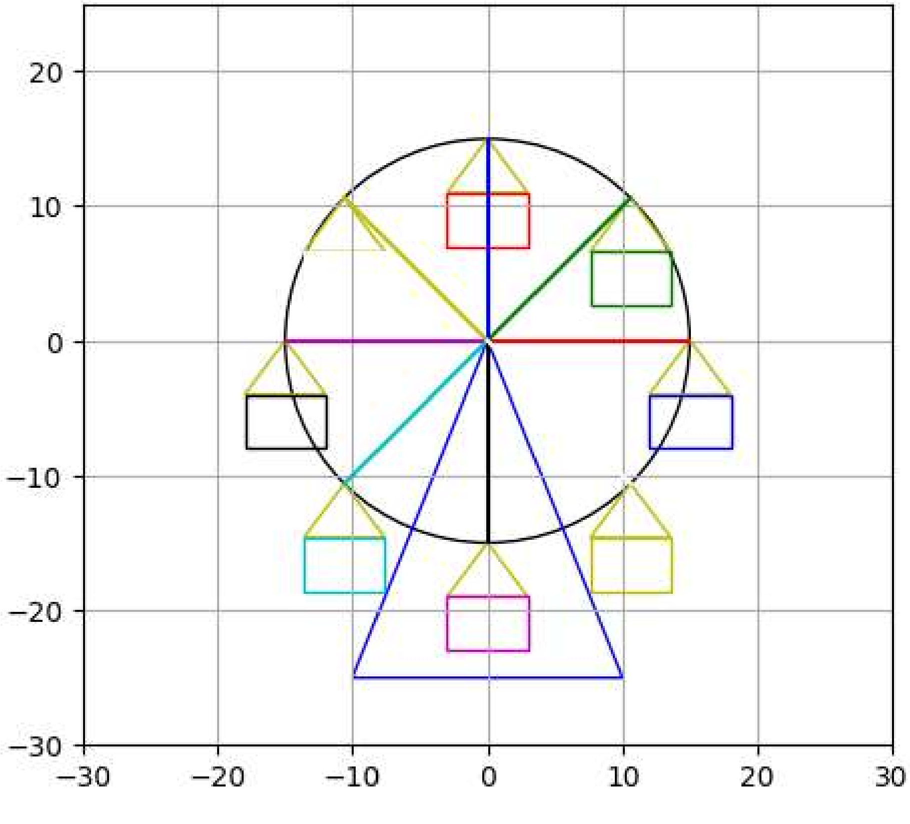
## 3. 繪製函數



In [ ]:

```
import matplotlib.pyplot as plt
import numpy as np
import matplotlib.patches as patches

fig,ax = plt.subplots()


n = 8
angle = 360 / n
radius = 15
```

```
circle = patches.Circle((0, 0), radius=radius, fill=False)
ax.add_patch(circle)

# 繪製三角形
triangle = patches.Polygon([[0, 0], [10, -25], [-10, -25]], fill=False, color='b')
ax.add_patch(triangle)

color = ['r', 'g', 'b', 'y', 'm', 'c', 'k', 'w']

for i in range(n):
    plt.plot([0, radius*np.cos(np.deg2rad(angle*i))], [0, radius*np.sin(np.deg2rad(
    original_x = radius*np.sin(np.deg2rad(angle*i))
    original_y = radius*np.cos(np.deg2rad(angle*i))
    triangle = patches.Polygon([[original_x, original_y], [original_x - 3, origina
    ax.add_patch(triangle)
    square = patches.Rectangle((original_x - 3, original_y - 8), 6, 4, fill=False,
    ax.add_patch(square)





plt.gca().set_aspect('equal', adjustable='box')
plt.xlim(-30, 30)
plt.ylim(-30, 25)
plt.grid(True)
plt.show()
```



## 3. 計算卡方右尾面積與自由度對照表

In [ ]:

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

col_F = np.array([0.995, 0.99, 0.975, 0.95, 0.9, 0.1, 0.05, 0.025, 0.01, 0.005])
row_df = np.array([i for i in range(1, 31)])

degrees_of_freedom = pd.DataFrame()

for col_f_value in col_F:

    df = chi2.ppf(col_f_value, row_df)
    degrees_of_freedom[col_f_value] = df

degrees_of_freedom.index = row_df
degrees_of_freedom.columns = np.flip(col_F)
degrees_of_freedom.to_excel('chi2.xlsx')
print(degrees_of_freedom)
```

```
        0.005      0.010      0.025      0.050      0.100      0.900  \
1    7.879439   6.634897   5.023886   3.841459   2.705543   0.015791   
2   10.596635   9.210340   7.377759   5.991465   4.605170   0.210721   
3   12.838156  11.344867   9.348404   7.814728   6.251389   0.584374   
4   14.860259  13.276704  11.143287   9.487729   7.779440   1.063623   
5   16.749602  15.086272  12.832502  11.070498   9.236357   1.610308   
6   18.547584  16.811894  14.449375  12.591587  10.644641   2.204131   
7   20.277740  18.475307  16.012764  14.067140  12.017037   2.833107   
8   21.954955  20.090235  17.534546  15.507313  13.361566   3.489539   
9   23.589351  21.665994  19.022768  16.918978  14.683657   4.168159   
10  25.188180  23.209251  20.483177  18.307038  15.987179   4.865182   
11  26.756849  24.724970  21.920049  19.675138  17.275009   5.577785   
12  28.299519  26.216967  23.336664  21.026070  18.549348   6.303796   
13  29.819471  27.688250  24.735605  22.362032  19.811929   7.041505   
14  31.319350  29.141238  26.118948  23.684791  21.064144   7.789534   
15  32.801321  30.577914  27.488393  24.995790  22.307130   8.546756   
16  34.267187  31.999927  28.845351  26.296228  23.541829   9.312236   
17  35.718466  33.408664  30.191009  27.587112  24.769035  10.085186   
18  37.156451  34.805306  31.526378  28.869299  25.989423  10.864936   
19  38.582257  36.190869  32.852327  30.143527  27.203571  11.650910   
20  39.996846  37.566235  34.169607  31.410433  28.411981  12.442609   
21  41.401065  38.932173  35.478876  32.670573  29.615089  13.239598   
22  42.795655  40.289360  36.780712  33.924438  30.813282  14.041493   
23  44.181275  41.638398  38.075627  35.172462  32.006900  14.847956   
24  45.558512  42.979820  39.364077  36.415029  33.196244  15.658684   
25  46.927890  44.314105  40.646469  37.652484  34.381587  16.473408   
26  48.289882  45.641683  41.923170  38.885139  35.563171  17.291885   
27  49.644915  46.962942  43.194511  40.113272  36.741217  18.113896   
28  50.993376  48.278236  44.460792  41.337138  37.915923  18.939242   
29  52.335618  49.587884  45.722286  42.556968  39.087470  19.767744   
30  53.671962  50.892181  46.979242  43.772972  40.256024  20.599235   

        0.950      0.975      0.990      0.995  
1    0.003932   0.000982   0.000157   0.000039  
2    0.102587   0.050636   0.020101   0.010025  
3    0.351846   0.215795   0.114832   0.071722  
4    0.710723   0.484419   0.297109   0.206989  
5    1.145476   0.831212   0.554298   0.411742  
6    1.635383   1.237344   0.872090   0.675727  
7    2.167350   1.689869   1.239042   0.989256  
8    2.732637   2.179731   1.646497   1.344413  
9    3.325113   2.700389   2.087901   1.734933  
10   3.940299   3.246973   2.558212   2.155856  
11   4.574813   3.815748   3.053484   2.603222  
12   5.226029   4.403789   3.570569   3.073824  
13   5.891864   5.008751   4.106915   3.565035  
14   6.570631   5.628726   4.660425   4.074675  
15   7.260944   6.262138   5.229349   4.600916  
16   7.961646   6.907664   5.812212   5.142205  
17   8.671760   7.564186   6.407760   5.697217  
18   9.390455   8.230746   7.014911   6.264805  
19  10.117013   8.906516   7.632730   6.843971  
20  10.850811   9.590777   8.260398   7.433844  
21  11.591305  10.282898   8.897198   8.033653  
22  12.338015  10.982321   9.542492   8.642716  
23  13.090514  11.688552  10.195716   9.260425  
24  13.848425  12.401150  10.856361   9.886234  
25  14.611408  13.119720  11.523975  10.519652  
26  15.379157  13.843905  12.198147  11.160237  
27  16.151396  14.573383  12.878504  11.807587  
28  16.927875  15.307861  13.564710  12.461336  
29  17.708366  16.047072  14.256455  13.121149  
30  18.492661  16.790772  14.953457  13.786720  
```