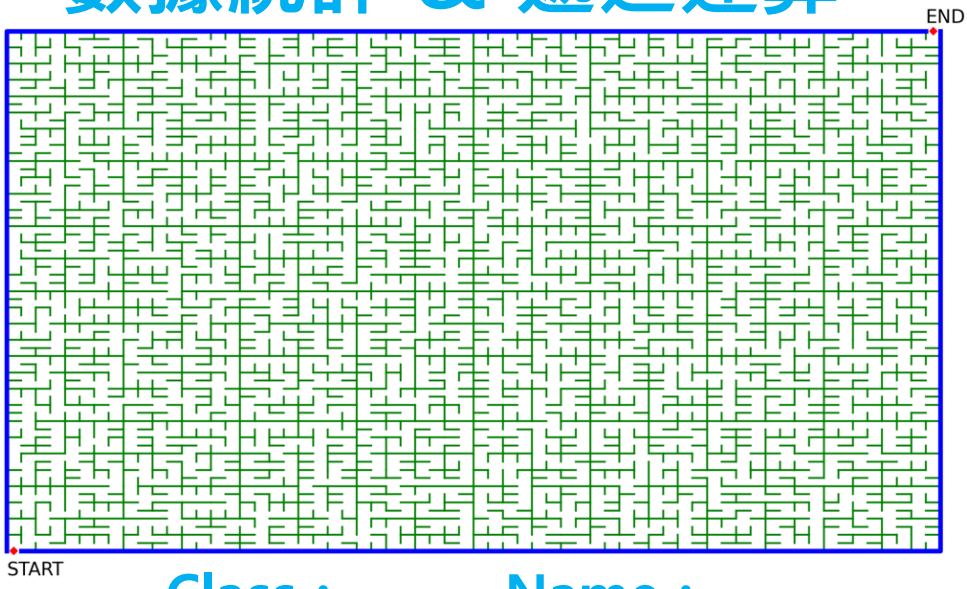
數據統計 & 遞迴運算

P1



Class：　　Name：

P2

線上平台登入與常用的功能

整數的算術運算

P3

```
a = 13
b = 10

print ("a加b=",a+b)
print ("a減b=",a-b)
print ("a乘b=",a*b)
print ("a除以b=",a/b)
print ("a除以b取整數值=",int(a/b))
print ("a除以b取商數=",a//b)
print ("a除以b取餘數=",a%b)
print ("a的b次方=",a**b)
```

試試看~可更改a,b值

重要喔~

以 , 做間隔

多運用 print 指令,可檢查程式是否有誤~

特殊函數~ 絕對值，正弦，餘弦，對數

P4

```
import numpy as np
pi=3.14159 #圓周率
x1=-5
x2=60*pi/180
x3=30*pi/180
x4=0.001

y1=abs(x1)
y2=np.sin(x2)
y3=np.cos(x3)
y4=np.log10(x4)

print("第1個值=",y1)
print("第2個值=",y2)
print("第3個值=",y3)
print("第4個值=",y4)
```

(1) $|-5| = ?$

(2) $\sin(60^0) = ?$

(3) $cos(30^0) = ?$

(4) $\log_{10} 0.001 = ?$

試寫程式~完成以下的計算

P5

(1) sin(30°)=

(2) cos(60°)=

(3) tan(45°)=

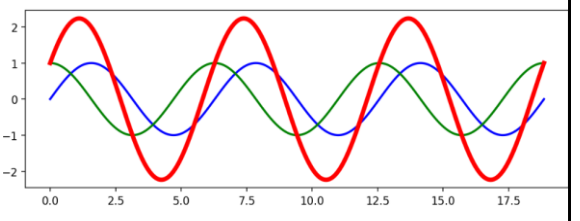
(4) $\log_2 3 =$

sin與cos之疊合

P6

```
import numpy as np
import matplotlib.pyplot as plt
plt.figure(figsize=(9,3),dpi=80)

pi=3.14159  # 圓周率
(A,B)=(2,1) # 亦可自行修改
x=np.linspace(0,6*pi,1001) #(起點,終點,共幾個點包含端點)

y1=np.sin(x)
y2=np.cos(x)
y3=A*np.sin(x)+B*np.cos(x)

plt.plot(x,y1,color="blue",linewidth=2)
plt.plot(x,y2,color="green",linewidth=2)
plt.plot(x,y3,color="red",linewidth=4)

plt.show()
```

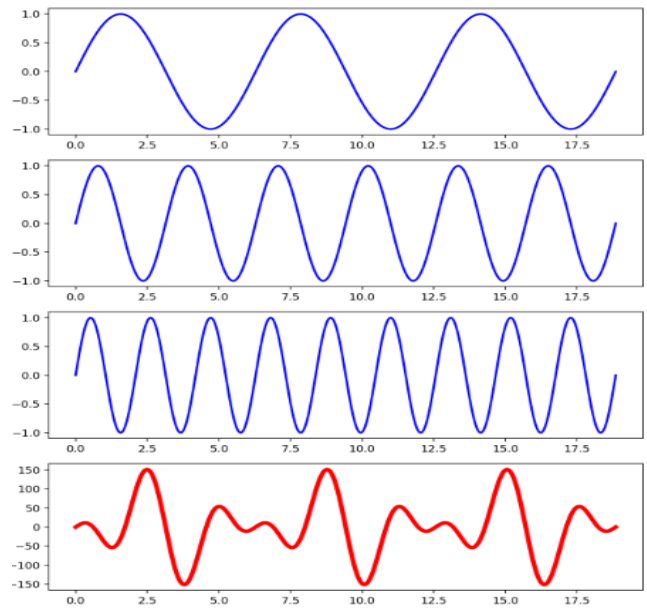
試試看~可更改A,B值



Making Waves 與波共舞

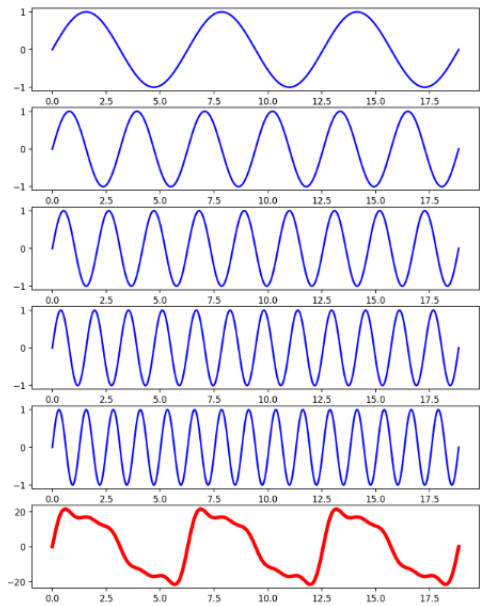
P7

```
import numpy as np
import matplotlib.pyplot as plt
plt.figure(figsize=(9,12),dpi=80)

pi=3.14159 # 圓周率
x=np.linspace(0,6*pi,1001)

plt.subplot(4,1,1)    #作圖1
y1=np.sin(x)
plt.plot(x,y1,color="blue",linewidth=2)

plt.subplot(4,1,2)    #作圖2
y2=np.sin(2*x)
plt.plot(x,y2,color="blue",linewidth=2)

plt.subplot(4,1,3)    #作圖3
y3=np.sin(3*x)
plt.plot(x,y3,color="blue",linewidth=2)

plt.subplot(4,1,4)    #作圖4 合成波
y4=30*y1-80*y2+60*y3
plt.plot(x,y4,color="red",linewidth=4)

plt.show()
```



試寫程式~完成以下的圖形

P8



提示 ~

y1=np.sin(x)

y2=np.sin(2*x)

y3=np.sin(3*x)

y4=np.sin(4*x)

y5=np.sin(5*x)

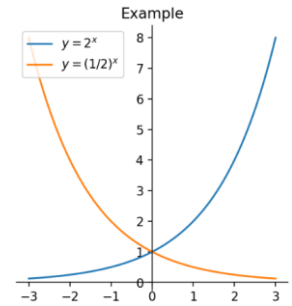
y6=20*y1+4*y2+5*y3+2*y4+2*y5

模擬鋼琴中央C的聲波

指數函數圖形

P9

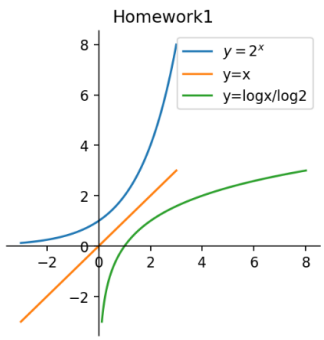
```
import numpy as np
import matplotlib.pyplot as plt
plt.figure(figsize=(4,4),dpi=80)
plt.title("Example")    #標題名稱

x=np.linspace(-3,3,100)  #(起點,終點,共幾個點包含端點)
y1=2**x
y2=(1/2)**x

plt.plot(x,y1,label="$y=2^x$")
plt.plot(x,y2,label="$y=(1/2)^x$")
plt.legend(loc="upper left") #展示每組數據對應的圖像名稱與位置

plt.show()
```

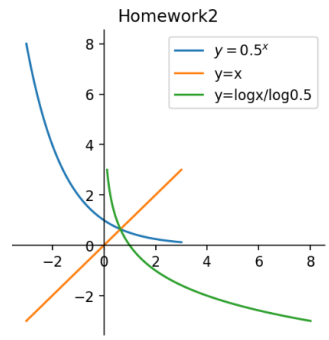


請在此插入坐標軸的指令

試寫程式~完成以下的圖形

P10





Homework1

提示 ~

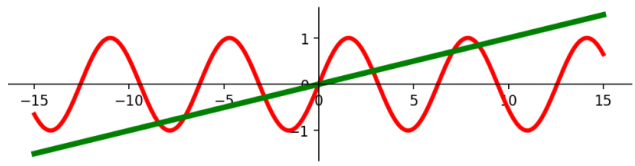
```
x=np.linspace(-3,3,100)  #(起點,終點,共幾個點包含端點)
y1=2**x
y2=x

plt.plot(x,y1,label="$y=2^x$")
plt.plot(x,y2,label="$y=x$")
plt.plot(y1,x,label="$y=logx/log2$")
```

Why ~

試寫程式~

P11

求方程式 sin(x)=0.1x 解的個數



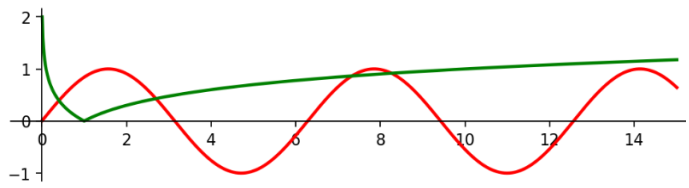
提示 ~

```
x=np.linspace(-15,15,1001) #(起點,終點,共幾個點包含端點)
y1=np.sin(x)
y2=0.1*x

plt.plot(x,y1,color="red",linewidth=2)
plt.plot(x,y2,color="green",linewidth=2)
```

試寫程式~

P12

求方程式 sin(x)= $|\log_{10} x|$ 解的個數



提示 ~

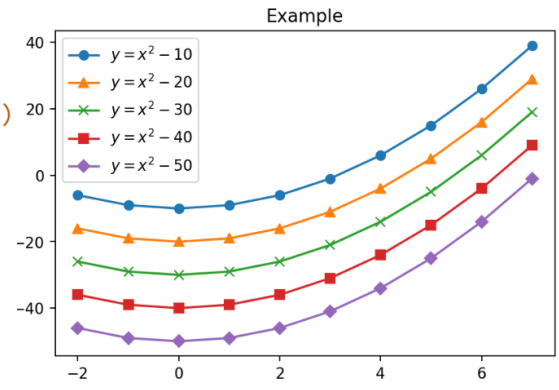
```
x=np.linspace(0.01,15.01,1001)  #(起點,終點,共幾個點包含端點)
y1=np.sin(x)
y2=abs(np.log10(x))

plt.plot(x,y1,color="red",linewidth=2)
plt.plot(x,y2,color="green",linewidth=2)
```

0.01 Why ~

數據分析~折線圖(函數資料)

P13

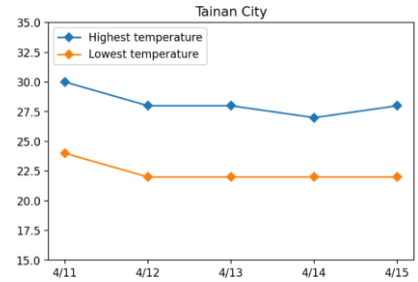
```
import numpy as np
import matplotlib.pyplot as plt
plt.figure(figsize=(6,4),dpi=80)
plt.title("Example")   #標題名稱

x=np.linspace(-2,7,10)  #(起點,終點,共幾個點包含端點)
y1=x**2-10
y2=x**2-20
y3=x**2-30
y4=x**2-40
y5=x**2-50

plt.plot(x,y1,marker="o",label="$y=x^2-10$")
plt.plot(x,y2,marker="^",label="$y=x^2-20$")
plt.plot(x,y3,marker="x",label="$y=x^2-30$")
plt.plot(x,y4,marker="s",label="$y=x^2-40$")
plt.plot(x,y5,marker="D",label="$y=x^2-50$")

plt.legend(loc="upper left") #展示每組數據對應的圖像名稱與位置
plt.show()
```



數據分析~折線圖(離散資料)

P14

```
import matplotlib.pyplot as plt
plt.figure(figsize=(6,4),dpi=80)
plt.title("Tainan City") #標題名稱

x=["4/11","4/12","4/13","4/14","4/15"]
y1=[30,28,28,27,28]    #每天之最高溫度
y2=[24,22,22,22,22]    #每天之最低溫度

plt.plot(x,y1,marker="D",label= "Highest temperature")
plt.plot(x,y2,marker="D",label= "Lowest temperature")

plt.legend(loc="upper left") #展示每組數據對應的圖像名稱與位置
plt.ylim(15,35)              #設定 y 軸的範圍
plt.show()
```

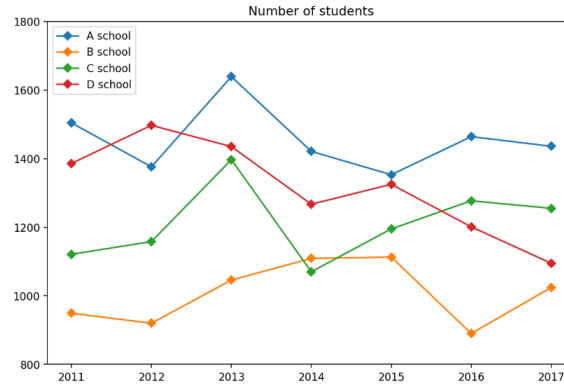


將下列各組數據~做出其折線圖

P15

Number of students

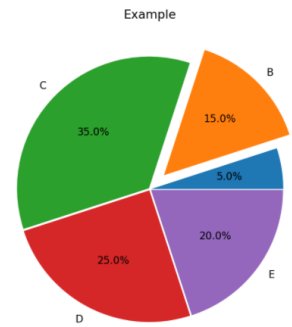
	2011	2012	2013	2014	2015	2016	2017
A school	1504	1376	1639	1421	1353	1464	1436
B school	949	920	1046	1109	1113	890	1024
C school	1121	1158	1397	1070	1195	1277	1255
D school	1386	1497	1435	1267	1325	1201	1095



數據分析~圓餅圖(離散資料)

P16

```
import matplotlib.pyplot as plt
plt.figure(figsize=(6,6),dpi=80)
plt.title("Example")  #標題名稱

x=[5, 15, 35, 25,20]              #數量
name=["A","B","C","D","E"]        #項目
e=[0.01,0.15,0.01,0.01,0.01]      #凸顯局部

plt.pie(x,labels=name,explode=e,autopct="%2.1f%%")
plt.show()
```



是用來顯示百分比~

P17

將下列數據~做出其圓餅圖

2020 Q1

Samsung	Huawei	Apple	Xiaomi	Oppo
72336	49846	44715	32825	28511

P18

循環語句 for (單層迴圈)

```
N=5
for i in range(1,N+1,1):
    print(i,"的平方數=",i*i)
print("執行完畢")
```

(起點,終點,間隔)
包含起點,但不含終點

也要注意 對齊~非常重要

循環語句 for (多層迴圈)

P19

```
M=3
N=6
for i in range(1,M+1,1):
    for j in range(1,N+1,1):
        print(i,"*",j,"=",i*j)
    print(" ")     #為美觀起見,故空1行
print("執行完畢")
```

(起點,終點,間隔)
包含起點,但不含終點

也要注意 對齊~非常重要

循環語句 for (多層迴圈)

P20

```
for i in ["黑桃","紅心","方塊"]:
    for j in ["1","2","3","4","5"]:
        print(i+j)
    print(" ")     #為美觀起見,故空1行
print("執行完畢")
```

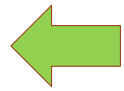
(迴圈也可以這樣用)

思考一下~~
i+j 這個運算特性

判斷語句：對與錯 True & False

P21

```
print(3>2)              #對
print(3>=2)             #對
print(3<2)              #錯
print(3<=2)             #錯

print(3==2 )            #錯
print(3!=2 )            #對

print(3>2 and 5>4)      #對
print(3>2 and 5<4)      #錯

print(3>2 or 5>4)       #對
print(3>2 or 5<4)       #對
print(3<2 or 5<4)       #錯
```

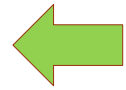


注意 ==與 != 的意義

條件語句 if

P22

```
x=66      #x是正整數

if (x!=55):
    print(x,"不等於55")

if (x==55):
    print(x,"等於55")

if (x%2==1):
    print(x,"是奇數")

if (x%2==0):
    print(x,"是偶數")

if (x>=10 and x<100):
    print(x,"是兩位數")
```



試試看~可更改 x 值

首數,尾數,首位數字

P23

```
import numpy as np
k=0.03          # k 是大於0的實數
t=np.log10(k)  #將 k 取對數值

if (t>=0 or t%1==0):
  a=int(t)              #首數
if (t<0 and t%1!=0):
  a=int(t)-1            #首數

b=t-a                   #尾數
c=int(k*(10**(-a)))     #首位數字

print(k,"的首數=",a)
print(k,"的尾數=",b)
print(k,"的首位數字=",c)
```

remove指令,append指令

P24

```
A=[1,2,3,4,5,6,7,8,9]
B=["Tom","John","Peter","Monica","Donna"]
C=[]

x=5
y="Alice"
z=100

A.remove(x)   #移除 x 的值
B.append(y)   #加入 y 的值
C.append(z)   #加入 z 的值

print("A=",A)
print("B=",B)
print("C=",C)
```

隨機亂數

P25

```
import random

S=["Tom","John","Peter","Verna","Monica"]

a=random.random()           #輸出0-1之間的隨機數
b=random.randint(-10,10)    #輸出 -10 到 10 之間(含端點)的隨機整數
c=random.choice(S)          #從序列中隨機抽1個值
A=random.sample(S,3)        #從序列中隨機抽3個值,並形成一個新序列
print (a)
print (b)
print (c)
print (A)
```

有相同物的排列數

P26

2個紅球、3個黃球，排成一列，求其排列數。

```
A=["R","Y"]
Ans=[]

#設x,y,z,u,v分別為第1,2,3,4,5 位置的顏色

for x in A:
    for y in A:
        for z in A:
            for u in A:
                for v in A:
                    T=[x,y,z,u,v]     #暫時放置的序列
                    C1=T.count("R")   #R的次數
                    C2=T.count("Y")   #Y的次數
                    if (C1==2 and C2==3):
                        Ans.append(x+y+z+u+v)
print ("排列=",Ans)
print ("方法數=",len(Ans))
```

P27

試寫程式~

(1) **2個紅球、3個黃球、2個綠球，排成一列，求其排列數。**

(2) **由6個數字0,0,1,1,2,2 排成的六位數．共有多少個？**

P28

試寫程式~

a、b、c 、d、e 等五人排一列，
求下列各情況之排列數：
(1)　a不排首位，b不排第二位。
(2)　a不排首位，b不排第二位，c不排第三位

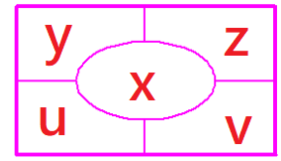
從「t o m a t o 」一字的 6個字母中，
任意選取 4 個排成一列．共有多少種排法？

答:102

#試寫程式~

P29

(1) 以4種顏色a,b,c,d塗右圖,4色全用且相鄰不同色，塗法有幾種?(圖形不可旋轉) 答:48

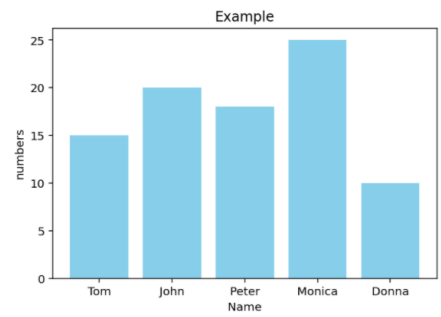


(2) 以4種顏色a,b,c,d塗右圖,4色全用且相鄰不同色，塗法有幾種?(圖形不可旋轉) 答:144

柱狀圖

P30

```
import matplotlib.pyplot as plt

x=["Tom","John","Peter","Monica","Donna"]   #各柱子的文字
y=[15, 20, 18, 25, 10]                      #各柱子的高度
plt.bar(x,y,color="skyblue")

plt.title("Example")    # 標題
plt.xlabel("Name")      # x 軸的文字
plt.ylabel("numbers")   # y 軸的文字
plt.show()
```



數學期望值 : 投擲1個公正骰子 (步驟1)

P31

```
import numpy as np
import matplotlib.pyplot as plt
import random
N=20                    #投擲總數
bag=[1,2,3,4,5,6]       #公正骰子的點數
Ans=[]                  #儲存每次的點數

for i in range(1,N+1,1):
    a=random.choice(bag)  #從袋子內取到的點數
    Ans.append(a)

print("投擲結果:",Ans)  #本指令為確認資料正確,事後可刪除
```

#數學期望值 : 投擲1個公正骰子 (步驟2)

P32

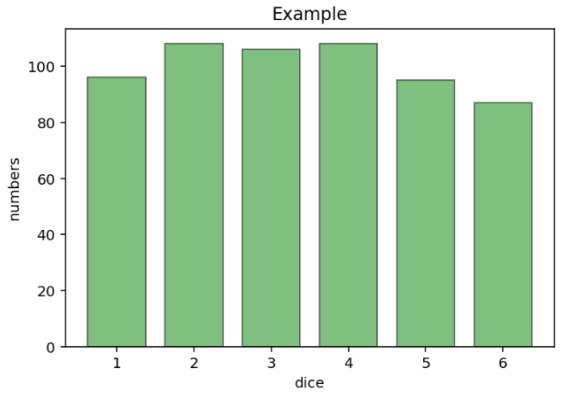
```
x=["1","2","3","4","5","6"]    #各柱子的文字
y=[]                           #準備儲存各柱子的高度
C1=Ans.count(1);y.append(C1)   #計算1的次數,並儲存
C2=Ans.count(2);y.append(C2)   #計算2的次數,並儲存
C3=Ans.count(3);y.append(C3)   #計算3的次數,並儲存
C4=Ans.count(4);y.append(C4)   #計算4的次數,並儲存
C5=Ans.count(5);y.append(C5)   #計算5的次數,並儲存
C6=Ans.count(6);y.append(C6)   #計算6的次數,並儲存

plt.bar(x,y,color="green")

plt.title("Example")    # 標題
plt.xlabel("dice")      # x 軸的文字
plt.ylabel("numbers")   # y 軸的文字
plt.show()
```

(本頁程式碼 續接上頁)~

數學期望值 : 投擲1個公正骰子 (步驟3)

P33

```
print("數學期望理論值=",3.5)
print("數學期望實驗值=",(1*C1+2*C2+3*C3+4*C4+5*C5+6*C6)/N)
```



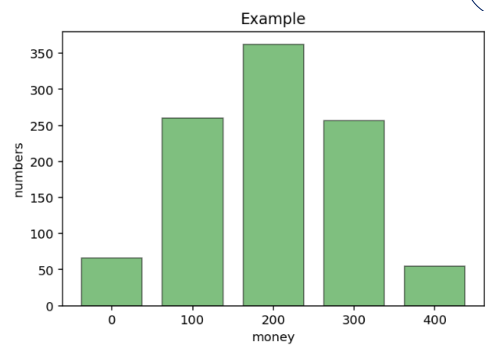

```
數學期望理論值= 3.5
數學期望實驗值= 3.4316666666666666
```

(本頁程式碼 續接上頁)~

試寫程式~

P34

同時投擲4個公正硬幣 1 次,若每出現1個正面可得100元.
試以前頁範例的步驟1,步驟2,步驟3完成數學期望值.




```
數學期望理論值= 200
數學期望實驗值= 197.5
```

提示 ~

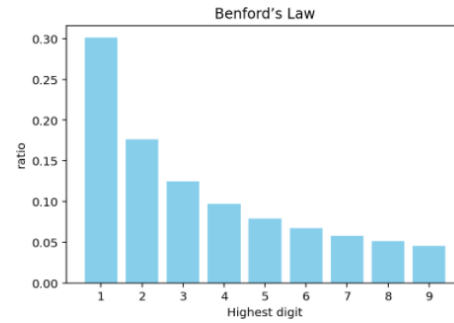
bag=[100 , 0]

a=random.choice(bag)+random.choice(bag)+random.choice(bag)+random.choice(bag)

Benford's Law 班佛定律

P35

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

x=["1","2","3","4","5","6","7","8","9"]  #各柱子的文字
a=np.linspace(1,9,9)  #(起點,終點,共幾個點包含端點)
y=np.log10(1+a)-np.log10(a)              #各柱子的高度
plt.bar(x,y,color="skyblue")

plt.title("Benford's Law")       # 標題
plt.xlabel("Highest digit")      # x 軸的文字
plt.ylabel("ratio")              # y 軸的文字
plt.show()
```



如何利用「班佛定律」偵破詐欺？

P36

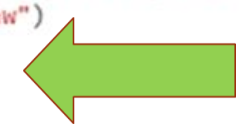
班佛定律：
在 10 進位制中，
以數 a 起頭的數出現的機率為
$\log_{10}(a+1)-\log_{10}(a)$

2002年，在美國Znetix/HMC上市詐騙案中，有七萬多筆支票跟匯款交易要查，人工全部看完交易是不太實際的事情。這時，鑑識會計專家就用了班佛定律，找出疑似虛假或重複的交易（即那些首位數字比例明顯高於班佛定律的交易），加速了整個調查的過程。

將 首位數比例 與 班佛定律 做對比

P37

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

x=["1","2","3","4","5","6","7","8","9"]  #各柱子的文字
y1=[0.3010,0.1761,0.1249,0.0969,0.0792,0.0669,0.0580,0.0512,0.0458]
plt.bar(x,y1,color="skyblue",label= "Benford's Law")

data=[ ]            #將要驗證的資料放在括號內
Ans=[]
for k in data:
    t=np.log10(k)  #將 k 取對數值
    if (t>=0 or t%1==0):
      a=int(t)                #首數
    if (t<0 and t%1!=0):
      a=int(t)-1              #首數
    b=t-a                     #尾數
    c=int(k*(10**(-a)))       #首位數字
    Ans.append(c)
```



要驗證的資料須大於0,且放在data=[ ] 的括號內,才能被執行~

將 首位數比例 與 班佛定律 做對比

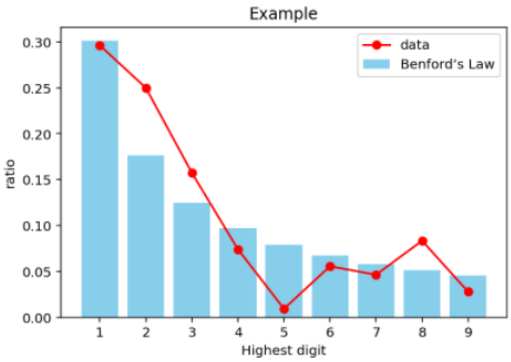
P38

(本頁程式碼 續接上頁)~

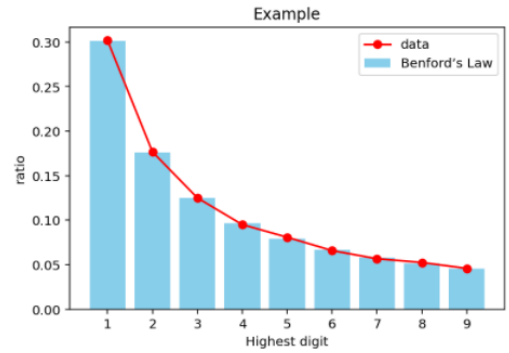
```
n=len(Ans)
y=[]                              #準備儲存各柱子的高度
C1=Ans.count(1);y.append(C1/n)    #計算1的次數,並儲存其比率
C2=Ans.count(2);y.append(C2/n)    #計算2的次數,並儲存其比率
C3=Ans.count(3);y.append(C3/n)    #計算3的次數,並儲存其比率
C4=Ans.count(4);y.append(C4/n)    #計算4的次數,並儲存其比率
C5=Ans.count(5);y.append(C5/n)    #計算5的次數,並儲存其比率
C6=Ans.count(6);y.append(C6/n)    #計算6的次數,並儲存其比率
C7=Ans.count(7);y.append(C7/n)    #計算7的次數,並儲存其比率
C8=Ans.count(8);y.append(C8/n)    #計算8的次數,並儲存其比率
C9=Ans.count(9);y.append(C9/n)    #計算9的次數,並儲存其比率

plt.plot(x,y,color="red",marker="o",label= "data")
plt.legend(loc="upper right")     #展示每組數據對應的圖像名稱與位置
plt.title("Example")              #標題
plt.xlabel("Highest digit")       # x 軸的文字
plt.ylabel("ratio")               # y 軸的文字
plt.show()
```

班佛定律之應用實例

P39



105/08/16 臺北捷運各站出站人數統計



費式數列前 1200 項數字

費氏數列

P40

```
def a(n):   #定義遞迴數列
    if (n==1) :return 1
    if (n==2) :return 1
    return a(n-1)+a(n-2)

#以下內容為主程式
N=10        #項數
Ans=[]

for i in range(1,N+1,1):
      Ans.append(a(i))

print("第",N,"項=",a(N))
print("前",N,"項=",Ans)
print("前",N,"項和=",sum(Ans))
```

$$a_1 = 1$$
$$a_2 = 1$$
$$a_n = a_{n-1} + a_{n-2}$$

指令 sum(Ans) 的用意為何?

P41

試寫程式~完成等差數列與級數

$$\begin{cases} a_1 = 8 \\ a_n = a_{n-1} + 5 \end{cases}$$

輸出的結果要有~

(1)列出第N項

(2)列出前N項

(3)列出前N項的和

提示 ~

```
def a(n):   #定義遞迴數列
    if (n==1) :return 8
    return a(n-1)+5
```

P42

#試寫程式~完成等比數列與級數

$$\begin{cases} a_1 = 1 \\ a_n = 2 \cdot a_{n-1} \end{cases}$$

輸出的結果要有~

(1)列出第N項

(2)列出前N項

(3)列出前N項的和

提示 ~

```
def a(n):   #定義遞迴數列
    if (n==1) :return 1
    return 2*a(n-1)
```

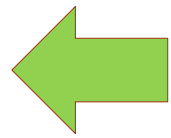
#組合數

P43

從 n 個不同事物中取出 k 個（ $0 \le k \le n$ ）為一組，

其組合數 C_k^n 為 $\dfrac{n!}{k!(n-k)!}$

```
def C(n,k):  #組合公式
    if (k==n) :return 1
    if (k==0) :return 1
    return C(n-1,k)+C(n-1,k-1)

#以下內容為主程式
n=6
for k in range(0,n+1,1):
        print("從",n,"個不同事物中取出",k,"個的組合數為:",C(n,k))
```

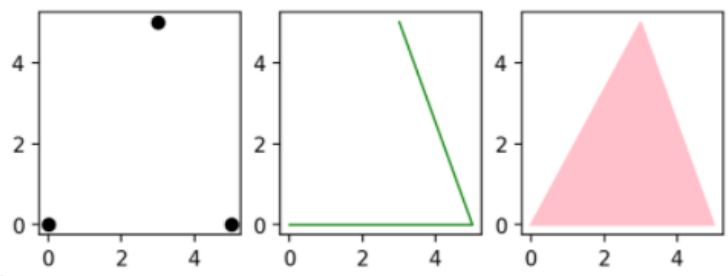


Why?

#描點,連線,塗色 (scatter , plot , fill)

P44

```
import matplotlib.pyplot as plt
plt.figure(figsize=(6,2),dpi=80)

#作圖1:描點
plt.subplot(1,3,1)
X=[0,5,3]
Y=[0,0,5]
plt.scatter(X,Y,color="black")

#作圖2:連線
plt.subplot(1,3,2)
X=[0,5,3]
Y=[0,0,5]
plt.plot(X,Y,color="green",linewidth=1)

#作圖3:塗色
plt.subplot(1,3,3)
X=[0,5,3]
Y=[0,0,5]
plt.fill(X,Y,color="pink")

plt.show()
```



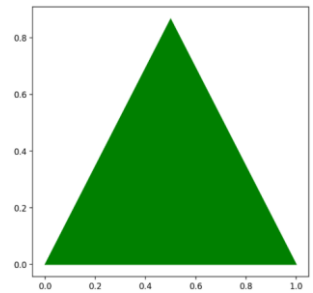
P45

Sierpinski triangle(謝爾賓斯基三角形)

P46

Sierpinski triangle(謝爾賓斯基三角形) 步驟 1

```
import matplotlib.pyplot as plt
plt.figure(figsize=(6,6),dpi=80)

(x1,y1)=(0,0)
(x2,y2)=(1,0)
(x3,y3)=(0.5,0.5*3**0.5)

X=[x1,x2,x3]
Y=[y1,y2,y3]
plt.fill(X,Y,color="green")

plt.show()
```



P47

Sierpinski triangle(謝爾賓斯基三角形) 步驟 2

```
import matplotlib.pyplot as plt
plt.figure(figsize=(6,6),dpi=80)
def F(x1,y1,x2,y2,x3,y3):  #定義遞迴程式
    if((x2-x1) <0.6) :return
    (x4,y4)=((x1+x2)/2,(y1+y2)/2)
    (x5,y5)=((x2+x3)/2,(y2+y3)/2)
    (x6,y6)=((x3+x1)/2,(y3+y1)/2)
    X=[x4,x5,x6]
    Y=[y4,y5,y6]
    plt.fill(X,Y,color="white")
    F(x1,y1,x4,y4,x6,y6)
    F(x4,y4,x2,y2,x5,y5)
    F(x6,y6,x5,y5,x3,y3)

#以下內容為主程式
(x1,y1)=(0,0)
(x2,y2)=(1,0)
(x3,y3)=(0.5,0.5*3**0.5)

X=[x1,x2,x3]
Y=[y1,y2,y3]
plt.fill(X,Y,color="green")
F(x1,y1,x2,y2,x3,y3)
plt.show()
```

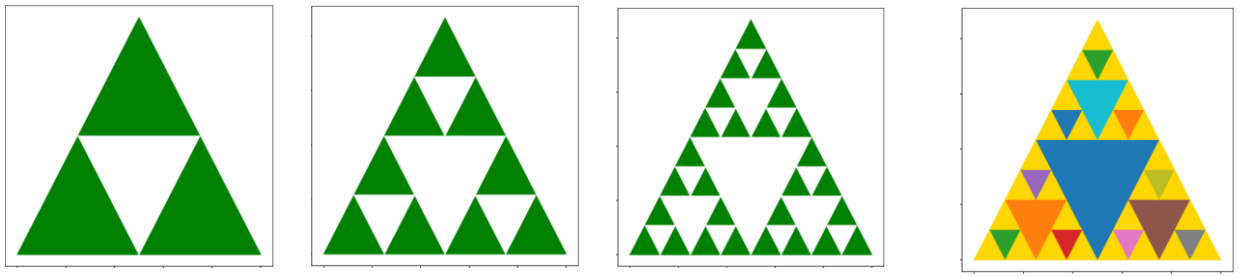
(1) 請完成左邊的程式碼~
(2) 試調整其值,其效果為何?

(3)繼續呼叫遞迴程式

呼叫遞迴程式

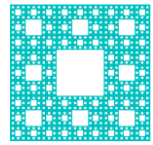
P48

試完成如下圖形~



P49

```
import matplotlib.pyplot as plt
plt.figure(figsize=(5,5),dpi=80)
def F(x1,y1,r):   #定義遞迴程式
    if(r<9) :return
    (x2,y2)=(x1+r/3,y1)
    (x3,y3)=(x2+r/3,y1)
    (x4,y4)=(x1,y1+r/3)
    (x5,y5)=(x4+r/3,y4)
    (x6,y6)=(x5+r/3,y4)
    (x7,y7)=(x1,y4+r/3)
    (x8,y8)=(x7+r/3,y7)
    (x9,y9)=(x8+r/3,y7)
    X=[x5,x6,x9,x8]
    Y=[y5,y6,y9,y8]
    plt.fill(X,Y,color="white")
    F(x1,y1,r/3)
    F(x2,y2,r/3)
    F(x3,y3,r/3)
    F(x4,y4,r/3)
    F(x6,y6,r/3)
    F(x7,y7,r/3)
    F(x8,y8,r/3)
    F(x9,y9,r/3)

#以下內容為主程式
(x1,y1,r)=(0,0,81)   #正方形左下的坐標與邊長
X=[x1,x1+r,x1+r,x1]
Y=[y1,y1,y1+r,y1+r]
plt.fill(X,Y,color="blue")
F(x1,y1,r)
plt.show()
```



(1)請完成左邊的程式碼~
(2) 試調整其值,其效果為何?

P50

Koch curve 科赫曲線

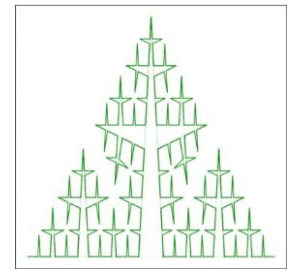
#三角形雪花

Koch curve 科赫曲線

P51

```
import numpy as np
import matplotlib.pyplot as plt
plt.figure(figsize=(6,6),dpi=80)

def F(x0,y0,r,a):
    if r<0.5 :return
    x1=x0+0.45*r*np.cos(a)                ; y1=y0+0.45*r*np.sin(a)
    x2=x1+0.45*r*np.cos(a+83.62*pi/180)   ; y2=y1+0.45*r*np.sin(a+83.62*pi/180)
    x3=x0+0.55*r*np.cos(a)                ; y3=y0+0.55*r*np.sin(a)
    x4=x0+r*np.cos(a)                     ; y4=y0+r*np.sin(a)

    X=[x0,x4]
    Y=[y0,y4]
    plt.plot(X,Y,color="green",linewidth=1)    #畫線

    X=[x1,x3]
    Y=[y1,y3]
    plt.plot(X,Y,color="white",linewidth=1)    #擦掉

    X=[x1,x2,x3]
    Y=[y1,y2,y3]
    plt.plot(X,Y,color="green",linewidth=1)    #畫線

    F(x0,y0,0.45*r,a)
    F(x1,y1,0.45*r,a+83.62*pi/180)
    F(x2,y2,0.45*r,a-83.62*pi/180)
    F(x3,y3,0.45*r,a)

#以下內容為主程式
pi=3.14159      #圓周率
(x0,y0,r,a)=(0,0,1,0)  #(基準點之x,y坐標,長度,方向角)
F(x0,y0,r,a)
plt.show()
```

試調整其值,其效果為何?



#三角形雪花

P52

```
import numpy as np
import matplotlib.pyplot as plt
plt.figure(figsize=(6,6),dpi=80)

def F(x0,y0,r,a):  #定義遞迴程式
    if r<0.4 :return
    x1=x0+(1/3)*r*np.cos(a)             ; y1=y0+(1/3)*r*np.sin(a)
    x2=x1+(1/3)*r*np.cos(a-pi/3)        ; y2=y1+(1/3)*r*np.sin(a-pi/3)
    x3=x0+(2/3)*r*np.cos(a)             ; y3=y0+(2/3)*r*np.sin(a)
    x4=x0+r*np.cos(a)                   ; y4=y0+r*np.sin(a)

    X=[x0,x4]
    Y=[y0,y4]
    plt.plot(X,Y,color="blue",linewidth=2)    #畫線

    X=[x1,x3]
    Y=[y1,y3]
    plt.plot(X,Y,color="white",linewidth=3)   #擦掉

    X=[x1,x2,x3]
    Y=[y1,y2,y3]
    plt.plot(X,Y,color="blue",linewidth=2)    #畫線

    F(x0,y0,r/3,a)
    F(x1,y1,r/3,a-pi/3)
    F(x2,y2,r/3,a+pi/3)
    F(x3,y3,r/3,a)

#以下內容為主程式
pi=3.14159              #圓周率
(x0,y0,r,a)=(0,0,1,0)   #(基準點之x,y坐標,長度,方向角)
F(x0,y0,r,a)
F(x0+r,y0,r,a+2*pi/3)
F(x0+0.5*r,y0+0.5*r*3**0.5,r,a+4*pi/3)
plt.show()
```

(1)請完成左邊的程式碼~

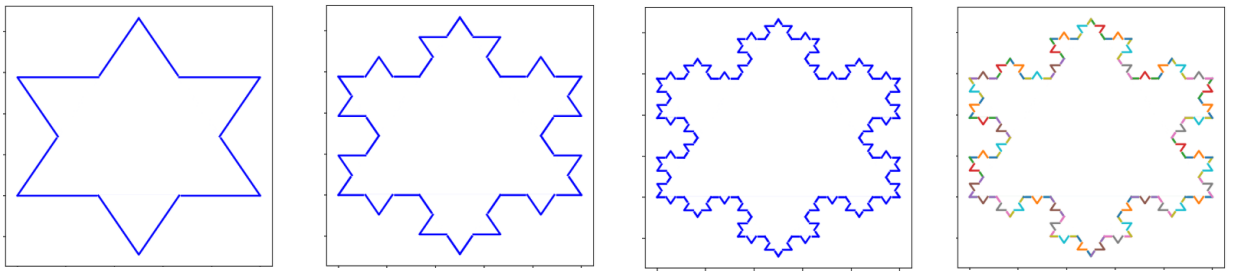
(2) 試調整其值,其效果為何?

繼續呼叫遞迴程式

呼叫遞迴程式

#試完成如下圖形~

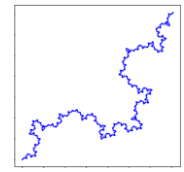
P53



#模擬朝東北方向之海岸線~

P54

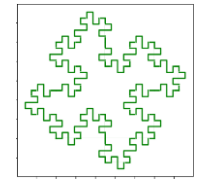
```
import numpy as np
import matplotlib.pyplot as plt
import random
plt.figure(figsize=(6,6),dpi=80)
bag=[1,-1]
def F(x0,y0,r,a):  #定義遞迴程式
    if r<0.03 :return
    k=random.choice(bag)
    x1=x0+(1/3)*r*np.cos(a)        ; y1=y0+(1/3)*r*np.sin(a)
    x2=x1+(1/3)*r*np.cos(a-k*pi/3) ; y2=y1+(1/3)*r*np.sin(a-k*pi/3)
    x3=x0+(2/3)*r*np.cos(a)        ; y3=y0+(2/3)*r*np.sin(a)
    x4=x0+r*np.cos(a)              ; y4=y0+r*np.sin(a)

    X=[x0,x4]
    Y=[y0,y4]
    plt.plot(X,Y,color="blue",linewidth=2)   #畫線

    X=[x1,x3]
    Y=[y1,y3]
    plt.plot(X,Y,color="white",linewidth=3)  #擦掉

    X=[x1,x2,x3]
    Y=[y1,y2,y3]
    plt.plot(X,Y,color="blue",linewidth=2)   #畫線

    F(x0,y0,r/3,a)
    F(x1,y1,r/3,a-k*pi/3)
    F(x2,y2,r/3,a+k*pi/3)
    F(x3,y3,r/3,a)
#以下內容為主程式
pi=3.14159                #圓周率
(x0,y0,r,a)=(0,0,1,pi/4)  #(基準點之x,y坐標,長度,方向角)
F(x0,y0,r,a)
plt.show()
```



與三角形雪花程式有何差別?

P55

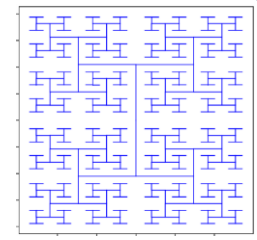
(1)請完成左邊的程式碼~
(2) 試調整其值,其效果為何?



```
import numpy as np
import matplotlib.pyplot as plt
plt.figure(figsize=(5,5),dpi=80)

def F(x0,y0,r,a):      #定義遞迴程式
    if r<0.25 :return
    x1=x0+0.25*r*np.cos(a)            ; y1=y0+0.25*r*np.sin(a)
    x2=x1+0.25*r*np.cos(a+pi/2)       ; y2=y1+0.25*r*np.sin(a+pi/2)
    x3=x2+0.25*r*np.cos(a)            ; y3=y2+0.25*r*np.sin(a)
    x4=x0+0.5*r*np.cos(a)             ; y4=y0+0.5*r*np.sin(a)
    x5=x4+0.25*r*np.cos(a-pi/2)       ; y5=y4+0.25*r*np.sin(a-pi/2)
    x6=x5+0.25*r*np.cos(a)            ; y6=y5+0.25*r*np.sin(a)
    x7=x0+0.75*r*np.cos(a)            ; y7=y0+0.75*r*np.sin(a)
    x8=x0+r*np.cos(a)                 ; y8=y0+r*np.sin(a)

    X=[x0,x8]
    Y=[y0,y8]
    plt.plot(X,Y,color="green",linewidth=2)    #畫線

    X=[x1,x7]
    Y=[y1,y7]
    plt.plot(X,Y,color="white",linewidth=3)    #擦掉

    X=[x1,x2,x3,x5,x6,x7]
    Y=[y1,y2,y3,y5,y6,y7]
    plt.plot(X,Y,color="green",linewidth=2)    #畫線

    F(x0,y0,r/4,a)
    F(x1,y1,r/4,a+pi/2)
    F(x2,y2,r/4,a)
    F(x3,y3,r/4,a-pi/2)
    F(x4,y4,r/4,a-pi/2)
    F(x5,y5,r/4,a)
    F(x6,y6,r/4,a+pi/2)
    F(x7,y7,r/4,a)
#以下內容為主程式
pi=3.14159                #圓周率
(x0,y0,r,a)=(0,0,1,0)     #(基準點之x,y坐標,長度,方向角)
F(x0,y0,r,a)
F(x0+r,y0,r,a+pi/2)
F(x0+r,y0+r,r,a+pi)
F(x0,y0+r,r,a+3*pi/2)
plt.show()
```

H 樹~

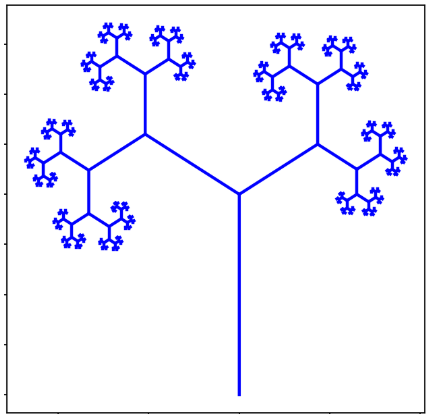
P56

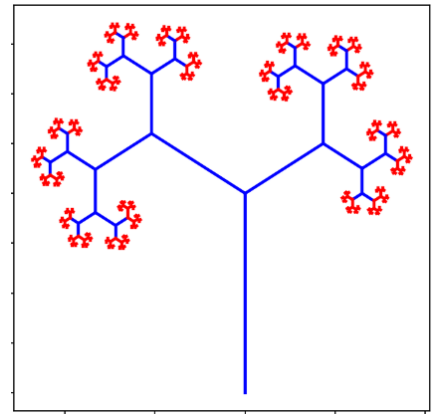


```
import numpy as np
import matplotlib.pyplot as plt
plt.figure(figsize=(5,5),dpi=80)

def F(x0,y0,r,a):  #定義遞迴程式
    if r <1 :return
    x1=x0+r*np.cos(a) ; y1=y0+r*np.sin(a)

    X=[x0,x1]
    Y=[y0,y1]
    plt.plot(X,Y,color="blue",linewidth=1)  #畫線

    F(x1,y1,0.7*r,a-pi/2)
    F(x1,y1,0.7*r,a+pi/2)
#以下內容為主程式
pi=3.14159                #圓周率
(x0,y0,r,a)=(0,0,15,pi/2)  #(基準點之x,y坐標,長度,方向角)
F(x0,y0,r,a)
F(x0,y0,r,a+pi)
plt.show()
```

#試完成如下圖形~

P57





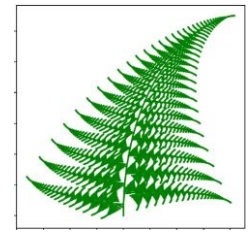
提示 ~

```
F(x1,y1,0.5*r,a-pi/3)
F(x1,y1,0.6*r,a+pi/3)
```

How ~

蕨葉~

P58



```
import numpy as np
import matplotlib.pyplot as plt
plt.figure(figsize=(5,5),dpi=80)

def F(x0,y0,r,a): #定義遞迴程式
    if r <1 :return
    x1=x0+0.04*r*np.cos(a)       ; y1=y0+0.04*r*np.sin(a)
    x2=x1+0.04*r*np.cos(a-pi/90) ; y2=y1+0.04*r*np.sin(a-pi/90)

    X=[x0,x1,x2]
    Y=[y0,y1,y2]
    plt.plot(X,Y,color="green",linewidth=2)    #畫圖

    F(x1,y1,0.3*r,a+pi*4/9)
    F(x2,y2,0.3*r,a-pi*41/90)
    F(x2,y2,0.9*r,a-pi/90)
#以下內容為主程式
pi=3.14159      #圓周率
(x0,y0,r,a)=(0,0,90,pi/2)  #(基準點之x,y坐標,長度,方向角)
F(0,0,90,pi/2)
plt.show()
```

隨機迷宮設計

P59

迷宮的邊界與出入口~

P60

```
import matplotlib.pyplot as plt
import random
plt.figure(figsize=(12,8),dpi=80)
#以下內容為遞迴程式,要保留許多空間


#以下內容為主程式
(x0,y0,n)=(0,0,5)  #基準點之x,y坐標,n是2~5的整數
bag=[1,1,1,0]      #抽籤桶內有3個開,1個關
#F(x0,y0,n)

plt.plot([x0+1,x0+2**n,x0+2**n],[y0,y0,y0+2**n],color="blue",linewidth=5)
plt.plot([x0,x0,x0+2**n-1],[y0,y0+2**n,y0+2**n],color="blue",linewidth=5)
plt.scatter(0.5,0,marker="D",color="red")                #迷宮入口
plt.scatter(2**n-0.5,2**n,marker="D",color="red")        #迷宮出口
plt.text(0,-3,"START", fontsize=20)
plt.text(2**n-1.5,2**n+1,"END", fontsize=20)
plt.ylim(-5,2**n+5)              #設定 y 軸的範圍
plt.show()
```

迷宮的遞迴程式~

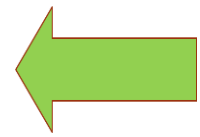
P61

```
def F(x0,y0,n):  #定義遞迴程式
    if n<1:return
    x1=x0+2**(n-1)          ;y1=y0
    x2=x0+2**n              ;y2=y0
    x3=x0                   ;y3=y0+2**(n-1)
    x4=x1                   ;y4=y3
    x5=x2                   ;y5=y3
    x6=x0                   ;y6=y0+2**n
    x7=x1                   ;y7=y6
    x8=x2                   ;y8=y6
    A=random.sample(bag,4)
    if A[0]==0:  #十字之西
        plt.plot([x3,x4],[y3,y4],color="green",linewidth=2)
    if A[1]==0:  #十字之東
        plt.plot([x4,x5],[y4,y5],color="green",linewidth=2)
    if A[2]==0:  #十字之南
        plt.plot([x1,x4],[y1,y4],color="green",linewidth=2)
    if A[3]==0:  #十字之北
        plt.plot([x4,x7],[y4,y7],color="green",linewidth=2)
```

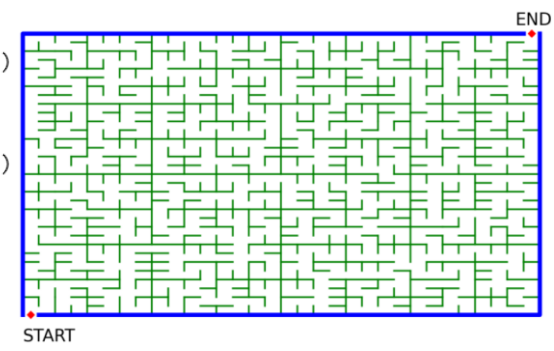
迷宮的遞迴程式(程式碼 續接上頁)~

P62

```
    if A[0]==1:  #十字之西
        a=random.randint(x3,x4-1)
        plt.plot([x3,a],[y3,y3],color="green",linewidth=2)
        plt.plot([a+1,x4],[y3,y4],color="green",linewidth=2)
    if A[1]==1:  #十字之東
        a=random.randint(x4,x5-1)
        plt.plot([x4,a],[y4,y4],color="green",linewidth=2)
        plt.plot([a+1,x5],[y4,y5],color="green",linewidth=2)
    if A[2]==1:  #十字之南
        a=random.randint(y1,y4-1)
        plt.plot([x1,x1],[y1,a],color="green",linewidth=2)
        plt.plot([x1,x4],[a+1,y4],color="green",linewidth=2)
    if A[3]==1:  #十字之北
        a=random.randint(y4,y7-1)
        plt.plot([x4,x4],[y4,a],color="green",linewidth=2)
        plt.plot([x4,x7],[a+1,y7],color="green",linewidth=2)

    F(x0,y0,n-1)
    F(x1,y1,n-1)
    F(x3,y3,n-1)
    F(x4,y4,n-1)
```



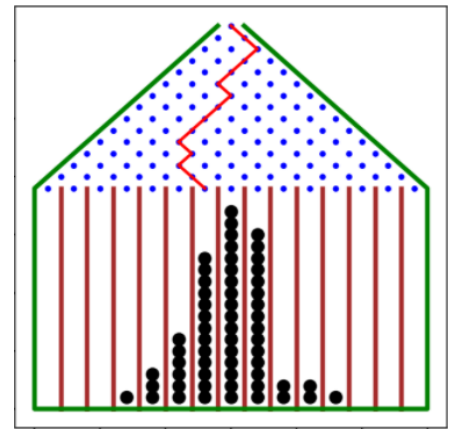
左邊界要對齊上頁的 if



彈珠檯,二項分佈~

P63

```
import random
import numpy as np
import matplotlib.pyplot as plt
plt.figure(figsize=(5,5),dpi=80)
H=15    #有H層的彈珠檯,H必須為奇數
K=H*1.2 #底座高為K
N=60    #有N顆彈珠,N值不可太大
R=H-1   #有R個路口,決定向右下或左下移動
T=[]    #儲存每一次彈珠落在哪個x坐標位置的暫存區

for i in range(1,H+1,1):
    for j in range(1,i+1,1):
        x=-i+2*j-1      #釘子的x坐標
        y=H+1-i+K       #釘子的y坐標
        plt.scatter(x,y,color="blue",marker=".")  #畫出釘子

for i in range(1,H-1+1,1):
    X=[2*i-H,2*i-H]
    Y=[  0   ,  K+1]
    plt.plot(X,Y,color="brown",linewidth=3)    #畫出底座內的隔板

X=[ -1,-1*H,-1*H,H,  H,  1]
Y=[H+K, K+1,   0,0,K+1,H+K]
plt.plot(X,Y,color="green",linewidth=3)        #畫出邊界
```



彈珠檯,二項分佈(程式碼 續接上頁)~

P64

```
for i in range(1,N+1,1): #執行N個彈珠的隨機路徑,並儲存最後位置
    X=[0];Y=[H+K]        #彈珠路徑的X與Y坐標與起點坐標
    for j in range(1,R+1,1):
        dice=[1,-1,1,-1,1,-1]
        r=random.choice(dice)
        x=X[j-1]+r ; X.append(x)
        y=Y[j-1]-1 ; Y.append(y)
    T.append(X[H-1])                    #儲存最後位置
plt.plot(X,Y,color="red",linewidth=2)  #只會畫出最後1次的彈珠路徑圖

for i in range(1,H+1,1):  #計算次數,並且放入序列儲存
    C=T.count(2*i-H-1)
    if (C!=0):
        for j in range(1,C+1,1):    #在底座畫出彈珠
            plt.scatter(2*i-H-1,j,color="black",marker="o",s=60)
plt.show()
```