IC252: Data Science II

by Adarsh Santoria(b21176)

QUESTIONS WITH SOLUTIONS DISPLAYED IN TERMINAL AND CODE

1. Simulate a fair coin from the throw of a fair die in three different ways. These can be:

• Method 1: Output H if d = 1, 2, or 3 and T if d = 4, 5, or 6

• Method 2: Output H if d = 1 and T if d = 2 and don't output anything for other values of d. This is a wasteful method.

• Method 3: Your own method, different from the above.

How will you be sure that the output is correct? Suppose your function is called die2coin. If you call this function N number of times (⇑ 10, 000 and above), and count the number of times it gave H and the number of times it gave T, then we can decide if die2coin is correct. Expected output: A plot, with proper labels, that convinces you that the generated coin is fair. Required input: Accept the type of method used (1,2 or 3), N

Interface of the function: die2coin(int m, int N), where m is the type of method used; N, the number of times to repeat; and returns 'H' or 'T'.

Spyder

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\HP

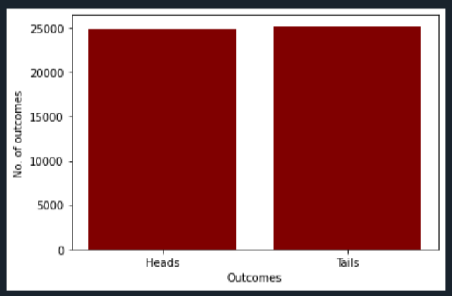
C:\Users\HP\untitled16.py

untitled2.py | untitled3.py | untitled14.py | untitled15.py | untitled16.py*

```
import matplotlib.pyplot as plt          #importing required modules
from random import randint as r
def die2coin(m ,N):                      #making a function
    H=0                                  #make variable H & T to respectively
    T=0                                  #initialise number of heads and tails from 0
    for i in range(N):                   #run a loop for number of throws
        n=r(1,6)                         #randomly take any number from 1 to 6 as in die
        if( m == 1):                     #checking the type of method
            if (n == 1) or(n == 2) or (n == 3):
                H = H+1                  #H or T based on situation by 1
            else:
                T=T+1
        if(m==2):
            if (n == 1):
                H = H+1
            elif(n == 2):
                T=T+1
        if(m==3):
            if (n == 2) or (n == 3):
                H = H+1
            elif (n==4) or (n==5):
                T=T+1
    print("Head occurs ",H,"times")      #printing total no. of H & T
    print("Tail occurs ",T,"times")
    #Ploting the bar graph
    plt.bar(["Heads","Tails"],[H,T], color ='maroon')
    plt.xlabel("Outcomes")               #labelling the graph
    plt.ylabel("No. of outcomes")
    plt.show()
m=int(input("type of method:"))
N=int(input("no. of times to repeat:",))
die2coin(m,N)
```

Console 4/A

```
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [M
Type "copyright", "credits" or "license" for more information.

IPython 7.31.1 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/HP/untitled16.py', wdir='C:/Users/HP'

type of method:1

no. of times to repeat:50000
Head occurs  24834 times
Tail occurs  25166 times
```




```
In [2]:
```

IPython Console

LSP Python: ready

Type here to search

32°C Haz

2. Same as the previous question, but this time, generate a biased coin from a fair die. Plots are required as before. You can just use one method.

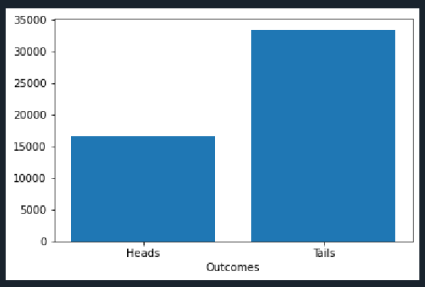
Expected output: A plot, with proper labels, that convinces you that the generated coin is biased.

Required input: N, the number of times to repeat.

Use a similar function interface: die2BiasedCoin(int N); returns ‘H’ or ‘T’.

```
# -*- coding: utf-8 -*-
"""
Created on Sat Mar 12 14:07:49 2022

@author: HP
"""
import matplotlib.pyplot as plt        #importing required modules
from random import randint as r
def die2BiasedCoin(N):                 #making a function
    H=0                                #make variable H & T to respectively
    T=0                                #initialise number of heads and tails from 0
    for i in range(N):                 #run a loop for number of throws
        n=r(1,6)                       #randomly take any number from 1 to 6 as in die
        if (n == 2) or (n == 3):       #taking a biased situation where 2 out of 6 cause Head
            H = H+1                    #H or T based on situation by 1
        else:
            T=T+1
    print("Head occurs ",H,"times")    #printing total no. of H & T
    print("Tail occurs ",T,"times")
    #Ploting the bar graph
    plt.bar(["Heads","Tails"],[H,T])
    plt.xlabel("Outcomes")             #labelling the graph
N=int(input("no. of times to repeat:"))
die2BiasedCoin(N)
```

```
Python 3.7.9 (tags/v3.7.9:13c94747c7, Aug 17 2020, 18:58:18) [MSC
Type "copyright", "credits" or "license" for more information.

IPython 7.31.1 -- An enhanced Interactive Python.

In [1]: runfile('C:/Users/HP/untitled17.py', wdir='C:/Users/HP')

no. of times to repeat:50000
Head occurs  16635 times
Tail occurs  33365 times
```




```
In [2]:
```

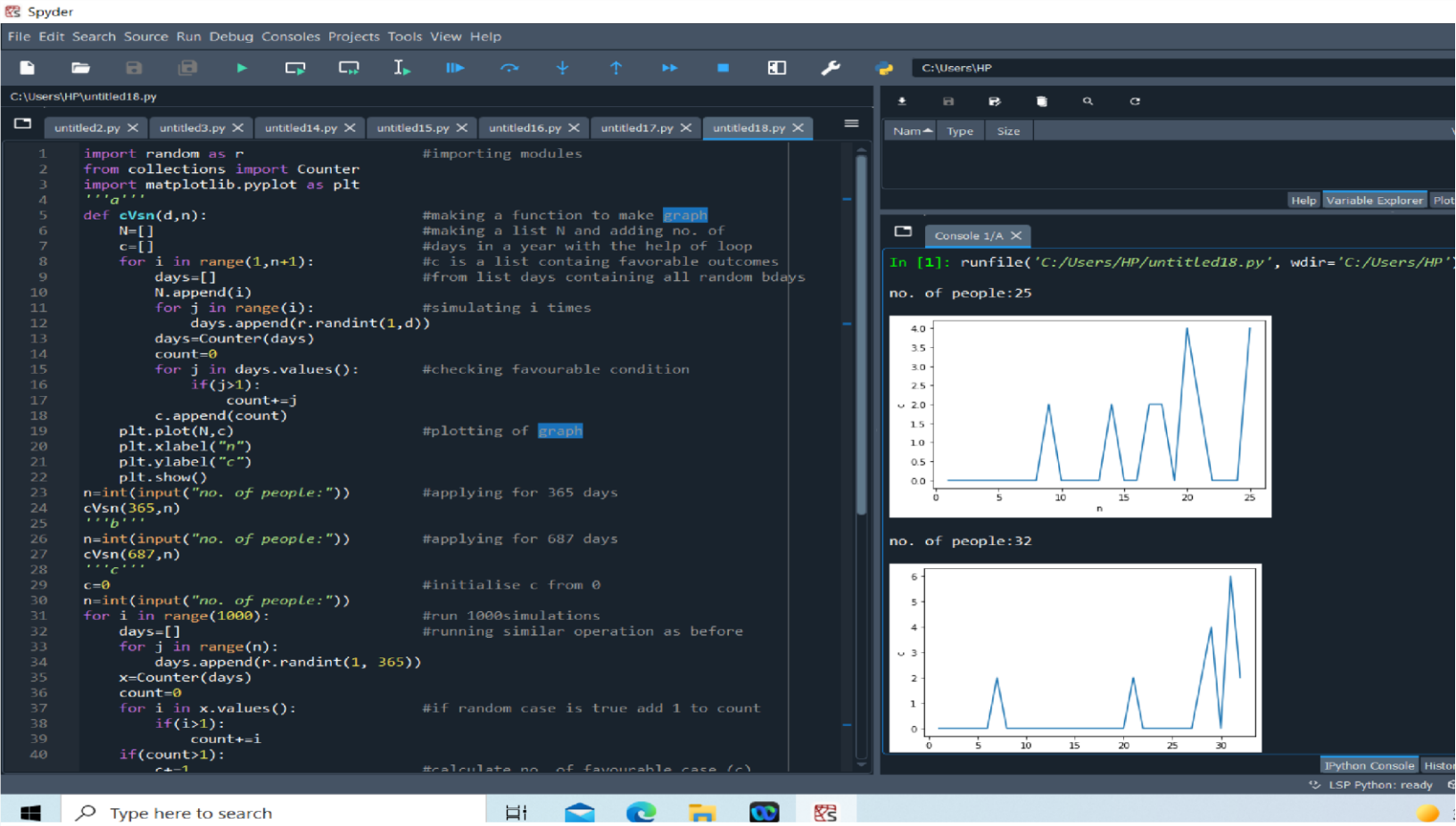
3.This question is derived from the birthday paradox. Let the number of people in the room be n. Generate a random number between 1 and 365, n times. This simulates n birthdays. Count how many common birthdays are present between at least two people, and let this be denoted by c. Plot c versus n, as n varies from 1 to 366 for the following cases:

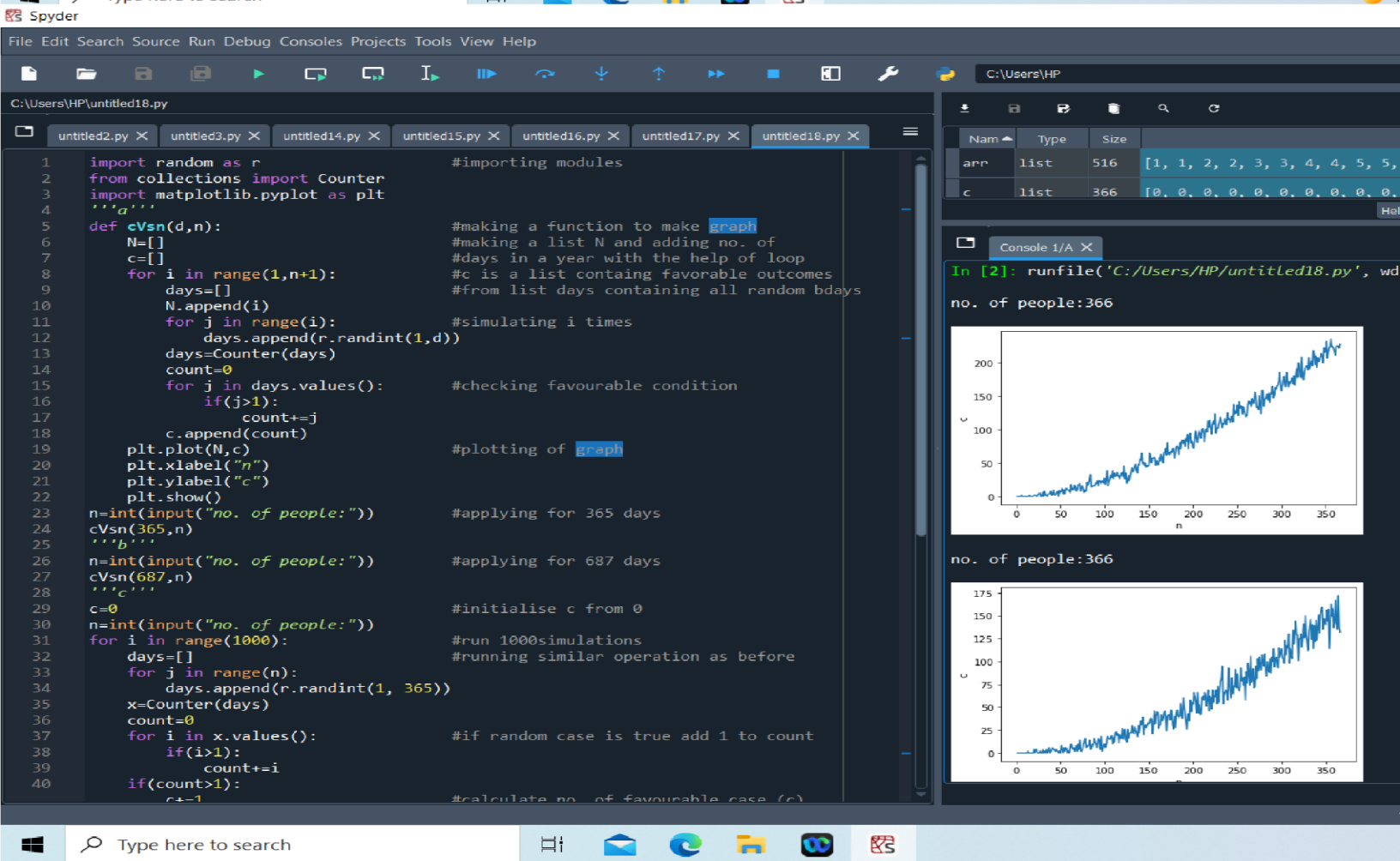
(a) When each birthday is equally likely. c should be 2 when n is around 25 or so.

(b) When the birthdays are computed on Mars. Each Martin year is 687 days. c should be 2 for n around 32.

(c) For n around 50, there is a high chance that c is at least 2. Demonstrate this by simulating this situation1000 times and computing the average probability. You should get the average probability close to 0.99. This basically means that in a group of 50 people, you can be almost sure that two of them share the same birthday.

(d) When birthdays between 1-150 are twice as likely as 151-365.





Spyder

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\HP

C:\Users\HP\untitled18.py

untitled2.py | untitled3.py | untitled14.py | untitled15.py | untitled16.py | untitled17.py | untitled18.py

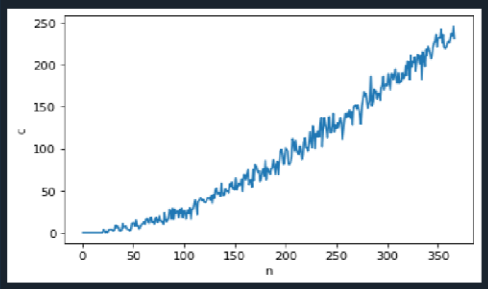
```
28  '''c'''
29  c=0                                 #initialise c from 0
30  n=int(input("no. of people:"))
31  for i in range(1000):               #run 1000simulations
32      days=[]                         #running similar operation as before
33      for j in range(n):
34          days.append(r.randint(1, 365))
35      x=Counter(days)
36      count=0
37      for i in x.values():            #if random case is true add 1 to count
38          if(i>1):
39              count+=i
40      if(count>1):
41          c+=1                        #calculate no. of favourable case (c)
42  print("Average probabilty that c is atleast 2 is",(c/1000))
43  #probability is printed with the help of it's formula
44  '''d'''
45  N=[]
46  c=[]                                #similar operations are performed as in a
47  arr=[]
48  for i in range(1,367):
49      N.append(i)                     #but probability of days between 1 and 150
50      if(i<151):                      #are twice  as compared to between 151 and 365
51          arr.append(i)               #as arr conatains the prior souble
52          arr.append(i)
53      else:
54          arr.append(i)
55  for n in range(1,367):
56      days=[]
57      for i in range(n):
58          days.append(r.choice(arr))
59      x=Counter(days)
60      count=0
61      for i in x.values():
62          if(i>1):
63              count=count+i
64      c.append(count)                 #plotting of graph
65  plt.plot(N,c)
66  plt.xlabel("n")
67  plt.ylabel("c")
```

Nam	Type	Size	
arr	list	516	[1, 1, 2, 2, 3, 3, 4, 4, 5, 5, .
c	list	366	[0. 0. 0. 0. 0. 0. 0. 0. 0. 0. .

Help

Console 1/A

```
no. of people:50
Average probabilty that c is atleast 2 is 0.964
```




```
In [3]:
```

Spyder

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\HP

C:\Users\HP\untitled18.py

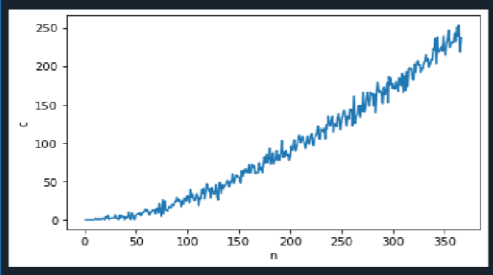
untitled2.py | untitled3.py | untitled14.py | untitled15.py | untitled16.py | untitled17.py | untitled18.py

```
28  '''c'''
29  c=0                                 #initialise c from 0
30  n=int(input("no. of people:"))
31  for i in range(1000):               #run 1000simulations
32      days=[]                         #running similar operation as before
33      for j in range(n):
34          days.append(r.randint(1, 365))
35      x=Counter(days)
36      count=0
37      for i in x.values():            #if random case is true add 1 to count
38          if(i>1):
39              count+=i
40      if(count>1):
41          c+=1                        #calculate no. of favourable case (c)
42  print("Average probabilty that c is atleast 2 is",(c/1000))
43  #probability is printed with the help of it's formula
44  '''d'''
45  N=[]
46  c=[]                                #similar operations are performed as in a
47  arr=[]
48  for i in range(1,367):
49      N.append(i)                     #but probability of days between 1 and 150
50      if(i<151):                      #are twice  as compared to between 151 and 365
51          arr.append(i)               #as arr conatains the prior souble
52          arr.append(i)
53      else:
54          arr.append(i)
55  for n in range(1,367):
56      days=[]
57      for i in range(n):
58          days.append(r.choice(arr))
59      x=Counter(days)
60      count=0
61      for i in x.values():
62          if(i>1):
63              count=count+i
64      c.append(count)                 #plotting of graph
65  plt.plot(N,c)
66  plt.xlabel("n")
67  plt.ylabel("c")
```

Nam	Type	Size	
arr	list	516	[1, 1, 2, 2, 3, 3, 4, 4, 5, 5,
c	list	366	[0. 0. 0. 0. 0. 0. 0. 0. 0. 0.

Console 1/A

```
no. of people:366
Average probabilty that c is atleast 2 is 1.0
```




```
In [4]:
```

REVISED 3. This question is derived from the birthday paradox. Let the number of people in the room be n. Estimate the probability pn that out of n people, at least one match occurs (in other words, at least two people share a birthday.)

Plot pn versus n for $2 \leq n \leq 100$ for the following cases.

(a) When each birthday is equally likely. In earth, the probability is about 0.5 for n = 23.

(b) When the birthdays are computed on Mars. Each Martin year is 669 days. The probability is about 0.5 when n = 31.

(c) When birthdays between 1-150 are twice as likely as 151-365.

Hint: The experiment is binary (outcome is success or failure.) We define success if there is a match. Determine if there is match by counting the number of occurrences for each birthday. If there is no match, then it is a failure. A useful Python class for counting is Collections.counter. Also, remember to repeat the experiment a large number of times (I did it 10000 times for each n.) The whole code is less than 50 lines

Spyder

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\HP

C:\Users\HP\untitled19.py

untitled2.py | untitled3.py | untitled14.py | untitled15.py | untitled18.py | untitled19.py

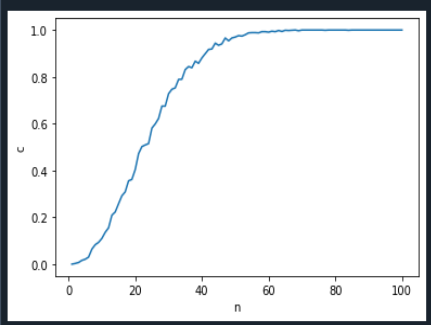
```
# -*- coding: utf-8 -*-
"""
Created on Sun Mar 13 01:01:14 2022

@author: HP
"""
import random as r                      #importing modules
from collections import Counter
import matplotlib.pyplot as plt
def bday(n,d):                          #making a function
    N=[]                                #making a list N and adding no. of
    c=[]                                #days in a year with the help of loop
    for i in range(1,n+1):              #c is a list containg favorable outcomes
        N.append(i)
        t=0
        for k in range(1000):           #running 1000 simulations
            days=[]
            count=False
            for j in range(1,i+1):
                days.append(r.randint(1,d))
            da=Counter(days)            #randomly taking a day
            for j in da.values():       #checking favourable condition
                if(j>1):
                    count=True
                    break
            if count:                   #counting no. of favourable cases
                t+=1
        c.append(t/1000)                #appending probability to the list
    plt.plot(N,c)                       #plotting of graph
    plt.xlabel("n")
    plt.ylabel("c")
    plt.show()
'''a'''
bday(100,365)                           #100 people in 365 days pf year
'''b'''
bday(100,669)                           #100 people in 669 days of year
'''c'''
N=[]                                #making a list N and adding no. of
c=[]
arr=[]
```

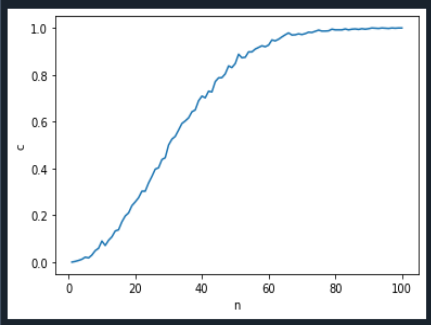
Nam	Type	Size	Value
arr	list	200	[1, 1, 2, 2, 3, 3, 4, 4, 5, 5, ...]

Help | Variable Explorer | Plots | Files

Console 1/A


```
In [4]: runfile('C:/Users/HP/untitled19.py', wdir='C:/Users/HP')
```





IPython Console | History

LSP Python: ready

Type here to search

33°C H

Spyder

File Edit Search Source Run Debug Consoles Projects Tools View Help

C:\Users\HP

C:\Users\HP\untitled19.py

untitled2.py | untitled3.py | untitled14.py | untitled15.py | untitled18.py | untitled19.py*

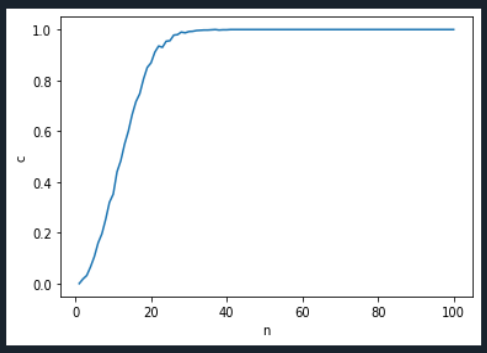
```
    plt.xlabel("n")
    plt.ylabel("c")
    plt.show()
'''a'''
bday(100,365)                       #100 people in 365 days pf year
'''b'''
bday(100,669)                       #100 people in 669 days of year
'''c'''
N=[]                                #making a list N and adding no. of
c=[]
arr=[]                             #n is number of people
n=int(input("number of people:"))   #similar operations are performed
for i in range(1,n+1):              #but probability of days between 1 and 150
    if(i<151):                      #are twice  as compared to between 151 and 365
        arr.append(i)               #as arr conatains the prior double
        arr.append(i)
    else:
        arr.append(i)               #days in a year with the help of loop
for i in range(1,n+1):              #c is a list containg favorable outcomes
    N.append(i)
    t=0
    for k in range(1000):          #running 1000 simulations
        days=[]
        count=False
        for j in range(1,i+1):
            days.append(r.choice(arr))
        da=Counter(days)           #randomly taking a day
        for j in da.values():      #checking favourable condition
            if(j>1):
                count=True
                break
        if count:                  #counting no. of favourable cases
            t+=1
    c.append(t/1000)               #appending probability to the list
plt.plot(N,c)                      #plotting of graph
plt.xlabel("n")
plt.ylabel("c")
plt.show()
```

Nam	Type	Size	
arr	list	200	[1, 1, 2, 2, 3, 3, 4, 4, 5, 5,

Help

Console 2/A

```
number of people:100
```




```
In [2]:
```

Type here to search