```
#  Experiment for drum m1:

for covid patient


total are 100 people(Universe)

Sample Space = S = { r1-7, r8-14, r15+ , d}
                      25     20     50     5


events are: :

all recovered people : E1 = { r1-7, r8-14, r15+ }
                       P[E1] = 25 + 20 +50 / 100 = 95/100

recovered in 14 days : E2 = { r1-7, r8-14 }
                       P[E2] = 25 + 20 /100 = 45/100

recovered in 20 days : E3
we dont have proper data : so its not a well defined event
so
                      45/100  <= P[E3] <= 95/100


```

In [1]:

```
# sample space : collection of all possible outcomes
# event : collection of any number of outcomes:

# Event is a subset of Sample space
```

Coin :

fair coin

```
Coin :
fair coin

S = {H,T}

P[H] = 1/2
P[T] = 1/2

```

```
Coin toss twice:

sample space: S = { HH,HT,TH,TT }

heads : E1 = { HH,HT,TH }
        P[E1] = 3/4
both heads or tail : E2 = { HH,TT }
                     P[E2] = 2/4


```

Outcomes + Sample_space + Event + Probability

In [ ]:

```

```

In [ ]:

```

```

In [ ]:

```

```

```
'''
for Dice:

sample space = S = {1,2,3,4,5,6}

example of events for Dice example
E1 = {1,2}
E2 = {1}
E3 = {1,3,5}

'''
```

In [ ]:

```

```

```
dice :
S = {1,2,3,4,5,6}

E = {1,2,3}
E' = E^c = {4,5,6}  E compliment, all element other than in E
```

dice 2 times :

```
dice 2 times :

sample space :
{
11 12 13 14 15 16
21 22 23 24 25 26
31 32 33 34 35 36
41 42 43 44 45 45
51 52 53 54 55 56
61 62 63 64 65 66
}
total 36 outcomes

E : dice1 is 3 : E = {31,32,33,34,35,36}
                 P[E] = 6/36 =1/6
F : dice1 + +dice2 == 7
                 F = {16,25,34,43,52,61}
                 P[F] = 6/36 =1/6
G : dice1**2 + dice2**2 == 25
                 G = {34,43}
                 P[G] = 2/36 = 1/18
```

In [ ]:

```

```

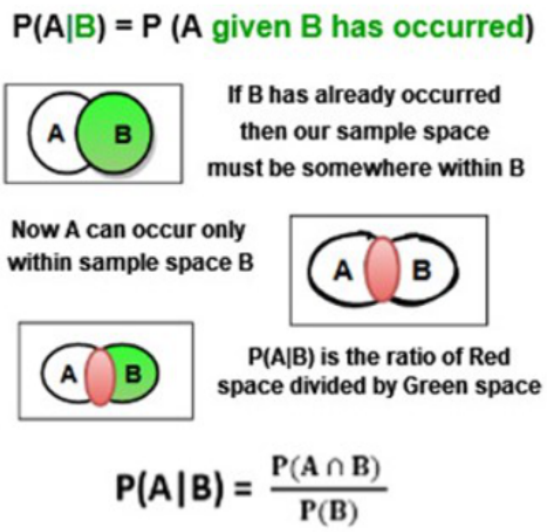
In [ ]:

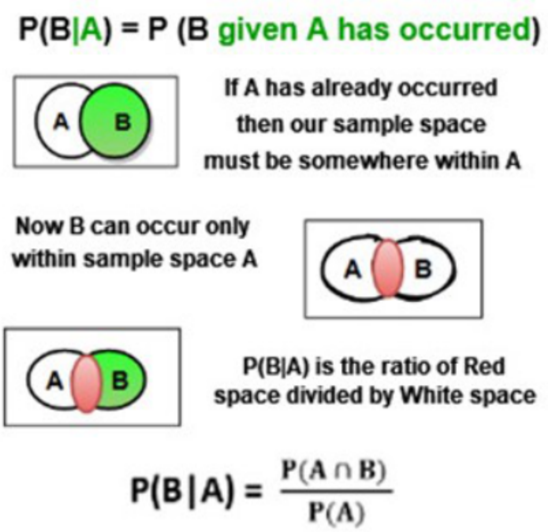
```

```

Covid example

```
# Covid example

U : 1000 all covid patients
m1 : 100 given medicine
S : 700 survived


probability os patient given medicine
P[m1] = 100/1000 = 0.1

probability of survival
```

```
P[S] = 700/1000 = 0.7

people who servived and given medicine is : m1 intersection S
                                            m1 ∩ S = 95 (given in data)
          probability of survival who given  medicine = 95/1000
          P[ m1 ∩ S ] = 95/1000



now  , if we want to know, the effectiveness of drug :
we need to find number of people survived who were given drug
we need to find probability of survival after given the drug,
probability wrt the people given medicine(condition)

P[S|m1] = probability of # of survival given # of people medicine
          = P[S ∩ m1] / P[m1]
        probability  of S given m1 . (conditional probability)
        = (95/1000) / (100/1000)
        = 95 / 100 = 0.95


So , probability of Survival is 0.7
and probability of survival given medicine  is 0.95
there are higher chances survival with medicine :

```






```
# general representation of events A and B

P[A|B] = P[A ∩ B] / P[B]
P[B|A] = P[A ∩ B] / P[A]

```

```
Conditional Probability :

P[A|B] = P[A ∩ B] / P[B]
P[A ∩ B] = P[B|A]*P[A]


P[A|B] = P[A ∩ B] / P[B]
       = P[B|A]*P[A] / P[B]

```

independent events

```
what if :

probability os patient given medicine
P[m1] = 100/1000 = 0.1

```

```
probability of survival
P[S] = 700/1000 = 0.7

P[ m1 ∩ S ] = 70/1000

P[S|m1] = probability of # of survival given # of people medicine
        = P[S ∩ m1] / P[m1]
        = 70 / 100 = 0.7


So , probability of Survival is 0.7  = probability of survival given medicine  is 0.7  ( P[S|m1] = P[S] )

there for no significant probability difference

That means the m1 and S are independent events :
P[S|m1] = P[S]
```

```
P[A|B] = P[A] defination of indepedence

in above example :

P[S|m1] = P[S]

P[S ∩ m1] / P[m1] = P[S]
```

P[A|B] = P[A] defination of indepedence

```
# Independent Events :
A and B are independent Events if :

P[A|B] == P[A]
P[B|A] == P[B]

P[A ∩ B] = P[A]*P[B]

"""
ratio of A happeneing inside B
is equal to
the ration of B happening inside the whole universe

"""

```

dice 2 times : independece

```
dice 2 times : independece

sample space :
{
11 12 13 14 15 16
21 22 23 24 25 26
31 32 33 34 35 36
41 42 43 44 45 45
51 52 53 54 55 56
61 62 63 64 65 66
}
total 36 outcomes

E : dice1 is 3 : E = {31,32,33,34,35,36}
                 P[E] = 6/36 =1/6
F : dice1 + +dice2 == 7
                 F = {16,25,34,43,52,61}
                 P[F] = 6/36 =1/6
G : dice1**2 + dice2**2 == 25
                 G = {34,43}
                 P[G] = 2/36 = 1/18
E ∩ F = {34}
P[E ∩ F] = 1/36
```

```

P[E|F] = P[E ∩ F]/ P[F]
       = (1/36)  / (6/36)
       = 1/6
       = P[E]

So , E and F are independent events
two dice are independent events
```

```
     P[E|F]       = P[E ∩ F]                 /     P[F]
conditional_prob  = num_favourable_samples  / num_of_conditioned_samples


```

In [2]:

```python
trials = 1100
num_favourable_samples = 0    # E and F
num_conditional_samples = 0   # F

for t in range(trials):
    dice1 = np.random.choice([1,2,3,4,5,6])
    dice2 = np.random.choice([1,2,3,4,5,6])

    if dice1 + dice2 == 7 :
        num_conditional_samples += 1
        if  dice1 == 3:
            num_favourable_samples+= 1
conditional_probability = num_favourable_samples / num_conditional_samples
conditional_probability

```

Out[2]: 0.21839080459770116

In [3]:

```python
expected_value = 1/6
expected_value
```

Out[3]: 0.16666666666666666

dice and coin together : independence:

```
S = { H1,H2,H3,H4,H5,H6,
      T1,T2,T3,T4,T5,T6 }

take an experimental events  :
E = coin is head = { H1,H2,H3,H4,H5,H6 }   = 6/12 = 1/2 = P[E]
F = dice is 3 = { H3,T3 }                  = 2/12 = 1/6 = P[F]
E ∩ F = {H3}              P[E ∩ F] = 1/12


probability of E given F:

P[E|F] = P[E ∩ F]/ P[F]
       = (1/12)  / (1/6)
       = 6 / 12
       = 1 / 2
       = P[E]

probability of F given E:

P[F|E] = P[E ∩ F]/ P[E]
       = (1/12)  / (1/2)
       = 2 / 12
       = 1 / 6
       = P[F]



So , E and F are independent events
       So , dice and coin toss are independent events
```

In [4]:

```
trials = 1000
num_favourable_samples = 0    # E and F
num_conditional_samples = 0   # F

for t in range(trials):
    coin = np.random.choice(["H","T"])
    dice = np.random.choice([1,2,3,4,5,6])

    if dice == 3 :
        num_conditional_samples += 1
        if  coin == "H":
            num_favourable_samples+= 1
conditional_probability = num_favourable_samples / num_conditional_samples
conditional_probability
            # P[E|F] = P[E ∩ F]/ P[F]
```

Out[4]: 0.4861878453038674

In [5]:

```
expected_value = 1/2   # P[E|F] = P[E ∩ F]/ P[F]
expected_value
```

Out[5]: 0.5

In [6]:

```
trials = 1000
num_favourable_samples = 0    # E and F
num_conditional_samples = 0   # F

for t in range(trials):
    coin = np.random.choice(["H","T"])
    dice = np.random.choice([1,2,3,4,5,6])

    if dice == 3 and coin == "H" :
        num_favourable_samples += 1
    if coin == "H":
        num_conditional_samples += 1
conditional_probability = num_favourable_samples / num_conditional_samples
conditional_probability
            # P[F|E] = P[E ∩ F]/ P[E]
```

Out[6]: 0.15415821501014199

In [7]:

```
expected_value = 1/6   # P[F|E] = P[E ∩ F]/ P[E]
expected_value
```

Out[7]: 0.16666666666666666

In [ ]:

In [ ]:

In [ ]:

In [8]:

```
# A ∩ B
# A ∪ B
# A ⊂ B
```

In [ ]:

Dice :

```
Dice :

S = {1,2,3,4,5,6}

E1 = {1,2}     probability of E1 : P[E1] = 2/6
E2 = {1,3,5}   probability of E2 : P[E2] = 3/6

probability of Event E1 and E2 :
E1 ∩ E2 = {1}
P[E1 ∩ E2] = 1/6

probability of Event E1 or E2 :

E1 ∪ E2 = {1,2,3,5}

P[ E1 ∪ E2 ] = 4/6

{1,2}  ∪ {1,3,5} = {1,2} + {1,3,5} - {1} # as 1 is intersectio of both events :
so :

P[ E1 ∪ E2 ] = P[E1] + P[E2] - P[E1 ∩ E2] = 4/6
               {1,2}   {1,3,5}     {1}
             = 2/6    + 3/6     - 1/6

             = 4/6
```

In [9]:
```
(2/6)    + (3/6  )  - (1/6)
```

Out[9]: 0.6666666666666666

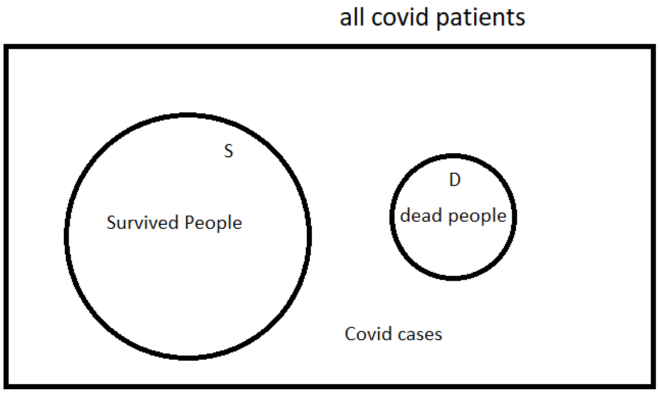
In [10]:
```
4/6
```

Out[10]: 0.6666666666666666

Covid Example

∩∪

```
Covid Example :

U : all patients
S : survived (recovered)
C : cases  (symptomatic )
d : died

S ∩ d = {}   :
S ∩ c = {}   :     Mutually Exclusive Events
c ∩ d = {}   :

S ∪ c ∪ d = Universe    Mutually Exhausted Events
```




```
 A ∩ A' = {}
 A ∪ A' = S
so , A and A' are mtually exclusive and exhaustive
```

In [ ]:

```

```

In [ ]:

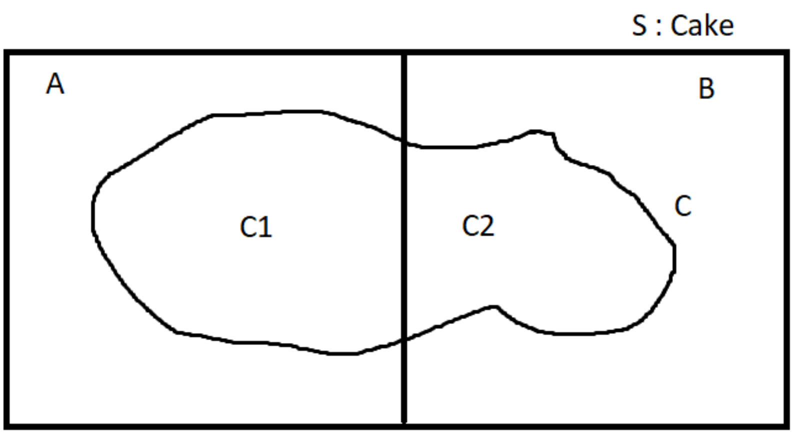
```

```

P[A or B or C] = P[A] + P[B] + P[C] - P[A and B] + P[B and C] + P[A and C] + P[A and B and C]

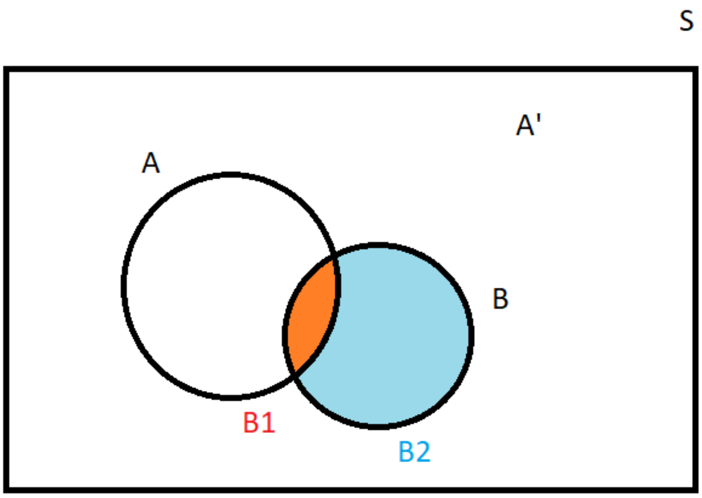
∩∪

Cake example :

```
Universe U = SampleSpace S = Cake

cut the cake in two pieces A and B:


there's a cream on cake S : called C

both pieces of cream C1 and C2

A ∩ B = {}  # mutually exclusive
A ∪ B = whole cake : S : # mutually exhaustive

A and B are mutually exhuastive and exclusive events

```



```
C1 is intersection of whole cream and part A cake
C2 is intersection of whole cream and part B cake

C1 = C ∩ A
C2 = C ∩ B

whole cream : C = C1 ∪ C2
and   C1 ∩ C2 = {}

P[C] = P[C1]+P[C2] - P[C1 ∩ C2]
     = P[C1]+P[C2] -  0
     = P[C1]+P[C2]
     = P[C ∩ A] + P[C ∩ B]

as per conditional probability :
we can replace :
P[C ∩ A] = P[C|A]*P[A]     = P[A|C]*P[C] / P[A]
P[C ∩ B] = P[C|B]*P[B]     = P[B|C]*P[B] / P[B]




P[C] = P[C1]+P[C2] - P[C1 ∩ C2]
     = P[C1]+P[C2] -  0
     = P[C1]        + P[C2]
     = P[C ∩ A]     + P[C ∩ B]
     = P[C|A]*P[A] + P[C|B]*P[B]
```




```
B = B1 ∪ B2

P[B] = P[B1]        +  P[B2]
     = P[B ∩ A]     +  P[ B ∩ A']
     = P[B|A]*P[A]  +  P[B|A']*P[A']
```

```

A ∩ A' = {}
A ∪ A' = S

so , A and A' are mtually exclusive and exhaustive

P[A|B] = P[A ∩ B]/P[B] = P[B|A]*P[A] / P[B]

       = P[B|A]*P[A]   /  ( P[B|A]*P[A] + P[B|A']*P[A'])
```

In [ ]:

```
```

In [ ]:

```
```

In [ ]:

```
```

```
P[C] = P[C1]+P[C2] - P[C1 ∩ C2]
     = P[C1]+P[C2] -  0
     = P[C1]       + P[C2]
     = P[C ∩ A]    + P[C ∩ B]
     = P[C|A]*P[A] + P[C|B]*P[B]
```

In [ ]:

```
```

In [ ]:

```
```

Covid case

In [ ]:

```
```

```
people : heathy or discesed
    test : +ve or -ve
```

```
P[H] = 0.9 #(90% population is heathy)
that means P[D] = 1- P[H] = 0.1 (disceased)



P[-ve|H] = 0.75    P[-ve|D] = 0.25

P[+ve|H] = 0.25    P[+ve|D] = 0.75

```

```
# if we get a +ve , what is the probability that i m still healthy

P[H|+ve]
```

In [11]:

```
# P[A|B] = P[A ∩ B]/P[B] = P[B|A]*P[A] / P[B]

#         = P[B|A]*P[A]   /  ( P[B|A]*P[A] + P[B|A']*P[A'])

```

```
P[H|+ve] = P[+ve|H]*P[H]   /  ( P[+ve|H]*P[H] + P[+ve|D]*P[D])
         = (0.25 * 0.9)  /  (0.25*0.9 + 0.75*0.1 )
         = 0.75
```

In [12]:

```
(0.25 * 0.9)  /  ((0.25*0.9) + ( 0.75*0.1 ))
```

Out[12]: 0.7499999999999999

In [ ]:

```
```

In [ ]:

```
```

```

P[D] = 0.6   p
P[H] = 0.4   1-p
```

```
P[+ve|D] = 0.9  d
P[+ve|H] = 0.1  h
P[D|+ve] = ?

P[D|+ve] = P[+ve|D]*P[D]   /  ( P[+ve|D]*P[D] + P[+ve|H]*P[H])
         ( 0.9 * 0.6) /      (  ( 0.9 * 0.6)  + (0.1 * 0.4))

           (d*p) / ((d*p)+(h*(1-p)))

```

In [13]:

```
( 0.9 * 0.6) /      (  ( 0.9 * 0.6)  + (0.1 * 0.4))
```

Out[13]: 0.9310344827586207

In [14]:

```
# # h|+VE  = [+VE|h]  * [h ]   /   [+VE|h]  * [h ] + [+VE|D]  * [D]
#          (0.25  * 0.9) /   (( 0.25*0.9) + (0.75 * 0.1))
```

In [15]:

```
def solve(p,d,h):
    # YOUR CODE GOES HERE

    return  np.round((d*p) / ((d*p)+(h*(1-p))),3)

```

In [16]:

```
solve(0.6,0.9,0.1)
```

Out[16]: 0.931

In [ ]:

In [17]:

```
(0.25  * 0.9) /   (( 0.25*0.9) + (0.75 * 0.1))
```

Out[17]: 0.7499999999999999

In [ ]:

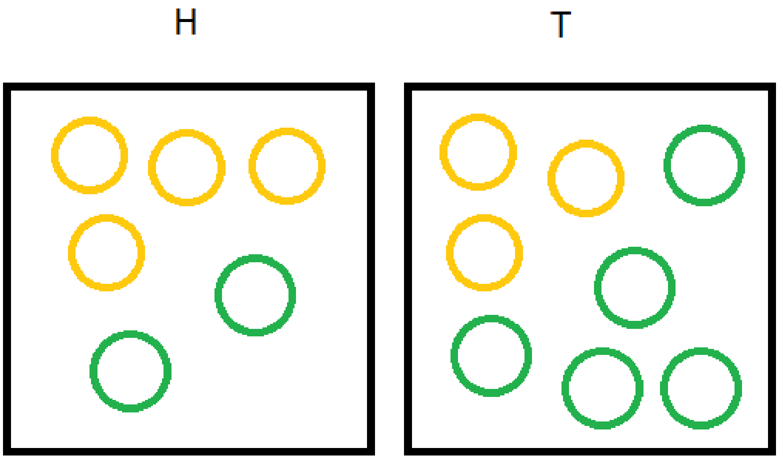
In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:




```
two boxes :
1st contains  4 yellow and 2 green balls
2nd contains 5 green and 3 yellow balls.

we toss a coin
if H : we select 1st box
if T : we select 2nd box

if the ball so chosen is yello , what is the probability that coin toss was heads.

P[H|Y] = ?

```

```
P[H|Y] = P[Y|H]*P[H]
         / P[Y|H]*P[H] + P[Y|T]*P[T]

       =  ((4/6)*(1/2))
    /(((4/6)*(1/2))+((3/8)*(1/2)))
```

In [18]:
```python
((4/6)*(1/2))/(((4/6)*(1/2))+((3/8)*(1/2)))
```

Out[18]: 0.64

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]: