In [1]:

```
# prob of getting 3 when die is rolled
# Ans :p =1/6

#n(s)= (1,2,3,4,5,6)
ns= 6

#n(A) = getting 3 = {3}
na = 1

#p(A)
pa =na/ns
print("probability of getting 3 is :",pa)
```

```
probability of getting 3 is : 0.16666666666666666
```

In [2]:

```
# prob of atleast getting one head when a coin is tossed

# 5 ={HHH,TTT,HTH,THH,TTH,THT,HTT,HHT}

ns= 8
na = 7
#p(A)
pa=na/ns
print("Probability of getting atleast one head when a coin is toss is :",pa)
```

```
Probability of getting atleast one head when a coin is toss is : 0.875
```

In [3]:

```
# example :3
# A glass jar contain 5 red , 3 blue  and 2 green jelly beans, what is probablity that it i

# s ={r1,r2,r3,r4,r4,r5,b1,b2,b3,g1,g2}
ns= 10

#A = it is not blue ={r1,r2,r3,r4,r4,r5,g1,g2}
#p(A)
pa=na/ns
print("Probability of the jelly bean is not blue is :",pa)
```

```
Probability of the jelly bean is not blue is : 0.7
```

In [4]:

```
# example :4
# if  the prob that person A will be alive i 20 years is 0.7
# and the prob of person B will be alive in 20 years is 0.5
# what is prob that they both will be alive in  20 years?

# these are independent event ,50

p= 0.7*0.5
print("prob they both will is alive in 20 years is :",p)
```

```
prob they both will is alive in 20 years is : 0.35
```

In [5]:

```
# A die is tossed twice , find the prob of getting 4  or 5 on first toss and 1,2,3 in secon
def event_prob(n,s):
    return n/s
```

In [6]:

```
# example 5

# A die i tossed  twise find the prob of getting a 4 or 5 on the first toss
# and a 1,2 or 3 in second tossed

# using my func

# prob of getting a 4 or 5 on the first toss ,2 event total sample is 6
pa= event_prob(2,6)


# prob of getting a 1,2 or 3 on second  toss 3 event total sample is 6
pb=event_prob(3,6)


# total prob of both will be
p=pa*pb
print("prob of getting a4 or 5 on the first toss and a 1,2 or 3 in the second toss is :",p)
```

```
prob of getting a4 or 5 on the first toss and a 1,2 or 3 in the second toss
is : 0.16666666666666666
```

In [7]:

```
# example :6
# A



#S = {5 white ,3 black , 2 green }
ns = 10

# event 1 {getting white}
na=5
pa= event_prob(na,10)

#event 2 {getting black}
nb= 3
pb= event_prob(na,9)

#event 3 {getting green}
nc= 2
pc= event_prob(na,8)
p=pa*pb*pc

print("prob of obtaining white ,black amd green in order is:",p)
```

```
prob of obtaining white ,black amd green in order is: 0.1736111111111111
```

In [8]:

```
# example :7
# sample space
cards =52

# calculate the prob of drawing a heart or a club

hearts =13
clubs= 13

#13/52 + 13/52
hearts_or_clubs = event_prob(hearts,cards )+ event_prob(clubs,cards)
print("prob of drawing a hearts or a clubs is :",hearts_or_clubs)
```

```
prob of drawing a hearts or a clubs is : 0.5
```

In [9]:

```
# ex 8

ace = 4
king = 4
queen = 4
ace_or_queen = event_prob(ace,cards)+event_prob(king,card)+event_prob(queen,cards)
print(ace)
print(king)
print(queen)
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_1924/3862312216.py in <module>
      4 king = 4
      5 queen = 4
----> 6 ace_or_queen = event_prob(ace,cards)+event_prob(king,card)+event_pro
b(queen,cards)
      7 print(ace)
      8 print(king)

NameError: name 'card' is not defined
```

In [10]:

```
#ex 9
hearts = 13
ace= 4
ace_of_heart=1
heart_or_ace=event_prob(heart,cards) + event_prob(ace,cards)- event_prob(ace_of_heart,cards
print("the prob of drawing a heart or an ace is :",round(heart_or_ace,1))

#round (heart_or_ace,1)
#1-1 element after decimal
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_1924/2564729771.py in <module>
      3 ace= 4
      4 ace_of_heart=1
----> 5 heart_or_ace=event_prob(heart,cards) + event_prob(ace,cards)- event_
prob(ace_of_heart,cards)
      6 print("the prob of drawing a heart or an ace is :",round(heart_or_ac
e,1))
      7 

NameError: name 'heart' is not defined
```

In [13]:

```
# ex 10

# cal prob of drawing red card or face cards
# 13 heart + 13 dimond
red = 26

# king+queen+j each set of 4 which is 3*4=12
face_card = 12

# but there is red face card {3 heart +3 dimond}
red_face= 6

red_or_face=event_prob(red,card)+event_prob(face_card,card)-event_prob(red_face,card)

print("prob of drawing a red card or face card is:",round(red_or_face,2))
# round(red_or_face)
# 2 element after dimond
```

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp/ipykernel_1924/74681778.py in <module>
     11 red_face= 6
     12
---> 13 red_or_face=event_prob(red,card)+event_prob(face_card,card)-event_pr
ob(red_face,card)
     14
     15 print("prob of drawing a red card or face card is:",round(red_or_fac
e,2))

NameError: name 'card' is not defined
```

In [14]:

```
# ex 13
# prob of not geting 5 when die is rolled.
# s +{1,2,3,4,5,,6}
ns = 6

# getting 5
na=1

pa= event_prob(na,ns)
print("prob of not getting 5 is :",round(1-event_prob(na,ns),2))
# both the way are same
# prob of not getting 5 is - prob of getting 5
print("prob of not getting 5 is :",round(1-pa,2))
```

```
prob of not getting 5 is : 0.83
prob of not getting 5 is : 0.83
```

In [15]:

```
# ex 12(conditional prob)

# suppose you draw 2 card from deck
# you win if you get ace given thatb you draw a jack is first draw

card= 52

j= 4
ace= 4

# prob of jack in first draw
pj= event_prob(j,card)

#prob of ace in second draw and not putting previous card
pa= event_prob(ace,51)

# formula  of conditional prob
pa_given_pj =(pa*pj)/pj
print(round(pa_given_pj,2))
print(pa_given_pj)
```

```
0.08
0.0784313725490196
```

In [16]:

```
import pandas as pd
import numpy  as np

df= pd.read_csv("student-mat - student-mat.csv")
```

In [17]:

```
df.head()
```

Out[17]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	fr
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	5	
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	4	
3	GP	F	15	U	GT3	T	4	2	health	services	...	3	
4	GP	F	16	U	GT3	T	3	3	other	other	...	4	

5 rows × 33 columns

In [18]:

```
df.tail()
```

Out[18]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel
390	MS	M	20	U	LE3	A	2	2	services	services	...	5
391	MS	M	17	U	LE3	T	3	1	services	services	...	2
392	MS	M	21	R	GT3	T	1	1	other	other	...	5
393	MS	M	18	R	LE3	T	3	2	services	other	...	4
394	MS	M	19	U	LE3	T	1	1	other	at_home	...	3

5 rows × 33 columns

calculate the probability a student gets A(80% +)in maths given they miss 10 or more classes.

we are concerned with the columns absences (number of absences)and G3(final grad from 0 to -20)

adding the boolean columns called grade_A

In [19]:

```
df['grade_A'] = np.where(df['G3']*5 >=80,1,0)
```

Makeing

In [20]:

```
df["high_absences"]=np.where(df["absences"] >=10,1,0)
```

In [ ]:

In [21]:

```
df['count']=1
```

In [ ]:

In [22]:

```
df= df[['grade_A','high_absences','count']]
df.head()
```

Out[22]:

	grade_A	high_absences	count
0	0	0	1
1	0	0	1
2	0	1	1
3	0	0	1
4	0	0	1

In [ ]:

```

```

In [23]:

```
final = pd.pivot_table(
   df,
   values='count',
   index=['grade_A'],
   columns=['high_absences'],
   aggfunc=np.size,
   fill_value =0)
```

In [24]:

```
print(final)
```

```
high_absences    0   1
grade_A
0              277  78
1               35   5
```

A= getting high grades B= high absence

In [25]:

```
# pa = (35+3)/(35+5+277+78)\

# a= final.iloc[0,1]
# a

pa = [(final.iloc[0,1] + final.iloc[1,1]) / (final.iloc[0,0] + final.iloc[0,1] + final.iloc
pa
```

Out[25]:

```
[0.21012658227848102]
```

In [26]:

```
# pb=(78+5)/(35+5+277+78)
# pa

pb = [(final.iloc[0,1] + final.iloc[1,1]) / (final.iloc[0,0] + final.iloc[0,1] + final.iloc
pb
```

Out[26]:

```
[0.21012658227848102]
```

In [29]:

```
# pa_or_pb =5 /(35+5+277+78)
# pa_or_pb

pa_or_pb = (final.iloc[1,1]) / (final.iloc[0,0] + final.iloc[0,1] + final.iloc[1,0] + final
pa_or_pb
```

Out[29]:

```
0.012658227848101266
```

In [30]:

```
A_given_B = pa_or_pb / pb
A_given_B
```

Out[30]:

```
array([0.06024096])
```

In [31]:

```
print(A_given_B)
```

```
[0.06024096]
```

In [ ]: