In [3]:

```
#probability of getting 3 when a die is rolled
ns=6 #n(S) = {1,2,3,4,5,6}
na=1 #n(A) ={3}
pa= na/ns #P(A)
print("probability of getting 3 is:",pa)
```

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

In [4]:

```
#Probability of atleast getting one head when a coin is tossed thrice
ns=8 #n(s) = {HHH,HTH,HHT,THH,TTT,THH,TTH,THT}
na=7 #n(A) ={HHH,HTH,HHT,THH,TTH,THT,THH}
pa=na/ns #P(A)
print("prob. of getting head is:",pa)
```

```
prob. of getting head is: 0.875
```

In [5]:

```
# A glass jar contain 5 red , 3 blue and 2 green jelly beans.
#if a jelly bean is chosen at random from the jar,
# what is the prob. that it is not blue?

ns=10 #n(s) = {R,R,R,R,R,B,B,B,G,G}
na= 7 #n(A) = {R,R,R,R,R,G,G}
pa=na/ns #p(A)
print("prob. of not getting blue:",pa)
```

```
prob. of not getting blue: 0.7
```

In [6]:

```
# If the probability that person A  will be alive in 20 years
# is 0.7 and the probability that person B will be alive in
# 20 years is 0.5, what is the probability that they will
# both be alive in 20 years?
# These are independent event so,
P=0.7*0.5
print("probabilty that they will be alive after 20 years is:", P)
```

```
probabilty that they will be alive after 20 years is: 0.35
```

In [9]:

```
def event_probability(n,s):
    return n/s
```

In [10]:

```
#A fair die is tossed twice, Find the probability of getting
#a 4 or 5 on the first tosse and a 1,2, or 3 in the second toss.
pa = event_probability(2,6)
pb = event_probability(3,6)
p = pa*pb
print("probability of getting a 4 or 5 on the first toss and a 1,2 or 3 in second toss is :
```

```
probability of getting a 4 or 5 on the first toss and a 1,2 or 3 in second t
oss is : 0.16666666666666666
```

In [12]:

```
# A bag contains 5 white marbles, 3 black marbles and 2 green marbles,
# in each draw, a marble is drawn from the bag
# and not replaced. In three draws, find the prob. of obtaining white,
# black and green in that order
pa = event_probability(5,10)
pb = event_probability(3,9)
pc = event_probability(2,8)
print("prob. of obtaining white,black and green in the order:",pa ,pb, pc)
```

```
prob. of obtaining white,black and green in the order: 0.5 0.333333333333333
3 0.25
```

In [14]:

```
# sample space
cards = 52
# calculate the prob.  of drawing a heart or a
hearts = 13
clubs = 13
heart_or_club = event_probability(hearts,cards) + event_probability(clubs,cards)
print(heart_or_club)
```

```
0.5
```

In [21]:

```
# calculate the prob. of drawing an ace , king, or a queen
cards=52
ace = 4
king = 4
queen = 4
ace_king_or_queen = event_probability(ace,cards)+event_probability(king,cards)+event_probab
print(round(ace_king_or_queen,2))
```

```
0.23
```

In [20]:

```
cards = 52
heart = 13
ace = 4
ace_of_heart = 1
heart_or_ace = event_probability(heart,cards)+event_probability(ace,cards)-event_probabilit
print(round(heart_or_ace,1))
```

```
0.3
```

In [23]:

```
#calculate the prob. of drawing red cards or face cards
red = 26
facecards = 12
red_facecards = 6
red_or_facecards = event_probability(red,cards)+event_probability(facecards,cards)-event_pr
print(round(red_or_facecards,2))
```

```
0.62
```

In [25]:

```
# prob. of not getting 5 when a fair die is rolled
ns=6 #n(s) = {1,2,3,4,5,6}
na=1 #n(a) ={5}
pa=1-na/ns
print("prob. of not getting 5 is:",pa)
```

```
prob. of not getting 5 is: 0.8333333333333334
```

In [26]:

```
#suppose you draw 2 cards from deck.
# you win if you get ace given that you draw a jack in first draw
# we used conditional prob.
cards = 52
j = 4
ace = 4
pj = event_probability(j,52)
pa = event_probability(ace,51)
pa_given_pj=(pa*pj)/pj
print(pa_given_pj)
```

```
0.0784313725490196
```

In [1]:

```
# conditional probability P(A/B)=P(A & B)/P(B)

import pandas as pd
import numpy as np
df = pd.read_csv("C:/Users/MSCIT/Downloads/student-mat - student-mat.csv")
df.head(3)
```

Out[1]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	famrel	fre
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 rows × 33 columns

In [2]:

```
len(df)
```

Out[2]:

395

In [5]:

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

Out[5]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	freetime
0	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	3
1	GP	F	17	U	GT3	T	1	1	at_home	other	...	3
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	3
3	GP	F	15	U	GT3	T	4	2	health	services	...	2
4	GP	F	16	U	GT3	T	3	3	other	other	...	3
...	...	...	...	...	...	...	...	...	...	...	...	..
390	MS	M	20	U	LE3	A	2	2	services	services	...	5
391	MS	M	17	U	LE3	T	3	1	services	services	...	4
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	...	2

395 rows × 34 columns

In [7]:

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

Out[7]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	goout	Dalc
)	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	4	1
	GP	F	17	U	GT3	T	1	1	at_home	other	...	3	1
!	GP	F	15	U	LE3	T	1	1	at_home	other	...	2	2
3	GP	F	15	U	GT3	T	4	2	health	services	...	2	1
4	GP	F	16	U	GT3	T	3	3	other	other	...	2	1
.	...	...	...	...	...	...	...	...	...	...	...	...	...
)	MS	M	20	U	LE3	A	2	2	services	services	...	4	4
	MS	M	17	U	LE3	T	3	1	services	services	...	5	3
!	MS	M	21	R	GT3	T	1	1	other	other	...	3	3
3	MS	M	18	R	LE3	T	3	2	services	other	...	1	3
4	MS	M	19	U	LE3	T	1	1	other	at_home	...	3	3

rows × 35 columns

In [9]:

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

Out[9]:

	school	sex	age	address	famsize	Pstatus	Medu	Fedu	Mjob	Fjob	...	Dalc	Walc
)	GP	F	18	U	GT3	A	4	4	at_home	teacher	...	1	1
I	GP	F	17	U	GT3	T	1	1	at_home	other	...	1	1
2	GP	F	15	U	LE3	T	1	1	at_home	other	...	2	3
3	GP	F	15	U	GT3	T	4	2	health	services	...	1	1
4	GP	F	16	U	GT3	T	3	3	other	other	...	1	2
.	...	...	...	...	...	...	...	...	...	...	...	...	...
)	MS	M	20	U	LE3	A	2	2	services	services	...	4	5
I	MS	M	17	U	LE3	T	3	1	services	services	...	3	4
2	MS	M	21	R	GT3	T	1	1	other	other	...	3	3
3	MS	M	18	R	LE3	T	3	2	services	other	...	3	4
4	MS	M	19	U	LE3	T	1	1	other	at_home	...	3	3

rows × 36 columns

In [17]:

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

Out[17]:

	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 [19]:

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

In [20]:

```
print(final)
```

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

In [24]:

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

Out[24]:

```
0.10126582278481013
```

In [34]:

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

Out[34]:

```
0.21012658227848102
```

In [36]:

```
p_c=(final.iloc[1,1])/total
p_c
```

Out[36]:

```
0.012658227848101266
```

In [39]:

```
P_d=p_c/p_b
P_d
```

Out[39]:

```
0.060240963855421686
```

In [ ]:

```

```