Itertools - Functions creating iterators for efficient looping

An arrangement of a set of n objects in a given order is called a permutation of the objects (taken all at a time).

Practice with permutations

In [1]:

```
from itertools import permutations
print(list(permutations(['a','b','c'], 3)))
```

```
[('a', 'b', 'c'), ('a', 'c', 'b'), ('b', 'a', 'c'), ('b', 'c', 'a'), ('c', 'a', 'b'), ('c
', 'b', 'a')]
```

In [3]:

```
from itertools import permutations
print(list(permutations(['a','b','c'], 2)))
```

```
[('a', 'b'), ('a', 'c'), ('b', 'a'), ('b', 'c'), ('c', 'a'), ('c', 'b')]
```

In [4]:

```
from itertools import permutations
print(list(permutations(['a','b','c'], 1)))
```

```
[('a',), ('b',), ('c',)]
```

In [2]:

```
from itertools import permutations
l = list(permutations(range(1, 5)))
print (l)
```

```
[(1, 2, 3, 4), (1, 2, 4, 3), (1, 3, 2, 4), (1, 3, 4, 2), (1, 4, 2, 3), (1, 4, 3, 2), (2, 1
, 3, 4), (2, 1, 4, 3), (2, 3, 1, 4), (2, 3, 4, 1), (2, 4, 1, 3), (2, 4, 3, 1), (3, 1, 2, 4
), (3, 1, 4, 2), (3, 2, 1, 4), (3, 2, 4, 1), (3, 4, 1, 2), (3, 4, 2, 1), (4, 1, 2, 3), (4,
1, 3, 2), (4, 2, 1, 3), (4, 2, 3, 1), (4, 3, 1, 2), (4, 3, 2, 1)]
```

In [3]:

```
from itertools import permutations
l = list(permutations(range(1, 6)))
print (l)
```

```
[(1, 2, 3, 4, 5), (1, 2, 3, 5, 4), (1, 2, 4, 3, 5), (1, 2, 4, 5, 3), (1, 2, 5, 3, 4), (1,
2, 5, 4, 3), (1, 3, 2, 4, 5), (1, 3, 2, 5, 4), (1, 3, 4, 2, 5), (1, 3, 4, 5, 2), (1, 3, 5,
2, 4), (1, 3, 5, 4, 2), (1, 4, 2, 3, 5), (1, 4, 2, 5, 3), (1, 4, 3, 2, 5), (1, 4, 3, 5, 2)
, (1, 4, 5, 2, 3), (1, 4, 5, 3, 2), (1, 5, 2, 3, 4), (1, 5, 2, 4, 3), (1, 5, 3, 2, 4), (1,
5, 3, 4, 2), (1, 5, 4, 2, 3), (1, 5, 4, 3, 2), (2, 1, 3, 4, 5), (2, 1, 3, 5, 4), (2, 1, 4,
3, 5), (2, 1, 4, 5, 3), (2, 1, 5, 3, 4), (2, 1, 5, 4, 3), (2, 3, 1, 4, 5), (2, 3, 1, 5, 4)
, (2, 3, 4, 1, 5), (2, 3, 4, 5, 1), (2, 3, 5, 1, 4), (2, 3, 5, 4, 1), (2, 4, 1, 3, 5), (2,
4, 1, 5, 3), (2, 4, 3, 1, 5), (2, 4, 3, 5, 1), (2, 4, 5, 1, 3), (2, 4, 5, 3, 1), (2, 5, 1,
3, 4), (2, 5, 1, 4, 3), (2, 5, 3, 1, 4), (2, 5, 3, 4, 1), (2, 5, 4, 1, 3), (2, 5, 4, 3, 1)
, (3, 1, 2, 4, 5), (3, 1, 2, 5, 4), (3, 1, 4, 2, 5), (3, 1, 4, 5, 2), (3, 1, 5, 2, 4), (3,
1, 5, 4, 2), (3, 2, 1, 4, 5), (3, 2, 1, 5, 4), (3, 2, 4, 1, 5), (3, 2, 4, 5, 1), (3, 2, 5,
1, 4), (3, 2, 5, 4, 1), (3, 4, 1, 2, 5), (3, 4, 1, 5, 2), (3, 4, 2, 1, 5), (3, 4, 2, 5, 1)
, (3, 4, 5, 1, 2), (3, 4, 5, 2, 1), (3, 5, 1, 2, 4), (3, 5, 1, 4, 2), (3, 5, 2, 1, 4), (3,
5, 2, 4, 1), (3, 5, 4, 1, 2), (3, 5, 4, 2, 1), (4, 1, 2, 3, 5), (4, 1, 2, 5, 3), (4, 1, 3,
2, 5), (4, 1, 3, 5, 2), (4, 1, 5, 2, 3), (4, 1, 5, 3, 2), (4, 2, 1, 3, 5), (4, 2, 1, 5, 3)
, (4, 2, 3, 1, 5), (4, 2, 3, 5, 1), (4, 2, 5, 1, 3), (4, 2, 5, 3, 1), (4, 3, 1, 2, 5), (4,
3, 1, 5, 2), (4, 3, 2, 1, 5), (4, 3, 2, 5, 1), (4, 3, 5, 1, 2), (4, 3, 5, 2, 1), (4, 5, 1,
2, 3), (4, 5, 1, 3, 2), (4, 5, 2, 1, 3), (4, 5, 2, 3, 1), (4, 5, 3, 1, 2), (4, 5, 3, 2, 1)
, (5, 1, 2, 3, 4), (5, 1, 2, 4, 3), (5, 1, 3, 2, 4), (5, 1, 3, 4, 2), (5, 1, 4, 2, 3), (5,
1, 4, 3, 2), (5, 2, 1, 3, 4), (5, 2, 1, 4, 3), (5, 2, 3, 1, 4), (5, 2, 3, 4, 1), (5, 2, 4,
1, 3), (5, 2, 4, 3, 1), (5, 3, 1, 2, 4), (5, 3, 1, 4, 2), (5, 3, 2, 1, 4), (5, 3, 2, 4, 1)
```

```
, (5, 3, 4, 1, 2), (5, 3, 4, 2, 1), (5, 4, 1, 2, 3), (5, 4, 1, 3, 2), (5, 4, 2, 1, 3), (5,
4, 2, 3, 1), (5, 4, 3, 1, 2), (5, 4, 3, 2, 1)]
```

Practice with combinations

In [8]:

```
from itertools import combinations
print(list(combinations(['a','b','c', 'd', 'e'], 1)))
```

```
[('a',), ('b',), ('c',), ('d',), ('e',)]
```

In [9]:

```
from itertools import combinations
print(list(combinations(['a','b','c', 'd', 'e'], 2)))
```

```
[('a', 'b'), ('a', 'c'), ('a', 'd'), ('a', 'e'), ('b', 'c'), ('b', 'd'), ('b', 'e'), ('c'
, 'd'), ('c', 'e'), ('d', 'e')]
```

In [10]:

```
from itertools import combinations
print(list(combinations(['a','b','c', 'd', 'e'], 3)))
```

```
[('a', 'b', 'c'), ('a', 'b', 'd'), ('a', 'b', 'e'), ('a', 'c', 'd'), ('a', 'c', 'e'), ('a
', 'd', 'e'), ('b', 'c', 'd'), ('b', 'c', 'e'), ('b', 'd', 'e'), ('c', 'd', 'e')]
```

In [11]:

```
from itertools import combinations
print(list(combinations(['a','b','c', 'd', 'e'], 4)))
```

```
[('a', 'b', 'c', 'd'), ('a', 'b', 'c', 'e'), ('a', 'b', 'd', 'e'), ('a', 'c', 'd', 'e'),
('b', 'c', 'd', 'e')]
```

In [12]:

```
from itertools import combinations
print(list(combinations(['a','b','c', 'd', 'e'], 5)))
```

```
[('a', 'b', 'c', 'd', 'e')]
```

In [14]:

```
from itertools import combinations_with_replacement
print(list(combinations_with_replacement(['a','b','c', 'd', 'e'], 2)))
```

```
[('a', 'a'), ('a', 'b'), ('a', 'c'), ('a', 'd'), ('a', 'e'), ('b', 'b'), ('b', 'c'), ('b'
, 'd'), ('b', 'e'), ('c', 'c'), ('c', 'd'), ('c', 'e'), ('d', 'd'), ('d', 'e'), ('e', 'e'
)]
```

In [21]:

```
from itertools import product
print(list(product('ABCD', repeat = 2)))
```

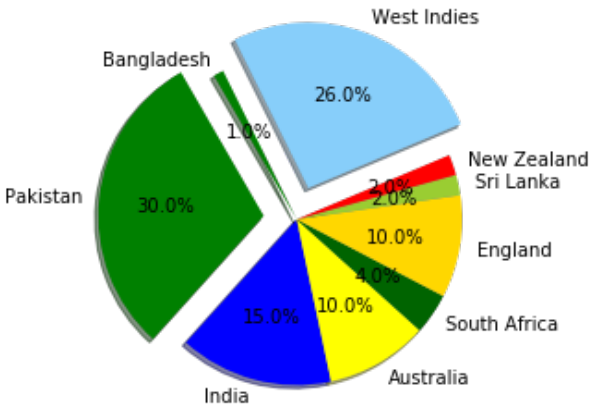
```
[('A', 'A'), ('A', 'B'), ('A', 'C'), ('A', 'D'), ('B', 'A'), ('B', 'B'), ('B', 'C'), ('B'
, 'D'), ('C', 'A'), ('C', 'B'), ('C', 'C'), ('C', 'D'), ('D', 'A'), ('D', 'B'), ('D', 'C'
), ('D', 'D')]
```

Predicted team performances for the T20 World Cup 2016

In [80]:

```
import matplotlib.pyplot as plt

# Data to plot
```

```
labels = ['Pakistan', 'India', 'Australia', 'South Africa', 'England',  'Sri Lanka', 'New Zealand', 'West Indies', 'Bangladesh']

sizes =  [0.3, 0.15, 0.1, 0.04, 0.10, 0.02, 0.02, 0.26, 0.01]

colors = ['green', 'blue', 'yellow', 'darkgreen', 'gold', 'yellowgreen', 'red', 'lightskyblue']

explode = (0.2, 0, 0, 0, 0, 0, 0, 0.2, 0)  # explode 1st slice and 2nd slice

# Plot
plt.pie(sizes, explode = explode, labels = labels, colors = colors, autopct = '%1.1f%%', shadow = True, startangle = 120)

plt.axis('equal')
plt.show()
```



Generate 5 real random numbers in the range 0 and 1(excluding 1).

In [21]:

```
import random
a =[]

for i in range(5):
        a.append(random.random())
print(a)
```

```
[0.11005091595922056, 0.48317080325764705, 0.8430215295683502, 0.22265371035249104, 0.21859876787841914]
```

Simulate the outcomes of 1000 biased coin tosses with prob[Head] = 0.3

In [83]:

```
import random
a =[]

for i in range(1000):
        a.append(random.random()<= 0.3)
#print(a)
print('Total number of heads = ', sum(a))
print('Probability of heads = ', sum (a) / 1000)
```

```
Total number of heads =  303
Probability of heads =  0.303
```

Print 10 integer random numbers in the range 0 - 1

In [84]:

```
import random
for i in range(10):
    print(random.randint(1, 6))
```

```
3
4
1
1
6
5
6
6
3
5
```

Binomial Probability Distribution

In [35]:

```
import matplotlib.pyplot as plt
from scipy.stats import binom

n = 5                           # Total number of coins

p = 0.5                         # Probability of head

x = [0, 1, 2, 3, 4, 5]          # Let x denotes number of heads

prob = binom.pmf(x, n, p)       # Compute probabilites corresponding to random variable x

print(prob)

sumOfProb = sum(prob)

print('Sum of probabilites is:', sumOfProb)


plt.xlabel('Total number of heads')    # Lable on x-axis i.e., Total number of heads
plt.ylabel('Probability')              # Lable on y-axis i.e., Probability of number of hea
ds (0, 1, 2, 3, 4, 5)
plt.bar(x,prob)
plt.show()
```

```
[ 0.03125  0.15625  0.3125   0.3125   0.15625  0.03125]
Sum of probabilites is: 1.0
```