Chap.-8 || Practise Book Que.

P.b.- 557

In [1]:
```
class Animal:
    count = 0 # class variable
    def __init__(self, a, l):
        self.arms = a
        self.legs = l
        Animal.count += 1
    def limbs(self):
        return (self.arms + self.legs)

spider = Animal(4,4)
spidlimbs = spider.limbs()
print('Limbs :',spidlimbs) # Limbs : 8

#----------------- Extra :--------------------
monkey = Animal(2,2)
print(monkey.arms) # 2

octopus = Animal(8,0)
print(octopus.legs) # 0

print(Animal.count) # 3
```

```
Limbs : 8
2
0
3
```

P.b.- 568

In [35]:

```
# ['apple', 'banana', 'find', 'dictionary', 'set', 'tuple', 'list', 'malayalam', 'nayan', 'grind', 'apricot']

class wordPlay():
    def __init__(self,l):
        self.l = l

    def words_with_length(self,s):
        nl = []
        for i in self.l:
            if(len(i) == s):
                nl.append(i)
        print(f"Length {s} : {nl}") # ['apple', 'tuple', 'nayan', 'grind']

    def starts_with(self,m):
        nl = []
        for i in self.l:
            if(i[0] == m):
                nl.append(i)
        print(f"Starts with {m} : {nl}")

    def ends_with(self,e):
        nl = []
        for i in self.l:
            if(i[-1] == e):
                nl.append(i)
        print(f"Ends with {e} : {nl}")

    def palindromes(self):
        nl = []
        for i in self.l:
            if(i == i[::-1]):
                nl.append(i)
        print(f"Palindrome : {nl}")

    def only(self,m):
        nl = []
        s1 = set(m)
        for word in self.l:
            s2 = set(word)
            if s1 == s2:
                nl.append(word)

        print(f"only '{m}' : {nl}")

    def avoids(self, a):
        nl = []
        for i in self.l:
            f = True
            for c in a:
                if(c in i):
                    f = False
                    break
            if f:
                nl.append(i)
        print(f"avoids ('{a}') : {nl}")

f = open('pb_568.txt')
l = f.read().split()
print(l)

ob1 = wordPlay(l)
ob1.words_with_length(5)
ob1.starts_with('a')
ob1.ends_with('d')
ob1.palindromes()
ob1.only('bna')
ob1.avoids('amkd')

# Output :
# ['apple', 'banana', 'find', 'dictionary', 'set', 'tuple', 'list', 'malayalam', 'nayan', 'grind', 'apricot']
# Length 5 : ['apple', 'tuple', 'nayan', 'grind']
# Starts with a : ['apple', 'apricot']
# Ends with d : ['find', 'grind']
# Palindrome : ['malayalam', 'nayan']
# only 'bna' : ['banana']
# avoids ('amkd') : ['set', 'tuple', 'list']
```

```
['apple', 'banana', 'find', 'dictionary', 'set', 'tuple', 'list', 'malayalam', 'nayan', 'grind', 'apricot']
Length 5 : ['apple', 'tuple', 'nayan', 'grind']
Starts with a : ['apple', 'apricot']
Ends with d : ['find', 'grind']
Palindrome : ['malayalam', 'nayan']
only 'bna' : ['banana']
avoids ('amkd') : ['set', 'tuple', 'list']
```

In [37]:
```python
# List Comprehension

x for x in self.l if len(x)== ln
```

P.b.- 572

In [ ]:
```python
class SQ:
    def __init__(self, initial_list=None):
        if initial_list is None:
            self.data = []
        else:
            self.data = initial_list

    def shift(self):
        if not self.data:
            raise IndexError("shift from empty list")
        return self.data.pop(0)

    def unshift(self, element):
        self.data.insert(0, element)

    def push(self, element):
        self.data.append(element)

    def pop(self):
        if not self.data:
            raise IndexError("pop from empty list")
        return self.data.pop()

    def remove(self):
        if not self.data:
            raise IndexError("remove from empty list")
        max_element = max(self.data)
        self.data.remove(max_element)
        return max_element

# Example usage:
sq = SQ([1, 2, 3, 4, 5])
print(sq.shift())  # Output: 1
sq.unshift(0)
print(sq.data)     # Output: [0, 2, 3, 4, 5]
sq.push(6)
print(sq.data)     # Output: [0, 2, 3, 4, 5, 6]
print(sq.pop())    # Output: 6
print(sq.data)     # Output: [0, 2, 3, 4, 5]
print(sq.remove()) # Output: 5
print(sq.data)     # Output: [0, 2, 3, 4]
```