In [1]:

```
###Exercise 1: Below and Above Average

import random

#Initializing list to use later
ls = []
upper = []
average = []
lower = []

#Taking input from the user
N = int(input("Please enter the count of random numbers to generate: "))
#For loop to generate N random numbers and appending them in list
for i in range(N):
    a = random.randint(1,10)
    ls.append(a)

#Computing average10
avg = sum(ls)/len(ls)

#For loop and if else condition, To find the values that are above average value
for j in range(len(ls)):
    if ls[j] > avg:
        upper.append(ls[j])
    elif ls[j] < avg:
        lower.append(ls[j])
    else:
        average.append(ls[j])

#Displaying above average, average(if any), and below average values
print("Above average values are:", upper)

if len(average) > 0:
    print("Average values are:", average)
print("Average:", avg)
print("Below average values are:", lower)
```

```
Above average values are: [7, 6, 9, 6]
Average: 4.3
Below average values are: [2, 2, 3, 3, 1, 4]
```

In [4]:

```
###Exercise 2: Random Lottery Numbers

from random import randrange

MIN_DIGIT = 1
MAX_DIGIT = 49
LOT_DIGIT = 6

#storing lottery ticket number
ticket_digit = []

#generate 6 digit distinct lottery numbers
for i in range(LOT_DIGIT):
    #generate number which is not in lottery ticket yet
    rand = randrange(MIN_DIGIT, MAX_DIGIT + 1)
    while rand in ticket_digit:
```

```
        rand = randrange(MIN_DIGIT, MAX_DIGIT + 1)

    #adding the distinct number to digit
    ticket_digit.append(rand)

#sorting and displaying in ascending order
ticket_digit.sort()
print("Your lottery numbers are: ", end="")
for n in ticket_digit:
    print (n, end=" ")
print()
```

```
Your lottery numbers are: 1 7 14 31 39 47
```

In [5]:

```
###Exercise 3:Remove Outliers

def removeout(data, num_out):
    #create new copy
    orgval = sorted(data)

    #remove largest value
    for i in range (num_out):
        orgval.pop()
    #remove smallest value
    for i in range (num_out):
        orgval.pop(0)

    return orgval
# read data from user and remove largest and smallest value
def main():
    #read values from user, blank line to stop reading
    vals = []
    s = input("Enter a value (enter without a value to stop): ")
    while s != "":
        num = float(s)
        vals.append(num)
        s = input("Enter a value (enter without a value to stop): ")
    #display result or error message
    if len(vals) < 4:
      print("You need to enter at least 4 values")
    else:
      print("Data without outliers: ", removeout(vals, 2))
      print("Original data: ", vals)
main()
```

```
Data without outliers:  [3.0, 4.0]
Original data:  [5.0, 1.0, 6.0, 2.0, 4.0, 3.0]
```

In [7]:

```
### execercise 04 -- modify and use this

# assumption is that both an empty list and a list with one item are sorted
def is_sorted(list_of_numbers):
    if sorted(list_of_numbers) == list_of_numbers:
        return True
    #Ok, not sorted ascending, lets check descending
    elif sorted(list_of_numbers, key=int, reverse=True) == list_of_numbers:
        return True
    #At this point we know it is not sorted
```

```
    return False

def main():
    list_of_numbers = []
    #We use this boolean to indicate that we are not done
    done = False
    while not done:
        number = int(input("Enter a number. 0 to exit: "))
        if number != 0:
            list_of_numbers.append(number)
        else:
            done = True
    print(list_of_numbers)

    if is_sorted(list_of_numbers)==True:
        print("This list is sorted.")
    else:
        print("This list is NOT sorted.")


if __name__ == '__main__':
    main()
```

```
[1, 2, 3, 4, 5]
This list is sorted.
```

In [8]:

```
###Exercise 5: Reverse Lookup

def reverseLookup(dictionary, search):
    keys = list(dictionary.keys()) # get list of keys in the dictionary
    values = list(dictionary.values()) # get list of keys in the dictionary
    mappedKeys = [] # make empty list
    for i in range(len(values)):
        # tranverse through values of the dictionary, if
        # any value matches with search value, then
        # append that key to mappedKeys list.
        if search == values[i]:
            mappedKeys.append(keys[i])
    # finally return mappedKeys
    return mappedKeys

def main():
    dictionary1 = {0:'A',1:'B',2:'B',3:'A',4:'B',5:'A'}
    dictionary2 = {'A':1,'B':2,'C':3}
    print(reverseLookup(dictionary1, 'A'))
    # multiple keys :- 'A' is mapped to keys 0,3,5
    print(reverseLookup(dictionary2, 3))
    # single key :- 3 is mapped to key 'C'
    print(reverseLookup(dictionary2, 4))
    # no key :- 4 is not mapped to any value in the dictionary

main()
```

```
[0, 3, 5]
['C']
[]
```

In [9]:

```
###Exercise 6:Two Dice Simulation
```

```
from random import randrange

num_rolls = 1000
roll_max = 6

def dice():
    # simulating dice
    dc1 = randrange(1, roll_max + 1)
    dc2 = randrange(1, roll_max + 1)

    return dc1 + dc2

def main():
    # expected proportion of outcome of rolling two dice
    expected = {2: 1/36, 3: 2/36, 4: 3/36, 5: 4/26, 6: 5/36, 7: 6/36, 8: 5/36, 9
    # sum of outcome by rolling two dice
    counts = {2: 0, 3: 0, 4: 0, 5: 0, 6: 0, 7: 0, 8: 0, 9: 0, 10: 0, 11: 0, 12:

    for i in range(num_rolls):
      d=dice()
      counts[d] = counts[d] + 1

    print("Total     Simulated      Expected")
    print("           Percent        Percent")
    for i in sorted(counts.keys()):
        print("%5d %11.2f  %8.2f" % \
             (i, counts[i] / num_rolls *100, expected[i] * 100))

main()
```

```
Total     Simulated      Expected
           Percent        Percent
    2        2.60      2.78
    3        4.90      5.56
    4        7.30      8.33
    5       10.40     15.38
    6       14.90     13.89
    7       17.60     16.67
    8       15.00     13.89
    9       11.00     11.11
   10        7.90      8.33
   11        5.80      5.56
   12        2.60      2.78
```

In [10]:

```
### Exercise 7:Write Out Numbers in English

# creating dictionaries
ones = ('Zero', 'One', 'Two', 'Three', 'Four', 'Five', 'Six', 'Seven', 'Eight',

twos = ('Ten', 'Eleven', 'Twelve', 'Thirteen', 'Fourteen', 'Fifteen', 'Sixteen',

tens = ('Twenty', 'Thirty', 'Forty', 'Fifty', 'Sixty', 'Seventy', 'Eighty', 'Nin

def process(number, index):

    if number=='0':
        return 'Zero'

    length = len(number)
```

```
    if(length > 3):
        return False

    number = number.zfill(3)
    words = ''

    hdigit = int(number[0])
    tdigit = int(number[1])
    odigit = int(number[2])

    words += '' if number[0] == '0' else ones[hdigit]
    words += ' Hundred ' if not words == '' else ''

    if(tdigit > 1):
        words += tens[tdigit - 2]
        words += ' '
        words += ones[odigit]

    elif(tdigit == 1):
        words += twos[(int(tdigit + odigit) % 10) - 1]

    elif(tdigit == 0):
        words += ones[odigit]

    if(words.endswith('Zero')):
        words = words[:-len('Zero')]
    else:
        words += ' '


    return words;

def getWords(number):
    length = len(str(number))

    if length>3:
        return 'This program supports upto 3 digit numbers.'

    count = length // 3 if length % 3 == 0 else length // 3 + 1
    copy = count
    words = []

    for i in range(length - 1, -1, -3):
        words.append(process(str(number)[0 if i - 2 < 0 else i - 2 : i + 1], cop
        count -= 1;

    final_words = ''
    for s in reversed(words):
        temp = s + ' '
        final_words += temp

    return final_words


# Taking input from user
number = int(input('Enter number between 0 and 999: '))
print('%d: %s' %(number, getWords(number)))
```

123: One Hundred Twenty Three

In [ ]: