**LIST OF PROGRAMS**

1. Compute the GCD of two numbers.

2. Find the square root of a number (Newton’s method)

3. Exponentiation (power of a number)

4. Linear search and Binary search

5. First n prime numbers

6. Find the maximum of a list of numbers

7. Selection sort, Insertion sort

8. Removing all the duplicate elements in a list

9. Merge sort, Quick sort

10. Multiply matrices

11. Programs that take command line arguments (word count)

12. Find the most frequent words in a text read from a file

1.

from math import \*

def euclid\_algo(x, y, verbose=True):

if x < y: # We want x >= y

return euclid\_algo(y, x, verbose)

print()

while y != 0:

if verbose: print('%s = %s \* %s + %s' % (x, floor(x/y), y, x % y))

(x, y) = (y, x % y)

if verbose: print('gcd is %s' % x)

return x

euclid\_algo(150, 304)

euclid\_algo(1000, 10)

euclid\_algo(150, 9)

Sample Output:

304 = 2 \* 150 + 4

150 = 37 \* 4 + 2

4 = 2 \* 2 + 0

gcd is 2

1000 = 100 \* 10 + 0

gcd is 10

150 = 16 \* 9 + 6

9 = 1 \* 6 + 3

6 = 2 \* 3 + 0

gcd is 3

2.

def check(x, guess):

return (abs(guess\*guess - x) < 0.001)

def newton(x, guess):

while not check(x, guess):

guess = (guess + (x/guess)) / 2.0

return guess

newton(16, 1)

def BabylonianAlgorithm(number):

if(number == 0):

return 0;

g = number/2.0;

g2 = g + 1;

while(g != g2):

n = number/ g;

g2 = g;

g = (g + n)/2;

return g;

print('The Square root of 0.3 =', BabylonianAlgorithm(0.3));

 Newton’s Method

Newton’s Method finds the root of the equation a2−n=0a2−n=0 using the iteration

xn+1=xn−f(xn)f′(xn)xn+1=xn−f(xn)f′(xn)

static double squareRootNewton(double n){

double precision = 0.001;

double guess = n/2;

double val = guess\*guess-n;

**while**(Math.abs(val) > precision){

guess = guess - val/(2\*guess); *// derivative is 2\*a*

val = guess\*guess - n;

}

**return** guess;

}

3.

from decimal import \*

x = Decimal('1.44')

print("Square root of ",x, " is :", x.sqrt())

print("exponential of ",x, " is :", x.exp())

Sample Output:

Square root of 1.44 is : 1.2

exponential of 1.44 is : 4.220695816996552825673328929

4.

**Linear search**

def Sequential\_Search(dlist, item):

pos = 0

found = False

while pos < len(dlist) and not found:

if dlist[pos] == item:

found = True

else:

pos = pos + 1

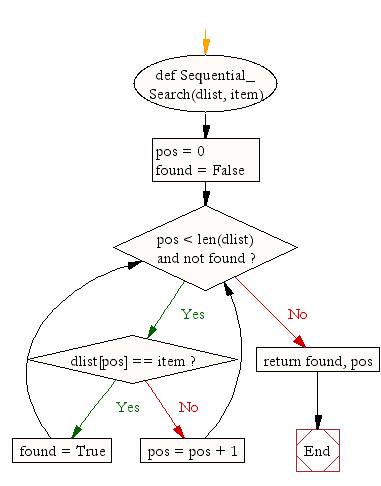
return found, pos

print(Sequential\_Search([11,23,58,31,56,77,43,12,65,19],31))

Sample Output:

(True, 3)

**Flowchart:**



**Binary search**

def binary\_search(item\_list,item):

first = 0

last = len(item\_list)-1

found = False

while( first<=last and not found):

mid = (first + last)//2

if item\_list[mid] == item :

found = True

else:

if item < item\_list[mid]:

last = mid - 1

else:

first = mid + 1

return found

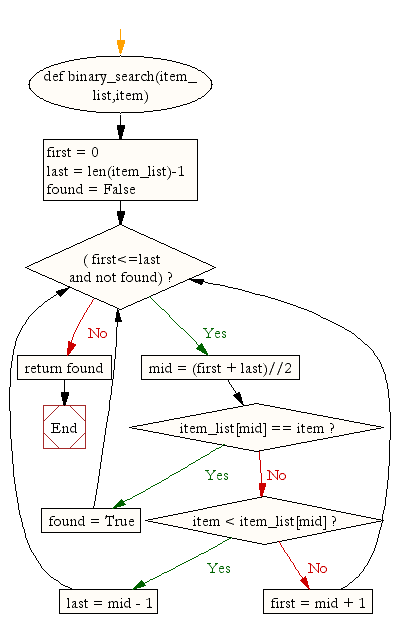
print(binary\_search([1,2,3,5,8], 6))

print(binary\_search([1,2,3,5,8], 5))

Sample Output:

False

True



5. **FINDING THE FIRST N PRIME NUMBERS**

|  |
| --- |
| def checkprime(n):      d=2      while d<=(n/2):          if n%d==0:              return 0          d+=1      return 1      def primeno(term):      numbers=[]      i=2      while len(numbers)<term:          if checkprime(i)==1:              numbers.append(i)          i+=1      return numbers |

**OUTPUT:**  
  
>>> primeno(25)  
[2, 3, 5, 7, 11, 13, 17, 19, 23, 29, 31, 37, 41, 43, 47, 53, 59, 61, 67, 71, 73, 79, 83, 89, 97]

6.

Write a Python program to get the largest number from a list.

def max\_num\_in\_list( list ):

max = list[ 0 ]

for a in list:

if a > max:

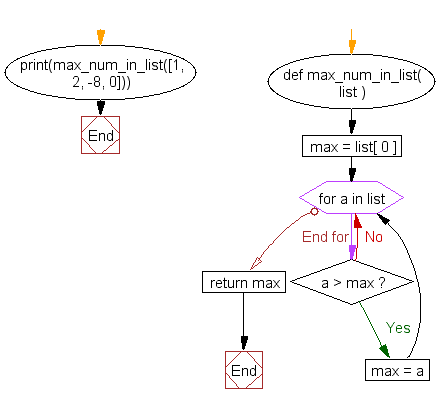
max = a

return max

print(max\_num\_in\_list([1, 2, -8, 0]))

Sample Output:

2



7.

Write a Python program to sort a list of elements using the selection sort algorithm.

def selectionSort(nlist):

for fillslot in range(len(nlist)-1,0,-1):

maxpos=0

for location in range(1,fillslot+1):

if nlist[location]>nlist[maxpos]:

maxpos = location

temp = nlist[fillslot]

nlist[fillslot] = nlist[maxpos]

nlist[maxpos] = temp

nlist = [14,46,43,27,57,41,45,21,70]

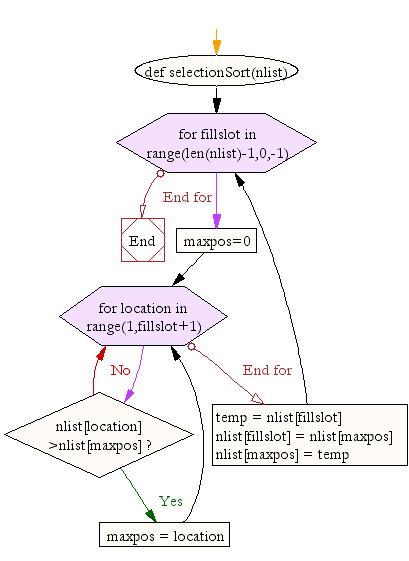
selectionSort(nlist)

print(nlist)

Sample Output:

[14, 21, 27, 41, 43, 45, 46, 57, 70]

**Flowchart:**



Write a Python program to sort a list of elements using the insertion sort algorithm.

def insertionSort(nlist):

for index in range(1,len(nlist)):

currentvalue = nlist[index]

position = index

while position>0 and nlist[position-1]>currentvalue:

nlist[position]=nlist[position-1]

position = position-1

nlist[position]=currentvalue

nlist = [14,46,43,27,57,41,45,21,70]

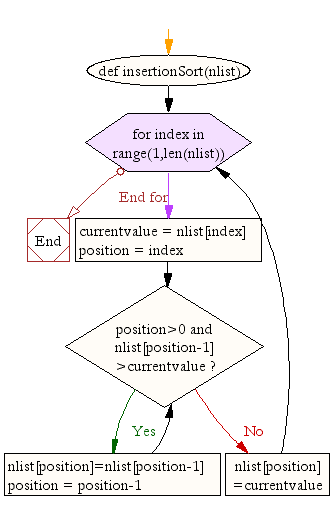
insertionSort(nlist)

print(nlist)

Sample Output:

[14, 21, 27, 41, 43, 45, 46, 57, 70]

**Flowchart :**



8.

Write a Python program to remove duplicates from a list.

a = [10,20,30,20,10,50,60,40,80,50,40]

dup\_items = set()

uniq\_items = []

for x in a:

if x not in dup\_items:

uniq\_items.append(x)

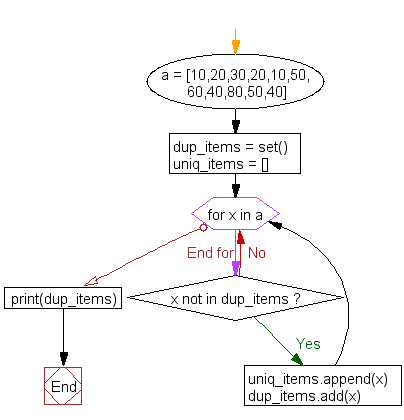
dup\_items.add(x)

print(dup\_items)

Sample Output:

{40, 10, 80, 50, 20, 60, 30}

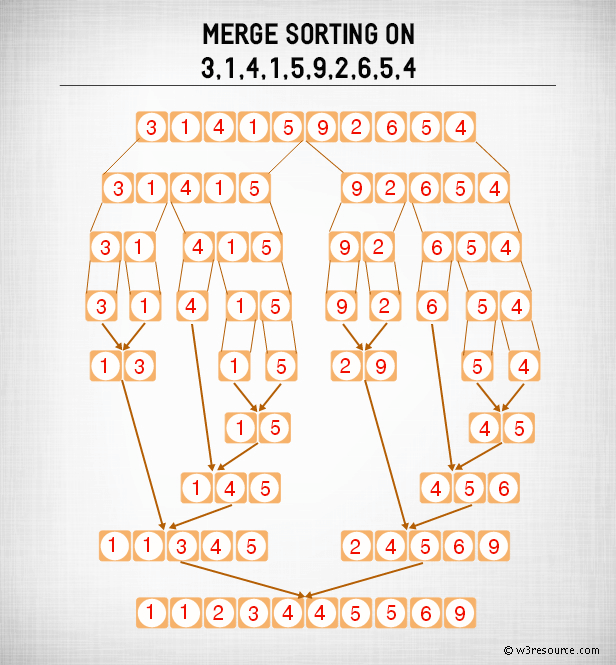
**Flowchart:**



9.

Merge sort, Quick sort

Write a Python program to sort a list of elements using the merge sort algorithm.



def mergeSort(nlist):

print("Splitting ",nlist)

if len(nlist)>1:

mid = len(nlist)//2

lefthalf = nlist[:mid]

righthalf = nlist[mid:]

mergeSort(lefthalf)

mergeSort(righthalf)

i=j=k=0

while i < len(lefthalf) and j < len(righthalf):

if lefthalf[i] < righthalf[j]:

nlist[k]=lefthalf[i]

i=i+1

else:

nlist[k]=righthalf[j]

j=j+1

k=k+1

while i < len(lefthalf):

nlist[k]=lefthalf[i]

i=i+1

k=k+1

while j < len(righthalf):

nlist[k]=righthalf[j]

j=j+1

k=k+1

print("Merging ",nlist)

nlist = [14,46,43,27,57,41,45,21,70]

mergeSort(nlist)

print(nlist)

ample Output:

Splitting [14, 46, 43, 27, 57, 41, 45, 21, 70]

Splitting [14, 46, 43, 27]

Splitting [14, 46]

Splitting [14]

Merging [14]

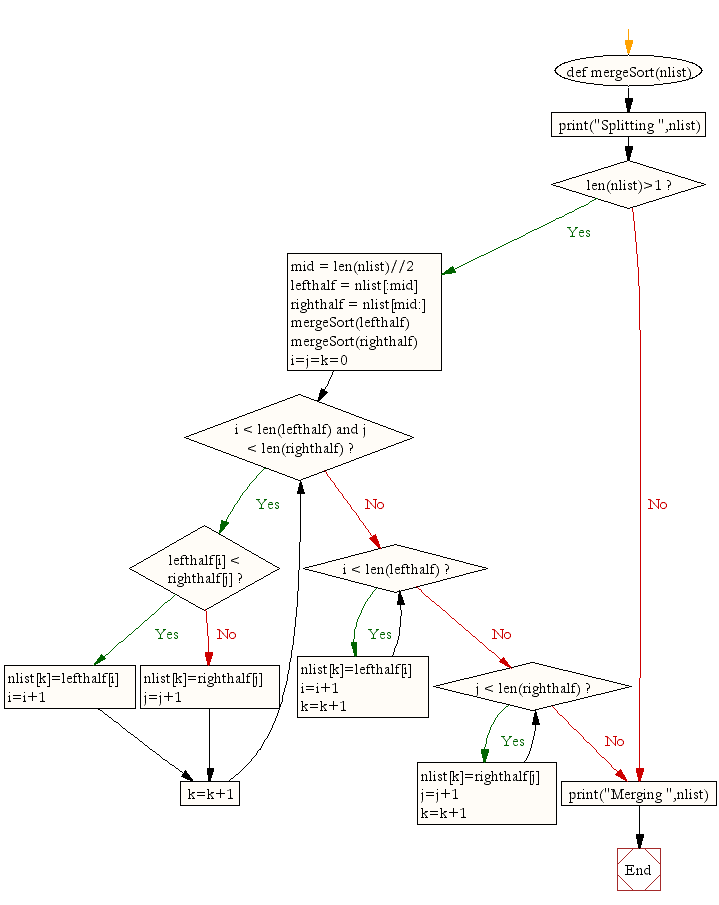
Splitting [46]

-------

Merging [14, 21, 27, 41, 43, 45, 46, 57, 70]

[14, 21, 27, 41, 43, 45, 46, 57, 70]

**Flowchart:**



Write a Python program to sort a list of elements using the quick sort algorithm.

def quickSort(data\_list):

quickSortHlp(data\_list,0,len(data\_list)-1)

def quickSortHlp(data\_list,first,last):

if first < last:

splitpoint = partition(data\_list,first,last)

quickSortHlp(data\_list,first,splitpoint-1)

quickSortHlp(data\_list,splitpoint+1,last)

def partition(data\_list,first,last):

pivotvalue = data\_list[first]

leftmark = first+1

rightmark = last

done = False

while not done:

while leftmark <= rightmark and data\_list[leftmark] <= pivotvalue:

leftmark = leftmark + 1

while data\_list[rightmark] >= pivotvalue and rightmark >= leftmark:

rightmark = rightmark -1

if rightmark < leftmark:

done = True

else:

temp = data\_list[leftmark]

data\_list[leftmark] = data\_list[rightmark]

data\_list[rightmark] = temp

temp = data\_list[first]

data\_list[first] = data\_list[rightmark]

data\_list[rightmark] = temp

return rightmark

data\_list = [54,26,93,17,77,31,44,55,20]

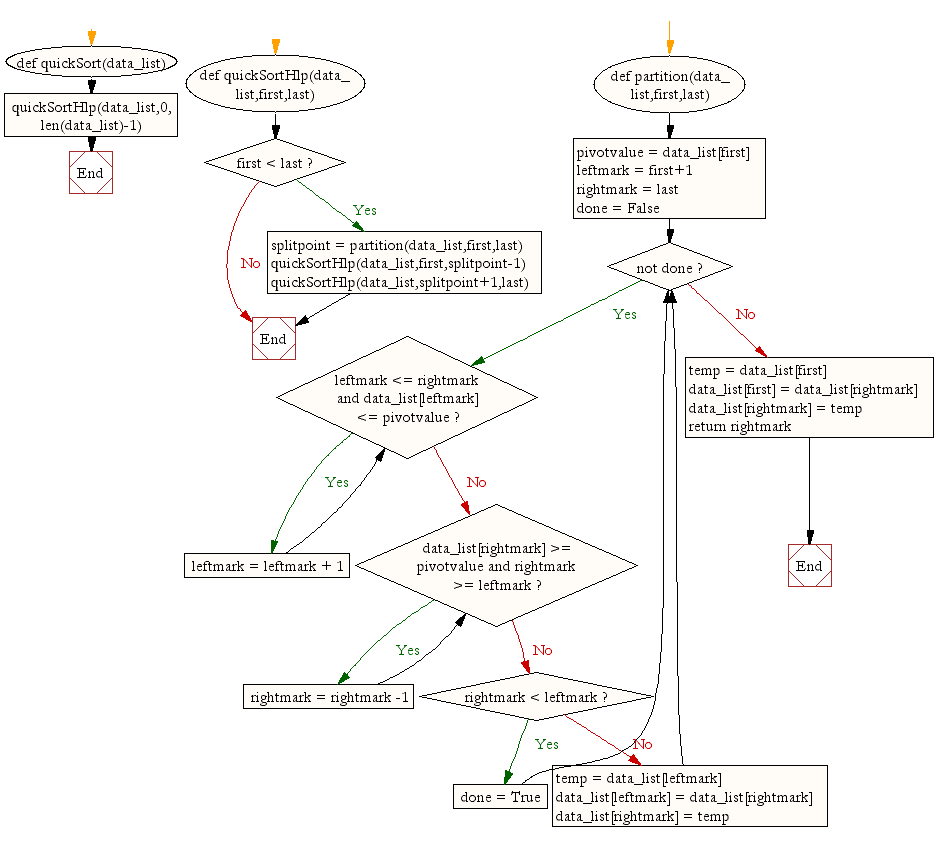
quickSort(data\_list)

print(data\_list)

Sample Output:

[17, 20, 26, 31, 44, 54, 55, 77, 93]

**Flowchart:**



10.

Write a Python program to multiply a 5x3 matrix by a 3x2 matrix and create a real matrix product.

import numpy as np

x = np.random.random((5,3))

print("First array:")

print(x)

y = np.random.random((3,2))

print("Second array:")

print(y)

z = np.dot(x, y)

print("Dot product of two arrays:")

print(z)

Sample Output:

First array:

[[ 0.44349753 0.81043761 0.00771825]

[ 0.64004088 0.86774612 0.19944667]

[ 0.61520091 0.24796788 0.93798297]

[ 0.22156999 0.61318856 0.82348994]

[ 0.91324026 0.13411297 0.00622696]]

Second array:

[[ 0.73873542 0.06448186]

[ 0.90974982 0.06409165]

[ 0.22321268 0.39147412]]

Dot product of two arrays:

[[ 1.06664562 0.08356133]

[ 1.30677176 0.17496452]

[ 0.88942914 0.42275803]

[ 0.90534318 0.37596252]

[ 0.79804212 0.06992065]]

11. Programs that take command line arguments (word count)

def string\_length(str1):

count = 0

for char in str1:

count += 1

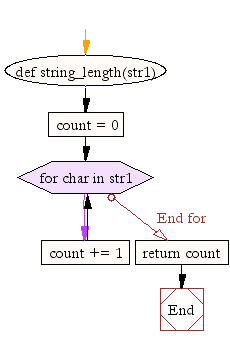
return count

**Output:**

>>> print(string\_length('Ganesh'))

6

**Flowchart:**



12.

Find the most frequent words in a text read from a file

rom collections import Counter

def word\_count(fname):

with open(fname) as f:

return Counter(f.read().split())

**Output**

**Sample.txt**

Hello how are you you

hello how are you

Hi how are you

>>> print("Number of words in the file :",word\_count("test.txt"))

Number of words in the file : Counter({'you': 4, 'how': 3, 'are': 3, 'Hello': 1, 'hello': 1, 'Hi': 1})