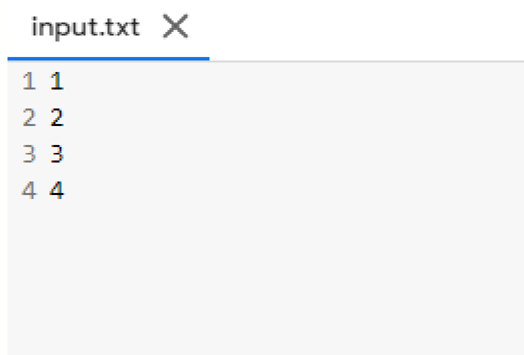
Problem #01:
Let A be the set {1, 2, 3, 4}. Write a program to find the ordered pairs are in the relation R1 $= \{(a, b) \mid a \text{ divides } b\}$ R2 $= \{(a, b) \mid a \leq b\}$



//input.txt
 1
 2
 3
 4

```
from itertools import product

with open("/content/sample_data/input.txt", "r", encoding="utf-8") as g:
 S = list(map(int, g.readlines()))
print("S= "+str(S))

res=[(i,j) for i,j in product(S,repeat=2) if i%j==0 or j%i==0]
res2=[(i,j) for i,j in product(S,repeat=2) if i<=j]
# printing result
print ("The pair list is for a/b : " + str(res))
print ("The pair list is for a<=b : " + str(res2))
```

Output:

```
S= [1, 2, 3, 4]
The pair list is for a/b : [(1, 1), (1, 2), (1, 3), (1, 4), (2, 1), (2, 2), (2, 4), (3, 1), (3, 3), (4, 1), (4, 2), (4, 4)]
The pair list is for a<=b : [(1, 1), (1, 2), (1, 3), (1, 4), (2, 2), (2, 3), (2, 4), (3, 3), (3, 4), (4, 4)]
```

Problem #02:
Suppose that $A = \{1, 2, 3\}$ and $B = \{1, 2\}$. Let R be the relation from A to B containing (a, b) if $a \in A$, $b \in B$ and $a > b$. Write a program to find the relation R and also represent this relation in matrix form.

```
import numpy as np
with open("/content/sample_data/input.txt", "r", encoding="utf-8") as g:
 list1 = list(map(int, g.readlines()))
with open("/content/sample_data/input.txt", "r", encoding="utf-8") as g:
 list2 = list(map(int, g.readlines()))

# using list comprehension
output = [(a, b) for a in list1
```

```
          for b in list2 if a > b]
output2 = [1 if a>b else 0 for a in list1
          for b in list2]

data = np.array(output2).reshape(4,4)

print(output)
print(data)
```

Output:

```
[(2, 1), (3, 1), (3, 2), (4, 1), (4, 2), (4, 3)]
[[0 0 0 0]
 [1 0 0 0]
 [1 1 0 0]
 [1 1 1 0]]
```

Problem #03: Write a program for the solution of graph coloring problem by Welch-Powell's algorithm.

```
def color_nodes(graph):
    color_map = {}
    # Consider nodes in descending degree
    for node in sorted(graph, key=lambda x: len(graph[x]), reverse=True):
        neighbor_colors = set(color_map.get(neigh) for neigh in graph[node])
        color_map[node] = next(
            color for color in range(len(graph)) if color not in neighbor_colors
        )
    return color_map
#Adjacent list
graph={'a':list('bcd'),'b': list('ac'),'c': list('abdef'),'d': list('ace'),'e': list('cdf'),'f': list('ce')}
print(color_nodes(graph))
```

Output:

```
{'c': 0, 'a': 1, 'd': 2, 'e': 1, 'b': 2, 'f': 2}
```

Problem #04: Write a program to find shortest path by Warshall's algorithm.

```
INF = 1000000000
def floyd_warshall(vertex, adjacency_matrix):
  # calculating all pair shortest path
  for k in range(0, vertex):
    for i in range(0, vertex):
        for j in range(0, vertex):
          # relax the distance from i to j by allowing vertex k as intermedi
ate vertex
```

```
        # consider which one is better, going through vertex k or the prev
ious value
        adjacency_matrix[i][j] = min(adjacency_matrix[i][j], adjacency_mat
rix[i][k] + adjacency_matrix[k][j])
  # pretty print the graph
  # o/d means the leftmost row is the origin vertex
  # and the topmost column as destination vertex
  print("o/d", end='')
  for i in range(0, vertex):
    print("\t{:d}".format(i+1), end='')
  print();
  for i in range(0, vertex):
    print("{:d}".format(i+1), end='')
    for j in range(0,vertex):
      print("\t{:d}".format(adjacency_matrix[i][j]), end='')
    print();
"""
input is given as adjacency matrix,
input represents this undirected graph
 A--1--B
 |    /
 3   /
 |  1
 | /
 C--2--D
should set infinite value for each pair of vertex that has no edge
 """
adjacency_matrix = [
          [  0,    5, INF, 10],
          [  INF,    0, 3, INF],
          [  INF,    INF, 0,    1],
          [INF, INF, INF,    0]
          ]
floyd_warshall(4, adjacency_matrix);
```

Output:

```
o/d   1      2      3      4
1     0      5      8      9
2     1000000000  0      3      4
3     1000000000  1000000000  0      1
4     1000000000  1000000000  1000000000  0
```

Source: https://iq.opengenus.org/floyd-warshall-algorithm-shortest-path-between-all-pair-of-nodes/

Problem #05: Suppose that the relations R1 and R2 on a set A are represented by the matrices

$M_{R1} = \begin{bmatrix} 1 & 1 & 1 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$ and $M_{R2} = \begin{bmatrix} 1 & 0 & 1 \\ 0 & 1 & 1 \\ 1 & 1 & 0 \end{bmatrix}$. Write a program to find the $M_{R1 \cup R2}$ and $M_{R1 \oplus R2}$.

```
def matrix_intersection(mat1, mat2):
    rows = len(mat1)
    cols = len(mat1[0])
    print('Rows=', rows, 'Cols=', cols)
    mat_inter = []
    for i in range(len(mat1)):
```

```
        mat_inter.append([mat1[i][j] and mat2[i][j] for j in
range(len(mat1[0]))])

    return mat_inter


def matrix_union(mat1, mat2):
    mat_union = []
    for i in range(len(mat1)):
        mat_union.append([mat1[i][j] or mat2[i][j] for j in
range(len(mat1[0]))])

    return mat_union



matrix1 = [[1, 0, 1],
           [1, 0, 0],
           [0, 1, 1]]
matrix2 = [[1, 0, 1],
           [0, 1, 1],
           [1, 0, 1]]

# print('Matrix Intersection', mat_inter)
print('First Matrix=', matrix1)
print('Second Matrix=', matrix2)

mi = matrix_intersection(matrix1, matrix2)
print('Matrix Intersection', mi)

mu = matrix_union(matrix1, matrix2)
print('Matrix Union', mu)
v = ['p', 'q', 'r']

r1 = []
for i in range(len(mi)):
    for j in range(len(mi[0])):
        if mi[i][j] == 1:
            r1.append((v[i], v[j]))

print(r1)

r2 = []
for i in range(len(mu)):
    for j in range(len(mu[0])):
        if mu[i][j] == 1:
            r2.append((v[i], v[j]))

print(r2)
```

Output:

First Matrix= [[1, 0, 1], [1, 0, 0], [0, 1, 1]]
Second Matrix= [[1, 0, 1], [0, 1, 1], [1, 0, 1]]
Rows= 3 Cols= 3
Matrix Intersection [[1, 0, 1], [0, 0, 0], [0, 0, 1]]
Matrix Union [[1, 0, 1], [1, 1, 1], [1, 1, 1]]
[('p', 'p'), ('p', 'r'), ('r', 'r')]
[('p', 'p'), ('p', 'r'), ('q', 'p'), ('q', 'q'), ('q', 'r'), ('r', 'p'), ('r', 'q'), ('r', 'r')]

Problem # 06: The following table gives the population of a town during the last six censuses. Write a Python program to find the population in the year of 1946 using Newton-Gregory forward interpolation formula.

Year:	1911	1921	1931	1941	1951	1961
Population:	12	15	20	27	39	52

Source Code:

```
# calculating u mentioned in the formula
def u_cal(u, n):

  temp = u;
  for i in range(1, n):
    temp = temp * (u - i);
  return temp;

# calculating factorial of given number n
def fact(n):
  f = 1;
  for i in range(2, n + 1):
    f *= i;
  return f;

# Driver Code

# Number of values given
n = 6;
x = [ 1911, 1921, 1931, 1941,1951,1961 ];

# y[][] is used for difference table
# with y[][0] used for input
y = [[0 for i in range(n)]
    for j in range(n)];
y[0][0] = 12;
y[1][0] = 15;
y[2][0] = 20;
y[3][0] = 27;
y[4][0] = 39;
y[5][0] = 52;

# Calculating the forward difference
# table
for i in range(1, n):
  for j in range(n - i):
    y[j][i] = y[j + 1][i - 1] - y[j][i - 1];

# Displaying the forward difference table
```

```python
for i in range(n):
  print(x[i], end = "\t");
  for j in range(n - i):
    print(y[i][j], end = "\t");
  print("");

# Value to interpolate at
value = 1946;

# initializing u and sum
sum = y[0][0];
u = (value - x[0]) / (x[1] - x[0]);
for i in range(1,n):
  sum = sum + (u_cal(u, i) * y[0][i]) / fact(i);

print("\nValue at", value,
  "is", round(sum, 6));
```

Output:

```
1911  12    3     2     0     3     -10
1921  15    5     2     3     -7
1931  20    7     5     -4
1941  27    12    1
1951  39    13
1961  52

Value at 1946 is 32.34375
```

Source: https://www.geeksforgeeks.org/newton-forward-backward-interpolation/

Problem # 07: Write a Python program to find $f(7.5)$ form the following table using Newton-Gregory backward interpolation formula.

x:	1	2	3	4	5	6	7	8
f(x):	1	8	27	64	125	216	343	512

```python
import math

# read input value from file
file_name = input("Enter file name with extension: ")  # code and input file should be in same folder
f = open(file_name, "r")
data = f.read()
print(data)
data = data.split()
x, y = [], []
for i, j in zip(data[0::2], data[1::2]):
    x.append(float(i))
    y.append(float(j))
inp = float(input("Enter value of x for interpolation: "))

# calculation of table
table = [y]
```

```
for l in range(len(y) - 1):
    yn = []
    for i, k in zip(y[1::1], y[0::1]):
        yn.append(i - k)
    table.append(yn)
    y = yn

# print table
formated_table = [["x", "f(x)", "∇f(x)"]]
for i in range(2, len(table)):
    formated_table[0].append("∇^" + str(i) + "f(x)")
for i in range(len(x)):
    row = []
    for j in range(len(table) - i):
        row.append(str(round(table[j][i], 5)))
    row.insert(0, str(x[i]))
    formated_table.append(row)
for row in formated_table:
    print(" \t".join(row))

# calculation of r
r = (inp - x[-1]) / (x[1] - x[0])

# result calculation
r_component = 1
partial_result = 0
for i in range(1, len(table)):
    r_component = r_component * (r + i - 1)
    partial_result = partial_result + (table[i][-1] * r_component) /
math.factorial(i)

final_result = table[0][-1] + partial_result
print("f(" + str(inp) + ") = ", final_result)
```

Output:

```
Enter file name with extension: data3.txt
1 1
2 8
3 27
4 64
5 125
6 216
7 343
8 512
Enter value of x for interpolation: 7.5
x       f(x)    ∇f(x)   ∇^2f(x) ∇^3f(x) ∇^4f(x) ∇^5f(x) ∇^6f(x) ∇^7f(x)
1.0     1.0     7.0     12.0    6.0     0.0     0.0     0.0     0.0
2.0     8.0     19.0    18.0    6.0     0.0     0.0     0.0
3.0     27.0    37.0    24.0    6.0     0.0     0.0
4.0     64.0    61.0    30.0    6.0     0.0
5.0     125.0   91.0    36.0    6.0
6.0     216.0   127.0   42.0
7.0     343.0   169.0
8.0     512.0
f(7.5) =  421.875
```

Problem # 08: Write a Python program to find the value of $f(15)$ from the following table using Newton's divided difference formula.

x:	4	5	7	10	11	13
f(x):	48	100	294	900	1210	2028

Source Code:

```
def proterm(i, value, x):
    pro = 1
    for j in range(i):
        pro = pro * (value - x[j])
    return pro


# Function for calculating divided difference table
def dividedDiffTable(x, y, n):
    for i in range(1, n):
        for j in range(n - i):
            y[j][i] = ((y[j][i - 1] - y[j + 1][i - 1]) / (x[j] - x[i + j]))
    return y


# Function for applying Newton's divided difference formula
def applyFormula(value, x, y, n):
    sum = y[0][0]

    for i in range(1, n):
        sum = sum + (proterm(i, value, x) * y[0][i])

    return sum

# Function for displaying divided difference table
def printDiffTable(y, n):
    for i in range(n):
        print(x[i], end="\t\t")
        for j in range(n - i):
            print(y[i][j], end="\t\t")
        print("")

# Driver Code
# number of inputs given
n = 6
y = [[0 for i in range(n)] for j in range(n)]
x = [4, 5, 7, 10, 11, 13]
print(x)
# y[][] is used for divided difference
# table where y[][0] is used for input
# Data from example 3 page no 90 vasistha
y[0][0] = 48
y[1][0] = 100
y[2][0] = 294
y[3][0] = 900
y[4][0] = 1210
y[5][0] = 2028
print(y)
# calculating divided difference table
y = dividedDiffTable(x, y, n)
```

```
# displaying divided difference table
printDiffTable(y, n)

# value to be interpolated
value = 15
# printing the value
print("\nValue at", value, "is", round(applyFormula(value, x, y, n), 2))

# value to be interpolated
value = 8
# printing the value
print("\nValue at", value, "is", round(applyFormula(value, x, y, n), 2))
```

Output:

```
[4, 5, 7, 10, 11, 13]
[[48, 0, 0, 0, 0, 0], [100, 0, 0, 0, 0, 0], [294, 0, 0, 0, 0, 0], [900, 0,
0, 0, 0, 0], [1210, 0, 0, 0, 0, 0], [2028, 0, 0, 0, 0, 0]]
4           48          52.0        15.0        1.0         -0.0        -0.0

5           100         97.0        21.0        1.0         -0.0
7           294         202.0       27.0        1.0
10          900         310.0       33.0
11          1210        409.0
13          2028

Value at 15 is 3150.0

Value at 8 is 448.0
```

Source: https://www.geeksforgeeks.org/newtons-divided-difference-interpolation-formula/

Problem # 09: Write a Python program to find the value of y when $x = 10$ from the following table using Lagrange's interpolation formula.

x:	**5**	**6**	**9**	**11**
y:	**12**	**13**	**14**	**16**

Source Code:

```
class Data:
  def __init__(self, x, y):
    self.x = x
    self.y = y

def interpolate(f: list, xi: int, n: int) -> float:

  result = 0.0
  for i in range(n):
    term = f[i].y
    for j in range(n):
      if j != i:
        term = term * (xi - f[j].x) / (f[i].x - f[j].x)
    result += term
```

```
  return result
if __name__ == "__main__":

  f = [Data(5, 12), Data(6, 13), Data(9, 14), Data(11, 16)]
  print("Value of f(10) is :", interpolate(f, 10, len(f)))
```
Output:
```
Value of f(10) is : 14.666666666666666
```
Source: https://www.geeksforgeeks.org/lagranges-interpolation/

Problem # 10: Write a Python program to find a real root of the equation $x^3 - 2x - 5 = 0$ using bisection method.

Source Code:
```
# Python program for implementation  of Bisection Method for solving
equations
# An example function whose solution is determined using Bisection Method.
# The function is x^3 - 2x - 5 = 0

def func(x):
    return x * x * x - 2 * x - 5


# Prints root of func(x) with error of EPSILON
def bisection(a, b):
    if func(a) * func(b) >= 0:
        print("You have not assumed right a and b\n")
        return

    c = a
    while (b - a) >= 0.0001:

        # Find middle point
        c = (a + b) / 2

        # Check if middle point is root
        if (func(c) == 0.0):
            break

        # Decide the side to repeat the steps
        if (func(c) * func(a) < 0):
            b = c
        else:
            a = c

    print("The value of root is : ", "%.4f" % c)

# Driver code
# Initial values assumed
a = -1
b = 3
bisection(a, b)

# This code is contributed
# by Anant Agarwal.
```
Output:
```
The value of root is :  2.0950
```
Source: https://www.geeksforgeeks.org/program-for-bisection-method/

Problem # 11: Write a Python program to find a real root of the equation $x^3 - 2x - 5 = 0$ using false position method.

Source Code:

```
MAX_ITER = 1000000

def func( x ):
  return (x * x * x - 2 * x -5)

def regulaFalsi( a , b):
  if func(a) * func(b) >= 0:
    print("You have not assumed right a and b")
    return -1
  c = a
  for i in range(MAX_ITER):

    c = (a * func(b) - b * func(a))/ (func(b) - func(a))
    if func(c) == 0:
      break
    elif func(c) * func(a) < 0:
      b = c
    else:
      a = c
  print("The value of root is : " , '%.4f' %c)
a =-200
b = 300
regulaFalsi(a, b)
```

Output:

```
The value of root is :  2.0946
```

Source: https://www.geeksforgeeks.org/program-for-method-of-false-position/