NATIONAL INSTITUTE OF TECHNOLOGY, DURGAPUR

SOFTWARE ENGINEERING LABORATORY

ASSIGNMENT 1

Name – Shubhabrata Ghosh

Roll – 20CS8003

Regn. – 20U10008

Sec – X Sub- CSS751

Dept- CSE(2020-2024)

Question

Write a program to store and display the CFG for given program segment.

i) Find maximal set of independent paths of the above generated CFG.

ii) Find the region and cyclomatic complexity of the CFG.

Write a program to detect loops from the generated CFG

Code

from pycfg.pycfg import PyCFG,CFGNode, slurp

import argparse

import tkinter as tk

from PIL import ImageTk, Image

def detectLoopInGraph(adj:dict, hasVisited:dict, currentVisited:dict, x:str):

    flag=False

    for node in adj[x]:

        if hasVisited[node]==0:

            hasVisited[node]=1

            currentVisited[node]=1

            flag=detectLoopInGraph(adj,hasVisited,currentVisited,node)

            if flag:

                return True

            currentVisited[node]=0

        elif currentVisited[node]==1:

            return True

    return False

def DFS(adj:dict, n:str, hasVisited:dict, pathList:list):

    flag=False

    for edge in adj[n]:

        if hasVisited[edge] == 0:

            flag=True

            hasVisited[edge]=1

            pathList.append(n)

            DFS(adj,edge[1],hasVisited,pathList)

            pathList.pop()

            hasVisited[edge]=0

    if not flag:

        pathList.append(n)

        st.append(pathList.copy())

        pathList.pop()

if \_\_name\_\_ == '\_\_main\_\_':

    parser = argparse.ArgumentParser()

    parser.add\_argument('pythonfile', help ='The python file to be analyzed')

    args = parser.parse\_args()

    arcs = []

    cfg = PyCFG()

    cfg.gen\_cfg(slurp(args.pythonfile).strip())

    g = CFGNode.to\_graph(arcs)

    g.draw(args.pythonfile + '.png', prog ='dot')

    print("Nodes are as follows:")

    print(g.nodes())

    adje = dict()

    hasVisited = dict()

    for x in g.nodes():

        adje[str(x)] = g.out\_edges(x)

        for y in adje[str(x)]:

            hasVisited[y]=0

    pathList = list()

    st = list()

    DFS(adje,'0',hasVisited,pathList)

    numberOfIndPaths=len(st)

    if(numberOfIndPaths>=1):

        print("Number of independent paths:",numberOfIndPaths)

        print("The independent paths are as follows: ")

        for s in st:

            k=1

            for i in s:

                if(k<len(s)):

                    print(f'{i}->',end="")

                else:

                    print(i)

                k=k+1

    adj=dict()

    currentVisited = dict()

    for x in g.nodes():

        adj[str(x)]=g.out\_neighbors(x)

        for y in adj[str(x)]:

            hasVisited[y]=0

            currentVisited[y]=0

    if detectLoopInGraph(adj,hasVisited,currentVisited,'0'):

        print("Loop was detected in CFG")

    else:

        print("No loop was detected in CFG")

    nodes = g.number\_of\_nodes()  # no. of nodes.

    edges = g.number\_of\_edges()  # no. of Edges.

    complexity = edges - nodes + 2       # Cyclomatic complexity

    print("Nodes\t\t\t",str(nodes))

    print("Edges\t\t\t",str(edges))

    print("Cyclomatic Complexity\t",str(complexity))

    print("Bounded regions\t\t",str(complexity-1))

    # Draw using tkinter.

    root = tk.Tk()

    root.title("Control Flow Graph")

    img1 = Image.open(str(args.pythonfile) + ".png") # PIL solution

    img1 = img1.resize((800, 600), Image.ANTIALIAS)

    img = ImageTk.PhotoImage(img1)

    background ="gray"

    panel = tk.Label(root, height = 600, image = img)

    panel.pack(side = "top", fill ="both", expand = "yes")

    frame = tk.Frame(root, bg = background)

    frame.pack(side ="bottom", fill ="both", expand = "yes")

    tk.Label(frame, text ="Nodes\t\t"+str(nodes), bg = background).pack()

    tk.Label(frame, text ="Edges\t\t"+str(edges), bg = background).pack()

    tk.Label(frame, text ="Cyclo Complexity\t"+str(complexity), bg = background).pack()

    tk.Label(frame, text ="Bounded regions\t"+str(complexity-1), bg = background).pack()

    root.mainloop()

Output:

For loop

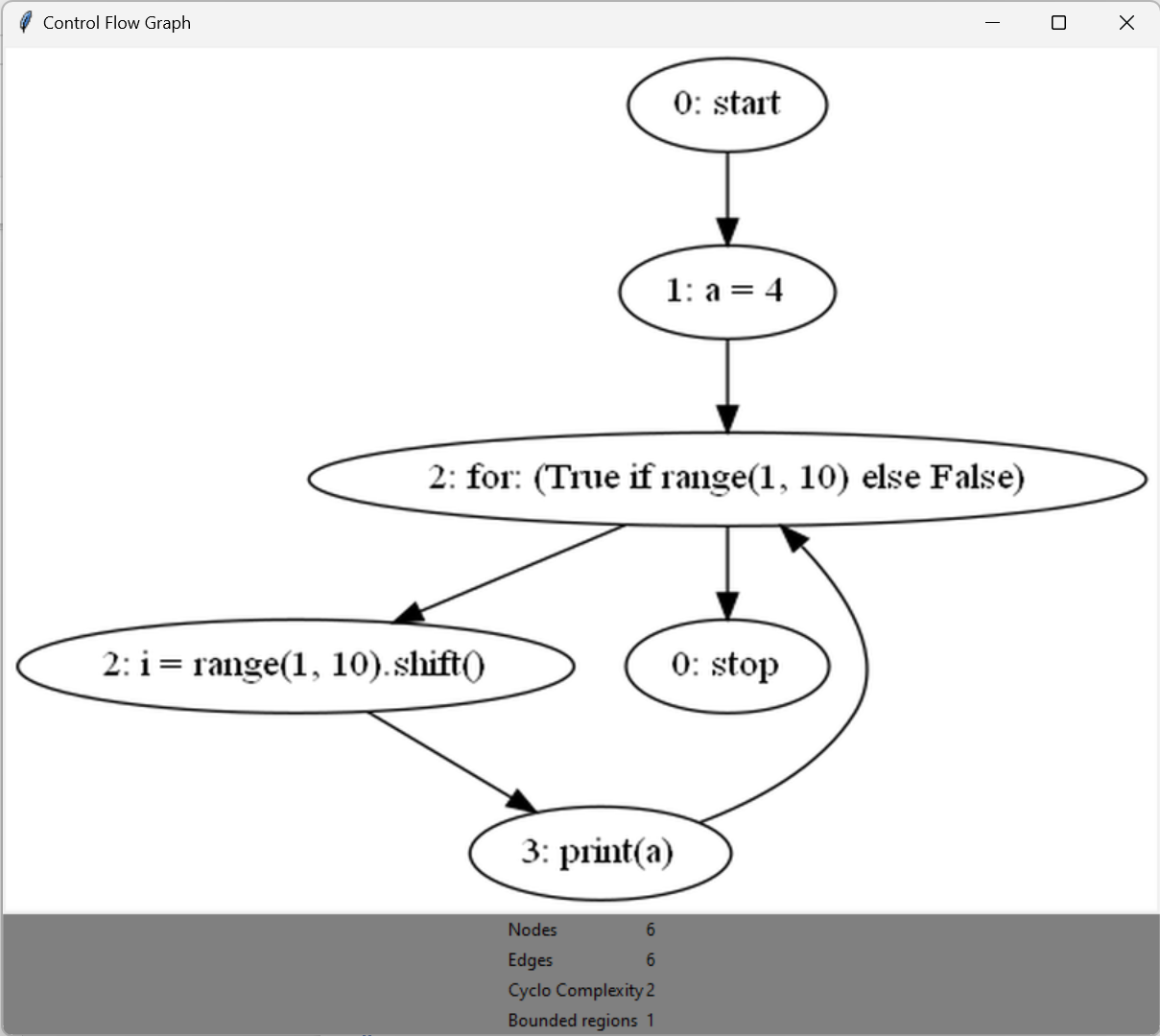
Code:

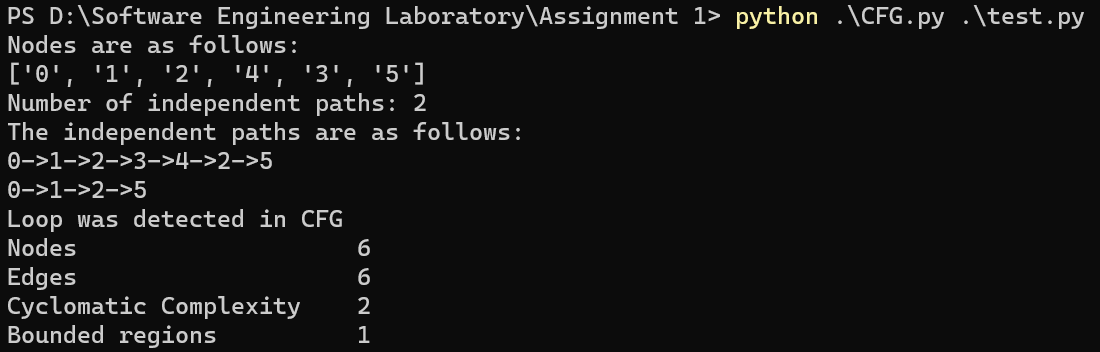
a=4

for i in range(1,10):

    print(a)

Output:





While Loop:

Code:

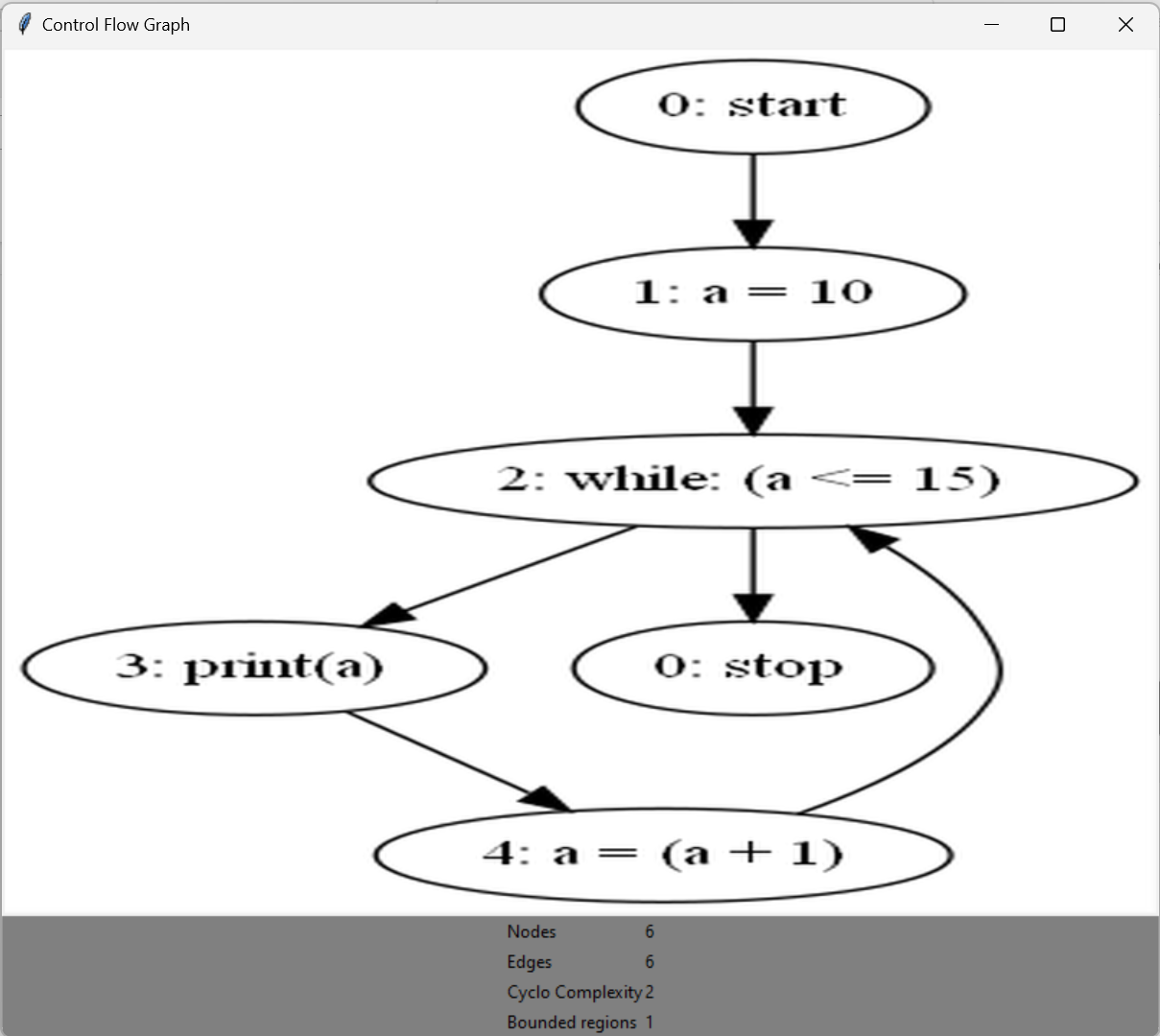
a=10

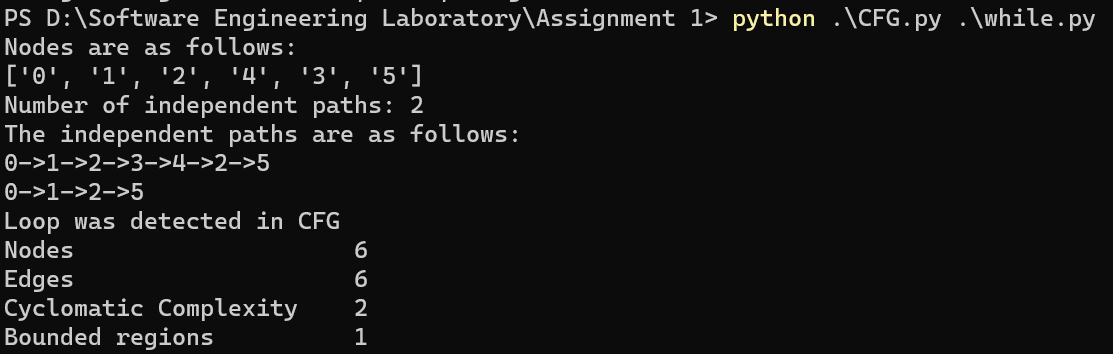
while(a<=15):

    print(a)

    a=a+1

Output:





Nested While Loop:

Code:

a=10

b=10

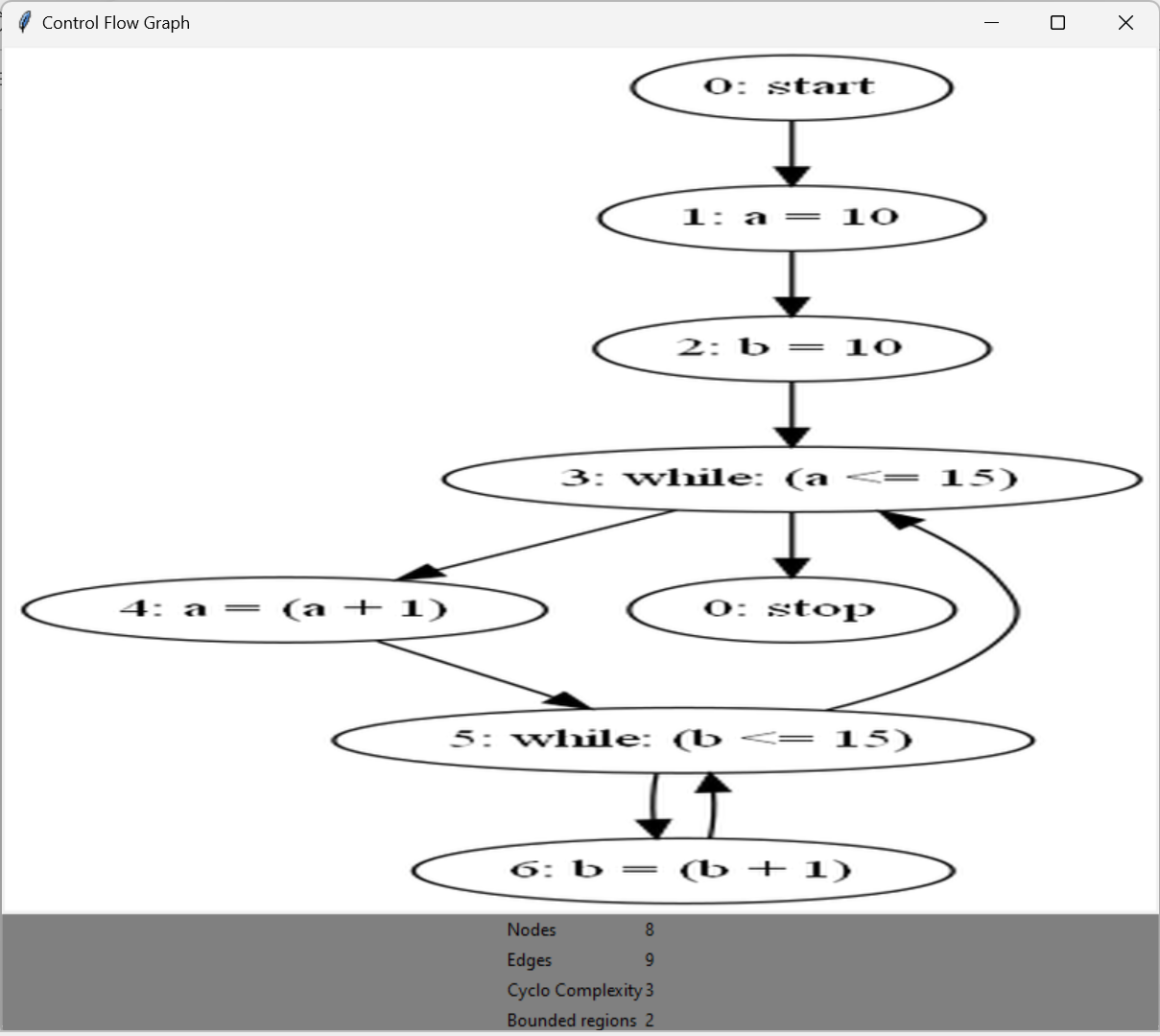
while(a<=15):

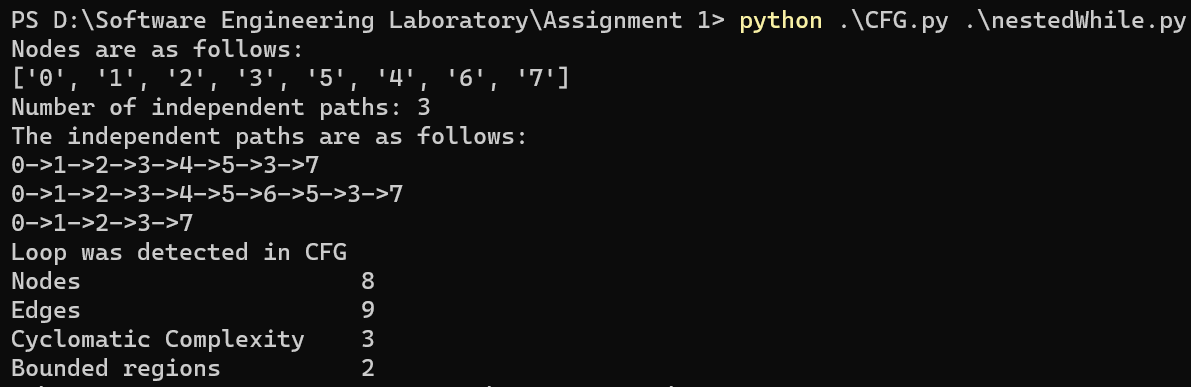
    a=a+1

    while(b<=15):

        b=b+1

Output:





Break statement:

Code:

a=10

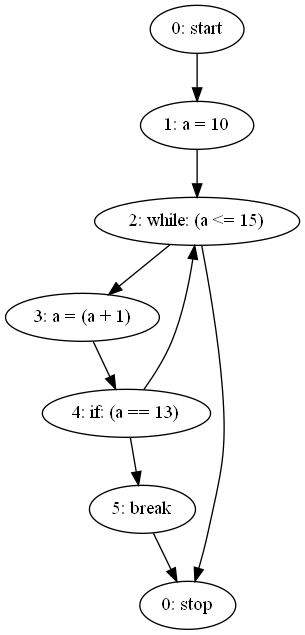
while(a<=15):

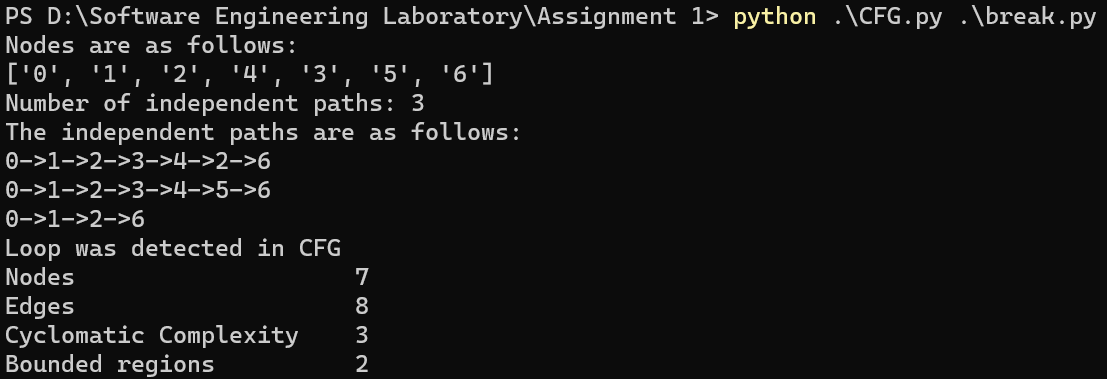
    a=a+1

    if(a==13):

        break

Output:





Continue statement

Code:

a=10

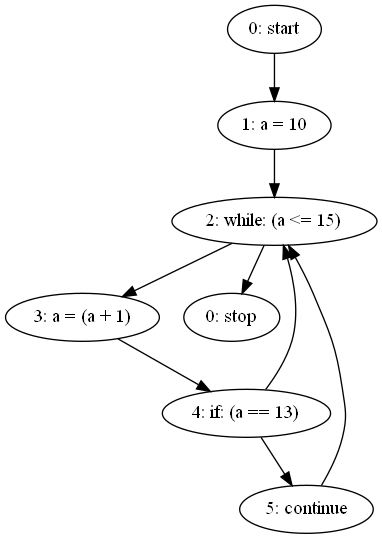
while(a<=15):

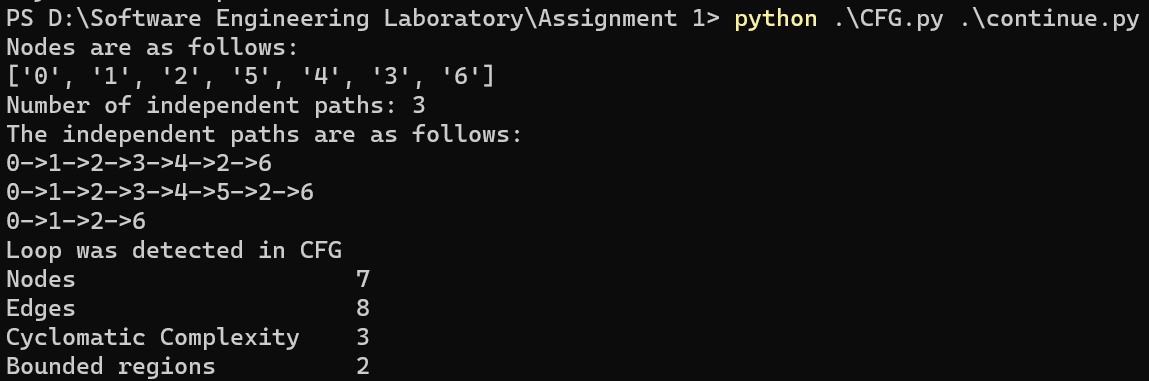
    a=a+1

    if(a==13):

        continue

Output:





All of the above:

Code:

x=1

y=1

z=1

while x<10:

    for i in range(1,10):

        y=y+1

        if z<2:

            z=z+3

            continue

    if x>5:

        break

Output:

