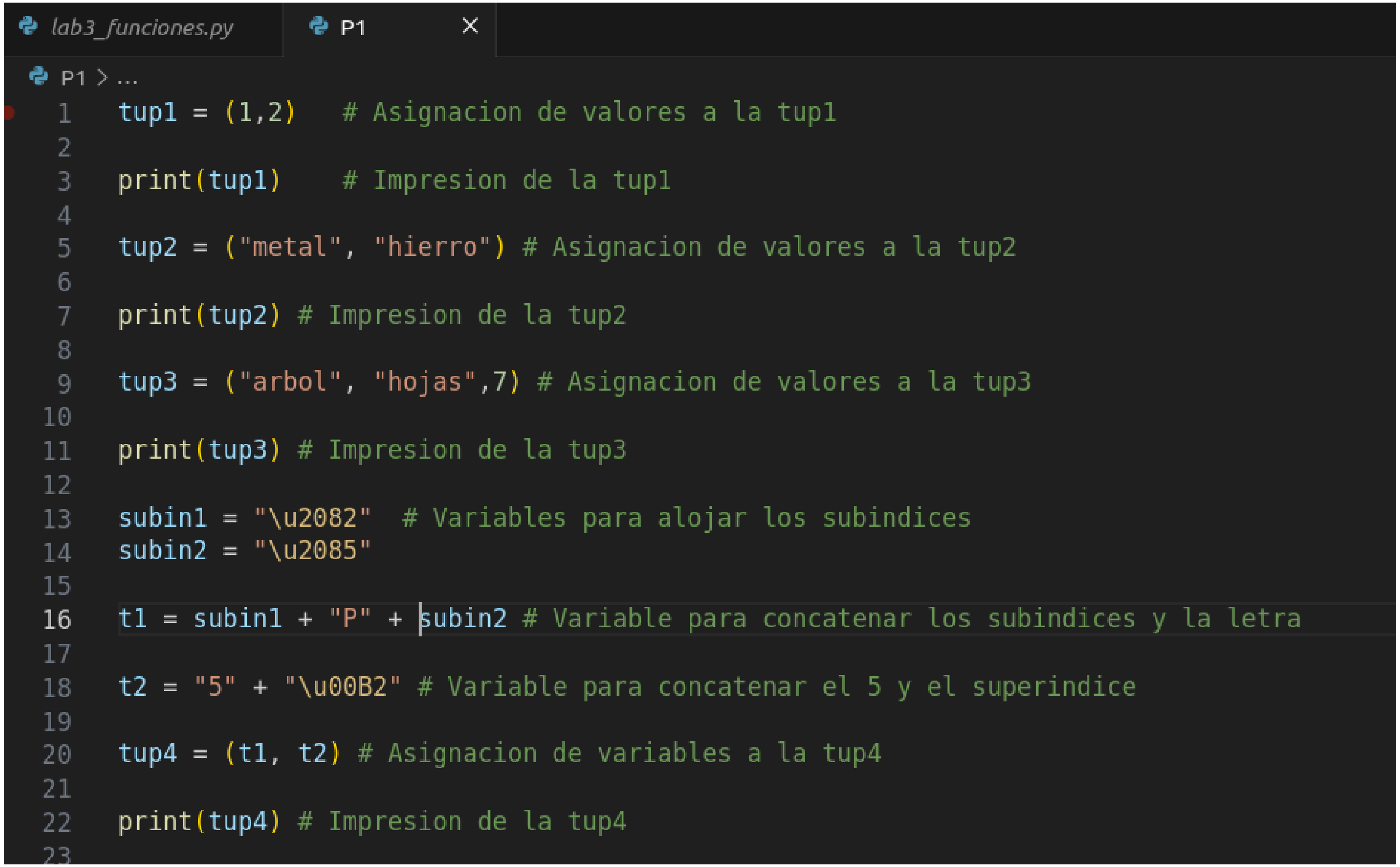
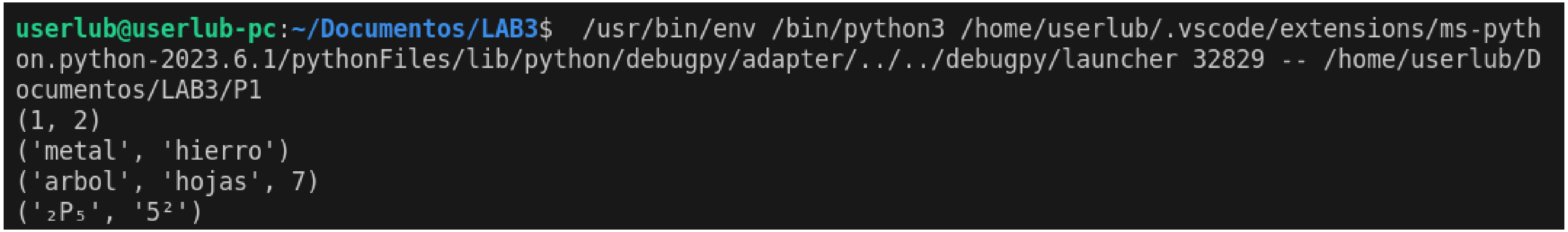
P1



```python
tup1 = (1,2)   # Asignacion de valores a la tup1

print(tup1)    # Impresion de la tup1

tup2 = ("metal", "hierro") # Asignacion de valores a la tup2

print(tup2) # Impresion de la tup2

tup3 = ("arbol", "hojas",7) # Asignacion de valores a la tup3

print(tup3) # Impresion de la tup3

subin1 = "\u2082"  # Variables para alojar los subindices
subin2 = "\u2085"

t1 = subin1 + "P" + subin2 # Variable para concatenar los subindices y la letra

t2 = "5" + "\u00B2" # Variable para concatenar el 5 y el superindice

tup4 = (t1, t2) # Asignacion de variables a la tup4

print(tup4) # Impresion de la tup4
```

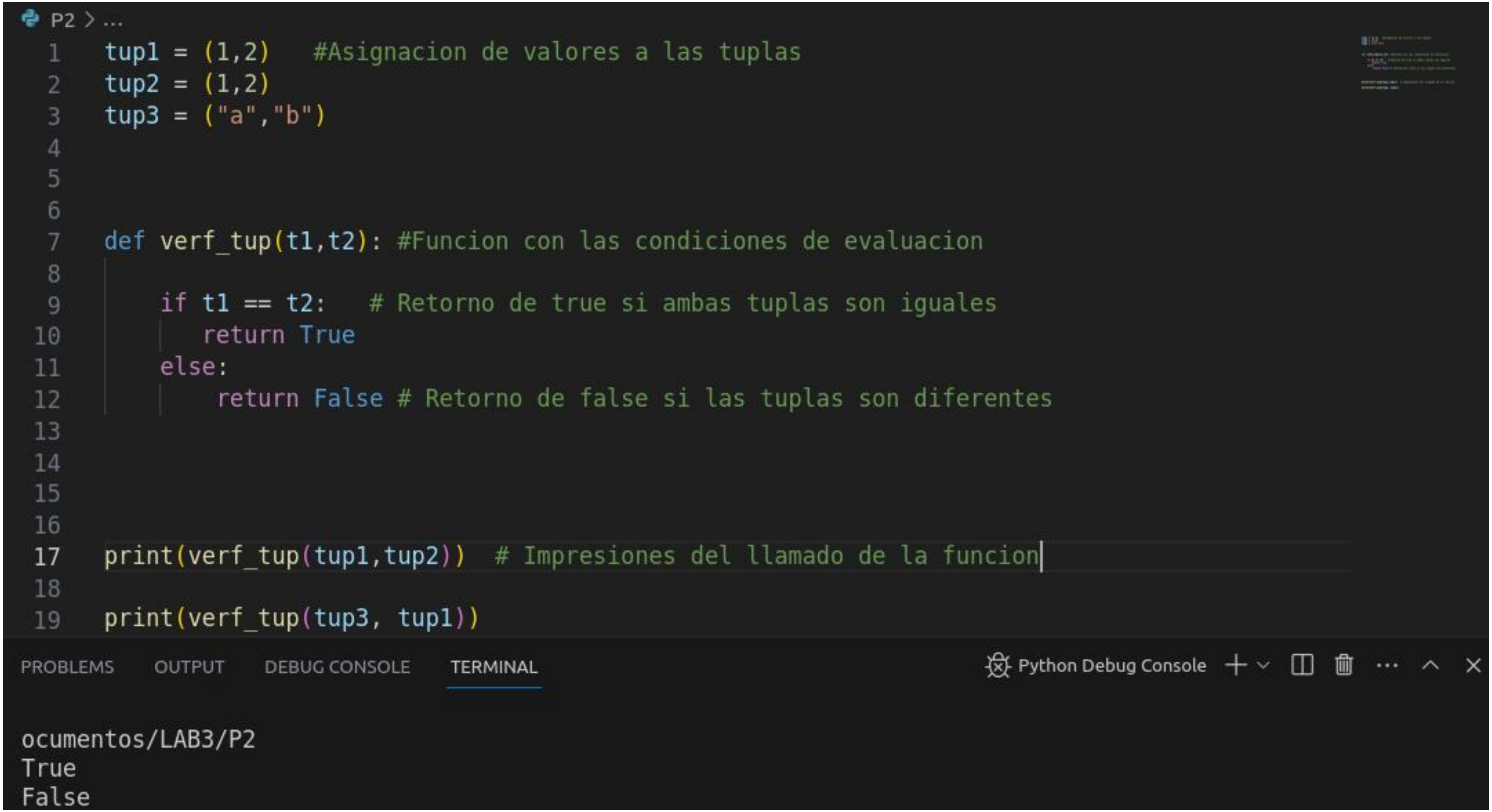
INFORME LAB #3 – E. DISCRETAS – JOY NELATON – 8-902-1282



```
userlub@userlub-pc:~/Documentos/LAB3$  /usr/bin/env /bin/python3 /home/userlub/.vscode/extensions/ms-python.python-2023.6.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 32829 -- /home/userlub/Documentos/LAB3/P1
(1, 2)
('metal', 'hierro')
('arbol', 'hojas', 7)
('₂P₅', '5²')
```

INFORME LAB #3 – E. DISCRETAS – JOY NELATON – 8-902-1282

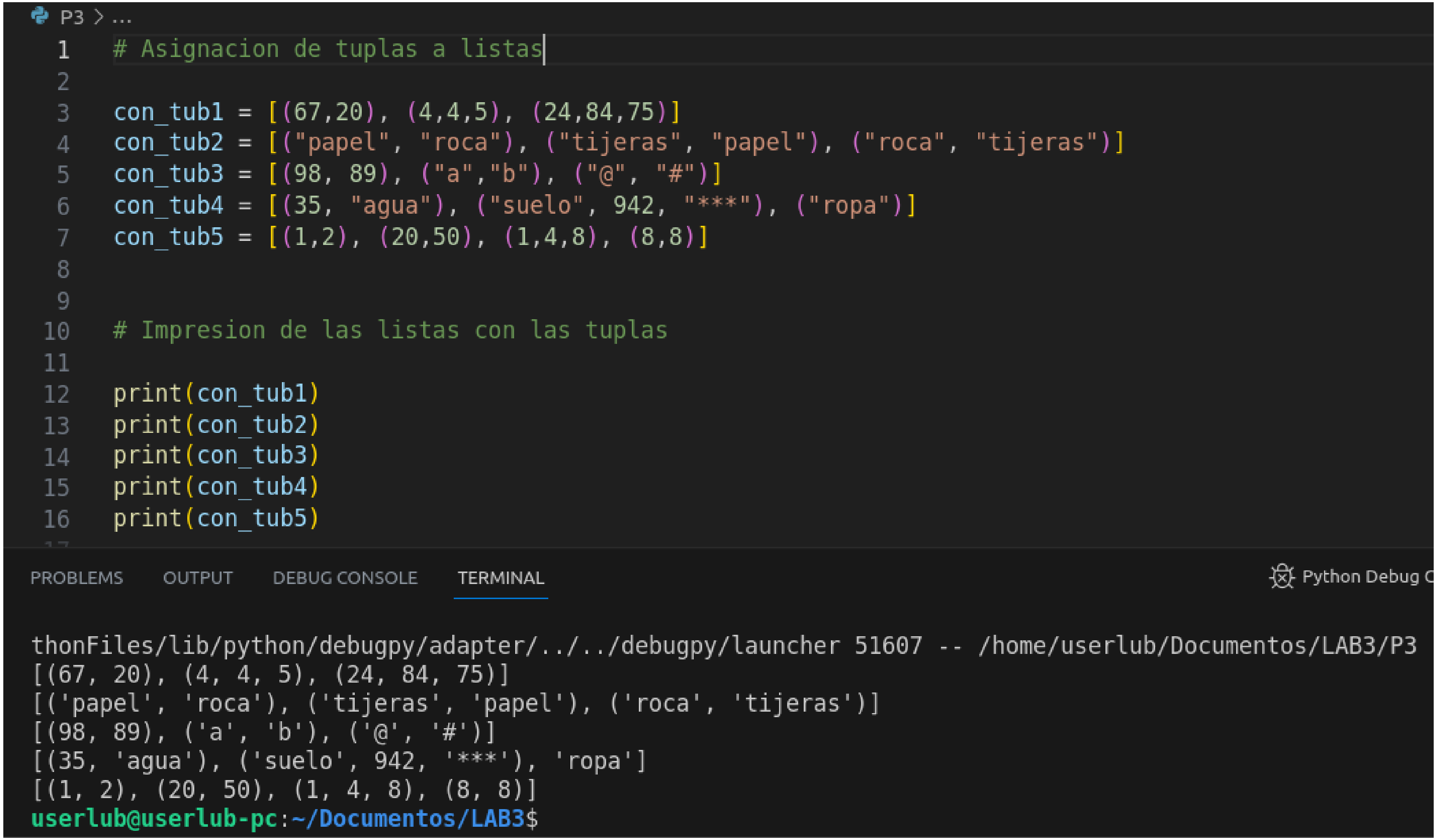
P2



```python
tup1 = (1,2)   #Asignacion de valores a las tuplas
tup2 = (1,2)
tup3 = ("a","b")



def verf_tup(t1,t2): #Funcion con las condiciones de evaluacion

    if t1 == t2:   # Retorno de true si ambas tuplas son iguales
        return True
    else:
         return False # Retorno de false si las tuplas son diferentes




print(verf_tup(tup1,tup2))  # Impresiones del llamado de la funcion

print(verf_tup(tup3, tup1))
```

```
ocumentos/LAB3/P2
True
False
```

P3

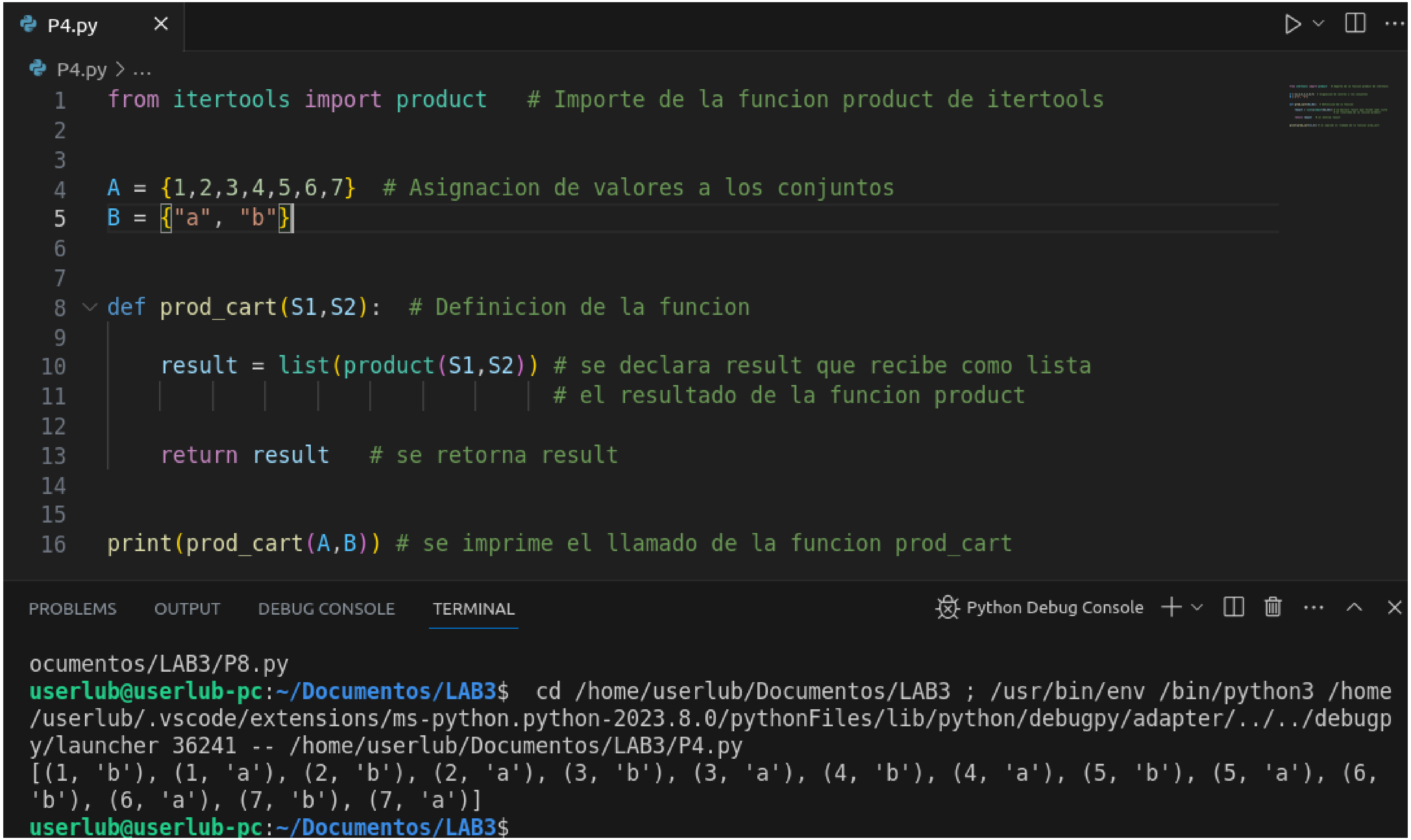


```python
# Asignacion de tuplas a listas

con_tub1 = [(67,20), (4,4,5), (24,84,75)]
con_tub2 = [("papel", "roca"), ("tijeras", "papel"), ("roca", "tijeras")]
con_tub3 = [(98, 89), ("a","b"), ("@", "#")]
con_tub4 = [(35, "agua"), ("suelo", 942, "***"), ("ropa")]
con_tub5 = [(1,2), (20,50), (1,4,8), (8,8)]


# Impresion de las listas con las tuplas

print(con_tub1)
print(con_tub2)
print(con_tub3)
print(con_tub4)
print(con_tub5)
```

```
thonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 51607 -- /home/userlub/Documentos/LAB3/P3
[(67, 20), (4, 4, 5), (24, 84, 75)]
[('papel', 'roca'), ('tijeras', 'papel'), ('roca', 'tijeras')]
[(98, 89), ('a', 'b'), ('@', '#')]
[(35, 'agua'), ('suelo', 942, '***'), 'ropa']
[(1, 2), (20, 50), (1, 4, 8), (8, 8)]
userlub@userlub-pc:~/Documentos/LAB3$
```

INFORME LAB #3 – E. DISCRETAS – JOY NELATON – 8-902-1282

P4

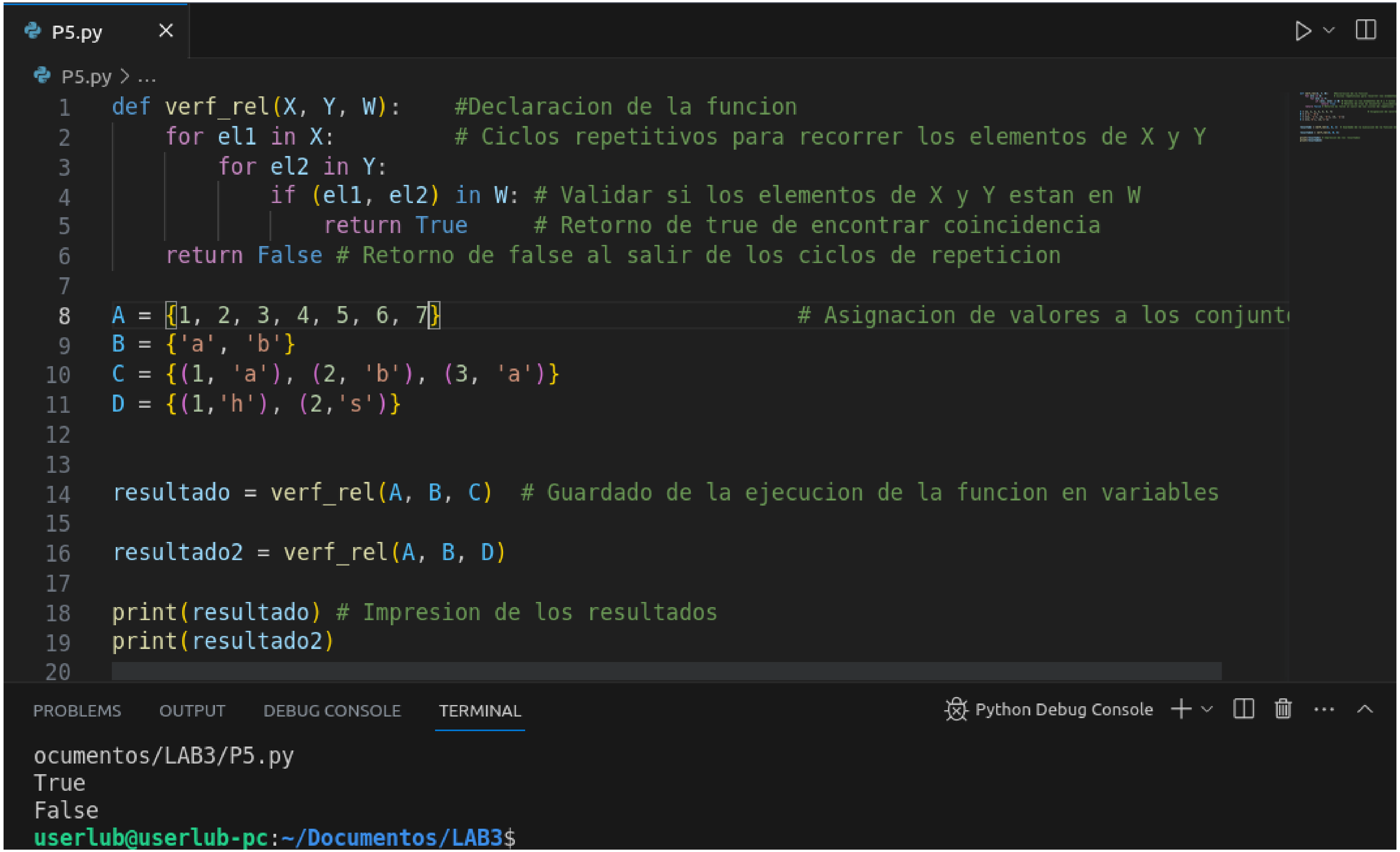


```python
from itertools import product   # Importe de la funcion product de itertools


A = {1,2,3,4,5,6,7}  # Asignacion de valores a los conjuntos
B = {"a", "b"}


def prod_cart(S1,S2):  # Definicion de la funcion

    result = list(product(S1,S2)) # se declara result que recibe como lista
                                  # el resultado de la funcion product

    return result   # se retorna result


print(prod_cart(A,B)) # se imprime el llamado de la funcion prod_cart
```

```
ocumentos/LAB3/P8.py
userlub@userlub-pc:~/Documentos/LAB3$  cd /home/userlub/Documentos/LAB3 ; /usr/bin/env /bin/python3 /home
/userlub/.vscode/extensions/ms-python.python-2023.8.0/pythonFiles/lib/python/debugpy/adapter/../../debugp
y/launcher 36241 -- /home/userlub/Documentos/LAB3/P4.py
[(1, 'b'), (1, 'a'), (2, 'b'), (2, 'a'), (3, 'b'), (3, 'a'), (4, 'b'), (4, 'a'), (5, 'b'), (5, 'a'), (6,
'b'), (6, 'a'), (7, 'b'), (7, 'a')]
userlub@userlub-pc:~/Documentos/LAB3$
```

INFORME LAB #3 – E. DISCRETAS – JOY NELATON – 8-902-1282

P5

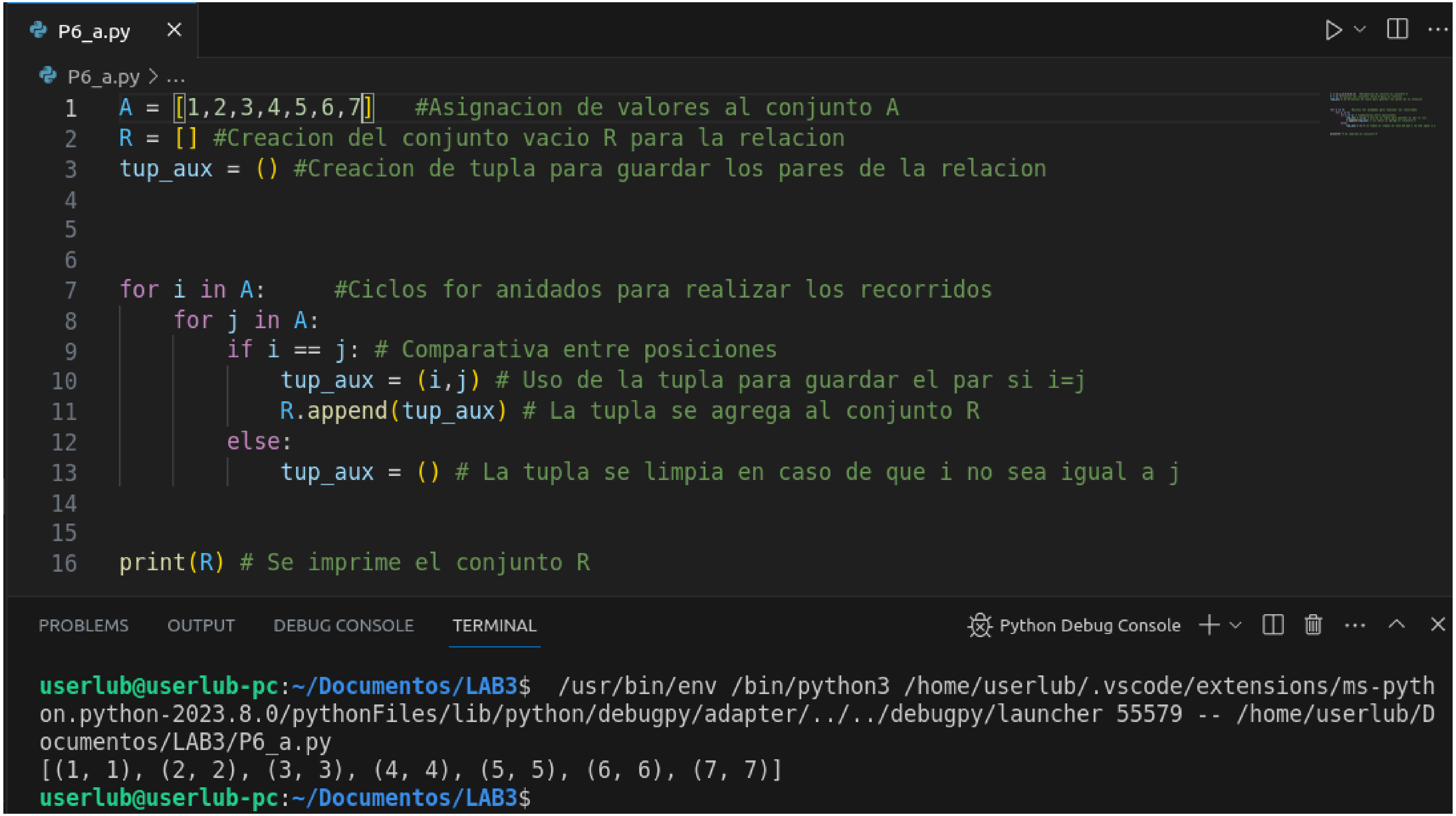


```python
def verf_rel(X, Y, W):    #Declaracion de la funcion
    for el1 in X:         # Ciclos repetitivos para recorrer los elementos de X y Y
        for el2 in Y:
            if (el1, el2) in W: # Validar si los elementos de X y Y estan en W
                return True     # Retorno de true de encontrar coincidencia
    return False # Retorno de false al salir de los ciclos de repeticion

A = {1, 2, 3, 4, 5, 6, 7}                           # Asignacion de valores a los conjunt
B = {'a', 'b'}
C = {(1, 'a'), (2, 'b'), (3, 'a')}
D = {(1,'h'), (2,'s')}


resultado = verf_rel(A, B, C)  # Guardado de la ejecucion de la funcion en variables

resultado2 = verf_rel(A, B, D)

print(resultado) # Impresion de los resultados
print(resultado2)
```

```
ocumentos/LAB3/P5.py
True
False
userlub@userlub-pc:~/Documentos/LAB3$
```

INFORME LAB #3 – E. DISCRETAS – JOY NELATON – 8-902-1282

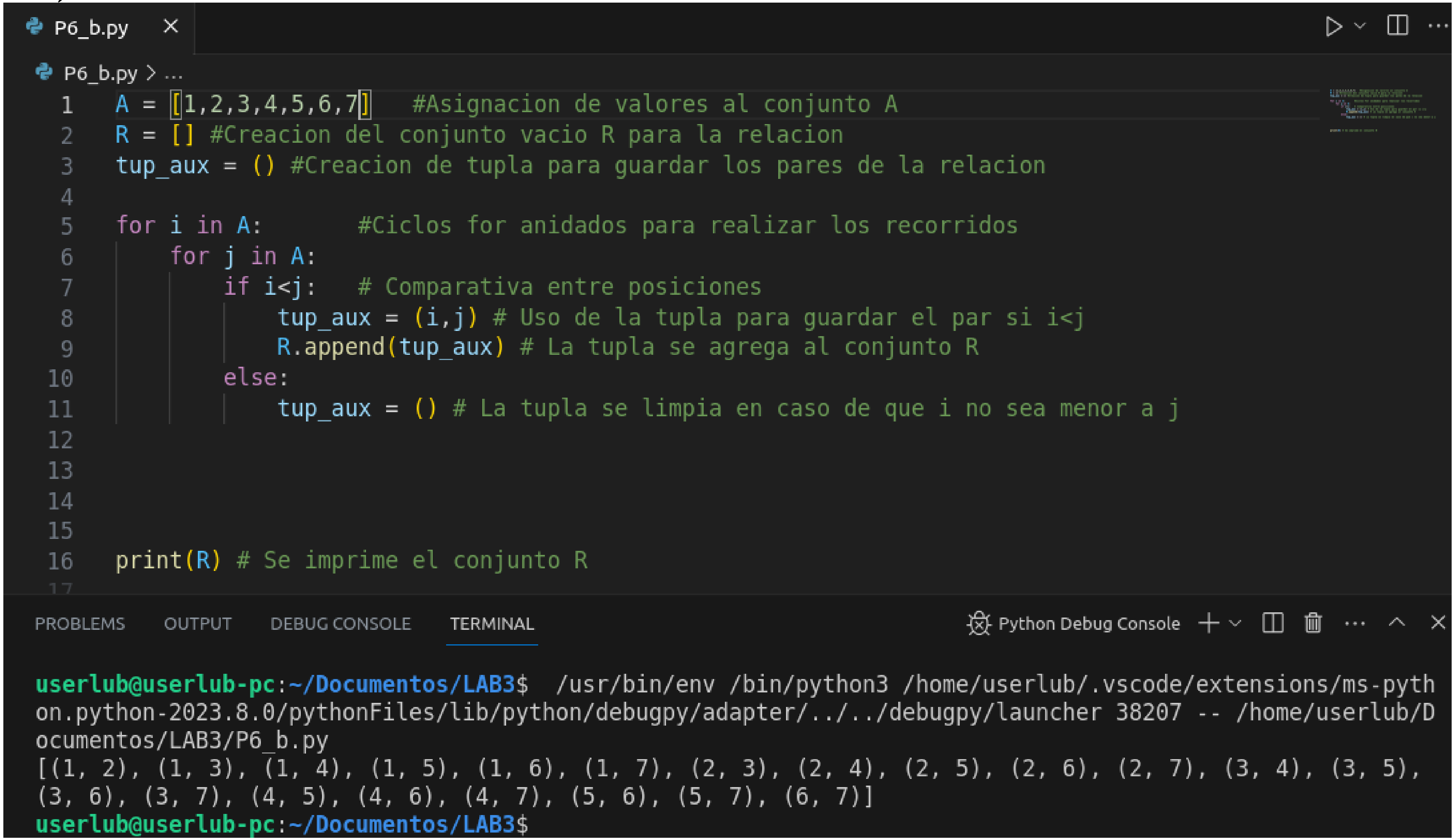
P6 a)



```python
A = [1,2,3,4,5,6,7]   #Asignacion de valores al conjunto A
R = [] #Creacion del conjunto vacio R para la relacion
tup_aux = () #Creacion de tupla para guardar los pares de la relacion



for i in A:     #Ciclos for anidados para realizar los recorridos
    for j in A:
        if i == j: # Comparativa entre posiciones
            tup_aux = (i,j) # Uso de la tupla para guardar el par si i=j
            R.append(tup_aux) # La tupla se agrega al conjunto R
        else:
            tup_aux = () # La tupla se limpia en caso de que i no sea igual a j


print(R) # Se imprime el conjunto R
```

```
userlub@userlub-pc:~/Documentos/LAB3$  /usr/bin/env /bin/python3 /home/userlub/.vscode/extensions/ms-python.python-2023.8.0/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 55579 -- /home/userlub/Documentos/LAB3/P6_a.py
[(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7)]
userlub@userlub-pc:~/Documentos/LAB3$
```

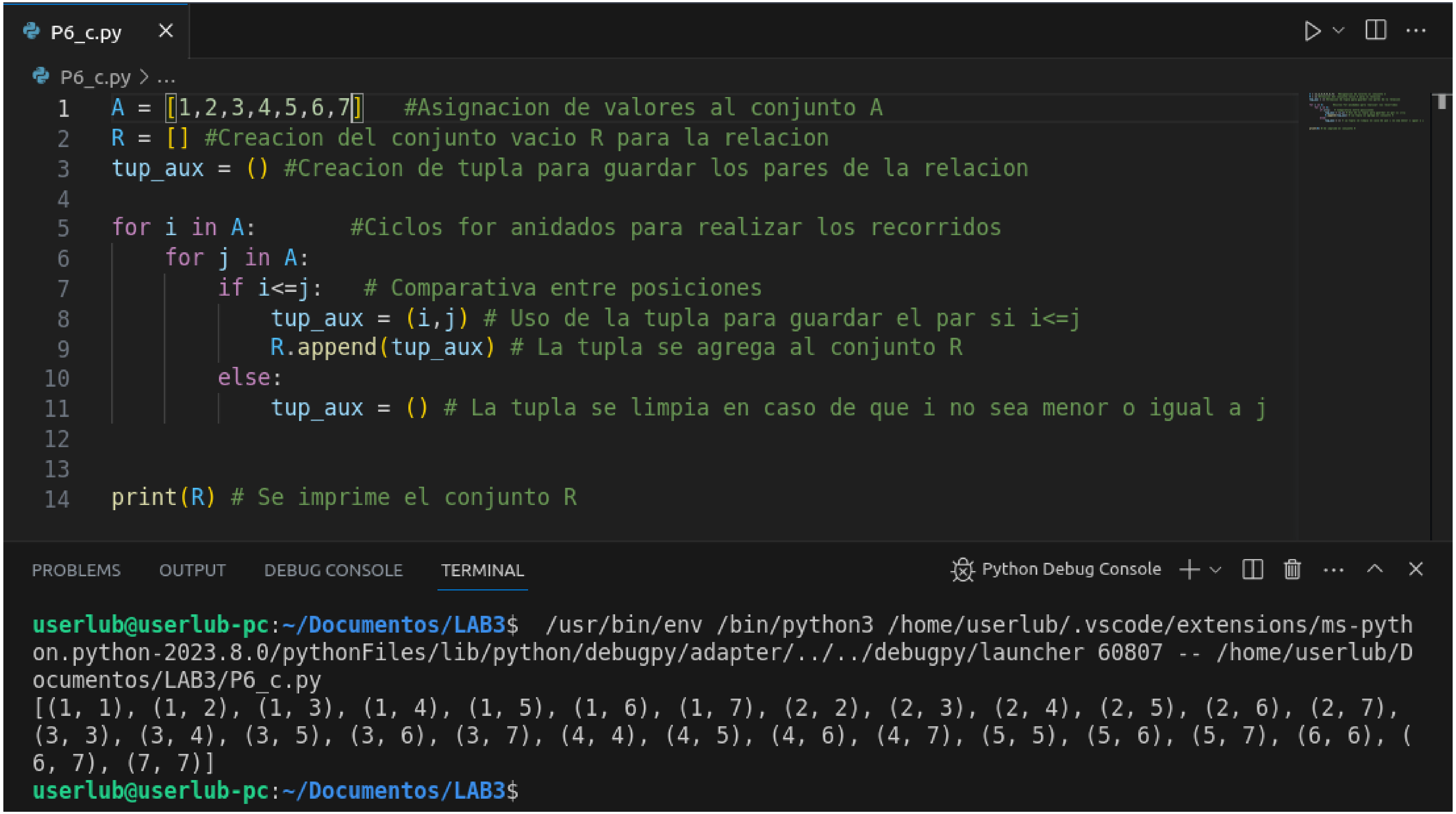
P6 b)



```python
A = [1,2,3,4,5,6,7]   #Asignacion de valores al conjunto A
R = [] #Creacion del conjunto vacio R para la relacion
tup_aux = () #Creacion de tupla para guardar los pares de la relacion

for i in A:       #Ciclos for anidados para realizar los recorridos
    for j in A:
        if i<j:   # Comparativa entre posiciones
            tup_aux = (i,j) # Uso de la tupla para guardar el par si i<j
            R.append(tup_aux) # La tupla se agrega al conjunto R
        else:
            tup_aux = () # La tupla se limpia en caso de que i no sea menor a j




print(R) # Se imprime el conjunto R
```

```
userlub@userlub-pc:~/Documentos/LAB3$  /usr/bin/env /bin/python3 /home/userlub/.vscode/extensions/ms-pyth
on.python-2023.8.0/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 38207 -- /home/userlub/D
ocumentos/LAB3/P6_b.py
[(1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 7), (2, 3), (2, 4), (2, 5), (2, 6), (2, 7), (3, 4), (3, 5),
(3, 6), (3, 7), (4, 5), (4, 6), (4, 7), (5, 6), (5, 7), (6, 7)]
userlub@userlub-pc:~/Documentos/LAB3$
```

P6 c)

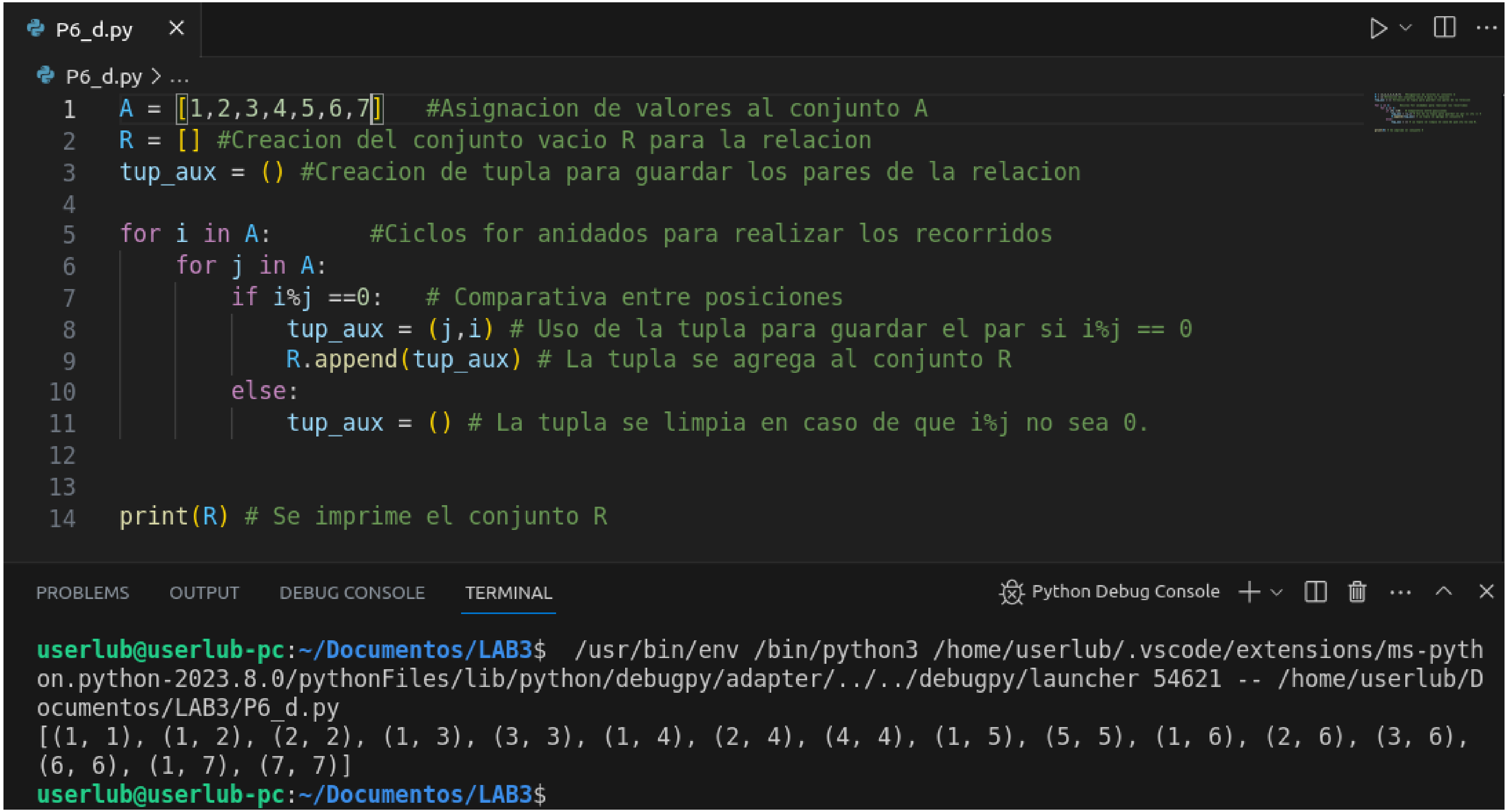


```python
A = [1,2,3,4,5,6,7]   #Asignacion de valores al conjunto A
R = [] #Creacion del conjunto vacio R para la relacion
tup_aux = () #Creacion de tupla para guardar los pares de la relacion

for i in A:       #Ciclos for anidados para realizar los recorridos
    for j in A:
        if i<=j:   # Comparativa entre posiciones
            tup_aux = (i,j) # Uso de la tupla para guardar el par si i<=j
            R.append(tup_aux) # La tupla se agrega al conjunto R
        else:
            tup_aux = () # La tupla se limpia en caso de que i no sea menor o igual a j


print(R) # Se imprime el conjunto R
```

```
userlub@userlub-pc:~/Documentos/LAB3$  /usr/bin/env /bin/python3 /home/userlub/.vscode/extensions/ms-pyth
on.python-2023.8.0/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 60807 -- /home/userlub/D
ocumentos/LAB3/P6_c.py
[(1, 1), (1, 2), (1, 3), (1, 4), (1, 5), (1, 6), (1, 7), (2, 2), (2, 3), (2, 4), (2, 5), (2, 6), (2, 7),
(3, 3), (3, 4), (3, 5), (3, 6), (3, 7), (4, 4), (4, 5), (4, 6), (4, 7), (5, 5), (5, 6), (5, 7), (6, 6), (
6, 7), (7, 7)]
userlub@userlub-pc:~/Documentos/LAB3$
```

INFORME LAB #3 – E. DISCRETAS – JOY NELATON – 8-902-1282

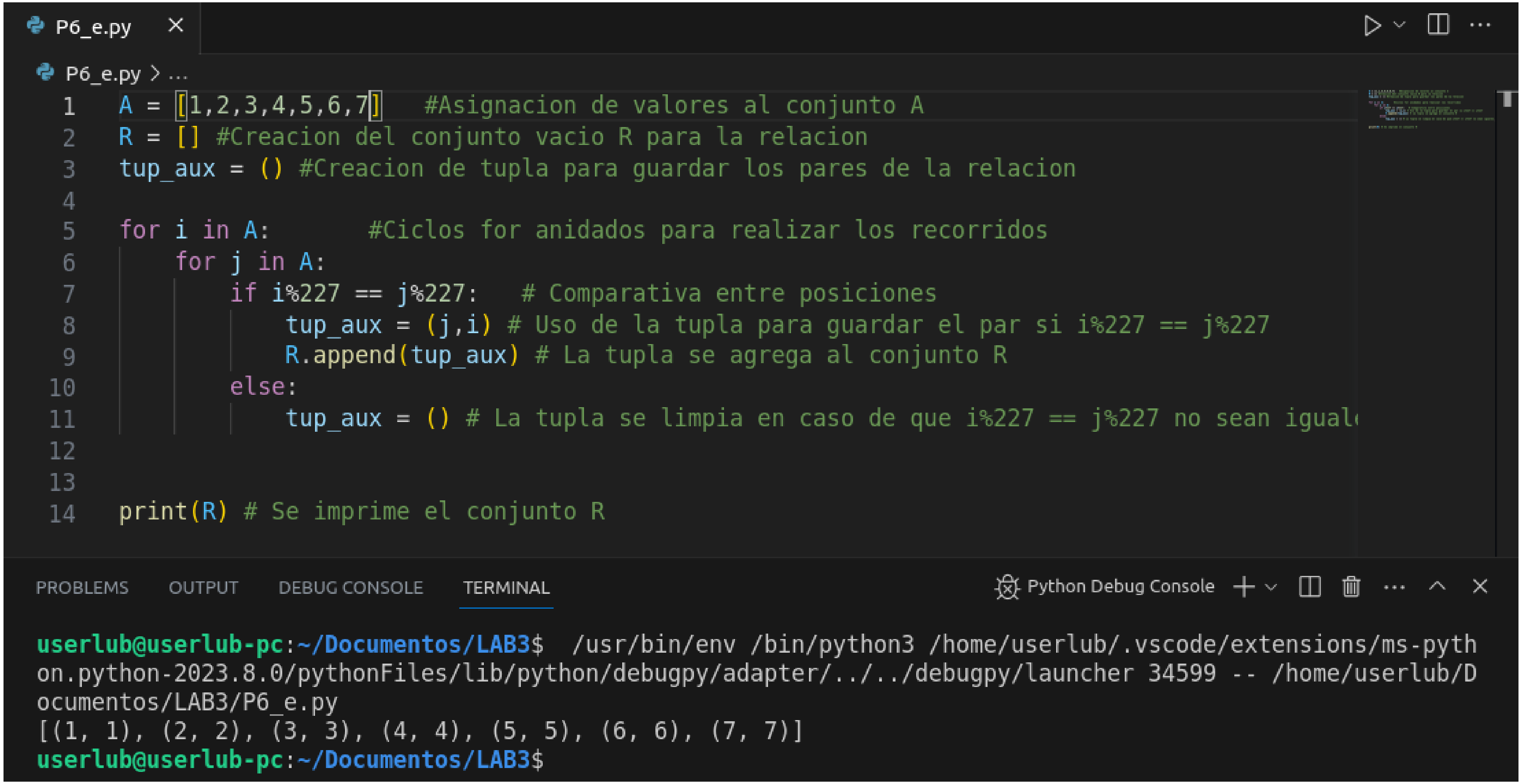
P6 d)



```python
A = [1,2,3,4,5,6,7]   #Asignacion de valores al conjunto A
R = [] #Creacion del conjunto vacio R para la relacion
tup_aux = () #Creacion de tupla para guardar los pares de la relacion

for i in A:        #Ciclos for anidados para realizar los recorridos
    for j in A:
        if i%j ==0:   # Comparativa entre posiciones
            tup_aux = (j,i) # Uso de la tupla para guardar el par si i%j == 0
            R.append(tup_aux) # La tupla se agrega al conjunto R
        else:
            tup_aux = () # La tupla se limpia en caso de que i%j no sea 0.


print(R) # Se imprime el conjunto R
```

```
userlub@userlub-pc:~/Documentos/LAB3$  /usr/bin/env /bin/python3 /home/userlub/.vscode/extensions/ms-python.python-2023.8.0/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 54621 -- /home/userlub/Documentos/LAB3/P6_d.py
[(1, 1), (1, 2), (2, 2), (1, 3), (3, 3), (1, 4), (2, 4), (4, 4), (1, 5), (5, 5), (1, 6), (2, 6), (3, 6), (6, 6), (1, 7), (7, 7)]
userlub@userlub-pc:~/Documentos/LAB3$
```

P6 e)

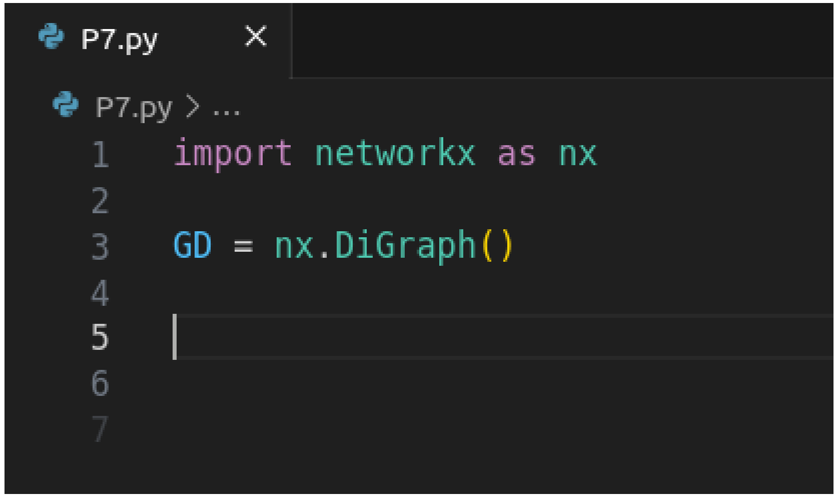


```python
A = [1,2,3,4,5,6,7]   #Asignacion de valores al conjunto A
R = [] #Creacion del conjunto vacio R para la relacion
tup_aux = () #Creacion de tupla para guardar los pares de la relacion

for i in A:        #Ciclos for anidados para realizar los recorridos
    for j in A:
        if i%227 == j%227:   # Comparativa entre posiciones
            tup_aux = (j,i) # Uso de la tupla para guardar el par si i%227 == j%227
            R.append(tup_aux) # La tupla se agrega al conjunto R
        else:
            tup_aux = () # La tupla se limpia en caso de que i%227 == j%227 no sean igual


print(R) # Se imprime el conjunto R
```

```
userlub@userlub-pc:~/Documentos/LAB3$  /usr/bin/env /bin/python3 /home/userlub/.vscode/extensions/ms-pyth
on.python-2023.8.0/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 34599 -- /home/userlub/D
ocumentos/LAB3/P6_e.py
[(1, 1), (2, 2), (3, 3), (4, 4), (5, 5), (6, 6), (7, 7)]
userlub@userlub-pc:~/Documentos/LAB3$
```

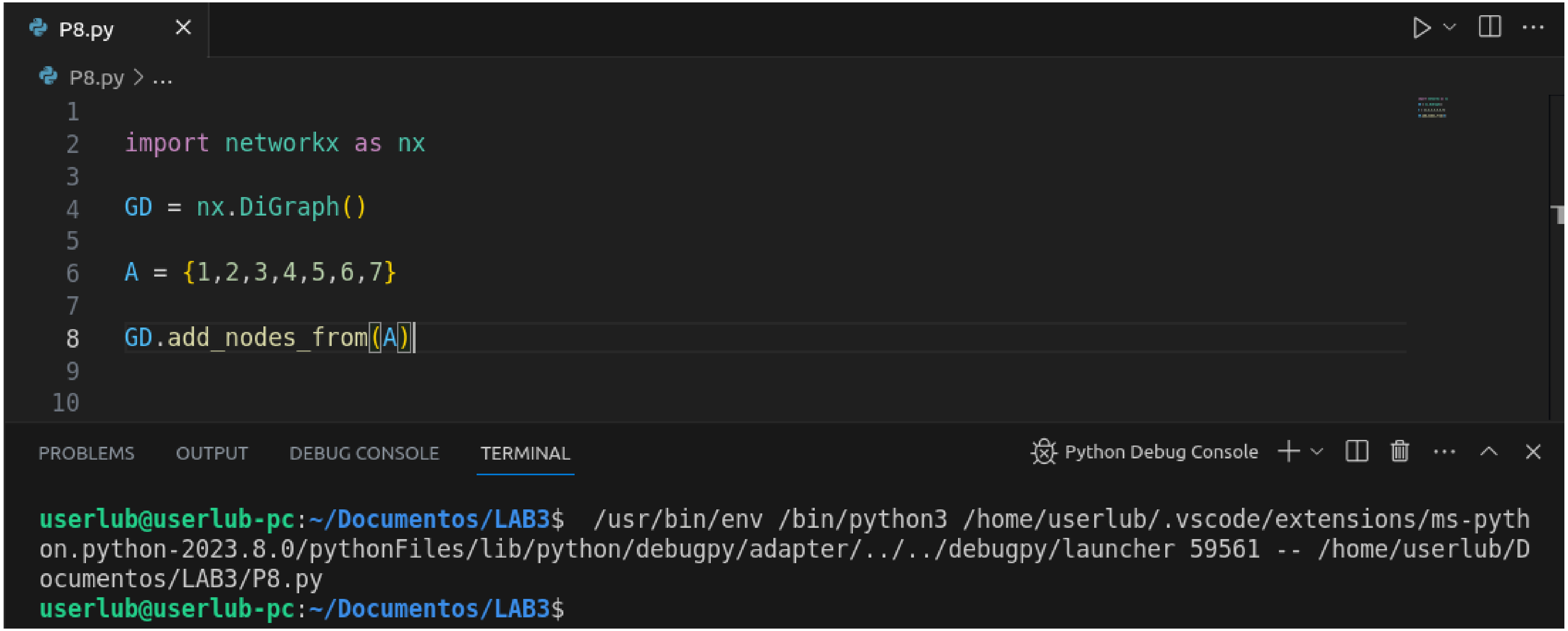
INFORME LAB #3 – E. DISCRETAS – JOY NELATON – 8-902-1282

P7



```python
import networkx as nx

GD = nx.DiGraph()

```

P8

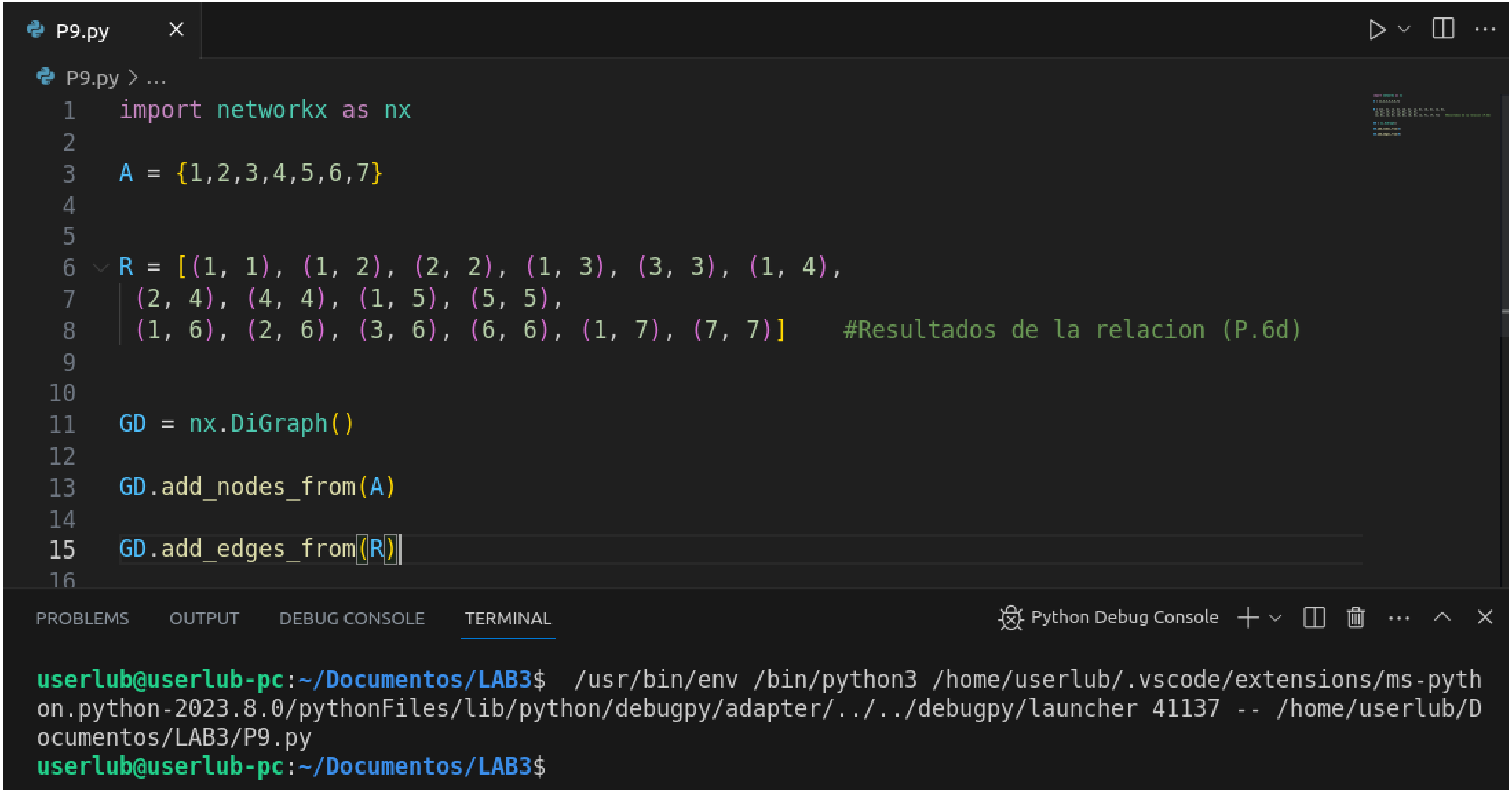


```python

import networkx as nx

GD = nx.DiGraph()

A = {1,2,3,4,5,6,7}

GD.add_nodes_from(A)


```

```
userlub@userlub-pc:~/Documentos/LAB3$  /usr/bin/env /bin/python3 /home/userlub/.vscode/extensions/ms-python.python-2023.8.0/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 59561 -- /home/userlub/Documentos/LAB3/P8.py
userlub@userlub-pc:~/Documentos/LAB3$
```

INFORME LAB #3 – E. DISCRETAS – JOY NELATON – 8-902-1282

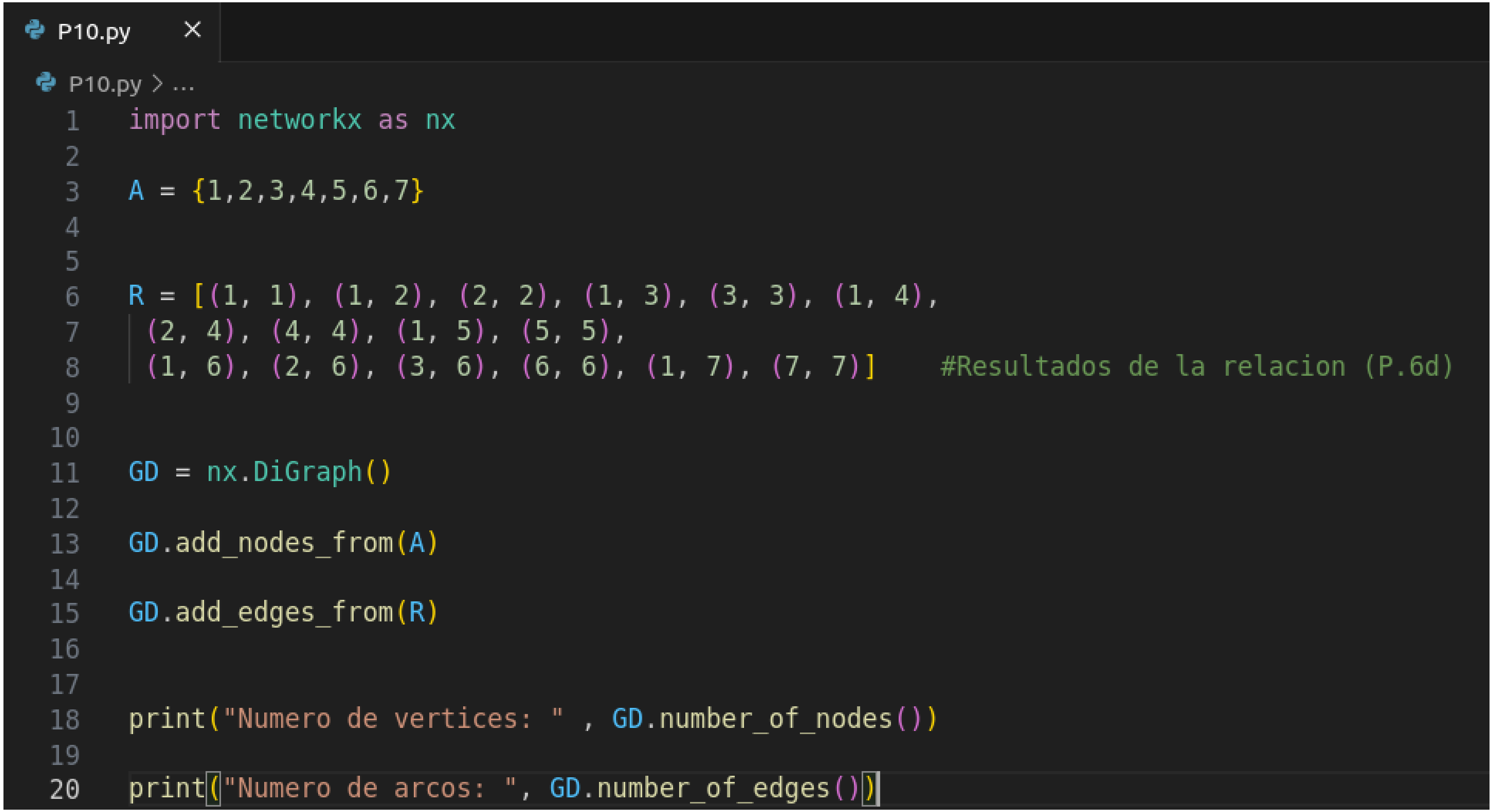
P9



```python
import networkx as nx

A = {1,2,3,4,5,6,7}


R = [(1, 1), (1, 2), (2, 2), (1, 3), (3, 3), (1, 4),
  (2, 4), (4, 4), (1, 5), (5, 5),
  (1, 6), (2, 6), (3, 6), (6, 6), (1, 7), (7, 7)]     #Resultados de la relacion (P.6d)


GD = nx.DiGraph()

GD.add_nodes_from(A)

GD.add_edges_from(R)
```

```
userlub@userlub-pc:~/Documentos/LAB3$  /usr/bin/env /bin/python3 /home/userlub/.vscode/extensions/ms-pyth
on.python-2023.8.0/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 41137 -- /home/userlub/D
ocumentos/LAB3/P9.py
userlub@userlub-pc:~/Documentos/LAB3$
```

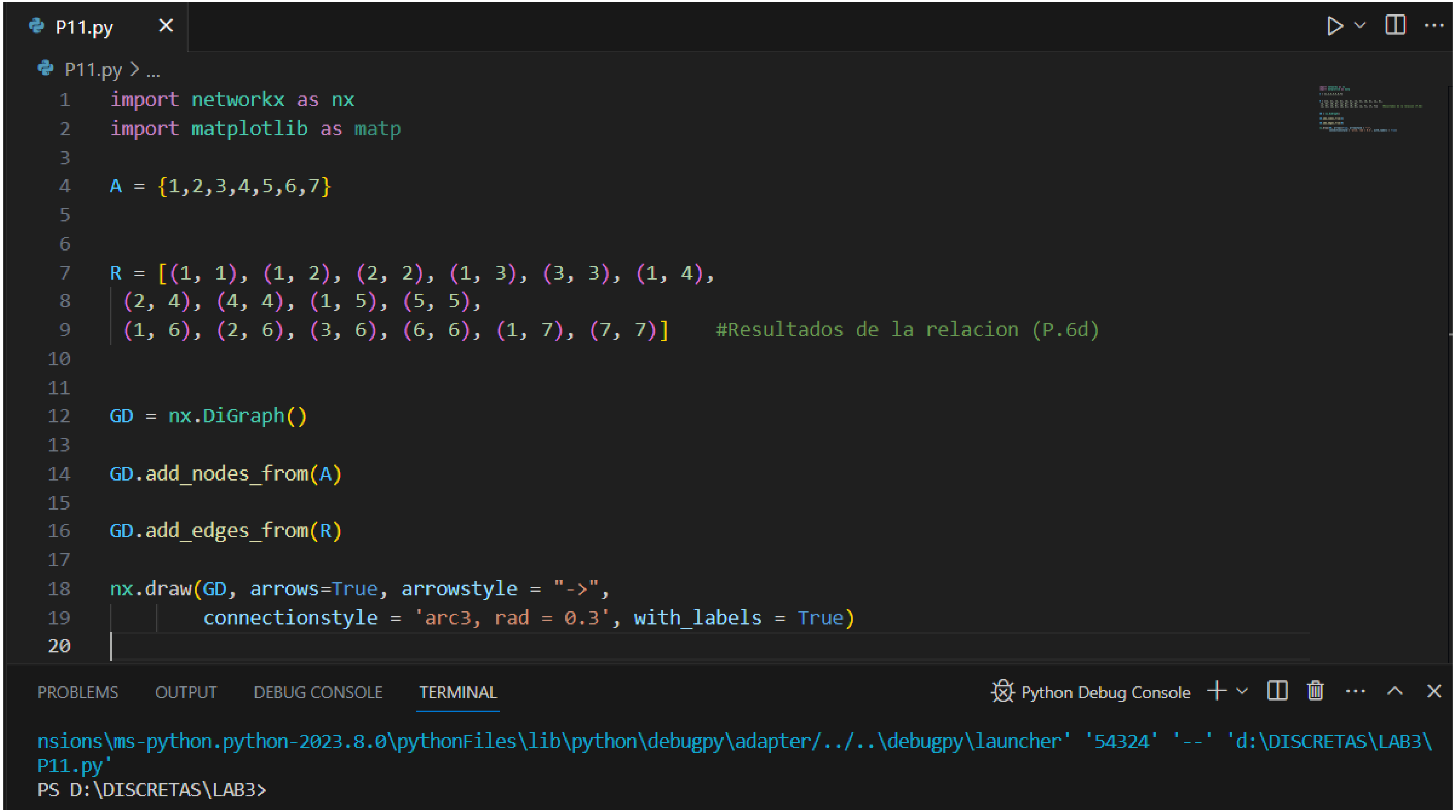
P10



```python
import networkx as nx

A = {1,2,3,4,5,6,7}


R = [(1, 1), (1, 2), (2, 2), (1, 3), (3, 3), (1, 4),
 (2, 4), (4, 4), (1, 5), (5, 5),
 (1, 6), (2, 6), (3, 6), (6, 6), (1, 7), (7, 7)]    #Resultados de la relacion (P.6d)


GD = nx.DiGraph()

GD.add_nodes_from(A)

GD.add_edges_from(R)


print("Numero de vertices: " , GD.number_of_nodes())

print("Numero de arcos: ", GD.number_of_edges())
```

INFORME LAB #3 – E. DISCRETAS – JOY NELATON – 8-902-1282

P11

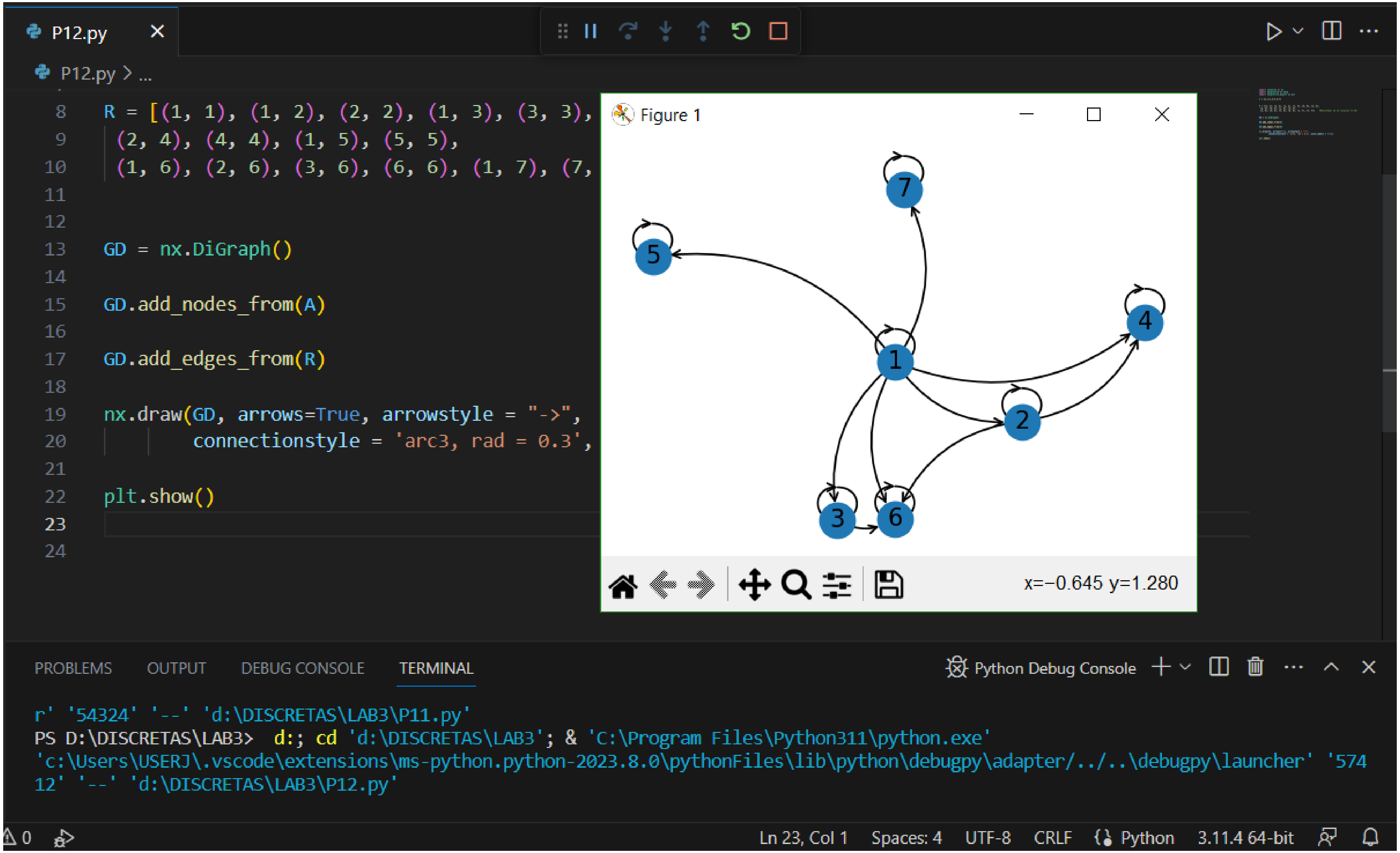


```python
import networkx as nx
import matplotlib as matp

A = {1,2,3,4,5,6,7}


R = [(1, 1), (1, 2), (2, 2), (1, 3), (3, 3), (1, 4),
 (2, 4), (4, 4), (1, 5), (5, 5),
 (1, 6), (2, 6), (3, 6), (6, 6), (1, 7), (7, 7)]    #Resultados de la relacion (P.6d)


GD = nx.DiGraph()

GD.add_nodes_from(A)

GD.add_edges_from(R)

nx.draw(GD, arrows=True, arrowstyle = "->",
        connectionstyle = 'arc3, rad = 0.3', with_labels = True)

```

```
nsions\ms-python.python-2023.8.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '54324' '--' 'd:\DISCRETAS\LAB3\
P11.py'
PS D:\DISCRETAS\LAB3>
```

INFORME LAB #3 – E. DISCRETAS – JOY NELATON – 8-902-1282

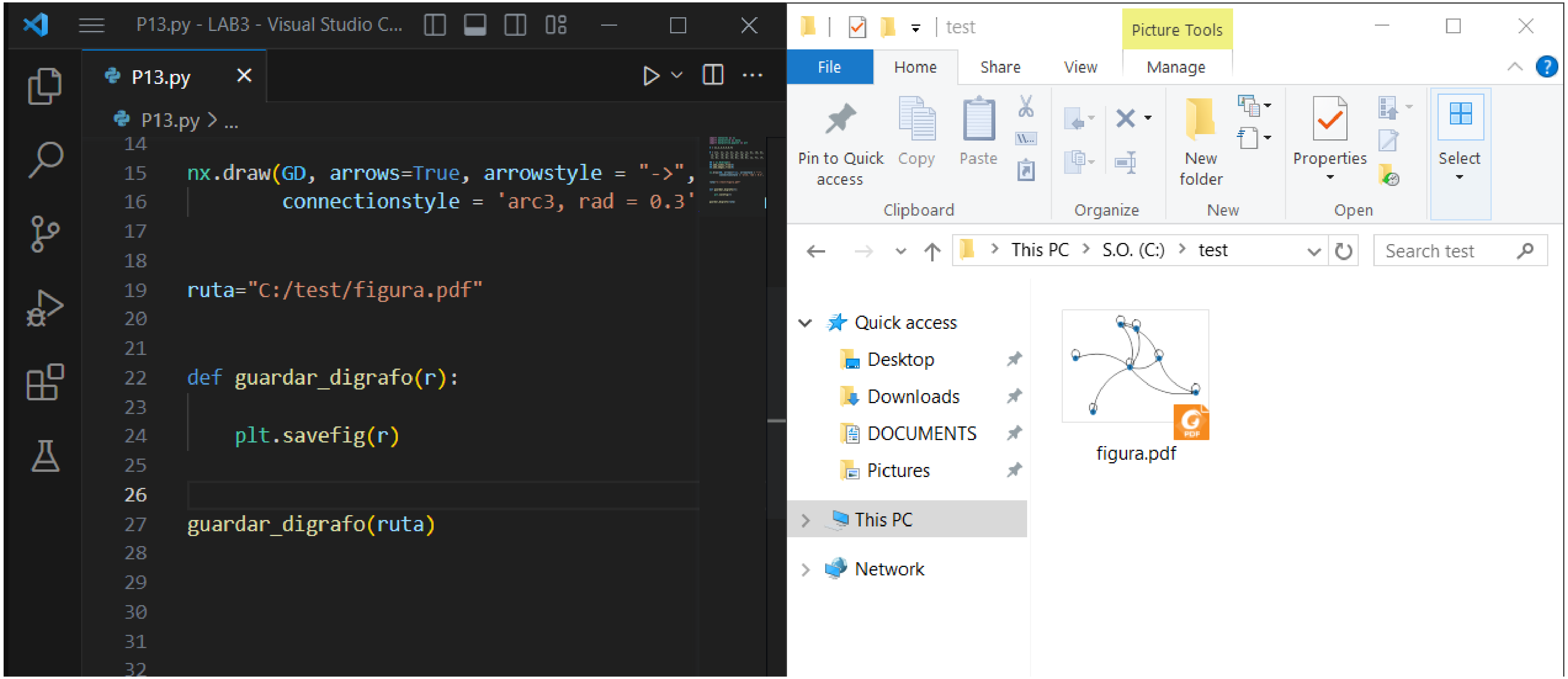
P12



```
R = [(1, 1), (1, 2), (2, 2), (1, 3), (3, 3),
 (2, 4), (4, 4), (1, 5), (5, 5),
 (1, 6), (2, 6), (3, 6), (6, 6), (1, 7), (7,


GD = nx.DiGraph()

GD.add_nodes_from(A)

GD.add_edges_from(R)

nx.draw(GD, arrows=True, arrowstyle = "->",
        connectionstyle = 'arc3, rad = 0.3',

plt.show()
```

```
r' '54324' '--' 'd:\DISCRETAS\LAB3\P11.py'
PS D:\DISCRETAS\LAB3>  d:; cd 'd:\DISCRETAS\LAB3'; & 'C:\Program Files\Python311\python.exe'
'c:\Users\USERJ\.vscode\extensions\ms-python.python-2023.8.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '574
12' '--' 'd:\DISCRETAS\LAB3\P12.py'
```

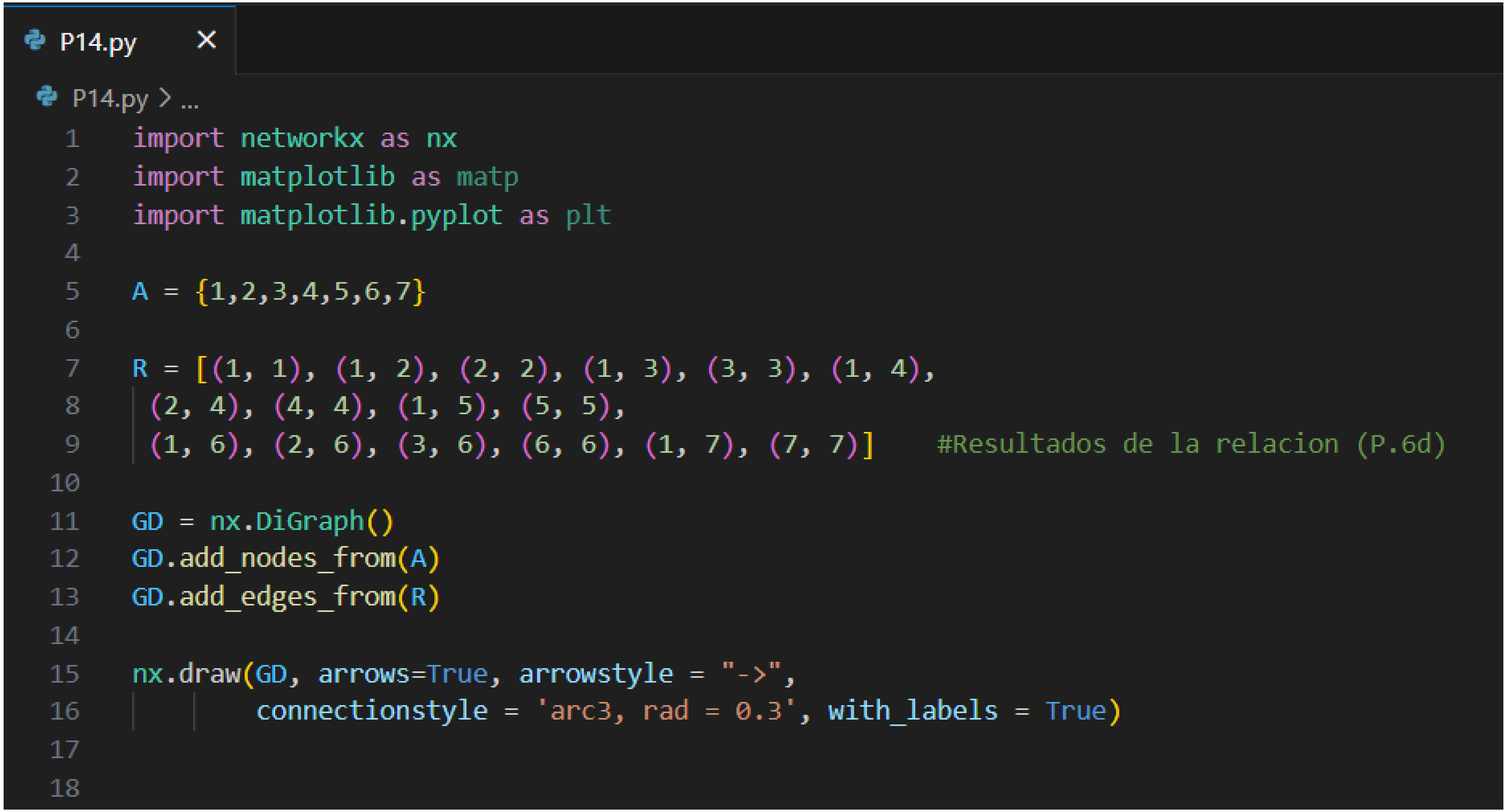
P13



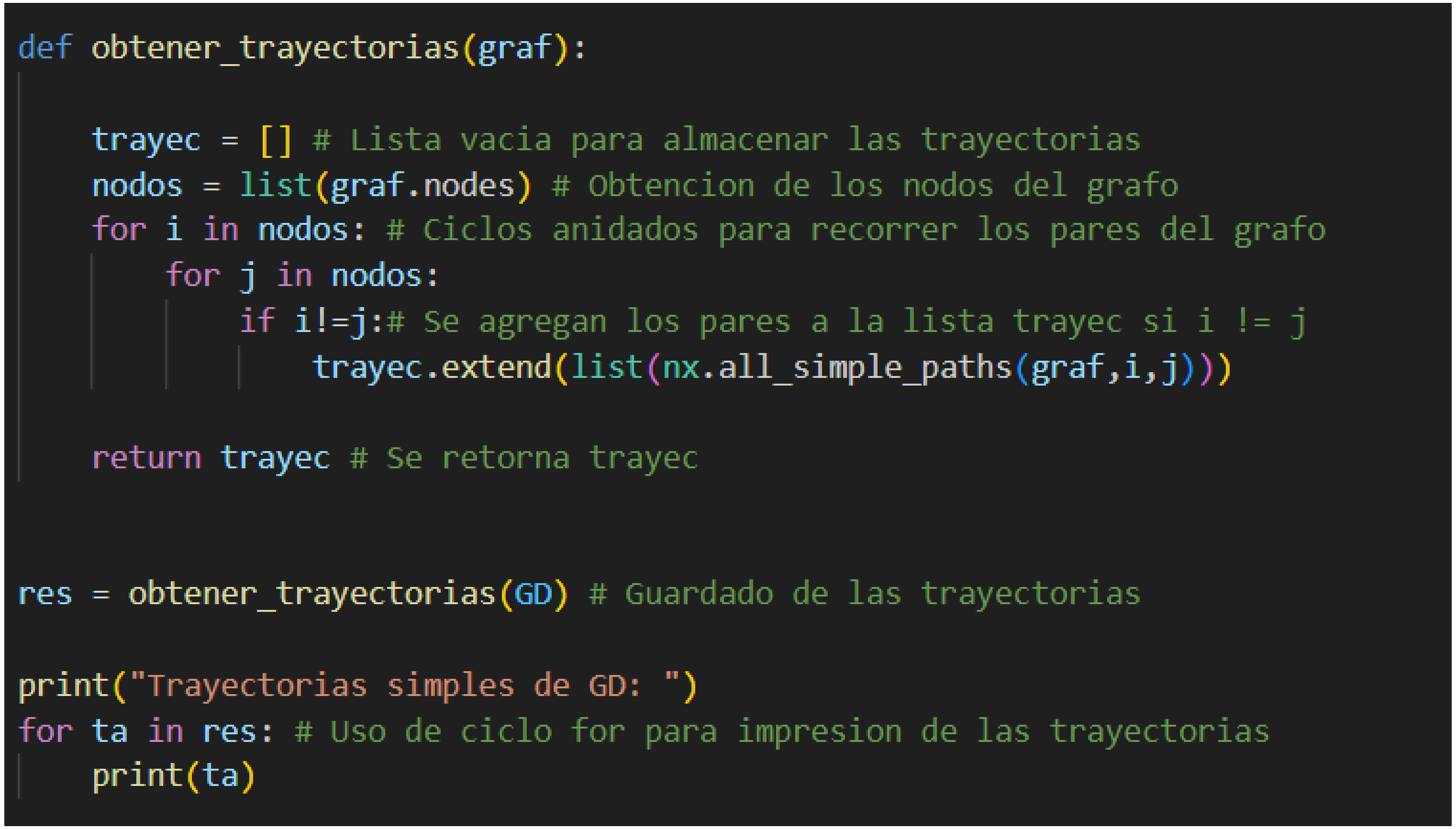
```
nx.draw(GD, arrows=True, arrowstyle = "->",
        connectionstyle = 'arc3, rad = 0.3'


ruta="C:/test/figura.pdf"


def guardar_digrafo(r):

    plt.savefig(r)


guardar_digrafo(ruta)
```

Guardado del digrafo en formato pdf en la carpeta test ubicada en disco local C.

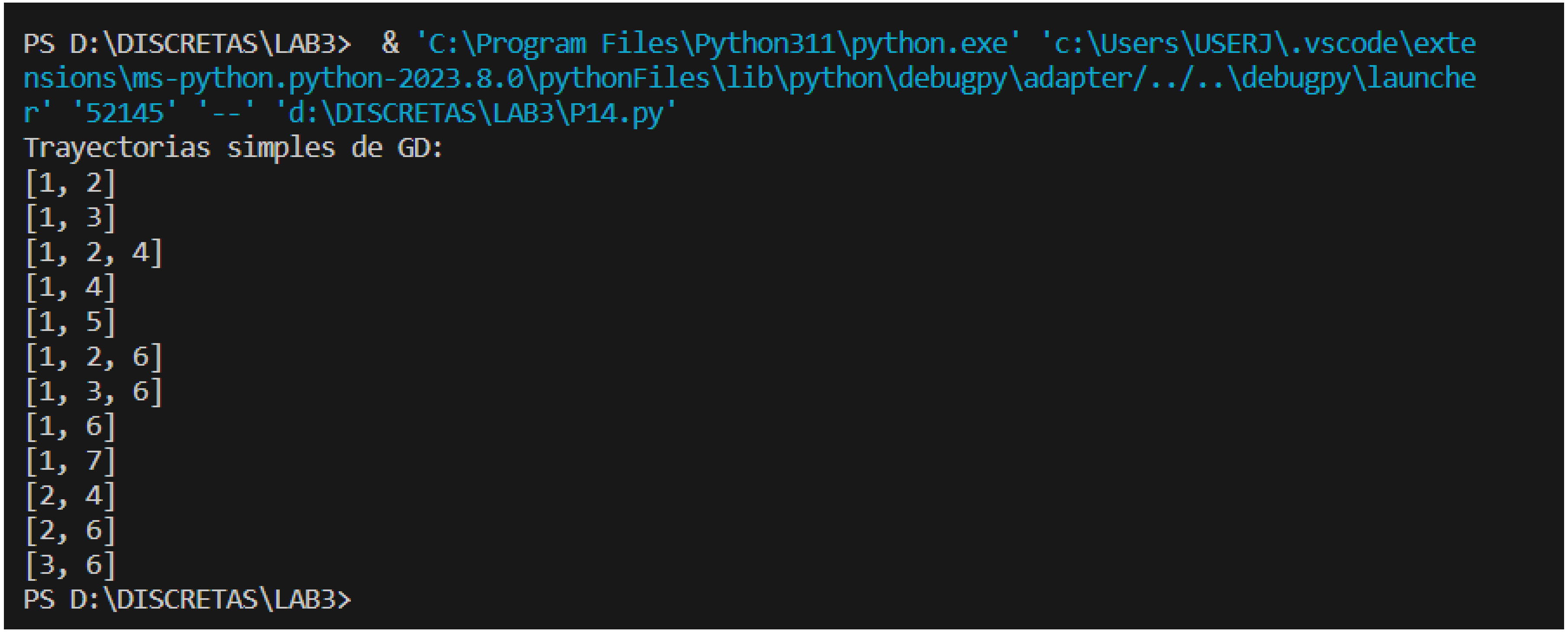
P14



```
import networkx as nx
import matplotlib as matp
import matplotlib.pyplot as plt

A = {1,2,3,4,5,6,7}

R = [(1, 1), (1, 2), (2, 2), (1, 3), (3, 3), (1, 4),
 (2, 4), (4, 4), (1, 5), (5, 5),
 (1, 6), (2, 6), (3, 6), (6, 6), (1, 7), (7, 7)]    #Resultados de la relacion (P.6d)

GD = nx.DiGraph()
GD.add_nodes_from(A)
GD.add_edges_from(R)

nx.draw(GD, arrows=True, arrowstyle = "->",
        connectionstyle = 'arc3, rad = 0.3', with_labels = True)


```

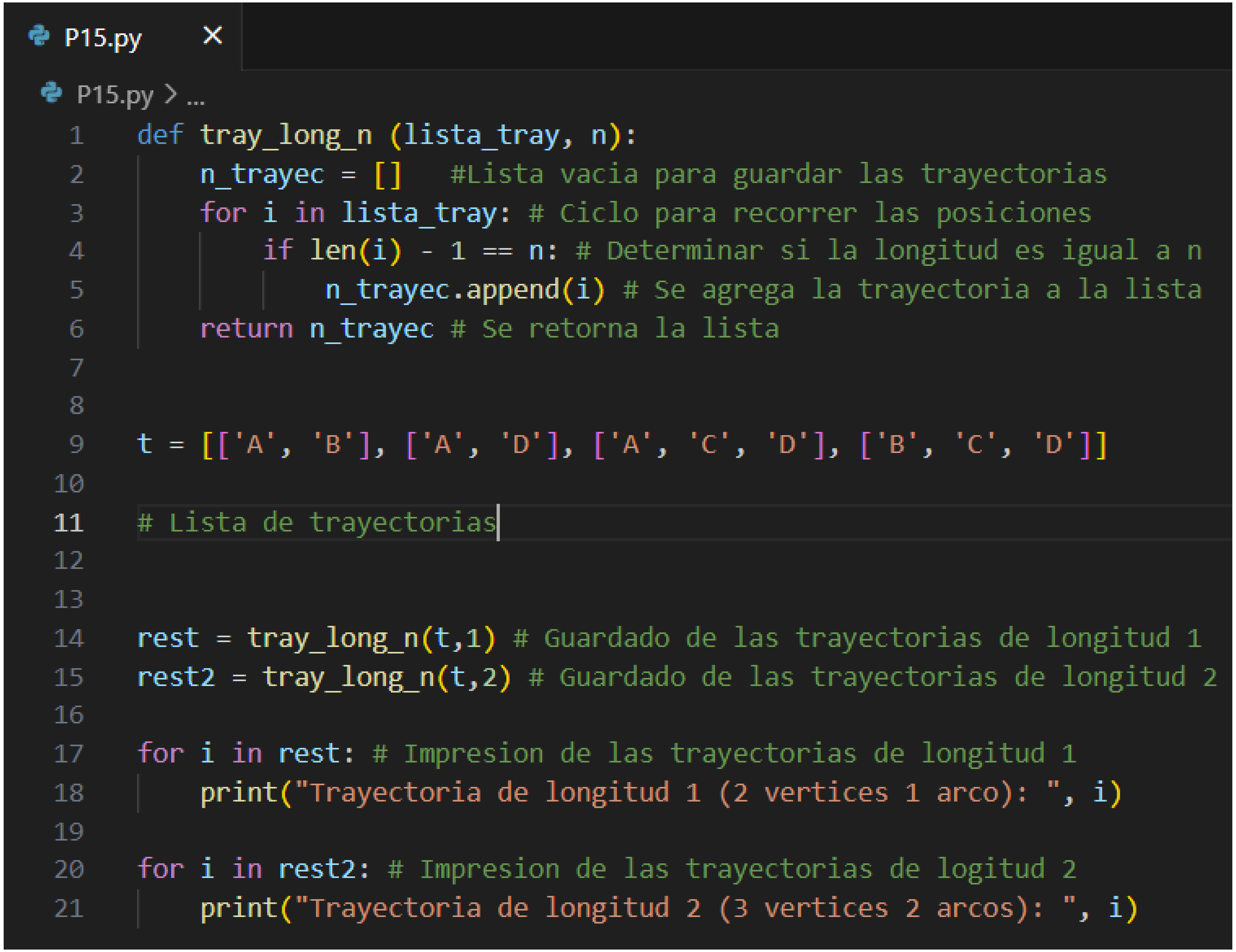


```python
def obtener_trayectorias(graf):

    trayec = [] # Lista vacia para almacenar las trayectorias
    nodos = list(graf.nodes) # Obtencion de los nodos del grafo
    for i in nodos: # Ciclos anidados para recorrer los pares del grafo
        for j in nodos:
            if i!=j:# Se agregan los pares a la lista trayec si i != j
                trayec.extend(list(nx.all_simple_paths(graf,i,j)))

    return trayec # Se retorna trayec


res = obtener_trayectorias(GD) # Guardado de las trayectorias

print("Trayectorias simples de GD: ")
for ta in res: # Uso de ciclo for para impresion de las trayectorias
    print(ta)
```

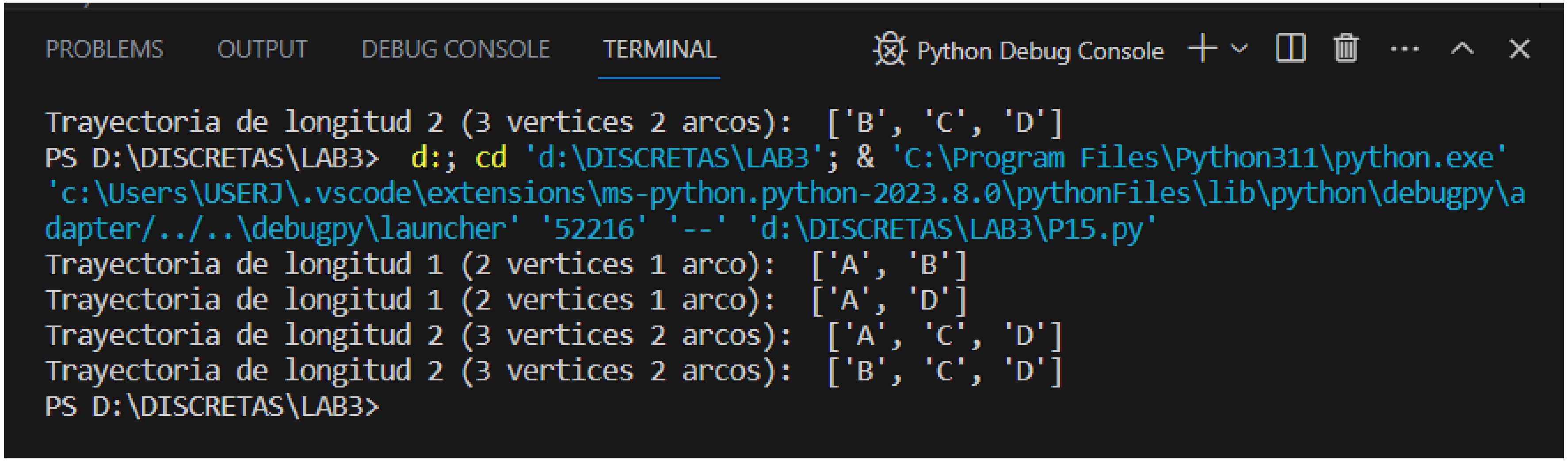


```
PS D:\DISCRETAS\LAB3>  & 'C:\Program Files\Python311\python.exe' 'c:\Users\USERJ\.vscode\exte
nsions\ms-python.python-2023.8.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launche
r' '52145' '--' 'd:\DISCRETAS\LAB3\P14.py'
Trayectorias simples de GD:
[1, 2]
[1, 3]
[1, 2, 4]
[1, 4]
[1, 5]
[1, 2, 6]
[1, 3, 6]
[1, 6]
[1, 7]
[2, 4]
[2, 6]
[3, 6]
PS D:\DISCRETAS\LAB3>
```

P15

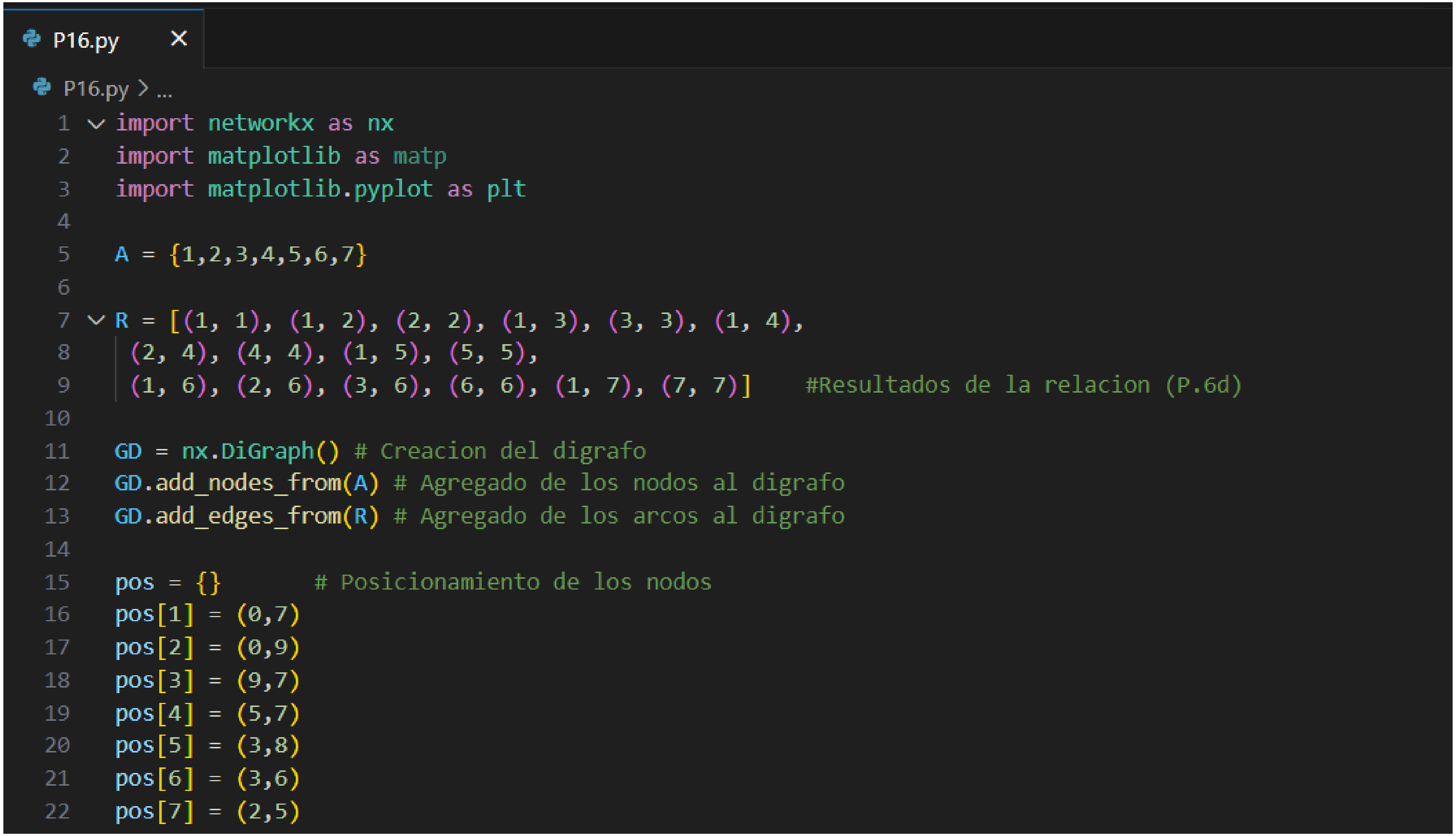


```python
def tray_long_n (lista_tray, n):
    n_trayec = []   #Lista vacia para guardar las trayectorias
    for i in lista_tray: # Ciclo para recorrer las posiciones
        if len(i) - 1 == n: # Determinar si la longitud es igual a n
            n_trayec.append(i) # Se agrega la trayectoria a la lista
    return n_trayec # Se retorna la lista


t = [['A', 'B'], ['A', 'D'], ['A', 'C', 'D'], ['B', 'C', 'D']]

# Lista de trayectorias


rest = tray_long_n(t,1) # Guardado de las trayectorias de longitud 1
rest2 = tray_long_n(t,2) # Guardado de las trayectorias de longitud 2

for i in rest: # Impresion de las trayectorias de longitud 1
    print("Trayectoria de longitud 1 (2 vertices 1 arco): ", i)

for i in rest2: # Impresion de las trayectorias de logitud 2
    print("Trayectoria de longitud 2 (3 vertices 2 arcos): ", i)
```

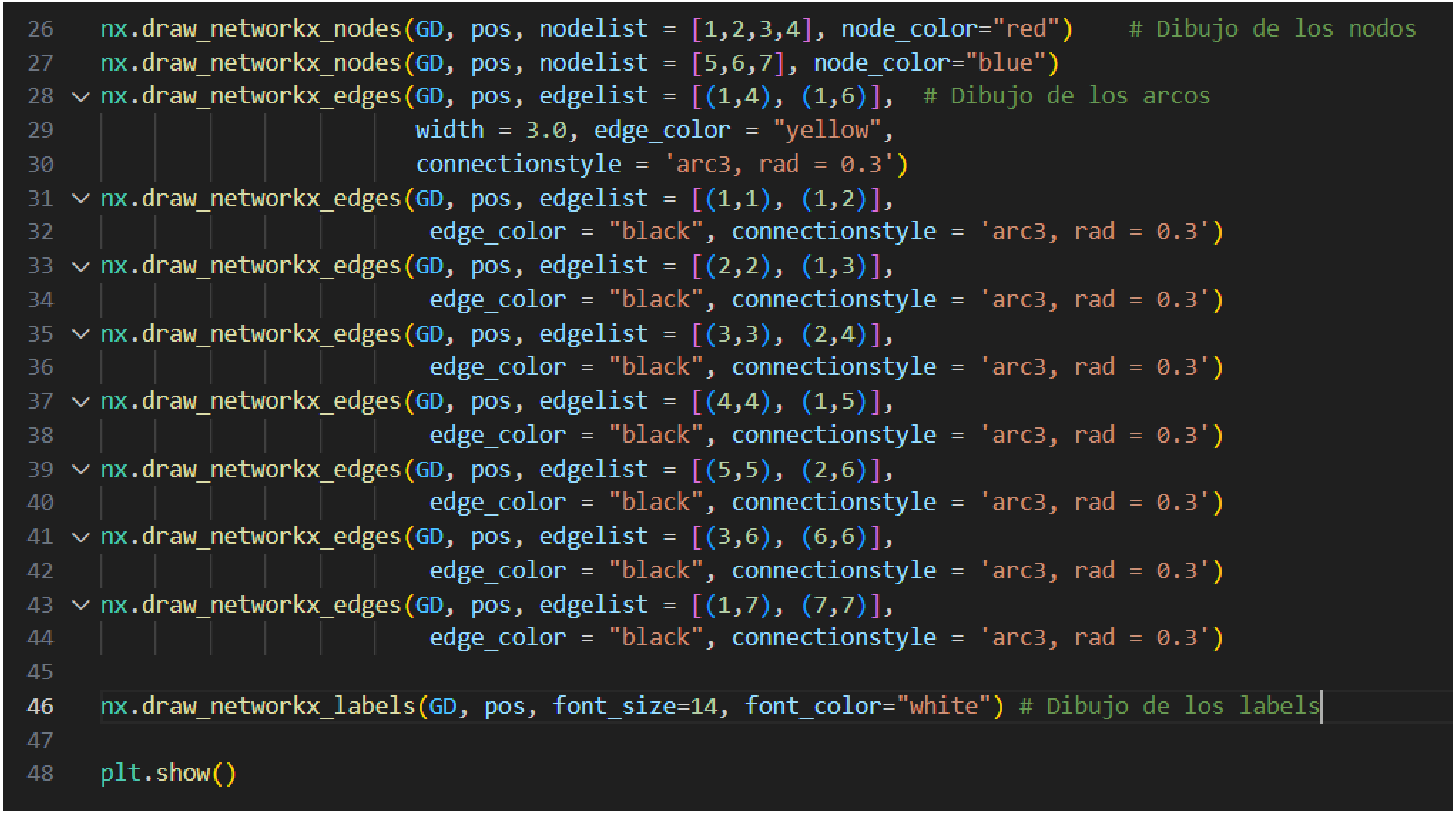


```
Trayectoria de longitud 2 (3 vertices 2 arcos):  ['B', 'C', 'D']
PS D:\DISCRETAS\LAB3>  d:; cd 'd:\DISCRETAS\LAB3'; & 'C:\Program Files\Python311\python.exe'
'c:\Users\USERJ\.vscode\extensions\ms-python.python-2023.8.0\pythonFiles\lib\python\debugpy\a
dapter/../..\debugpy\launcher' '52216' '--' 'd:\DISCRETAS\LAB3\P15.py'
Trayectoria de longitud 1 (2 vertices 1 arco):  ['A', 'B']
Trayectoria de longitud 1 (2 vertices 1 arco):  ['A', 'D']
Trayectoria de longitud 2 (3 vertices 2 arcos):  ['A', 'C', 'D']
Trayectoria de longitud 2 (3 vertices 2 arcos):  ['B', 'C', 'D']
PS D:\DISCRETAS\LAB3>
```

P16



```python
import networkx as nx
import matplotlib as matp
import matplotlib.pyplot as plt

A = {1,2,3,4,5,6,7}

R = [(1, 1), (1, 2), (2, 2), (1, 3), (3, 3), (1, 4),
 (2, 4), (4, 4), (1, 5), (5, 5),
 (1, 6), (2, 6), (3, 6), (6, 6), (1, 7), (7, 7)]    #Resultados de la relacion (P.6d)

GD = nx.DiGraph() # Creacion del digrafo
GD.add_nodes_from(A) # Agregado de los nodos al digrafo
GD.add_edges_from(R) # Agregado de los arcos al digrafo

pos = {}       # Posicionamiento de los nodos
pos[1] = (0,7)
pos[2] = (0,9)
pos[3] = (9,7)
pos[4] = (5,7)
pos[5] = (3,8)
pos[6] = (3,6)
pos[7] = (2,5)
```



```
nx.draw_networkx_nodes(GD, pos, nodelist = [1,2,3,4], node_color="red")    # Dibujo de los nodos
nx.draw_networkx_nodes(GD, pos, nodelist = [5,6,7], node_color="blue")
nx.draw_networkx_edges(GD, pos, edgelist = [(1,4), (1,6)],  # Dibujo de los arcos
                       width = 3.0, edge_color = "yellow",
                       connectionstyle = 'arc3, rad = 0.3')
nx.draw_networkx_edges(GD, pos, edgelist = [(1,1), (1,2)],
                        edge_color = "black", connectionstyle = 'arc3, rad = 0.3')
nx.draw_networkx_edges(GD, pos, edgelist = [(2,2), (1,3)],
                        edge_color = "black", connectionstyle = 'arc3, rad = 0.3')
nx.draw_networkx_edges(GD, pos, edgelist = [(3,3), (2,4)],
                        edge_color = "black", connectionstyle = 'arc3, rad = 0.3')
nx.draw_networkx_edges(GD, pos, edgelist = [(4,4), (1,5)],
                        edge_color = "black", connectionstyle = 'arc3, rad = 0.3')
nx.draw_networkx_edges(GD, pos, edgelist = [(5,5), (2,6)],
                        edge_color = "black", connectionstyle = 'arc3, rad = 0.3')
nx.draw_networkx_edges(GD, pos, edgelist = [(3,6), (6,6)],
                        edge_color = "black", connectionstyle = 'arc3, rad = 0.3')
nx.draw_networkx_edges(GD, pos, edgelist = [(1,7), (7,7)],
                        edge_color = "black", connectionstyle = 'arc3, rad = 0.3')

nx.draw_networkx_labels(GD, pos, font_size=14, font_color="white") # Dibujo de los labels

plt.show()
```

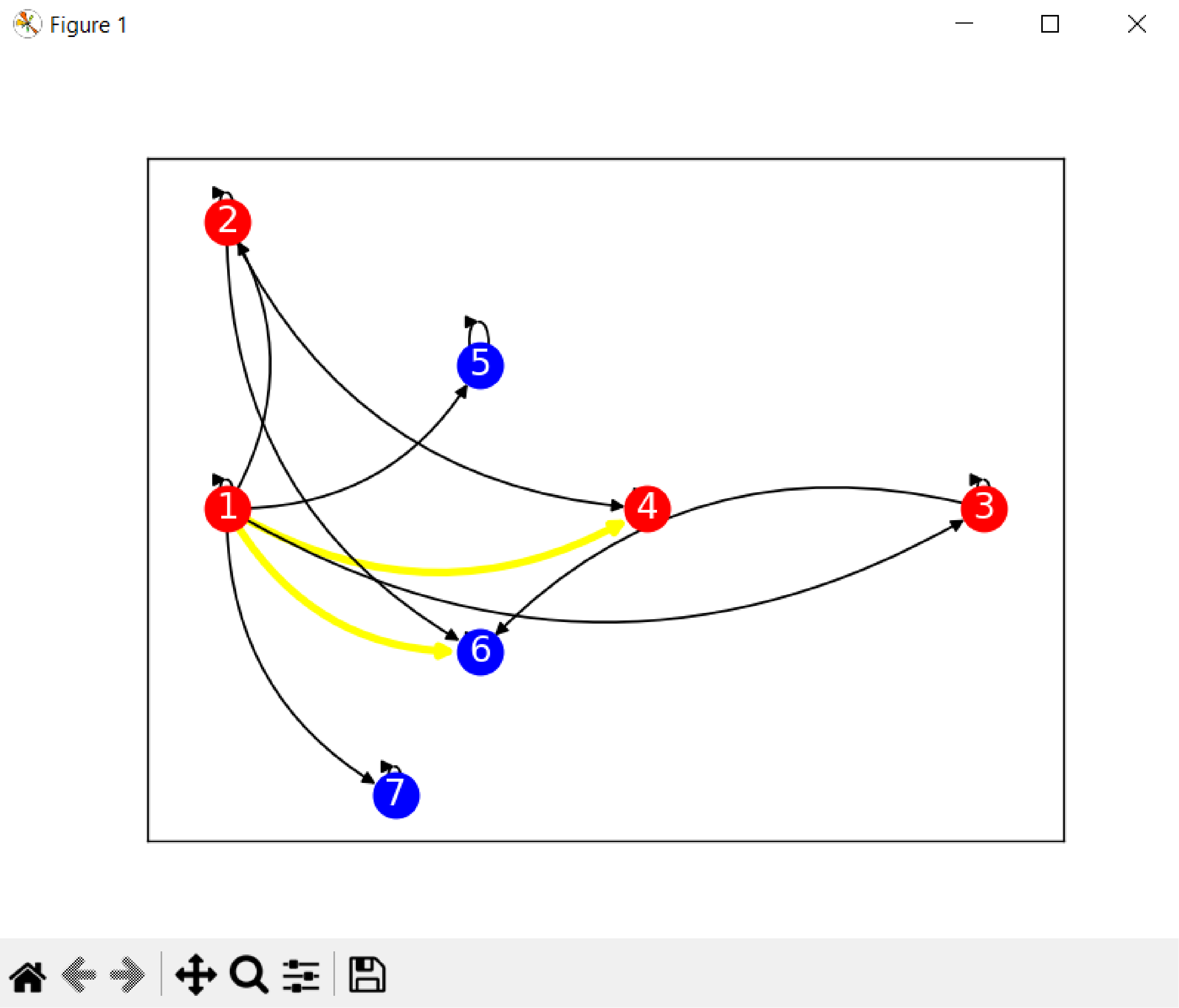
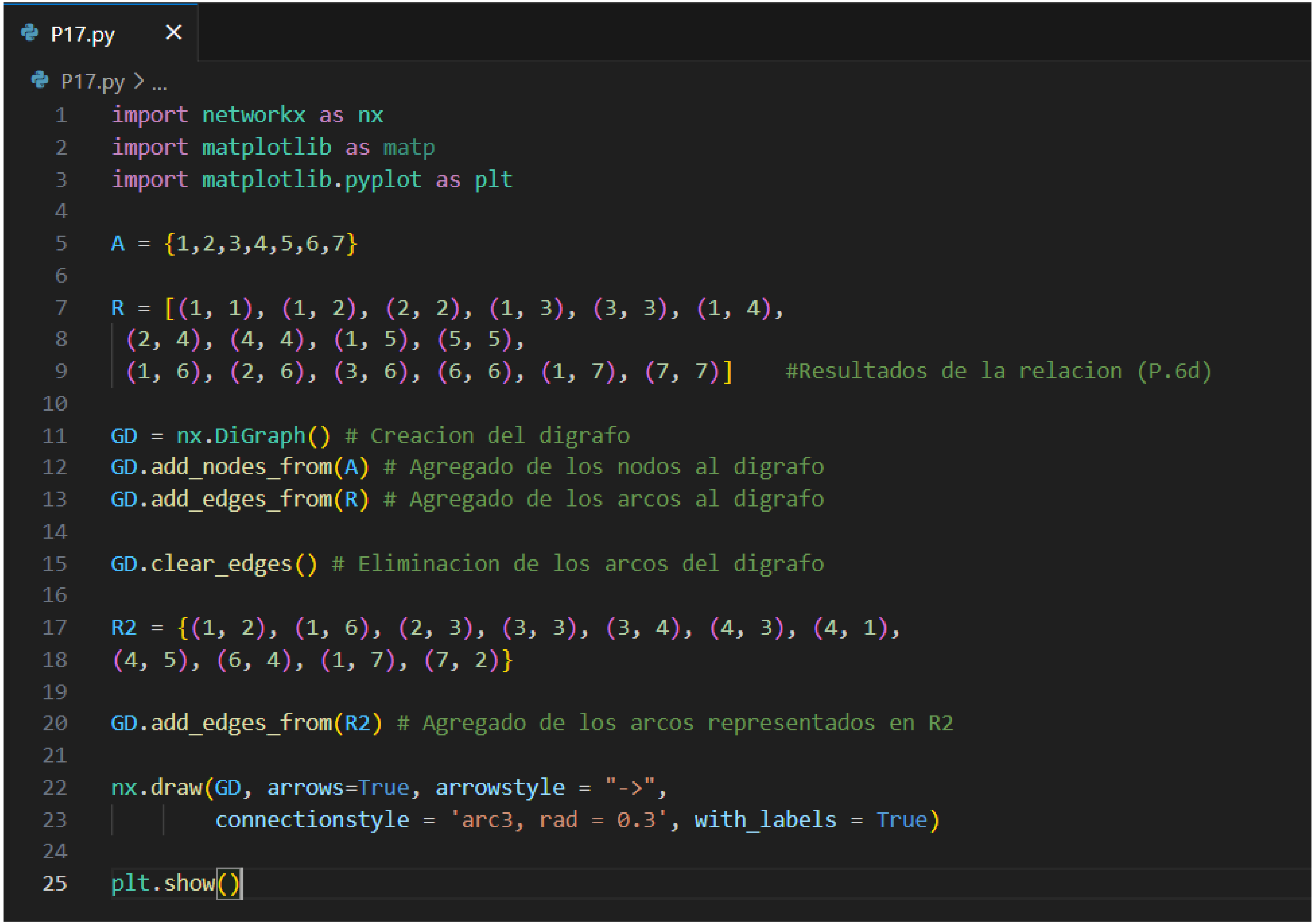


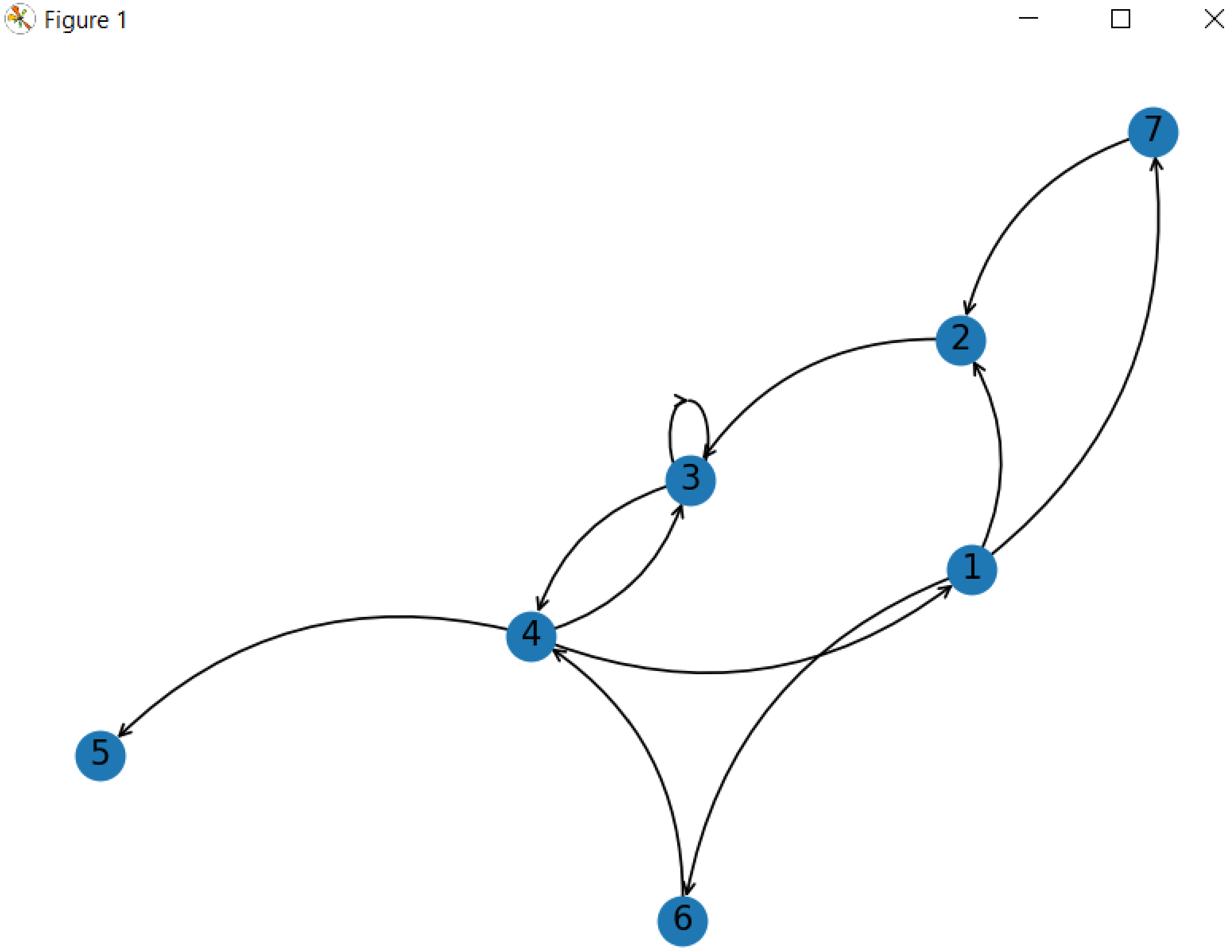
Figure 1
2
5
1
4
3
6
7

P17



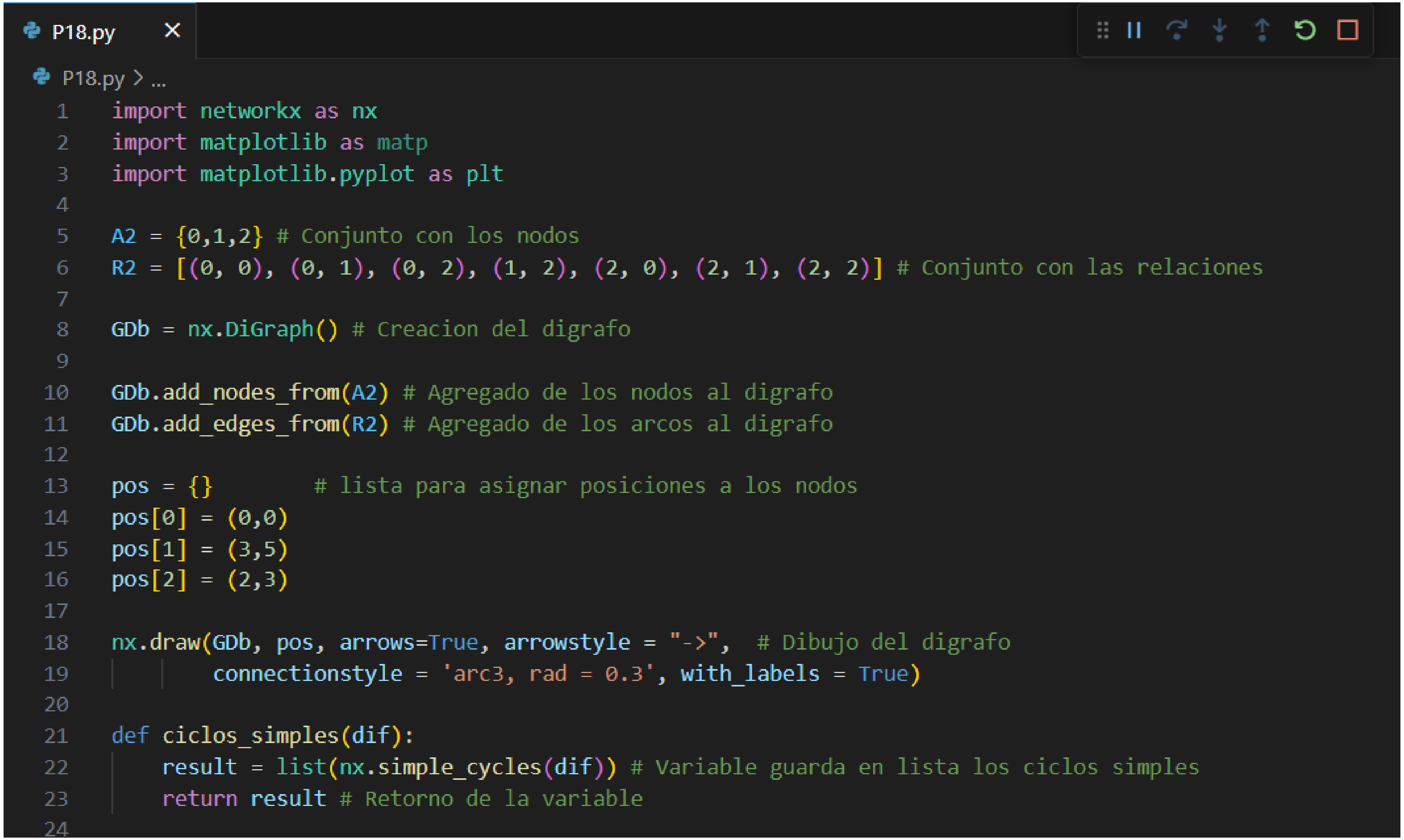
```python
import networkx as nx
import matplotlib as matp
import matplotlib.pyplot as plt

A = {1,2,3,4,5,6,7}

R = [(1, 1), (1, 2), (2, 2), (1, 3), (3, 3), (1, 4),
 (2, 4), (4, 4), (1, 5), (5, 5),
 (1, 6), (2, 6), (3, 6), (6, 6), (1, 7), (7, 7)]    #Resultados de la relacion (P.6d)

GD = nx.DiGraph() # Creacion del digrafo
GD.add_nodes_from(A) # Agregado de los nodos al digrafo
GD.add_edges_from(R) # Agregado de los arcos al digrafo

GD.clear_edges() # Eliminacion de los arcos del digrafo

R2 = {(1, 2), (1, 6), (2, 3), (3, 3), (3, 4), (4, 3), (4, 1),
(4, 5), (6, 4), (1, 7), (7, 2)}

GD.add_edges_from(R2) # Agregado de los arcos representados en R2

nx.draw(GD, arrows=True, arrowstyle = "->",
        connectionstyle = 'arc3, rad = 0.3', with_labels = True)

plt.show()
```

INFORME LAB #3 – E. DISCRETAS – JOY NELATON – 8-902-1282

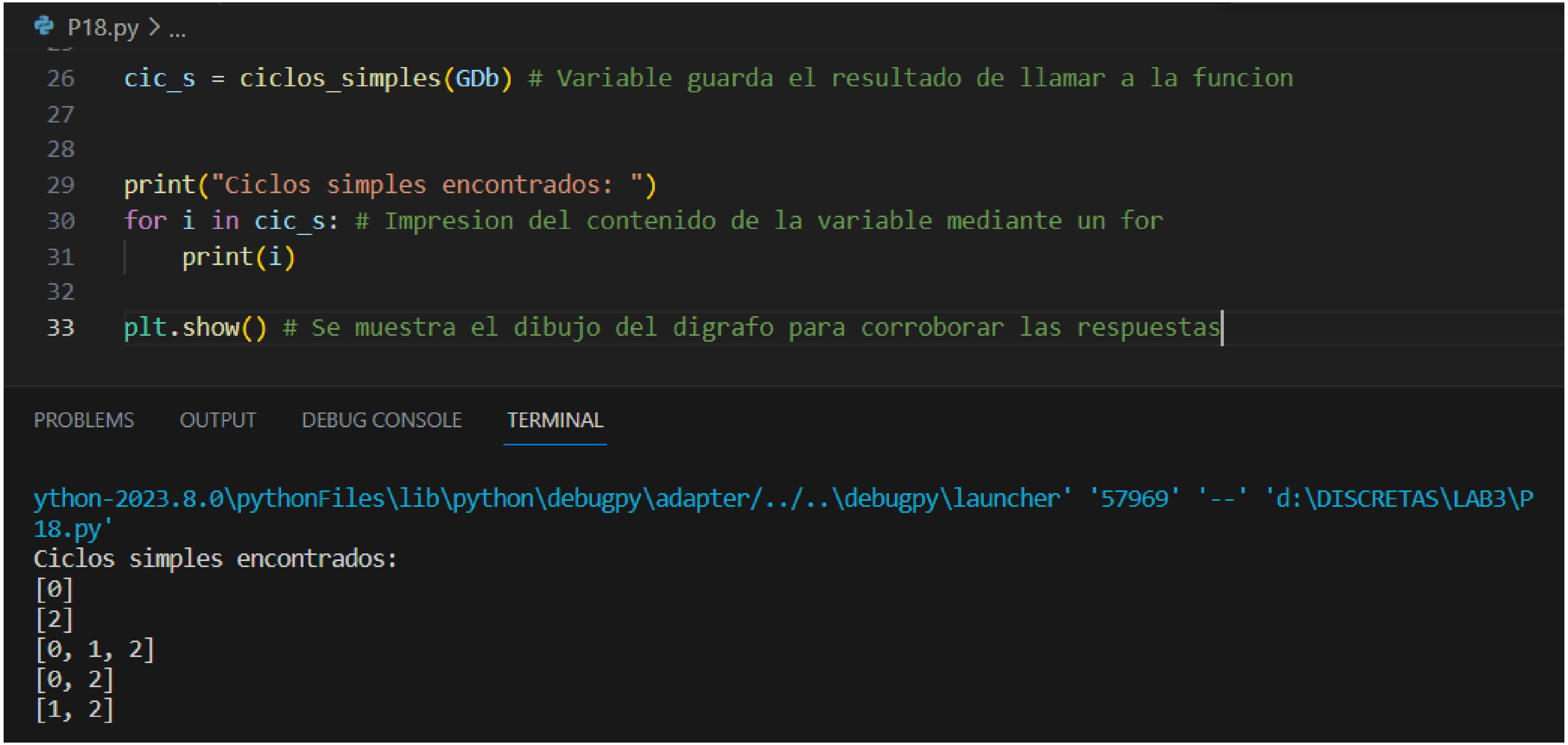


INFORME LAB #3 – E. DISCRETAS – JOY NELATON – 8-902-1282

P18



```python
import networkx as nx
import matplotlib as matp
import matplotlib.pyplot as plt

A2 = {0,1,2} # Conjunto con los nodos
R2 = [(0, 0), (0, 1), (0, 2), (1, 2), (2, 0), (2, 1), (2, 2)] # Conjunto con las relaciones

GDb = nx.DiGraph() # Creacion del digrafo

GDb.add_nodes_from(A2) # Agregado de los nodos al digrafo
GDb.add_edges_from(R2) # Agregado de los arcos al digrafo

pos = {}        # lista para asignar posiciones a los nodos
pos[0] = (0,0)
pos[1] = (3,5)
pos[2] = (2,3)

nx.draw(GDb, pos, arrows=True, arrowstyle = "->",  # Dibujo del digrafo
        connectionstyle = 'arc3, rad = 0.3', with_labels = True)

def ciclos_simples(dif):
    result = list(nx.simple_cycles(dif)) # Variable guarda en lista los ciclos simples
    return result # Retorno de la variable

```



```python
cic_s = ciclos_simples(GDb) # Variable guarda el resultado de llamar a la funcion


print("Ciclos simples encontrados: ")
for i in cic_s: # Impresion del contenido de la variable mediante un for
    print(i)

plt.show() # Se muestra el dibujo del digrafo para corroborar las respuestas
```

```
ython-2023.8.0\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '57969' '--' 'd:\DISCRETAS\LAB3\P
18.py'
Ciclos simples encontrados:
[0]
[2]
[0, 1, 2]
[0, 2]
[1, 2]
```

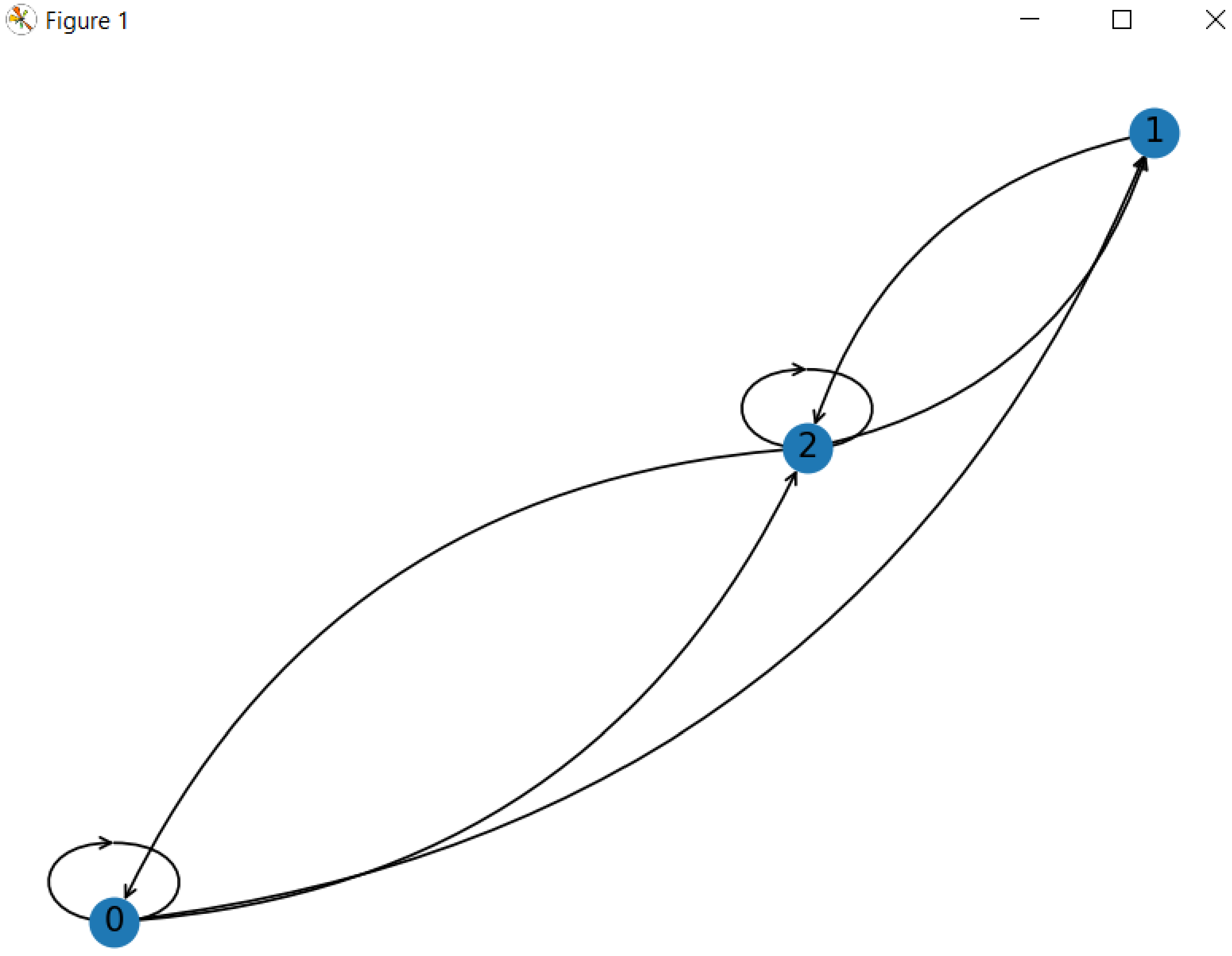
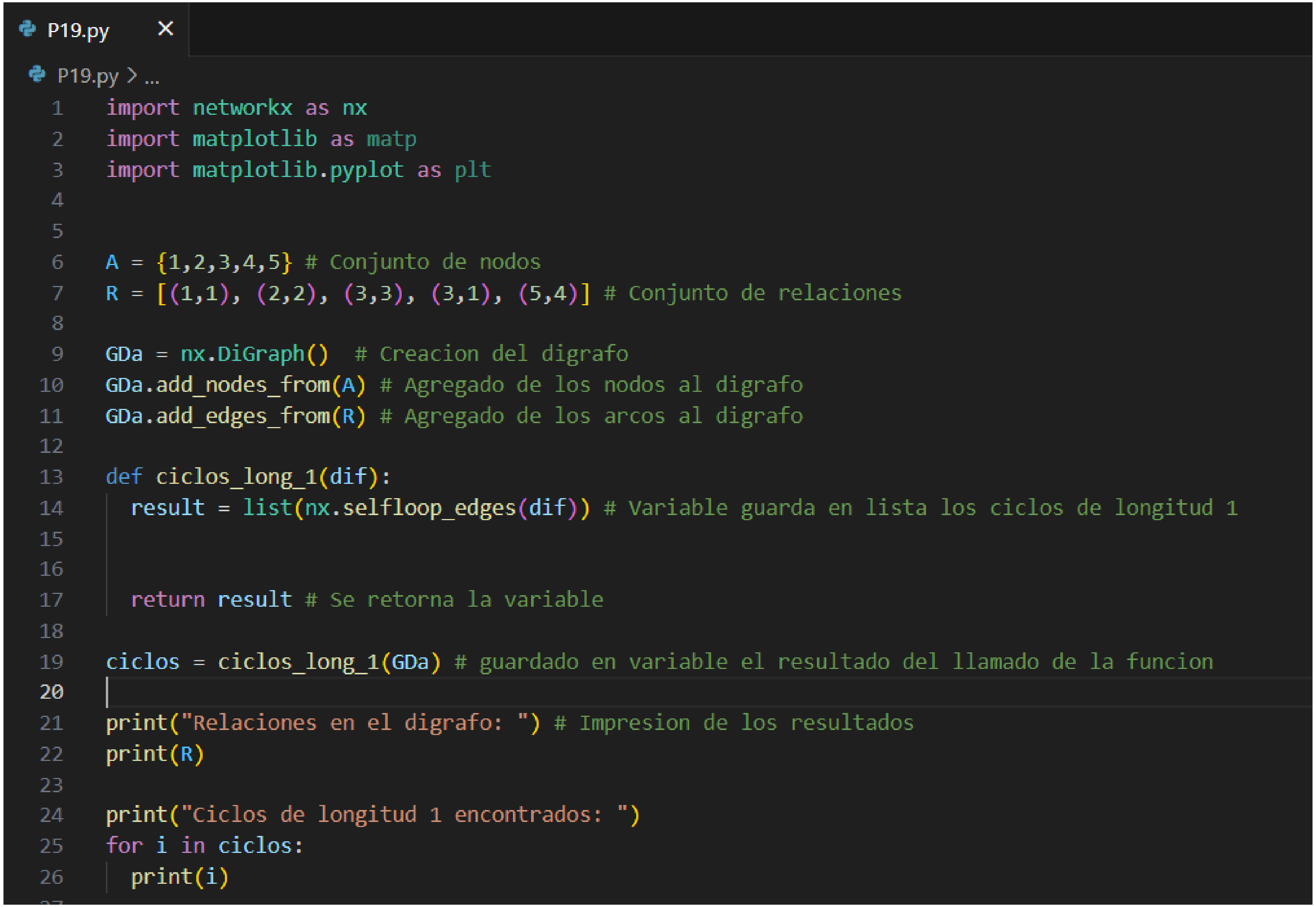


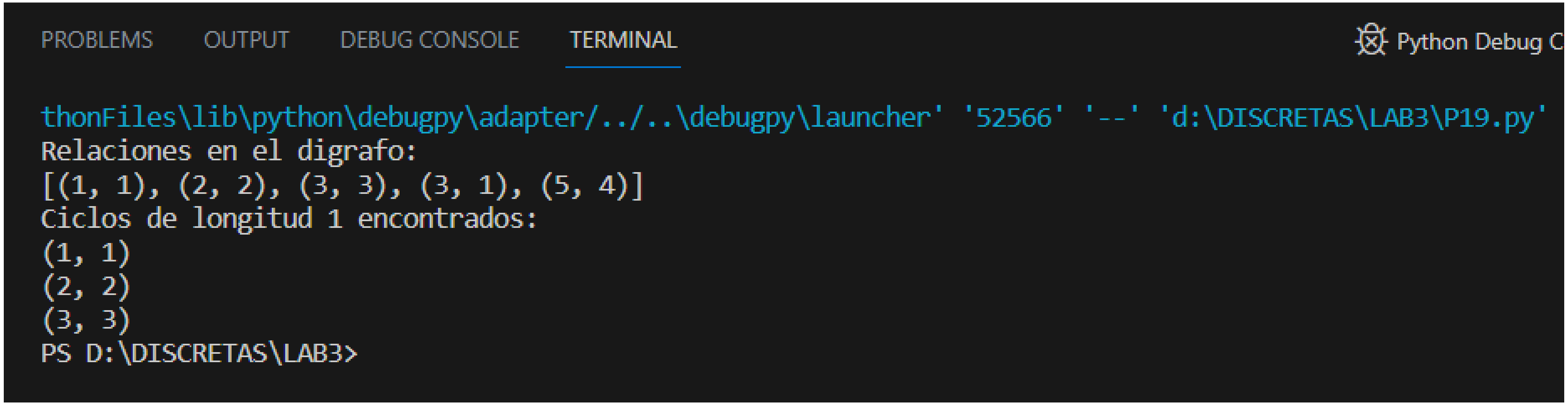
Figure 1
1
2
0

P19



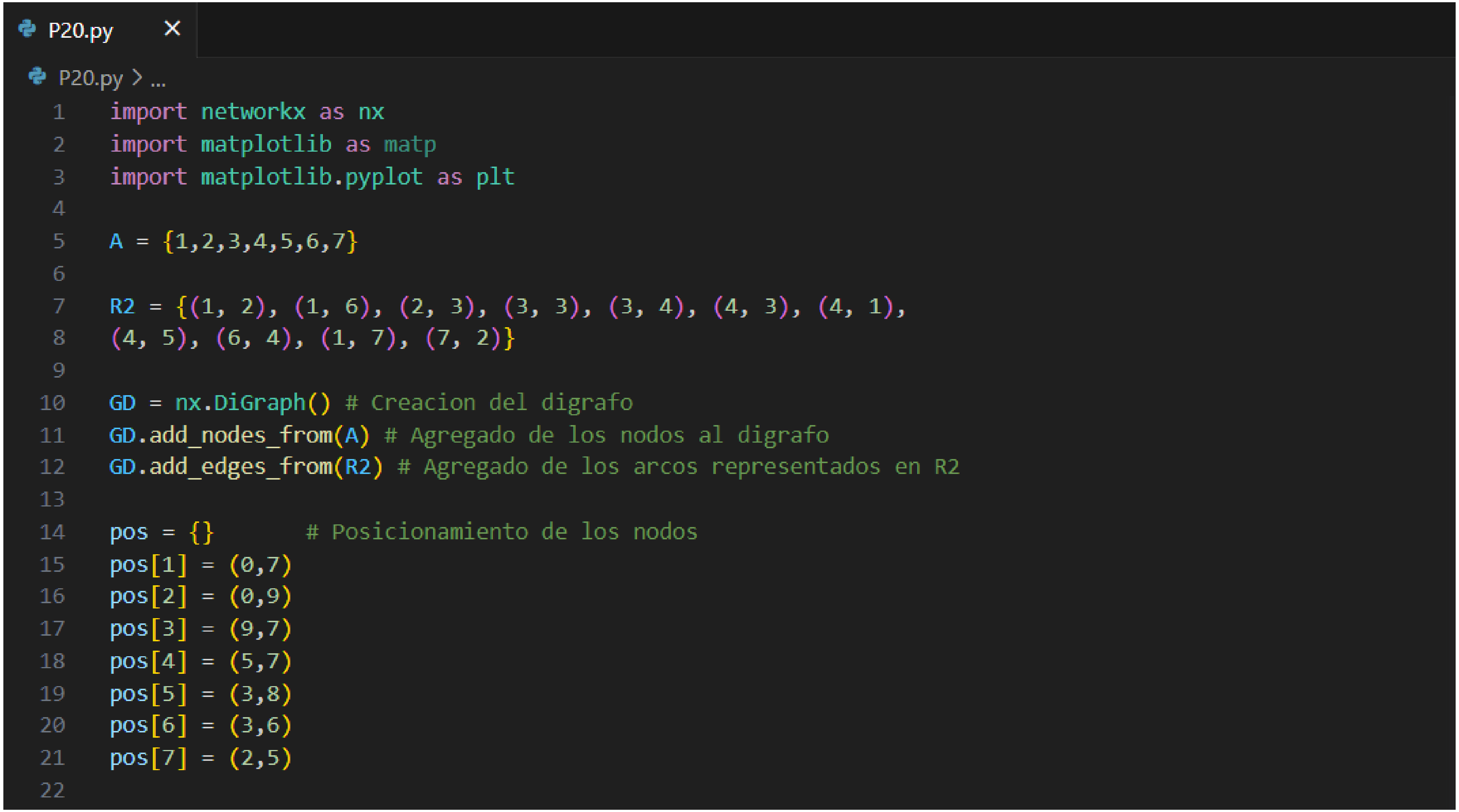
```python
import networkx as nx
import matplotlib as matp
import matplotlib.pyplot as plt


A = {1,2,3,4,5} # Conjunto de nodos
R = [(1,1), (2,2), (3,3), (3,1), (5,4)] # Conjunto de relaciones

GDa = nx.DiGraph()  # Creacion del digrafo
GDa.add_nodes_from(A) # Agregado de los nodos al digrafo
GDa.add_edges_from(R) # Agregado de los arcos al digrafo

def ciclos_long_1(dif):
    result = list(nx.selfloop_edges(dif)) # Variable guarda en lista los ciclos de longitud 1


    return result # Se retorna la variable

ciclos = ciclos_long_1(GDa) # guardado en variable el resultado del llamado de la funcion

print("Relaciones en el digrafo: ") # Impresion de los resultados
print(R)

print("Ciclos de longitud 1 encontrados: ")
for i in ciclos:
    print(i)
```

INFORME LAB #3 – E. DISCRETAS – JOY NELATON – 8-902-1282

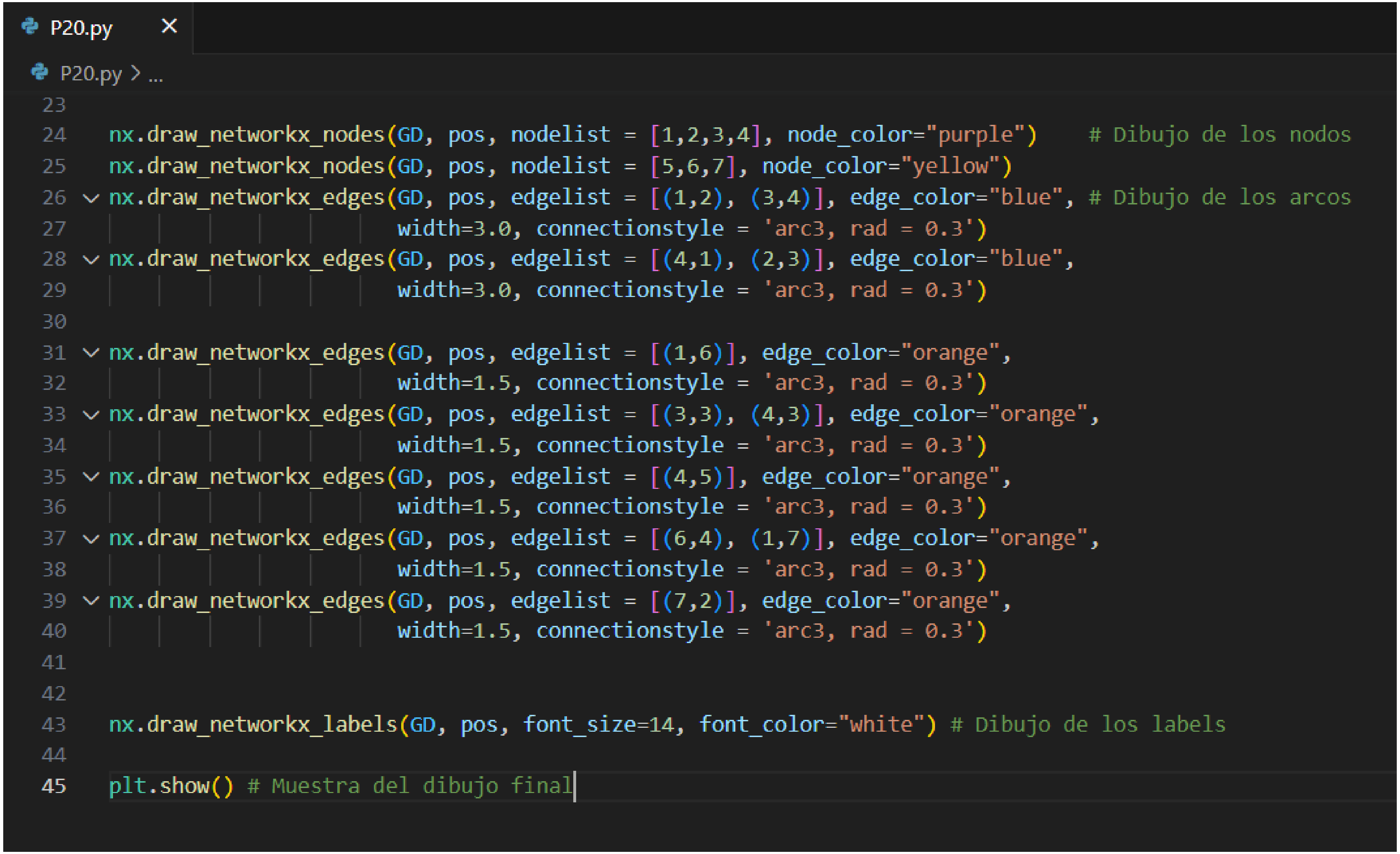


```
thonFiles\lib\python\debugpy\adapter/../..\debugpy\launcher' '52566' '--' 'd:\DISCRETAS\LAB3\P19.py'
Relaciones en el digrafo:
[(1, 1), (2, 2), (3, 3), (3, 1), (5, 4)]
Ciclos de longitud 1 encontrados:
(1, 1)
(2, 2)
(3, 3)
PS D:\DISCRETAS\LAB3>
```

P20

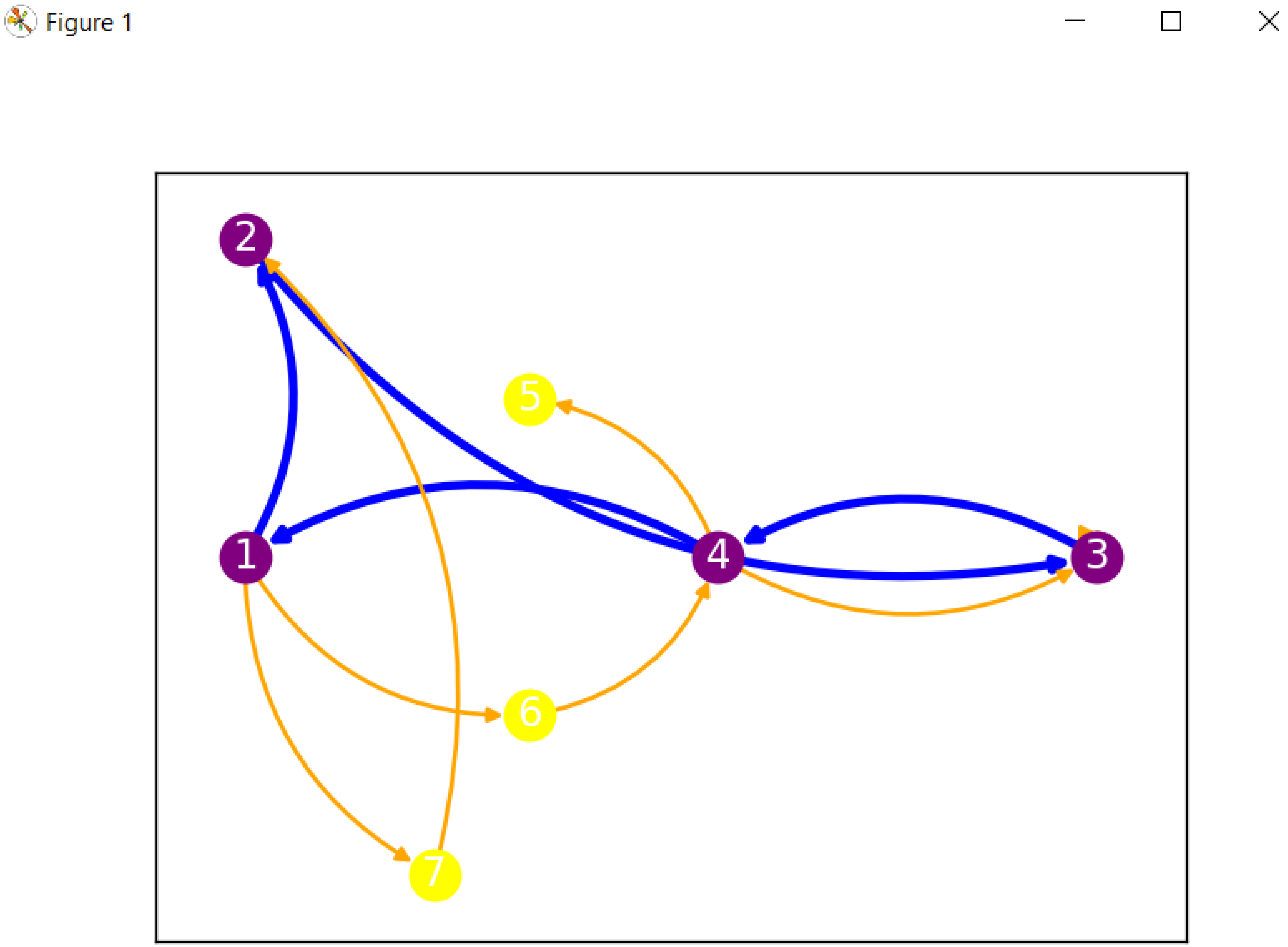


```python
import networkx as nx
import matplotlib as matp
import matplotlib.pyplot as plt

A = {1,2,3,4,5,6,7}

R2 = {(1, 2), (1, 6), (2, 3), (3, 3), (3, 4), (4, 3), (4, 1),
(4, 5), (6, 4), (1, 7), (7, 2)}

GD = nx.DiGraph() # Creacion del digrafo
GD.add_nodes_from(A) # Agregado de los nodos al digrafo
GD.add_edges_from(R2) # Agregado de los arcos representados en R2

pos = {}        # Posicionamiento de los nodos
pos[1] = (0,7)
pos[2] = (0,9)
pos[3] = (9,7)
pos[4] = (5,7)
pos[5] = (3,8)
pos[6] = (3,6)
pos[7] = (2,5)

```



```python

nx.draw_networkx_nodes(GD, pos, nodelist = [1,2,3,4], node_color="purple")    # Dibujo de los nodos
nx.draw_networkx_nodes(GD, pos, nodelist = [5,6,7], node_color="yellow")
nx.draw_networkx_edges(GD, pos, edgelist = [(1,2), (3,4)], edge_color="blue", # Dibujo de los arcos
                       width=3.0, connectionstyle = 'arc3, rad = 0.3')
nx.draw_networkx_edges(GD, pos, edgelist = [(4,1), (2,3)], edge_color="blue",
                       width=3.0, connectionstyle = 'arc3, rad = 0.3')

nx.draw_networkx_edges(GD, pos, edgelist = [(1,6)], edge_color="orange",
                       width=1.5, connectionstyle = 'arc3, rad = 0.3')
nx.draw_networkx_edges(GD, pos, edgelist = [(3,3), (4,3)], edge_color="orange",
                       width=1.5, connectionstyle = 'arc3, rad = 0.3')
nx.draw_networkx_edges(GD, pos, edgelist = [(4,5)], edge_color="orange",
                       width=1.5, connectionstyle = 'arc3, rad = 0.3')
nx.draw_networkx_edges(GD, pos, edgelist = [(6,4), (1,7)], edge_color="orange",
                       width=1.5, connectionstyle = 'arc3, rad = 0.3')
nx.draw_networkx_edges(GD, pos, edgelist = [(7,2)], edge_color="orange",
                       width=1.5, connectionstyle = 'arc3, rad = 0.3')


nx.draw_networkx_labels(GD, pos, font_size=14, font_color="white") # Dibujo de los labels

plt.show() # Muestra del dibujo final
```

INFORME LAB #3 – E. DISCRETAS – JOY NELATON – 8-902-1282



INFORME LAB #3 – E. DISCRETAS – JOY NELATON – 8-902-1282