

## P1

```
lab3_funciones.py  P1  X
P1 > ...
1  tup1 = (1,2)    # Asignacion de valores a la tup1
2
3  print(tup1)    # Impresion de la tup1
4
5  tup2 = ("metal", "hierro") # Asignacion de valores a la tup2
6
7  print(tup2) # Impresion de la tup2
8
9  tup3 = ("arbol", "hojas",7) # Asignacion de valores a la tup3
10
11 print(tup3) # Impresion de la tup3
12
13 subin1 = "\u2082" # Variables para alojar los subindices
14 subin2 = "\u2085"
15
16 t1 = subin1 + "P" + subin2 # Variable para concatenar los subindices y la letra
17
18 t2 = "5" + "\u00B2" # Variable para concatenar el 5 y el superindice
19
20 tup4 = (t1, t2) # Asignacion de variables a la tup4
21
22 print(tup4) # Impresion de la tup4
23
```

### 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-pyth
on.python-2023.6.1/pythonFiles/lib/python/debugpy/adapter/../../debugpy/launcher 32829 -- /home/userlub/D
ocumentos/LAB3/P1
(1, 2)
('metal', 'hierro')
('arbol', 'hojas', 7)
('₂P₅', '5²')
```

P2

```
P2 > ...
1  tup1 = (1,2)    #Asignacion de valores a las tuplas
2  tup2 = (1,2)
3  tup3 = ("a","b")
4
5
6
7  def verf_tup(t1,t2): #Funcion con las condiciones de evaluacion
8
9      if t1 == t2:    # Retorno de true si ambas tuplas son iguales
10         return True
11     else:
12         return False # Retorno de false si las tuplas son diferentes
13
14
15
16
17  print(verf_tup(tup1,tup2)) # Impresiones del llamado de la funcion
18
19  print(verf_tup(tup3, tup1))
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python Debug Console + - □ □ ... ^ ×

ocumentos/LAB3/P2  
True  
False

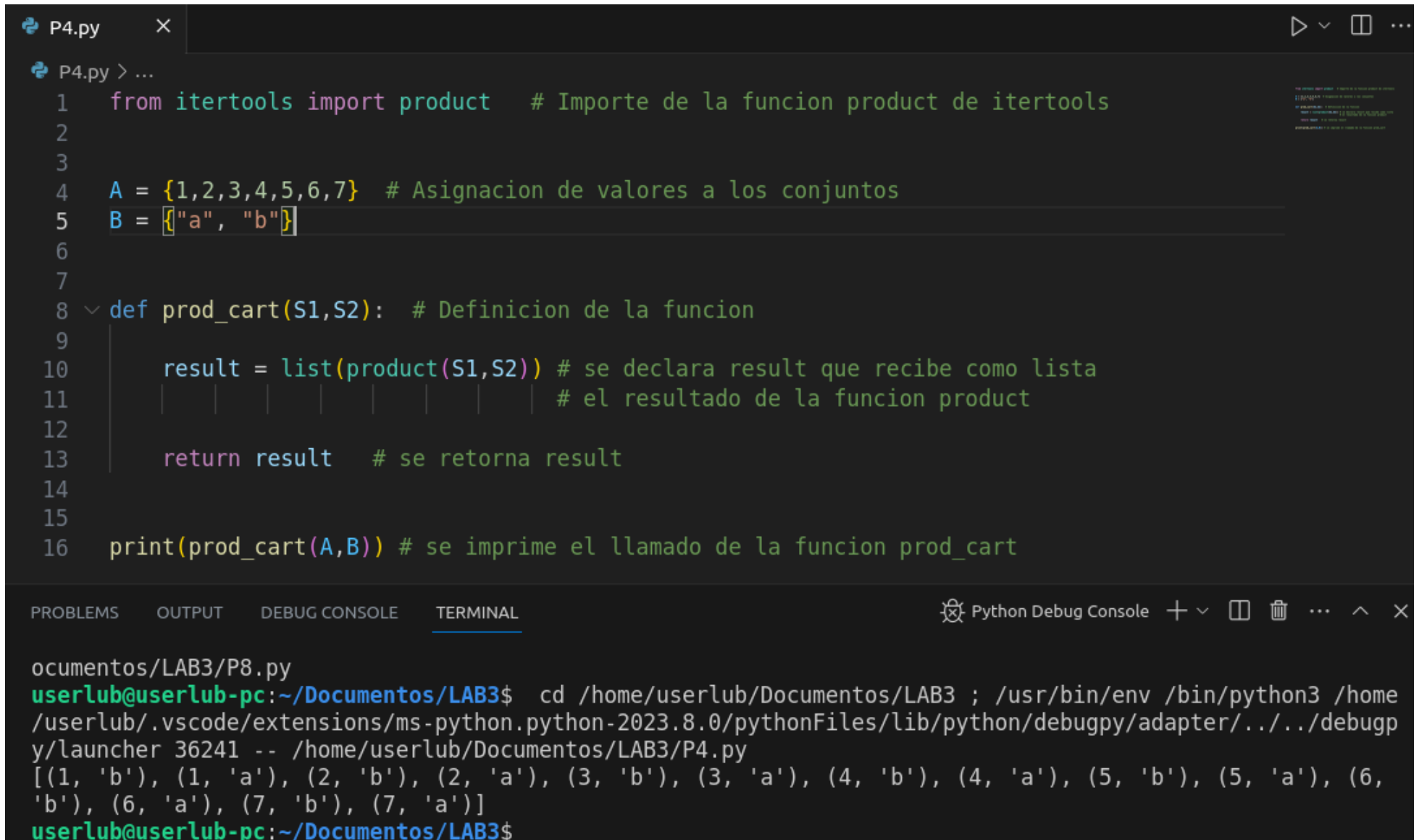
## P3

```
P3 > ...
1  # Asignacion de tuplas a listas
2
3  con_tub1 = [(67,20), (4,4,5), (24,84,75)]
4  con_tub2 = [("papel", "roca"), ("tijeras", "papel"), ("roca", "tijeras")]
5  con_tub3 = [(98, 89), ("a","b"), ("@", "#")]
6  con_tub4 = [(35, "agua"), ("suelo", 942, "***"), ("ropa")]
7  con_tub5 = [(1,2), (20,50), (1,4,8), (8,8)]
8
9
10 # Impresion de las listas con las tuplas
11
12 print(con_tub1)
13 print(con_tub2)
14 print(con_tub3)
15 print(con_tub4)
16 print(con_tub5)
17
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL   Python Debug C

```
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$
```

## P4



```
P4.py
P4.py > ...
1  from itertools import product  # Importe de la funcion product de itertools
2
3
4  A = {1,2,3,4,5,6,7}  # Asignacion de valores a los conjuntos
5  B = [{"a", "b"}]
6
7
8  def prod_cart(S1,S2):  # Definicion de la funcion
9
10     result = list(product(S1,S2))  # se declara result que recibe como lista
11     # el resultado de la funcion product
12
13     return result  # se retorna result
14
15
16  print(prod_cart(A,B))  # se imprime el llamado de la funcion prod_cart
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python Debug Console

```
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/adapters/../../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$
```

## P5

```
P5.py x
P5.py > ...
1 def verf_rel(X, Y, W): #Declaracion de la funcion
2     for el1 in X:      # Ciclos repetitivos para recorrer los elementos de X y Y
3         for el2 in Y:
4             if (el1, el2) in W: # Validar si los elementos de X y Y estan en W
5                 return True    # Retorno de true de encontrar coincidencia
6     return False # Retorno de false al salir de los ciclos de repeticion
7
8 A = {1, 2, 3, 4, 5, 6, 7} # Asignacion de valores a los conjuntos
9 B = {'a', 'b'}
10 C = {(1, 'a'), (2, 'b'), (3, 'a')}
11 D = {(1, 'h'), (2, 's')}
12
13
14 resultado = verf_rel(A, B, C) # Guardado de la ejecucion de la funcion en variables
15
16 resultado2 = verf_rel(A, B, D)
17
18 print(resultado) # Impresion de los resultados
19 print(resultado2)
20
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python Debug Console + - [ ] [X] ... ^

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

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

P6 a)

```
P6_a.py x
P6_a.py > ...
1  A = [1,2,3,4,5,6,7] #Asignacion de valores al conjunto A
2  R = [] #Creacion del conjunto vacio R para la relacion
3  tup_aux = () #Creacion de tupla para guardar los pares de la relacion
4
5
6
7  for i in A: #Ciclos for anidados para realizar los recorridos
8      for j in A:
9          if i == j: # Comparativa entre posiciones
10             tup_aux = (i,j) # Uso de la tupla para guardar el par si i=j
11             R.append(tup_aux) # La tupla se agrega al conjunto R
12         else:
13             tup_aux = () # La tupla se limpia en caso de que i no sea igual a j
14
15
16  print(R) # Se imprime el conjunto R
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python Debug Console + - [ ] [X] ... ^ X

```
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)

```
P6_b.py ×
P6_b.py > ...
1  A = [1,2,3,4,5,6,7] #Asignacion de valores al conjunto A
2  R = [] #Creacion del conjunto vacio R para la relacion
3  tup_aux = () #Creacion de tupla para guardar los pares de la relacion
4
5  for i in A:      #Ciclos for anidados para realizar los recorridos
6      for j in A:
7          if i<j:  # Comparativa entre posiciones
8              tup_aux = (i,j) # Uso de la tupla para guardar el par si i<j
9              R.append(tup_aux) # La tupla se agrega al conjunto R
10         else:
11             tup_aux = () # La tupla se limpia en caso de que i no sea menor a j
12
13
14
15
16  print(R) # Se imprime el conjunto R
17
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python Debug Console + - [ ] [X] ... ^ X

```
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 38207 -- /home/userlub/Documentos/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$
```



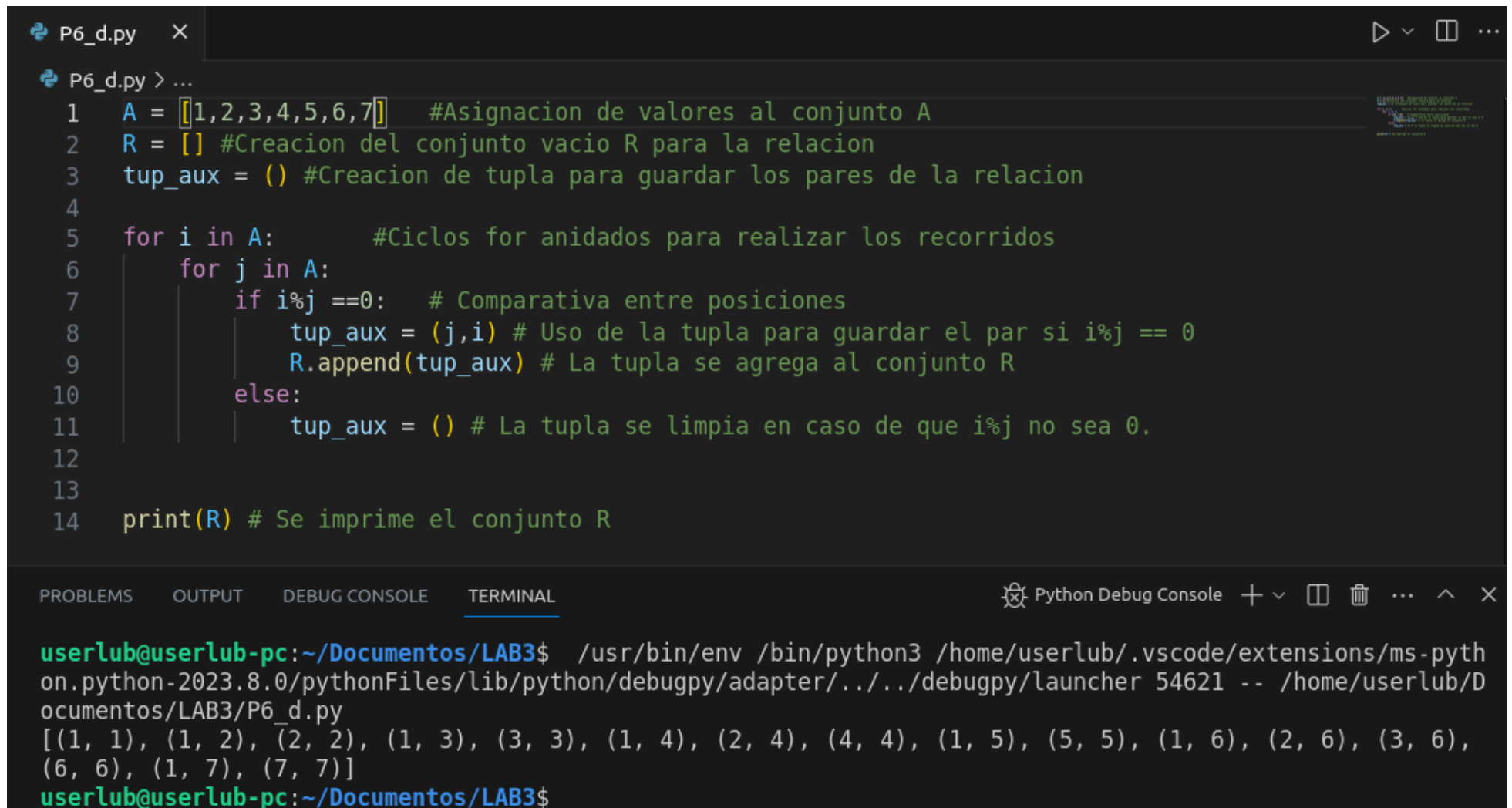
P6 c)

```
P6_c.py  X
P6_c.py > ...
1  A = [1,2,3,4,5,6,7] #Asignacion de valores al conjunto A
2  R = [] #Creacion del conjunto vacio R para la relacion
3  tup_aux = () #Creacion de tupla para guardar los pares de la relacion
4
5  for i in A:          #Ciclos for anidados para realizar los recorridos
6      for j in A:
7          if i<=j:      # Comparativa entre posiciones
8              tup_aux = (i,j) # Uso de la tupla para guardar el par si i<=j
9              R.append(tup_aux) # La tupla se agrega al conjunto R
10         else:
11             tup_aux = () # La tupla se limpia en caso de que i no sea menor o igual a j
12
13
14  print(R) # Se imprime el conjunto R
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python Debug Console + - [ ] [X] ... ^ X

```
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$
```

P6 d)



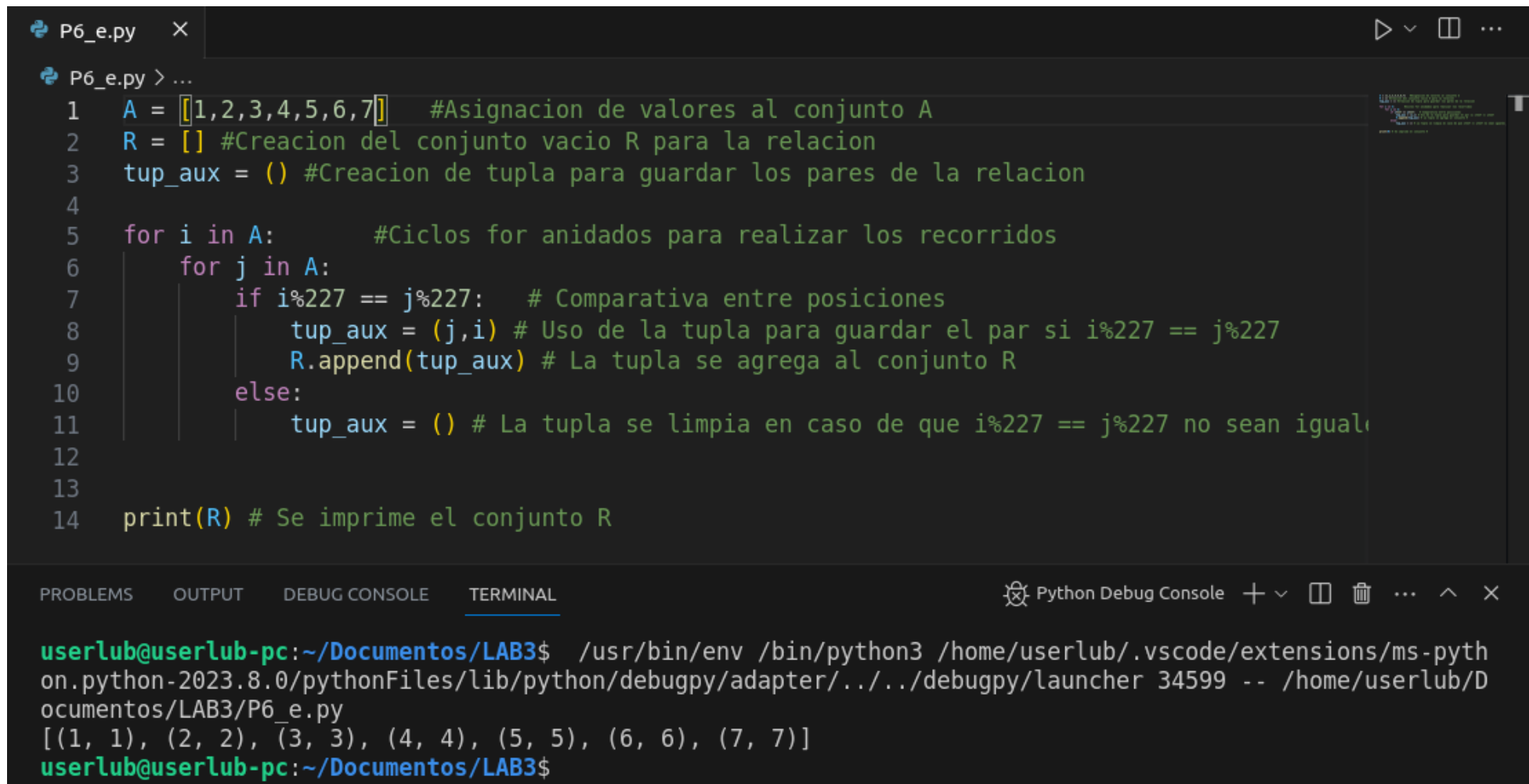
The image shows a VS Code editor window with a file named `P6_d.py`. The code is a Python script that creates a set `A` with values [1, 2, 3, 4, 5, 6, 7], an empty set `R`, and a tuple `tup_aux`. It then uses nested loops to iterate over `A` and `A` again, checking if `i % j == 0`. If true, it appends the tuple `(j, i)` to `R`. Otherwise, it resets `tup_aux` to an empty tuple. Finally, it prints the set `R`.

```
1 A = [1,2,3,4,5,6,7] #Asignacion de valores al conjunto A
2 R = [] #Creacion del conjunto vacio R para la relacion
3 tup_aux = () #Creacion de tupla para guardar los pares de la relacion
4
5 for i in A: #Ciclos for anidados para realizar los recorridos
6     for j in A:
7         if i%j ==0: # Comparativa entre posiciones
8             tup_aux = (j,i) # Uso de la tupla para guardar el par si i%j == 0
9             R.append(tup_aux) # La tupla se agrega al conjunto R
10        else:
11            tup_aux = () # La tupla se limpia en caso de que i%j no sea 0.
12
13
14 print(R) # Se imprime el conjunto R
```

The terminal output shows the command used to run the script and the resulting set `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)




The image shows a VS Code editor window with a file named `P6_e.py`. The code is a Python script that creates a set `A` with values `[1, 2, 3, 4, 5, 6, 7]`, an empty set `R`, and a tuple `tup_aux`. It then uses nested `for` loops to iterate over `A` and compare elements `i` and `j` using the modulo operation `i%227 == j%227`. If the condition is true, it appends the tuple `(j, i)` to `R`. Otherwise, it resets `tup_aux` to an empty tuple. Finally, it prints the set `R`.

```
1 A = [1,2,3,4,5,6,7] #Asignacion de valores al conjunto A
2 R = [] #Creacion del conjunto vacio R para la relacion
3 tup_aux = () #Creacion de tupla para guardar los pares de la relacion
4
5 for i in A: #Ciclos for anidados para realizar los recorridos
6     for j in A:
7         if i%227 == j%227: # Comparativa entre posiciones
8             tup_aux = (j,i) # Uso de la tupla para guardar el par si i%227 == j%227
9             R.append(tup_aux) # La tupla se agrega al conjunto R
10        else:
11            tup_aux = () # La tupla se limpia en caso de que i%227 == j%227 no sean iguales
12
13
14 print(R) # Se imprime el conjunto R
```

The terminal output shows the command to run the script and the resulting set `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$
```

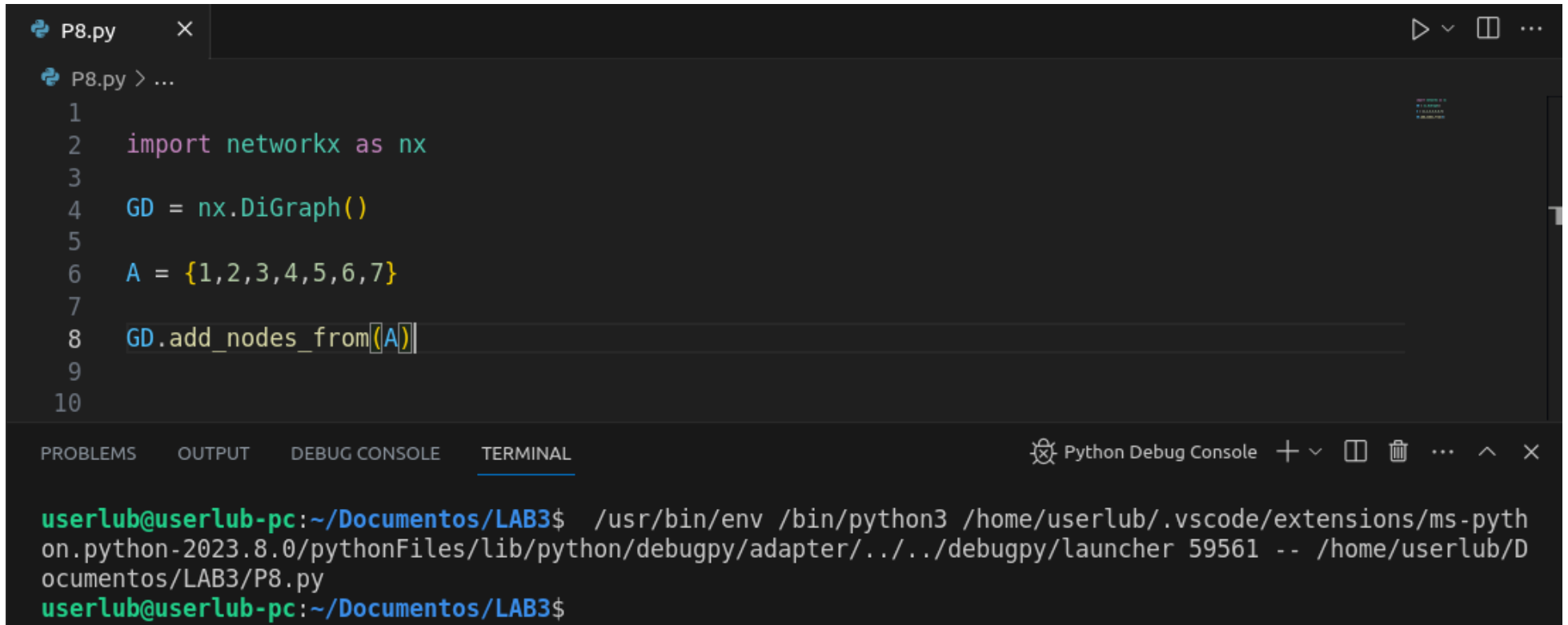
P7



```
P7.py ×  
P7.py > ...  
1 import networkx as nx  
2  
3 GD = nx.DiGraph()  
4  
5 |  
6  
7
```

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

### P8



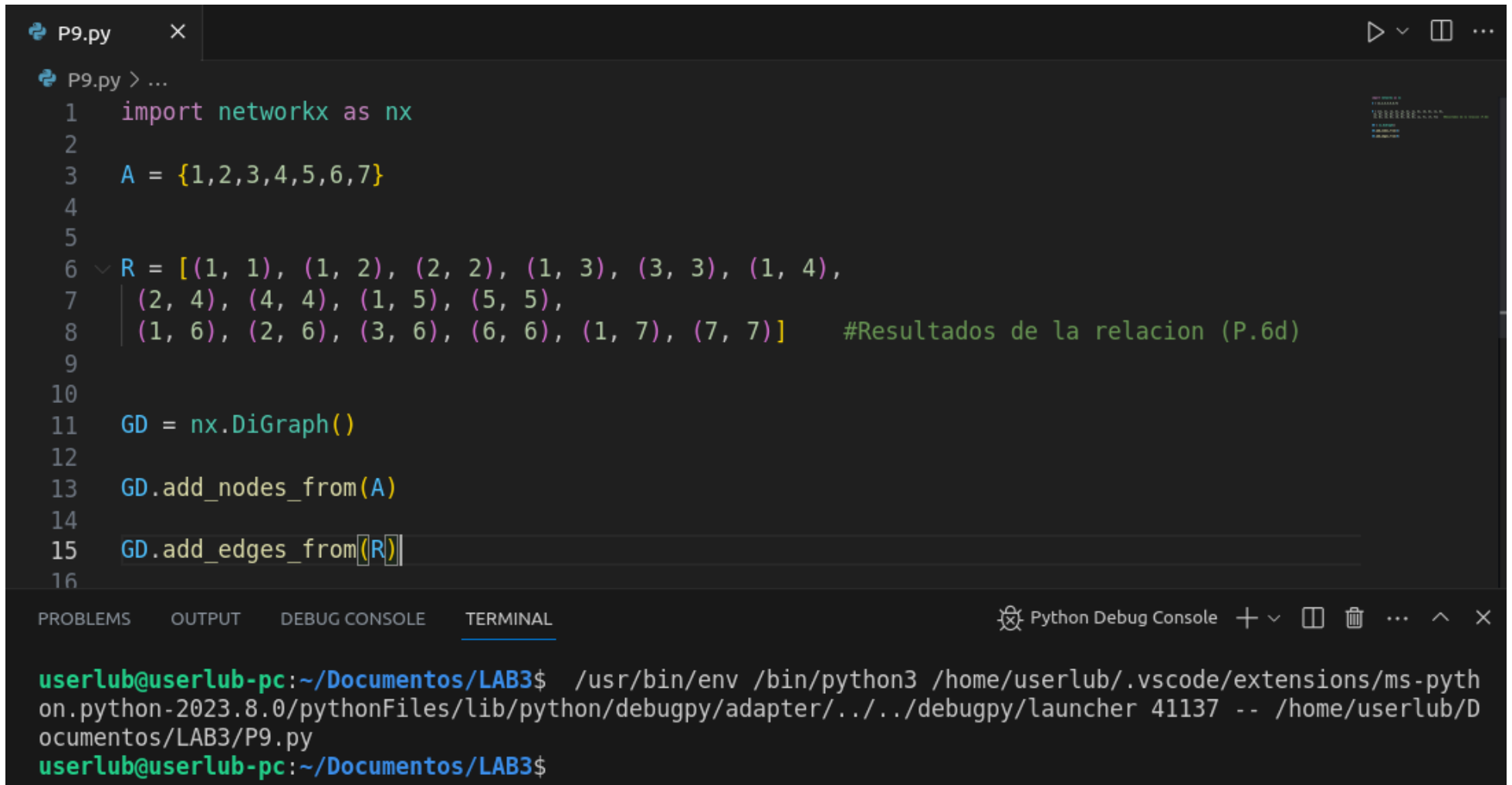
The image shows a screenshot of a Visual Studio Code editor window. The editor has a dark theme. At the top, there is a tab labeled 'P8.py' with a close button. Below the tab, the code in the editor is as follows:

```
1
2 import networkx as nx
3
4 GD = nx.DiGraph()
5
6 A = {1,2,3,4,5,6,7}
7
8 GD.add_nodes_from(A)
9
10
```

Below the editor, there is a panel with four tabs: 'PROBLEMS', 'OUTPUT', 'DEBUG CONSOLE', and 'TERMINAL'. The 'TERMINAL' tab is selected. The terminal shows the following command and output:

```
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$
```

## P9



The image shows a VS Code editor window with a file named P9.py. The code defines a set A and a relation R, then creates a directed graph GD and adds nodes and edges. The terminal shows the command to run the script, which is executed successfully.

```
P9.py
1 import networkx as nx
2
3 A = {1,2,3,4,5,6,7}
4
5
6 R = [(1, 1), (1, 2), (2, 2), (1, 3), (3, 3), (1, 4),
7      (2, 4), (4, 4), (1, 5), (5, 5),
8      (1, 6), (2, 6), (3, 6), (6, 6), (1, 7), (7, 7)] #Resultados de la relacion (P.6d)
9
10
11 GD = nx.DiGraph()
12
13 GD.add_nodes_from(A)
14
15 GD.add_edges_from(R)
16
```

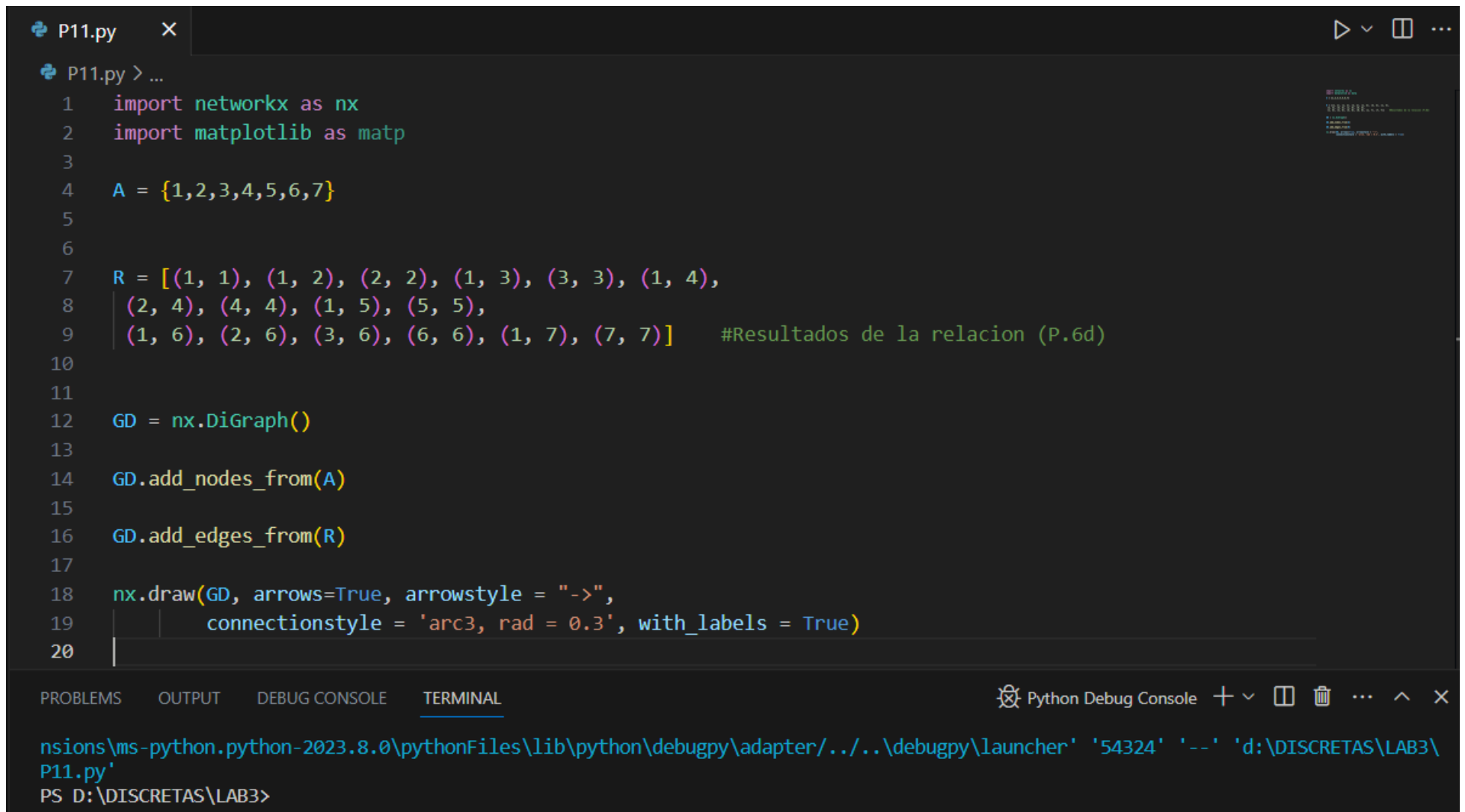
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python Debug Console

```
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 41137 -- /home/userlub/Documentos/LAB3/P9.py
userlub@userlub-pc:~/Documentos/LAB3$
```

## P10

```
P10.py ×
P10.py > ...
1  import networkx as nx
2
3  A = {1,2,3,4,5,6,7}
4
5
6  R = [(1, 1), (1, 2), (2, 2), (1, 3), (3, 3), (1, 4),
7       (2, 4), (4, 4), (1, 5), (5, 5),
8       (1, 6), (2, 6), (3, 6), (6, 6), (1, 7), (7, 7)]    #Resultados de la relacion (P.6d)
9
10
11 GD = nx.DiGraph()
12
13 GD.add_nodes_from(A)
14
15 GD.add_edges_from(R)
16
17
18 print("Numero de vertices: " , GD.number_of_nodes())
19
20 print("Numero de arcos: ", GD.number_of_edges())
```

## P11



```
P11.py X
P11.py > ...
1 import networkx as nx
2 import matplotlib as matp
3
4 A = {1,2,3,4,5,6,7}
5
6
7 R = [(1, 1), (1, 2), (2, 2), (1, 3), (3, 3), (1, 4),
8      (2, 4), (4, 4), (1, 5), (5, 5),
9      (1, 6), (2, 6), (3, 6), (6, 6), (1, 7), (7, 7)] #Resultados de la relacion (P.6d)
10
11
12 GD = nx.DiGraph()
13
14 GD.add_nodes_from(A)
15
16 GD.add_edges_from(R)
17
18 nx.draw(GD, arrows=True, arrowstyle = "->",
19         connectionstyle = 'arc3, rad = 0.3', with_labels = True)
20
```

PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python Debug Console + - □ □ ... ^ X

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



## P12

P12.py

```
8 R = [(1, 1), (1, 2), (2, 2), (1, 3), (3, 3),  
9      (2, 4), (4, 4), (1, 5), (5, 5),  
10     (1, 6), (2, 6), (3, 6), (6, 6), (1, 7), (7,  
11  
12  
13 GD = nx.DiGraph()  
14  
15 GD.add_nodes_from(A)  
16  
17 GD.add_edges_from(R)  
18  
19 nx.draw(GD, arrows=True, arrowstyle="->",  
20         connectionstyle='arc3, rad = 0.3',  
21  
22 plt.show()  
23  
24
```

Figure 1

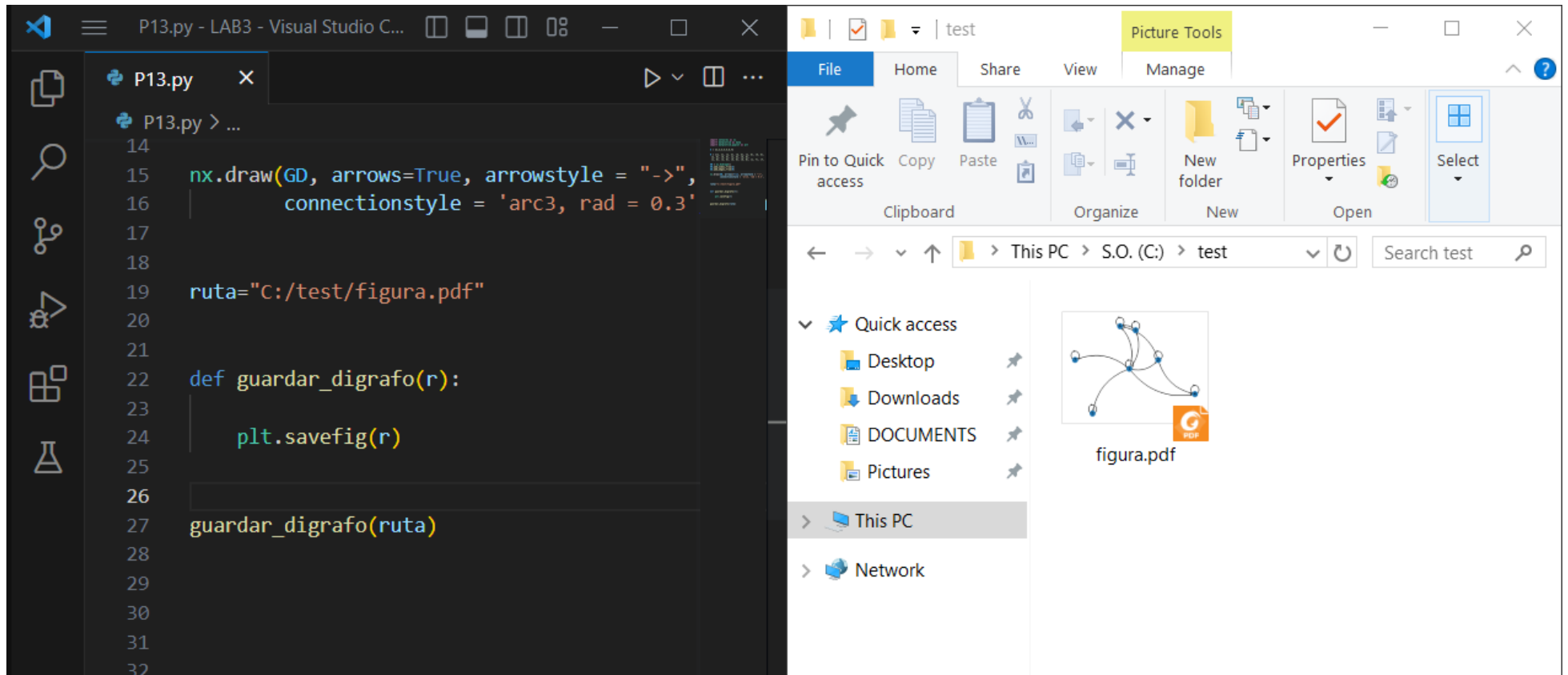
```
graph TD
    1((1)) --> 2((2))
    1((1)) --> 3((3))
    1((1)) --> 4((4))
    1((1)) --> 5((5))
    1((1)) --> 6((6))
    1((1)) --> 7((7))
    2((2)) --> 2((2))
    3((3)) --> 3((3))
    4((4)) --> 4((4))
    5((5)) --> 5((5))
    6((6)) --> 1((1))
    7((7)) --> 7((7))
```

Python Debug Console

```
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'
```

Ln 23, Col 1 Spaces: 4 UTF-8 CRLF Python 3.11.4 64-bit

## P13



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

P14

```
P14.py X
P14.py > ...
1  import networkx as nx
2  import matplotlib as matp
3  import matplotlib.pyplot as plt
4
5  A = {1,2,3,4,5,6,7}
6
7  R = [(1, 1), (1, 2), (2, 2), (1, 3), (3, 3), (1, 4),
8      (2, 4), (4, 4), (1, 5), (5, 5),
9      (1, 6), (2, 6), (3, 6), (6, 6), (1, 7), (7, 7)]    #Resultados de la relacion (P.6d)
10
11  GD = nx.DiGraph()
12  GD.add_nodes_from(A)
13  GD.add_edges_from(R)
14
15  nx.draw(GD, arrows=True, arrowstyle = "->",
16      |      connectionstyle = 'arc3, rad = 0.3', with_labels = True)
17
18
```

```
def obtener_trayectorias(graf):  
    trayec = [] # Lista vacia para almacenar las trayectorias  
    nodos = list(graf.nodos) # 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\extensions\ms-python.python-2023.8.0\pythonFiles\lib\python\debugpy\adapter\..\..\debugpy\launcher' '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

```
P15.py X
P15.py > ...
1 def tray_long_n (lista_tray, n):
2     n_trayec = [] #Lista vacia para guardar las trayectorias
3     for i in lista_tray: # Ciclo para recorrer las posiciones
4         if len(i) - 1 == n: # Determinar si la longitud es igual a n
5             n_trayec.append(i) # Se agrega la trayectoria a la lista
6     return n_trayec # Se retorna la lista
7
8
9 t = [['A', 'B'], ['A', 'D'], ['A', 'C', 'D'], ['B', 'C', 'D']]
10
11 # Lista de trayectorias
12
13
14 rest = tray_long_n(t,1) # Guardado de las trayectorias de longitud 1
15 rest2 = tray_long_n(t,2) # Guardado de las trayectorias de longitud 2
16
17 for i in rest: # Impresion de las trayectorias de longitud 1
18     print("Trayectoria de longitud 1 (2 vertices 1 arco): ", i)
19
20 for i in rest2: # Impresion de las trayectorias de longitud 2
21     print("Trayectoria de longitud 2 (3 vertices 2 arcos): ", i)
```

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

```
PROBLEMS OUTPUT DEBUG CONSOLE TERMINAL Python Debug Console + v [ ] [X] ... ^ X

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\ad
apter/../../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>
```

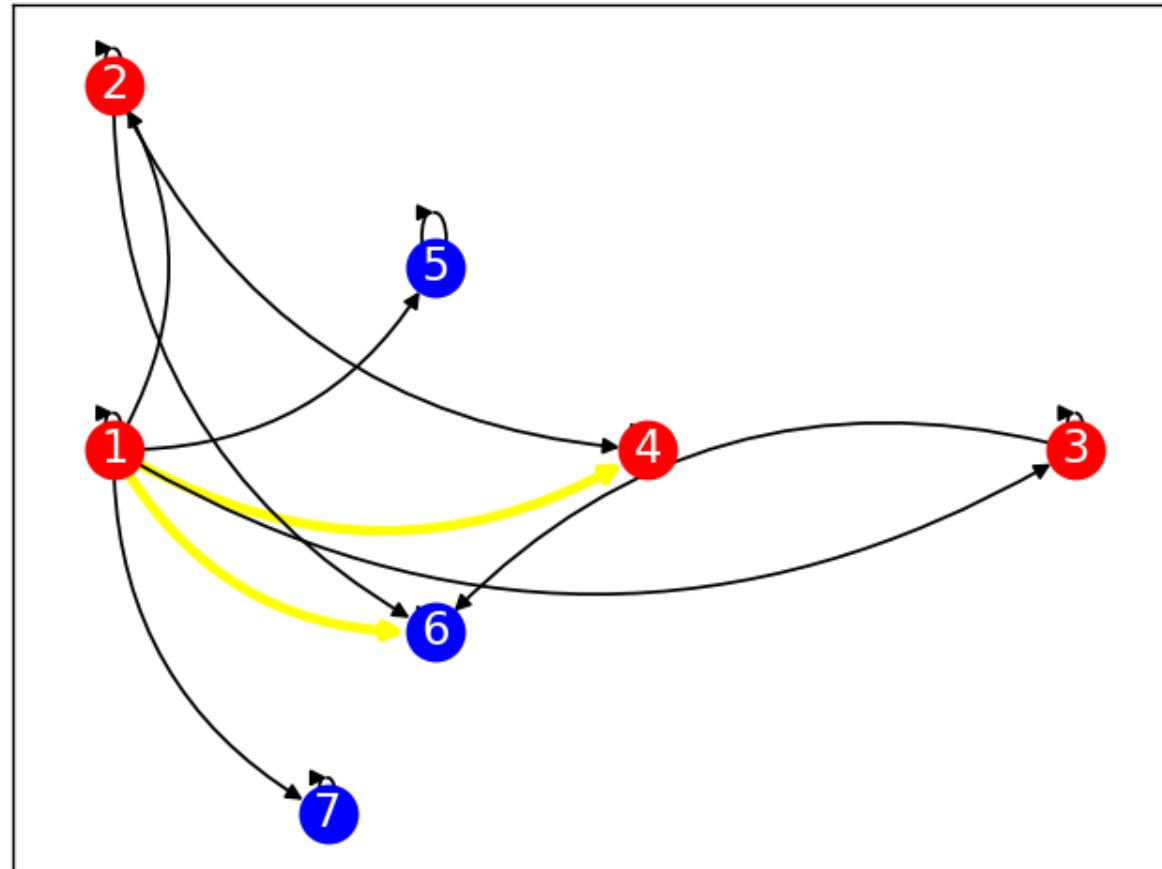
## P16

```
P16.py X
P16.py > ...
1  import networkx as nx
2  import matplotlib as matp
3  import matplotlib.pyplot as plt
4
5  A = {1,2,3,4,5,6,7}
6
7  R = [(1, 1), (1, 2), (2, 2), (1, 3), (3, 3), (1, 4),
8      (2, 4), (4, 4), (1, 5), (5, 5),
9      (1, 6), (2, 6), (3, 6), (6, 6), (1, 7), (7, 7)]    #Resultados de la relacion (P.6d)
10
11  GD = nx.DiGraph() # Creacion del digrafo
12  GD.add_nodes_from(A) # Agregado de los nodos al digrafo
13  GD.add_edges_from(R) # Agregado de los arcos al digrafo
14
15  pos = {}          # Posicionamiento de los nodos
16  pos[1] = (0,7)
17  pos[2] = (0,9)
18  pos[3] = (9,7)
19  pos[4] = (5,7)
20  pos[5] = (3,8)
21  pos[6] = (3,6)
22  pos[7] = (2,5)
```





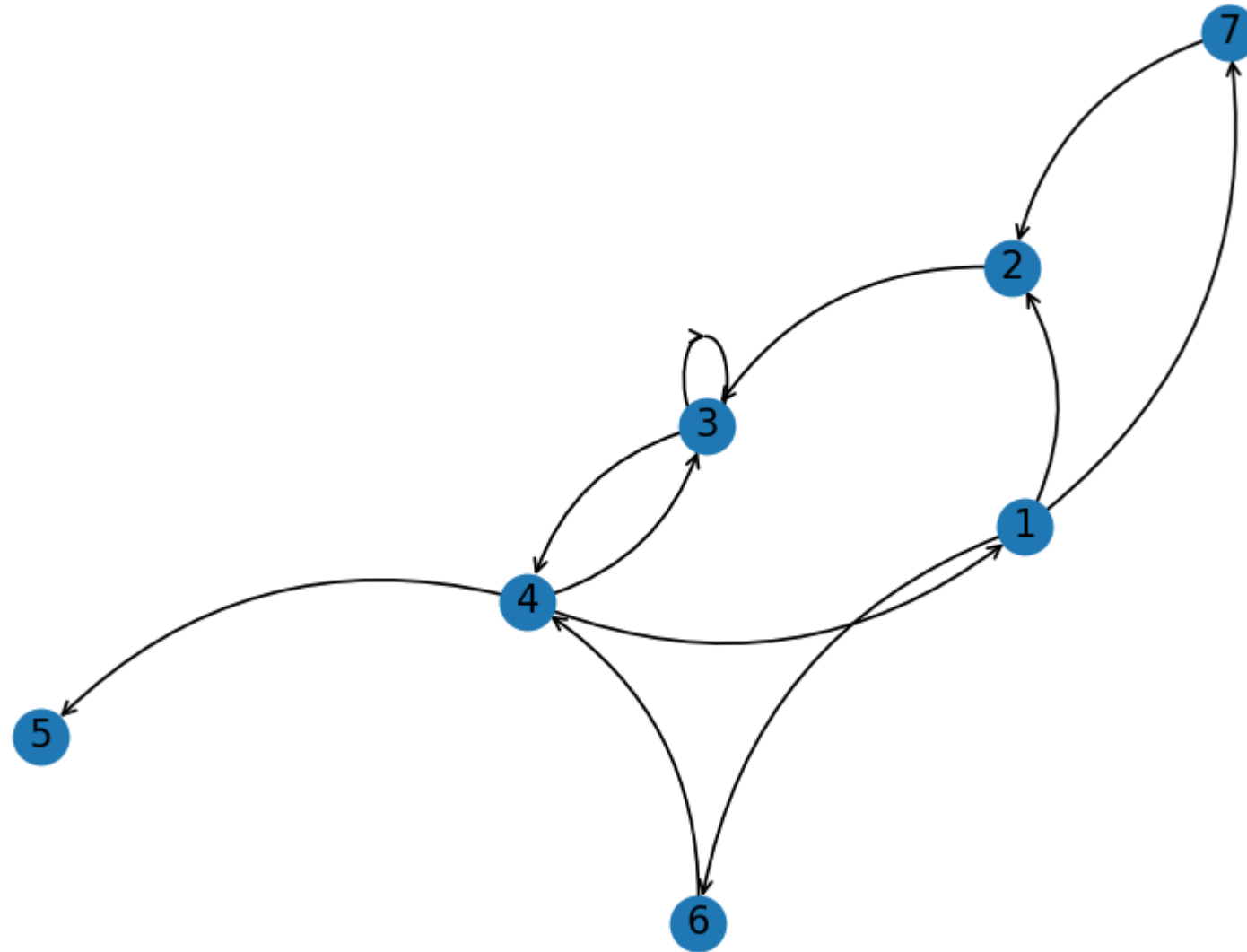
Figure 1



## P17

```
P17.py X
P17.py > ...
1  import networkx as nx
2  import matplotlib as matp
3  import matplotlib.pyplot as plt
4
5  A = {1,2,3,4,5,6,7}
6
7  R = [(1, 1), (1, 2), (2, 2), (1, 3), (3, 3), (1, 4),
8       (2, 4), (4, 4), (1, 5), (5, 5),
9       (1, 6), (2, 6), (3, 6), (6, 6), (1, 7), (7, 7)]    #Resultados de la relacion (P.6d)
10
11  GD = nx.DiGraph() # Creacion del digrafo
12  GD.add_nodes_from(A) # Agregado de los nodos al digrafo
13  GD.add_edges_from(R) # Agregado de los arcos al digrafo
14
15  GD.clear_edges() # Eliminacion de los arcos del digrafo
16
17  R2 = {(1, 2), (1, 6), (2, 3), (3, 3), (3, 4), (4, 3), (4, 1),
18        (4, 5), (6, 4), (1, 7), (7, 2)}
19
20  GD.add_edges_from(R2) # Agregado de los arcos representados en R2
21
22  nx.draw(GD, arrows=True, arrowstyle = "->",
23         connectionstyle = 'arc3, rad = 0.3', with_labels = True)
24
25  plt.show()
```

Figure 1



## P18

```
P18.py ×
P18.py > ...
1  import networkx as nx
2  import matplotlib as matp
3  import matplotlib.pyplot as plt
4
5  A2 = {0,1,2} # Conjunto con los nodos
6  R2 = [(0, 0), (0, 1), (0, 2), (1, 2), (2, 0), (2, 1), (2, 2)] # Conjunto con las relaciones
7
8  GDb = nx.DiGraph() # Creacion del digrafo
9
10 GDb.add_nodes_from(A2) # Agregado de los nodos al digrafo
11 GDb.add_edges_from(R2) # Agregado de los arcos al digrafo
12
13 pos = {} # lista para asignar posiciones a los nodos
14 pos[0] = (0,0)
15 pos[1] = (3,5)
16 pos[2] = (2,3)
17
18 nx.draw(GDb, pos, arrows=True, arrowstyle = "->", # Dibujo del digrafo
19         connectionstyle = 'arc3, rad = 0.3', with_labels = True)
20
21 def ciclos_simples(dif):
22     result = list(nx.simple_cycles(dif)) # Variable guarda en lista los ciclos simples
23     return result # Retorno de la variable
24
```

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

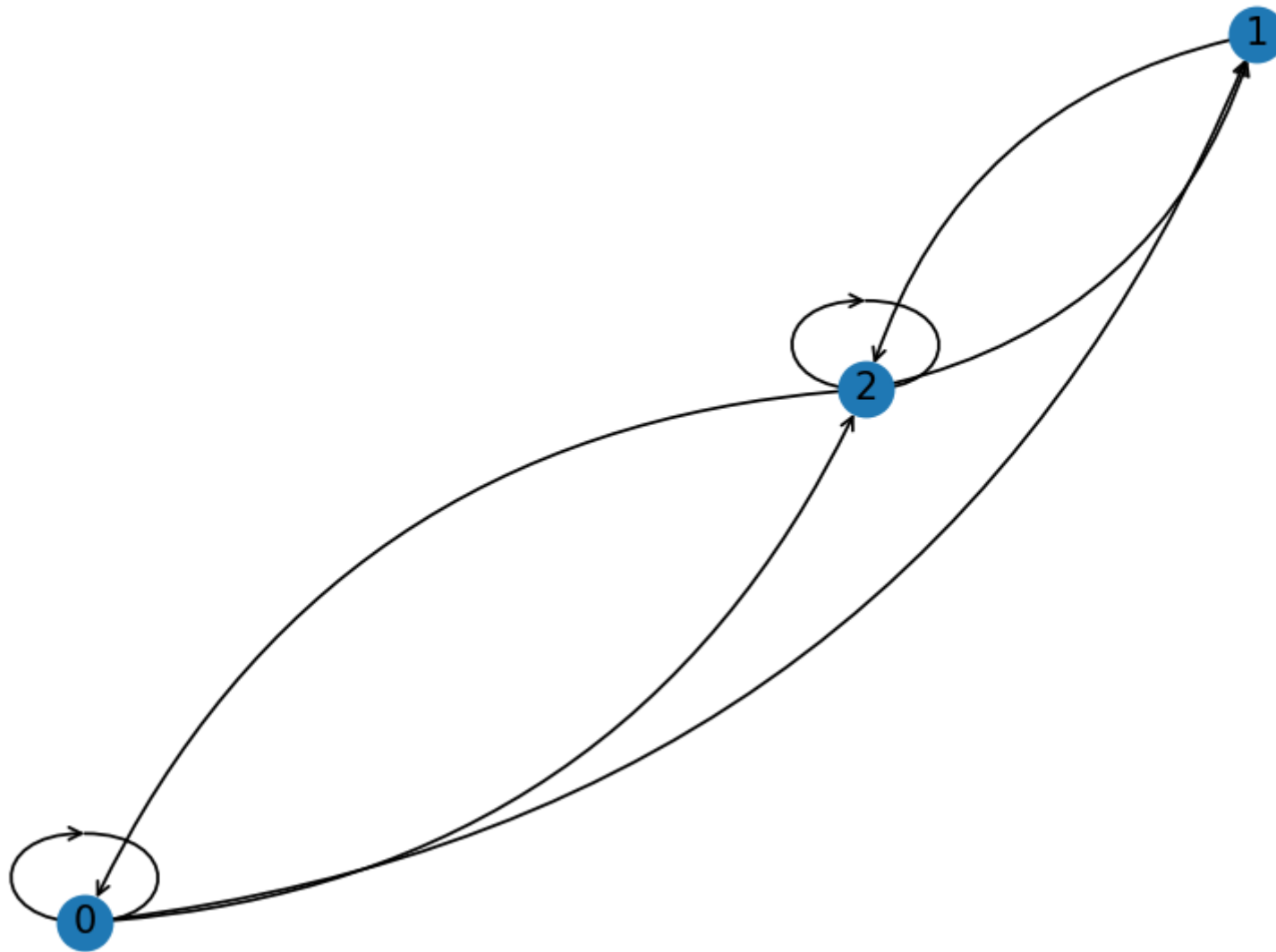
P18.py > ...

```
26  cic_s = ciclos_simples(GDb) # Variable guarda el resultado de llamar a la funcion
27
28
29  print("Ciclos simples encontrados: ")
30  for i in cic_s: # Impresion del contenido de la variable mediante un for
31      | print(i)
32
33  plt.show() # Se muestra el dibujo del digrafo para corroborar las respuestas
```

PROBLEMS   OUTPUT   DEBUG CONSOLE   TERMINAL

```
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



P19

```
P19.py X
P19.py > ...
1  import networkx as nx
2  import matplotlib as matp
3  import matplotlib.pyplot as plt
4
5
6  A = {1,2,3,4,5} # Conjunto de nodos
7  R = [(1,1), (2,2), (3,3), (3,1), (5,4)] # Conjunto de relaciones
8
9  GDa = nx.DiGraph() # Creacion del digrafo
10 GDa.add_nodes_from(A) # Agregado de los nodos al digrafo
11 GDa.add_edges_from(R) # Agregado de los arcos al digrafo
12
13 def ciclos_long_1(dif):
14     result = list(nx.selfloop_edges(dif)) # Variable guarda en lista los ciclos de longitud 1
15
16
17     return result # Se retorna la variable
18
19 ciclos = ciclos_long_1(GDa) # guardado en variable el resultado del llamado de la funcion
20
21 print("Relaciones en el digrafo: ") # Impresion de los resultados
22 print(R)
23
24 print("Ciclos de longitud 1 encontrados: ")
25 for i in ciclos:
26     print(i)
```




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

PROBLEMS

OUTPUT

DEBUG CONSOLE

TERMINAL


 Python Debug C

```
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

```
P20.py ×
P20.py > ...
1  import networkx as nx
2  import matplotlib as matp
3  import matplotlib.pyplot as plt
4
5  A = {1,2,3,4,5,6,7}
6
7  R2 = {(1, 2), (1, 6), (2, 3), (3, 3), (3, 4), (4, 3), (4, 1),
8  (4, 5), (6, 4), (1, 7), (7, 2)}
9
10 GD = nx.DiGraph() # Creacion del digrafo
11 GD.add_nodes_from(A) # Agregado de los nodos al digrafo
12 GD.add_edges_from(R2) # Agregado de los arcos representados en R2
13
14 pos = {}          # Posicionamiento de los nodos
15 pos[1] = (0,7)
16 pos[2] = (0,9)
17 pos[3] = (9,7)
18 pos[4] = (5,7)
19 pos[5] = (3,8)
20 pos[6] = (3,6)
21 pos[7] = (2,5)
22
```

P20.py

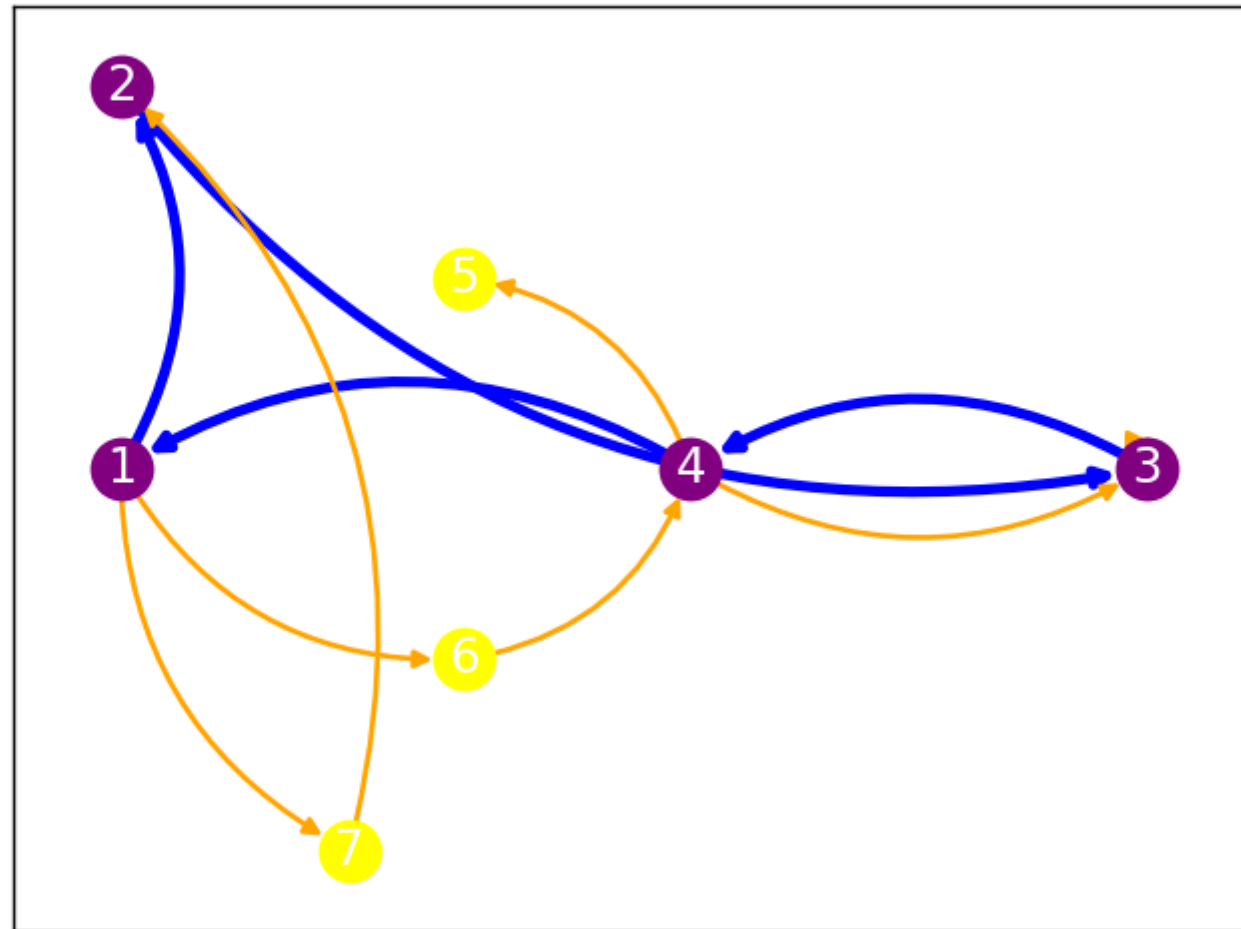
 P20.py > ...

```

23
24 nx.draw_networkx_nodes(GD, pos, nodelist = [1,2,3,4], node_color="purple") # Dibujo de los nodos
25 nx.draw_networkx_nodes(GD, pos, nodelist = [5,6,7], node_color="yellow")
26 ✓ nx.draw_networkx_edges(GD, pos, edgelist = [(1,2), (3,4)], edge_color="blue", # Dibujo de los arcos
27 | | | | | width=3.0, connectionstyle = 'arc3, rad = 0.3')
28 ✓ nx.draw_networkx_edges(GD, pos, edgelist = [(4,1), (2,3)], edge_color="blue",
29 | | | | | width=3.0, connectionstyle = 'arc3, rad = 0.3')
30
31 ✓ nx.draw_networkx_edges(GD, pos, edgelist = [(1,6)], edge_color="orange",
32 | | | | | width=1.5, connectionstyle = 'arc3, rad = 0.3')
33 ✓ nx.draw_networkx_edges(GD, pos, edgelist = [(3,3), (4,3)], edge_color="orange",
34 | | | | | width=1.5, connectionstyle = 'arc3, rad = 0.3')
35 ✓ nx.draw_networkx_edges(GD, pos, edgelist = [(4,5)], edge_color="orange",
36 | | | | | width=1.5, connectionstyle = 'arc3, rad = 0.3')
37 ✓ nx.draw_networkx_edges(GD, pos, edgelist = [(6,4), (1,7)], edge_color="orange",
38 | | | | | width=1.5, connectionstyle = 'arc3, rad = 0.3')
39 ✓ nx.draw_networkx_edges(GD, pos, edgelist = [(7,2)], edge_color="orange",
40 | | | | | width=1.5, connectionstyle = 'arc3, rad = 0.3')
41
42
43 nx.draw_networkx_labels(GD, pos, font_size=14, font_color="white") # Dibujo de los labels
44
45 plt.show() # Muestra del dibujo final

```

Figure 1



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