Módulo 1 - Capitulo 2

Unidad 10

Quiz 1

Cree una lista de primos que tenga como elementos números primos entre 2 y 10.

A continuación, utilice la indexación de la lista al primer elemento de la lista e imprima como se muestra a continuación.

Condición para la ejecución	1er elemento de `primos_lista` 2do 3ro
Tiempo	5 minutos

```
def isPrime(n):
    for i in range(2, n):
        if (n % i) == 0:
            return False
    return True

prime_list = []

for i in range(2, 10):
    if isPrime(i):
        prime_list.append(i)

for i in range(len(prime_list)):
    print(prime_list[i])
```

```
2
3
5
7
```

Quiz 2

Crear `primos_lista` que tenga como elementos números primos entre 1 y 10.

A continuación, utiliza el método append para añadir 11. Imprime lso resultados antes y despues de la adición como se muestra a continuación.

Condición para la ejecución	Números primos: [] Números primos despues de la adición: []

Tiempo	5 minutos

```
prime_list = []

for i in range(1, 10):
    if isPrime(i):
        prime_list.append(i)


print(f"Números primos: { prime_list }")

prime_list.append(11)
print(f"Números primos después de la adición: { prime_list }")
```

```
Números primos: [1, 2, 3, 5, 7]
Números primos después de la adición: [1, 2, 3, 5, 7, 11]
```

Quiz 3

Para la `list1` y la `list2`, utilice el bucle for anidado para multiplicar cada elemento de la `list1` y la `list2` y luego imprima el resultado con el resultado de la multiplicación de elementos.

Condicion para la ejecución	Declare `list1` y `list2` en la primera y segunda fila. Utilizar el bucle for anidado en la tercera y cuarta fila, y utilizar el bucle print en la quinta fila.
Tiempo	5 Minutos

```
list1 = [3, 5, 7]
list2 = [2, 3, 4, 5, 6]


for i in range(len(list1)):
    for j in range(len(list2)):
        print(f"{list1[i]} * {list2[j]} = { list1[i] * list2[j] }")
```

```
3 * 2 = 6
3 * 3 = 9
3 * 4 = 12
3 * 5 = 15
3 * 6 = 18
5 * 2 = 10
5 * 3 = 15
```

```
5 * 4 = 20
5 * 5 = 25
5 * 6 = 30
7 * 2 = 14
7 * 3 = 21
7 * 4 = 28
7 * 5 = 35
7 * 6 = 42
```

Quiz 4

Hay una lista con la cadena `s_lista = ['abc', 'bcd', 'bcdefg', 'abba', 'cddc', 'opq']`.
Implementa la siguiente función a esta lista.

No utilice la función `min` o el método `sort` para imprimir la cadena más corta de las cadenas de `s_list`.

(Si hay varias cadenas más cortas, imprime la cadena que muestra la primera como siguiente).

Ejemplo de salida:

La cadena más corta es:

```
abc
```

```
s_list = ['abc', 'bcd', 'bcdefg', 'abba', 'cddc', 'opq']

aux = 0

for i in range(len(s_list)):
    if len(s_list[aux]) <= len(s_list[i]):
        pass
    else:
        aux = i
        pass

print(f"La cadena más corta es: {s_list[aux]}")
```

```
La cadena más corta es: abc
```

Quiz 5

Hay una lista con la cadena `s_list = ['abc', 'bcd', 'bcdefg', 'abba', 'cddc', 'opq']`.
Implementa la siguiente función a esta lista.

No utilice la función `min` ni el método `sort` para imprimir la cadena más corta de las cadenas de `s_list`.

(Si hay varias cadenas más cortas, imprime la cadena que muestra la primera como la siguiente).

Ejemplo de salida:

La cadena más larga es:

```
bcdefg
```

```
s_list = ['abc', 'bcd', 'bcdefg', 'abba', 'cddc', 'opq']

aux = 0

for i in range(len(s_list)):
    if len(s_list[i]) > len(s_list[aux]):
        aux = i
        pass
    else:
        pass

print(f"La cadena más larga es {s_list[aux]}")
```

```
La cadena más larga es bcdefg
```

Quiz 6

Hay una lista con la cadena `s_list = ['abc', 'bcd', 'bcdefg', 'abba', 'cddc', 'opq']`.
Implementa la siguiente función a esta lista.

Del problema de programación por pares anterior, la longitud de `abc`, `bcd` y `opq` es igual a 3.

Asimismo, si las longitudes de las cadenas son iguales, escribe un programa que imprima las tres cadenas más cortas como sigue.

Utiliza la función `sort(key=len)` para ordernar las cadenas por su longitud y luego escribe un código.

Ejemplo de salida:

Las cádenas más cortas son:

```
abc, bcd, opq
```

```
s_list = ['abc', 'bcd', 'bcdefg', 'abba', 'cddc', 'opq']
s_list.sort(key=len)

min_len = len(s_list[0])

new_list = []

for i in range(len(s_list)):
    if len(s_list[i]) == min_len:
        new_list.append(s_list[i])

print(f"Las cadrenas más cortas son: {', '.join(new_list)}")
```

```
Las cadrenas más cortas son: abc, bcd, opq
```

Unidad 11

Quiz 1

Cree el diccionario `capital_dic` con solos siguientes elementos clave-valor de cadena. A continuación, utilice el diccionario `capital_dic` para escribir los resultados relativos a Corea en los siguientes elementos del diccionario.

Ejecución del programa Declaración de variables	Clave: Corea Valor: Seoul Clave: China Valor: Beijing Clave: USA Valor: Washington DC
Tiempo	5 minutos

Ejemplo de salida

```
Seoul
```

Escriba el código completo y el resultado esperado.

```
capital_dic = {
    "Corea": "Seoul",
    "China": "Beijing",
    "USA": "Washington DC"
}

capital_dic["Corea"]
```

```
'Seoul'
```

Quiz 2

Cree el diccionario `frutas_dic` que tiene elementos de los siguientes elementos clave-valor. A continuación, utilice este diccionario para imprimir el precio de cada fruta como se muestra a continuación:

Condiciones de Ejecución	El precio de una manzana es de 5000 KRW. El precio de un banano es de 4000 KRW. El precio de una uva es de 5300 krw. El precio de un melón es de 6500 krw.
Tiempo	5 Minutos

Escriba el código completo y el resultado esperado resulta en la nota

```
fruits_dic = {
    "Manzana": 5000,
    "Banano": 4000,
    "Uva": 5300,
    "Melon": 6500
}

for k, v in fruits_dic.items():
    print(f"El precio de una {k} es de {v} krw")
```

```
El precio de una Manzana es de 5000 krw
El precio de una Banano es de 4000 krw
El precio de una Uva es de 5300 krw
El precio de una Melon es de 6500 krw
```

Quiz 3

Cree el diccionario `frutas_dic` compuesto por pares clave-valor que incluyan (manzana, 6000), (melon, 3000), (banano, 5000), (naranja, 4000). A continuación, imprima todas las claves del diccioarion `frutas_dic` como tipo de lista, y examine si las claves `manzana` y `mango` se encuentran en el diccionario `frutas_dic`, e imprima como sigue:

Ejemplo de salida:

```
dic_keys(['manzana', 'banano', 'naranja'])
```

```
manzana is in fruits_dic

mango is not in fruits_dic
```

```
fruits_dic = {
    "manzana": 6000,
    "melon": 3000,
    "banano": 5000,
    "naranja": 4000
}

dic_keys = fruits_dic.keys()
print(dic_keys)


def evaluate(key):
    status = "is" if key in fruits_dic else "is not"
    print(f"{key} {status} in fruits_dic")

evaluate("manzana")
evaluate("mango")
```

```
dict_keys(['manzana', 'melon', 'banano', 'naranja'])
manzana is in fruits_dic
mango is not in fruits_dic
```

Unidad 12

Quiz 1

Predecir el resultado de la ejecución del siguiente código y proporcionar el resultado escrito a mano.

Condicion para la ejecución	Predecir el resultado del siguiente programa y proporcionar los resultados de la codificación a mano.
Tiempo	5 Minutos

Ejemplo de salida:

```
t1 = 'a', 'b', 'c'
t2 = ('a', 'b', 'c')
t3 = ('d', 'e')

print(t1 == t2)
print(t1 > t3)
print(t1 < t3)
```

```
print(t2 + t3)
print([ t2 + t3 ])
print(t1)
```

```
t1 = 'a', 'b', 'c'
t2 = ('a', 'b', 'c')
t3 = ('d', 'e')

print(f"1. {t1 == t2}")
print(f"2. {t1 > t3}")
print(f"3. {t1 < t3}")
print(f"4. {t2 + t3}")
print(f"5. {[ t2 + t3 ]}")
print(f"6. {t1}")
```

```
1. True
2. False
3. True
4. ('a', 'b', 'c', 'd', 'e')
5. [('a', 'b', 'c', 'd', 'e')]
6. ('a', 'b', 'c')
```

Quiz 2

La siguiente tupla registra las ventas diarias de una tienda durante 10 días. Escribe un código para imprimir cuántos días se redujeron las ventas en comparación con el día anterior. (Sugerencia compara los valores iterando los elementos con la sentencia de iteración).

Condicion para la ejecución	Registro diario de ventas: (100, 121, 120, 130, 140, 120, 122, 123, 190, En los últimos 10 días, 3 días han reducido las ventas en comparación con el día anterior.
Tiempo	10 Minutos

```
sales = (100, 121, 120, 130, 140, 120, 122, 123, 190, 150)

days = 0

for i in range(len(sales) - 1):
    if sales[i] > sales[i + 1]:
        days += 1
    else:
        pass

print(f"En los últimos días, {days} días han reducido las ventas en comparación
con el día anterior")
```

```
En los últimos días, 3 días han reducido las ventas en comparación con el día anterior
```

Quiz 3

Devuelve el elemento con el máximo número de ocurrencias. Cuando hay más de dos elementos frecuentes, imprime el número más alto.

Ejemplo de salida:

```
Tupla obtenida: (1, 2, 5, 4, 3, 2, 1, 9, 9, 3, 7, 3, 9)

El elemento más frecuente: 9
```

```
numbers = (1, 2, 5, 4, 3, 2, 1, 9, 9, 3, 7, 3, 9)

sorted_list = list(numbers)
sorted_list.sort()

occurrences = dict(
    zip(
        sorted_list,
        map(
            lambda x: sorted_list.count(x),
            sorted_list
        )
    )
)

max_occurrence = 0
most_frecuent_item = None

for k, v in occurrences.items():
    if v > max_occurrence:
        max_occurrence = v
        most_frecuent_item = k
    elif v == max_occurrence:
        if k > most_frecuent_item:
            most_frecuent_item = k

print(f"El elemento más frecuente: {most_frecuent_item}")
```

```
El elemento más frecuente: 9
```

Quiz 4

En el ejemplo de salida de abajo, hay tuplas que contienen elementos, así como tuplas vacías, cadenas vacías y listas vacías que no tienen elementos. Escribe un código para eliminar estas tuplas, cadenas y listas vacías de la lista dada a continuación. Sin embargo, no elimine la tupa `((),)` porque se considera que tiene una tupla vacía.

Ejemplo de salida:

```
Tupla obtenida: [(), (1,), [], 'abc', (), (), (1,), ('a',), ('a', 'b'), ((),), '']

Lista limpia: [(1,), 'abc', (1,), ('a',), ('a', 'b'), ((),)]
```

```
_tuple = [(), (1,), [], 'abc', (), (), (1,), ('a',), ('a', 'b'), ((),), '']

new_tuple = []

for i in range(len(_tuple) - 1):
    if len(_tuple[i]) != 0:
        new_tuple.append(_tuple[i])

print(f'Lista limpia: {new_tuple}')
```

```
Lista limpia: [(1,), 'abc', (1,), ('a',), ('a', 'b'), ((),)]
```

Unidad 13

Quiz 1

Poner las matrices bidimensionales `[[10, 20], [30, 40], [50, 60]]` en la variables `lista_matriz` y salida `50`. Haz la indexación correcta.

Condicion para la ejecución	50
Tiempo	5 Minutos

```
list_2d = [[10, 20], [30, 40], [50, 60]]

list_2d[2][0]
```

```
30
```

Quiz 2

Crea una lista 2D de tamaño 4 x 4 con valores que van de 1 a 16 e imprime todos los elementos utilizando el bucle for.

Condicion para la ejecución	1 2 3 4 5 6 7 8 9 10 11 12 13 14 15 16
Tiempo	5 Minutos

```
list_2d = [[j*4+i+1 for i in range(4)] for j in range(4)]

for i in list_2d:
    for j in i:
        print(j, end=' ')
    print()
```

```
1 2 3 4
5 6 7 8
9 10 11 12
13 14 15 16
```

Quiz 3

Escriba un programa que genere un arreglo multidimensional de tamaño n×n, basado en el número de entradas, al recibir dos o más n como entradas de los usuarios. En este caso, el contenido del arreglo debe mostrarse de manera que los valores de 0 y 1 se crucen en un patrón de cuadros.

Ejemplo de salida

```
10101
01010
10101
01010
10101
```

```
n = int(input("Ingrese el tamaño del arreglo (nxn): "))
```

```
matrix_of_0 = [[0 for i in range(n)] for j in range(n)]

for i in range(n):
    for j in range(n):
        if (i + j) % 2 == 0:
            matrix_of_0[i][j] = 1

for i in matrix_of_0:
    for j in i:
        print(j, end=' ')
    print()
```

```
Ingrese el tamaño del arreglo (nxn): 10
1 0 1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1 0 1
1 0 1 0 1 0 1 0 1 0
0 1 0 1 0 1 0 1 0 1
```

Unidad 14

Quiz 1

Vamos a crear un diccionario llamado `dict_person` con la siguiente información de contacto en su teléfono. Imprima esta información utilizando el bucle `for` para mostrar los resultados de salida que se muestran a continuación:

Condiciones de Ejecución	'Apellido': 'Doe', 'Primer nombre': 'David', 'Compañia': 'Samsung'
Tiempo	5 minutos

```
dict_person = {
    "Last Name": "Doe",
    "First Name": "David",
    "Company": "Samsung"
}

for k,v in dict_person.items():
    print(f"{k}: {v}")
```

```
Last Name: Doe
First Name: David
Company: Samsung
```

Quiz 2

Escribamos un programa que realice la gestión del inventario en una tienda de conveniencia. Para ello, el inventario de los artículos que se venden en las tiendas de conveniencia se almacena en el diccionario de artículos, como se muestra en el siguiente ejemplo.

Escriba un programa que reciba el nombre del artículo de los usuarios y devuelve el inventario del mismo. Suponga que se trata de una tienda de conveniencia muy pequeña y los artículos tratados son los siguientes:

Ejemplo de programa	Introduzca el nombre del artículo: Leche
Resultados de la ejecución	1
Tiempo	5 minutos

Items

```
items = {"Café": 7, "Lápiz": 3, "Vaso de papel": 2, "Leche": 1, "Coca-Cola": 4,
"Libro": 5}
```

```
items = {"Café": 7, "Lápiz": 3, "Vaso de papel": 2, "Leche": 1, "Coca-Cola": 4,
"Libro": 5}

item = input("Introduzca el nombre del artículo: ")

print(items[item])
```

```
Introduzca el nombre del artículo: Leche
1
```

Quiz 3

Actualicemos el programa para gestionar el inventario de las tiendas de coveniencia que resolvimos en el ejercicio anterior. En otras palabras, añadir código para aumentar o disminuir el inventario. También, hacer menús simples con la consulta de inventario, el almacenamiento y el envío.

```
items = {"Café": 7, "Lápiz": 3, "Vaso de papel": 2, "Leche": 1, "Coca-Cola": 4,
"Libro": 5}
```

Ejemplo de salida:

```
Seleccione el menú:
1) Comprobar existencias
2) Almacenamiento
3) Liberación
4) Salir

    > 1

[Comprobar existencias] Introduzca el artículo: leche
    - Existencias: 1

Seleccione el menú:
1) Comprobar existencias
2) Almacenamiento
3) Liberación
4) Salir

    > 3

[Liberación]: Introduzca el artículo y la cantidad: coca-cola 1

Seleccione el menú:
1) Comprobar existencias
2) Almacenamiento
3) Liberación
4) Salir

    > 1

[Comprobar existencias] Introduzca el artículo: coca-cola
    - Existencias: 3

Seleccione el menú:
1) Comprobar existencias
2) Almacenamiento
3) Liberación
4) Salir

    > 4

Se ha salido del programa
```

```
menu = '''
Seleccione el menú:
    1) Comprobar existencias
    2) Almacenamiento
    3) Liberación
```

```
    4) Salir

    > '''

items = {"Café": 7, "Lápiz": 3, "Vaso de papel": 2, "Leche": 1, "Coca-Cola": 4,
"Libro": 5}

opt = int(input(menu))

while opt in [1, 2, 3]:
    if opt == 1:
        art = input("\n[Comprobar existencias] Introduzca el artículo: ")
        print(f"\t - Existencias: {items[art]}\n")
        opt = int(input(menu))
    elif opt == 2:
        print("\n[Almacenamiento]: Introduzca el artículo y la cantidad: ")
        art, amount = input("\tArtículo: "), int(input("\tCantidad: "))
        items[art] = items[art] + amount
        opt = int(input(menu))
    elif opt == 3:
        print("\n[Liberación]: Introduzca el artículo y la cantidad: ")
        art, amount = input("\tArtículo: "), int(input("\tCantidad: "))
        items[art] = items[art] - amount
        opt = int(input(menu))
    else:
        print("\nOpción incorrecta\n")
        opt = int(input(menu))

print("Se ha salido del programa")
```

```
Seleccione el menú:
    1) Comprobar existencias
    2) Almacenamiento
    3) Liberación
    4) Salir

    > 1

[Comprobar existencias] Introduzca el artículo: Leche
     - Existencias: 1


Seleccione el menú:
    1) Comprobar existencias
    2) Almacenamiento
    3) Liberación
    4) Salir

    > 2
```

```
[Almacenamiento]: Introduzca el artículo y la cantidad:
    Artículo: Leche
    Cantidad: 5

Seleccione el menú:
    1) Comprobar existencias
    2) Almacenamiento
    3) Liberación
    4) Salir

    > 1

[Comprobar existencias] Introduzca el artículo: Leche
     - Existencias: 6


Seleccione el menú:
    1) Comprobar existencias
    2) Almacenamiento
    3) Liberación
    4) Salir

    > 3

[Liberación]: Introduzca el artículo y la cantidad:
    Artículo: Leche
    Cantidad: 2

Seleccione el menú:
    1) Comprobar existencias
    2) Almacenamiento
    3) Liberación
    4) Salir

    > 1

[Comprobar existencias] Introduzca el artículo: Leche
     - Existencias: 4


Seleccione el menú:
    1) Comprobar existencias
    2) Almacenamiento
    3) Liberación
    4) Salir

    > 4
Se ha salido del programa
```

Unidad 15

Quiz 1

Existe una tupla llamada `student_tup`, que tiene tres pares de elementos: número de identificación del estudiante, nombre y número de teléfono, como se muestra a continuación.

Modifique la tupla `student_tup` a continuación para crear a imprimir un diccionario del par (número de identificación del estudiante: [nombre, número de teléfono])

Condicion para la ejecución	estudiante_tup = (('211101', 'David Doe', '010-1234-4500'), ('211102', ' John Smith', '010-2230-6540'), ('211103', ' Jane Carter', '010-3232-7788')).
Tiempo	7 Minutos

Ejemplos de salida:

```
{ '211101' = ['David Doe', '010-1234-4500'] }
```

```
{ '211102' = ['John Smith', '010-2230-5540'] }
```

```
{ '211103' = ['Jane Carter', '010-3232-7788'] }
```

```
student_tuple = (
    ('211101', 'David Doe', '010-1234-4500'),
    ('211102', 'John Smith', '010-2230-6540'),
    ('211103', ' Jane Carter', '010-3232-7788')
)

student_dict = {}

for student in student_tuple:
    student_dict[student[0]] = [student[1], student[2]]

for k, v in student_dict.items():
    print(f"{k} = {v}")
```

```
211101 = ['David Doe', '010-1234-4500']
211102 = ['John Smith', '010-2230-6540']
211103 = [' Jane Carter', '010-3232-7788']
```

Quiz 2

Escriba un programa de información sobre el estudiante utilizando el `student_tuple` anterior para recibir el número de identificación del estudiante como entrada e imprimir el nombre y el número de teléfono del estudiante.

Condicion para la ejecución	Enter student ID number : 211101 Name : David Doe Phone number : 010-1234-4500
Tiempo	5 Minutos

```
student_tuple = (
    ('211101', 'David Doe', '010-1234-4500'),
    ('211102', 'John Smith', '010-2230-6540'),
    ('211103', ' Jane Carter', '010-3232-7788')
)

student_dict = {}

for student in student_tuple:
    student_dict[student[0]] = [student[1], student[2]]

student_id = input("Enter student ID number: ")

if student_id in student_dict:
    student_info = student_dict[student_id]
    print("Name:", student_info[0])
    print("Phone:", student_info[1])
else:
    print("Student not found.")
```

```
Enter student ID number: 211101
Name: David Doe
Phone: 010-1234-4500
```

Quiz 3

La lista `student_tuple` con tuplas como elementos es la que se muestra a continuación. La tupla, que es el elemento de esta tupla consiste en un (número de identificación del estudiante, nombre, número de teléfono). Usanto esto, haga un diccionario para (número de identificación del estudiante: nombre) e imprímalo.

Cuando pregunte por el número de identifación del estudiante, asegúrese de que el número de identificación del estudiante, el nombre y el número de teléfono se impriman como se muestra a continuación.

```
student_tuple = [('211101', 'David Doe', '010-123-1111'), ('211102', 'John Smith',
'010-123-2222’),('211103', 'Jane Carter', '010-123-3333')]
```

Ejemplo de salida:

```
211101: David Doe
211102: John Smith
211103: Jane Carter

Ingrese el número de identificación del estudiante: 211103

211103 es el estudiante Jane Carter y su número de teléfono es 010-123-3333
```

```
student_tuple = [('211101', 'David Doe', '010-123-1111'), ('211102', 'John Smith',
'010-123-2222'), ('211103', 'Jane Carter', '010-123-3333')]
student_dict = {}

for student in student_tuple:
    student_dict[student[0]] = student[1]

print("Student ID: Name")
for student_id, name in student_dict.items():
    print(f"{student_id}: {name}")

student_id = input("\nEnter student ID: ")

if student_id in student_dict:
    student_info = student_dict[student_id]
    for student in student_tuple:
        if student[0] == student_id:
            print(f"{student_id} is the student {student_info} and their phone
number is {student[2]}")
else:
    print("Student not found.")
```

```
Student ID: Name
211101: David Doe
211102: John Smith
211103: Jane Carter

Enter student ID: 211103
211103 is the student Jane Carter and their phone number is 010-123-3333
```

Unidad 16

Quiz 1

Utilice la función set para generar e imprimir el conjunto `s1` de la siguiente lista `lst`.

Condiciones variables del programa	lst = ['apple', 'mango', 'banana'] # A list of 3 fruit information
	s1 = {'apple', 'mango', 'banana'} # Set s1 generated from lst

Tiempo	7 Minutos

Ejemplo de salida:

```
s1 = {'banana', 'Manzana', 'Mango'}
```

```
lst = ['apple', 'mango', 'banana']

s1 = set(lst)
s1
```

```
{'apple', 'banana', 'mango'}
```

Quiz 2

Escriba los resultados computacionales de los dos conjuntos siguientes: Encuentra los resultados del `1)` a `7)`

Condición para la operación	s1 = {10, 20, 30, 40} s2 = {30, 40, 50, 60, 70} 1) s1 | s2 2) s1 & s2 3) s1 – s2 4) s1 ^ s2 5) s1.issubset(s2) 6) s1.issuperset(s2) 7) s1.isdisjoint(s2)
Tiempo	5 Minutos

```
s1 = {10, 20, 30, 40}
s2 = {30, 40, 50, 60, 70}
```

```
s1 | s2
```

```
{10, 20, 30, 40, 50, 60, 70}
```

```
s1.union(s2)
```

```
{10, 20, 30, 40, 50, 60, 70}
```

```
s1 & s2
```

```
{30, 40}
```

```
s1.intersection(s2)
```

```
{30, 40}
```

```
s1 - s2
```

```
{10, 20}
```

```
s1.difference(s2)
```

```
{10, 20}
```

```
s1 ^ s2
```

```
{10, 20, 50, 60, 70}
```

```
s1.symmetric_difference(s2)
```

```
{10, 20, 50, 60, 70}
```

```
s1.issubset(s2)
```

```
False
```

```
s1.issuperset(s2)
```

```
False
```

```
s1.isdisjoint(s2)
```

```
False
```

```
s2 - s1
```

```
{50, 60, 70}
```

```
s2.difference(s1)
```

```
{50, 60, 70}
```

Quiz 3

Hay una lista `mylist` con tupla(m, n) como elementos como se muestra a continuación. Si hay una tupla con valor(a, b) y los valores a, b introducidos por el usuario, imprime `Hay un elemento (a, b) en xth`. Si no hay (a, b) pero sí (b, a) imprime `No hay (a, b) pero sí (b, a) en yth`. Si no hay (a, b) ni (b, a), imprime `no hay tal elemento`:

```
mylist = [(1, 2), (4, 5), (4, 2), (3, 1), (9, 4)]
```

Ejemplo de salida:

Introduce dos enteros: 1 2

Hay (1,2) en el primero.

Introduce dos enteros: 5 4

No hay (5,4) pero hay (4,5) en el segundo.

Introduce dos enteros: 3 9

No hay (3,9) ni (9,3)

```
mylist = [(1, 2), (4, 5), (4, 2), (3, 1), (9, 4)]

_input = input("Introduce dos enteros separados por un espacio: ").split()

a = int(_input[0])
b = int(_input[1])

_tuple1 = (a, b)
_tuple2 = (b, a)

if _tuple1 in mylist:
    print(f"Hay {_tuple1} en el elemento #{mylist.index(_tuple1) + 1} de la
lista")
elif _tuple2 in mylist:
    print(f"No hay {_tuple1} pero hay {_tuple2} en el elemento #
{mylist.index(_tuple2) + 1} de la lista")
else:
    print(f"No hay {_tuple1} ni {_tuple2}")
```

```
Introduce dos enteros separados por un espacio: 5 4
No hay (5, 4) pero hay (4, 5) en el elemento #2 de la lista
```