Métodos de las listas (`list`):

- `append()` : Agrega un elemento al final de la lista.
- `clear()` : Elimina todos los elementos de la lista.
- `copy()` : Devuelve una copia superficial de la lista.
- `count()` : Devuelve el número de veces que aparece un elemento en la lista.
- `extend()` : Agrega los elementos de una lista (o cualquier iterable) al final de la lista.
- `index()` : Devuelve el índice del primer elemento con el valor especificado.
- `insert()` : Inserta un elemento en la posición especificada.
- `pop()` : Elimina el elemento en la posición especificada y lo devuelve.
- `remove()` : Elimina el primer elemento con el valor especificado.
- `reverse()` : Invierte el orden de la lista.
- `sort()` : Ordena la lista.

Métodos de las cadenas (`str`):

- `capitalize()` : Convierte el primer carácter en mayúscula.
- `casefold()` : Convierte la cadena a minúsculas.
- `center()` : Devuelve una cadena centrada.
- `count()` : Devuelve el número de ocurrencias de una subcadena.
- `encode()` : Codifica la cadena usando el método especificado.
- `endswith()` : Devuelve `True` si la cadena termina con el sufijo especificado.
- `expandtabs()` : Establece el tamaño de la tabulación de la cadena.
- `find()` : Devuelve el índice de la primera aparición de una subcadena.
- `format()` : Formatea la cadena.
- `index()` : Devuelve el índice de la primera aparición de una subcadena, genera un error si no se encuentra.
- `isalnum()` : Devuelve `True` si todos los caracteres de la cadena son alfanuméricos.
- `isalpha()` : Devuelve `True` si todos los caracteres de la cadena son alfabéticos.
- `isdigit()` : Devuelve `True` si todos los caracteres de la cadena son dígitos.
- `islower()` : Devuelve `True` si todos los caracteres de la cadena están en minúsculas.
- `isspace()` : Devuelve `True` si todos los caracteres de la cadena son espacios en blanco.
- `istitle()` : Devuelve `True` si la cadena es una cadena con título.
- `isupper()` : Devuelve `True` si todos los caracteres de la cadena están en mayúsculas.
- `join()` : Une los elementos de una lista usando la cadena como separador.
- `ljust()` : Devuelve una versión justificada a la izquierda de la cadena.
- `lower()` : Convierte la cadena a minúsculas.
- `lstrip()` : Elimina los espacios en blanco a la izquierda de la cadena.
- `partition()` : Devuelve una tupla donde la cadena se divide en tres partes.
- `replace()` : Reemplaza una cadena con otra.
- `rfind()` : Devuelve el índice de la última aparición de una subcadena.
- `rindex()` : Devuelve el índice de la última aparición de una subcadena, genera un error si no se encuentra.
- `rjust()` : Devuelve una versión justificada a la derecha de la cadena.
- `rpartition()` : Devuelve una tupla donde la cadena se divide en tres partes, comenzando por la derecha.
- `rsplit()` : Divide la cadena en una lista, comenzando por la derecha.
- `rstrip()` : Elimina los espacios en blanco a la derecha de la cadena.

- `split()`: Divide la cadena en una lista.
- `splitlines()`: Divide la cadena en líneas.
- `startswith()`: Devuelve `True` si la cadena comienza con el prefijo especificado.
- `strip()`: Elimina los espacios en blanco al principio y al final de la cadena.
- `swapcase()`: Intercambia mayúsculas y minúsculas.
- `title()`: Convierte la primera letra de cada palabra en mayúscula.
- `translate()`: Devuelve una cadena donde algunos caracteres especificados se reemplazan con el carácter descrito en un diccionario.
- `upper()`: Convierte la cadena a mayúsculas.
- `zfill()`: Rellena la cadena con ceros a la izquierda.

Métodos de los booleanos (`bool`):

Los objetos booleanos en Python son simples (solo `True` o `False`) y no tienen métodos específicos.

Métodos de los diccionarios (`dict`):

- `clear()`: Elimina todos los elementos del diccionario.
- `copy()`: Devuelve una copia superficial del diccionario.
- `fromkeys()`: Devuelve un diccionario con las claves y el valor especificados.
- `get()`: Devuelve el valor de la clave especificada.
- `items()`: Devuelve una lista que contiene una tupla para cada par clave-valor.
- `keys()`: Devuelve una lista que contiene las claves del diccionario.
- `pop()`: Elimina el elemento con la clave especificada y devuelve su valor.
- `popitem()`: Elimina el último par clave-valor insertado.
- `setdefault()`: Devuelve el valor de la clave especificada. Si la clave no existe, inserta la clave con el valor especificado.
- `update()`: Actualiza el diccionario con los pares clave-valor especificados.
- `values()`: Devuelve una lista que contiene los valores del diccionario.

Métodos de los conjuntos (`set`):

- `add()`: Agrega un elemento al conjunto.
- `clear()`: Elimina todos los elementos del conjunto.
- `copy()`: Devuelve una copia superficial del conjunto.
- `difference()`: Devuelve un conjunto que contiene la diferencia entre dos o más conjuntos.
- `difference_update()`: Elimina los elementos en común con otro conjunto.
- `discard()`: Elimina el elemento especificado del conjunto.
- `intersection()`: Devuelve un conjunto que contiene la intersección de dos o más conjuntos.
- `intersection_update()`: Actualiza el conjunto con la intersección de sí mismo y otro(s) conjunto(s).
- `isdisjoint()`: Devuelve `True` si dos conjuntos no tienen elementos en común.
- `issubset()`: Devuelve `True` si todos los elementos de un conjunto están presentes en otro conjunto.

- `issuperset()`: Devuelve `True` si un conjunto contiene todos los elementos de otro conjunto.
- `pop()`: Elimina un elemento del conjunto y lo devuelve.
- `remove()`: Elimina el elemento especificado del conjunto.
- `symmetric_difference()`: Devuelve un conjunto que contiene todos los elementos que no son comunes entre

  dos conjuntos.
- `symmetric_difference_update()`: Actualiza el conjunto con la diferencia simétrica de sí mismo y otro conjunto.
- `union()`: Devuelve un conjunto que contiene la unión de conjuntos.
- `update()`: Actualiza el conjunto con la unión de sí mismo y otro(s) conjunto(s).

Métodos de los objetos `range`:

- `start`: Devuelve el inicio del rango.
- `stop`: Devuelve el final del rango.
- `step`: Devuelve el tamaño del paso del rango.
- `__contains__`: Verifica si un valor está dentro del rango.
- `__getitem__`: Devuelve el elemento en el índice especificado.

Métodos de los objetos `tuple`:

Los objetos `tuple` son inmutables, por lo que solo tienen un conjunto limitado de métodos:

- `count()`: Devuelve el número de veces que aparece un elemento en la tupla.
- `index()`: Devuelve el índice del primer elemento con el valor especificado.

Métodos de los objetos `bytes`:

- `capitalize()`: Convierte el primer carácter en mayúscula.
- `center()`: Devuelve una cadena centrada.
- `count()`: Devuelve el número de ocurrencias de una subcadena.
- `decode()`: Decodifica la secuencia de bytes usando el codec especificado.
- `endswith()`: Devuelve `True` si la secuencia de bytes termina con el sufijo especificado.
- `expandtabs()`: Establece el tamaño de la tabulación de la secuencia de bytes.
- `find()`: Devuelve el índice de la primera aparición de una subcadena.
- `fromhex()`: Crea un objeto bytes desde una cadena hexadecimal.
- `hex()`: Convierte cada byte en dos caracteres hexadecimales.
- `index()`: Devuelve el índice de la primera aparición de una subcadena, genera un error si no se encuentra.
- `isalnum()`: Devuelve `True` si todos los bytes son alfanuméricos.
- `isalpha()`: Devuelve `True` si todos los bytes son alfabéticos.
- `isdigit()`: Devuelve `True` si todos los bytes son dígitos.
- `islower()`: Devuelve `True` si todos los bytes están en minúsculas.
- `isspace()`: Devuelve `True` si todos los bytes son espacios en blanco.
- `istitle()`: Devuelve `True` si la secuencia de bytes es una cadena con título.
- `isupper()`: Devuelve `True` si todos los bytes están en mayúsculas.

- `join()`: Une los elementos de una lista usando la secuencia de bytes como separador.
- `ljust()`: Devuelve una versión justificada a la izquierda de la secuencia de bytes.
- `lower()`: Convierte la secuencia de bytes a minúsculas.
- `lstrip()`: Elimina los espacios en blanco a la izquierda de la secuencia de bytes.
- `maketrans()`: Devuelve una tabla de traducción que se puede usar en `translate()`.
- `partition()`: Devuelve una tupla donde la secuencia de bytes se divide en tres partes.
- `replace()`: Reemplaza una secuencia de bytes con otra.
- `rfind()`: Devuelve el índice de la última aparición de una subcadena.
- `rindex()`: Devuelve el índice de la última aparición de una subcadena, genera un error si no se encuentra.
- `rjust()`: Devuelve una versión justificada a la derecha de la secuencia de bytes.
- `rpartition()`: Devuelve una tupla donde la secuencia de bytes se divide en tres partes, comenzando por la derecha.
- `rsplit()`: Divide la secuencia de bytes en una lista, comenzando por la derecha.
- `rstrip()`: Elimina los espacios en blanco a la derecha de la secuencia de bytes.
- `split()`: Divide la secuencia de bytes en una lista.
- `splitlines()`: Divide la secuencia de bytes en líneas.
- `startswith()`: Devuelve `True` si la secuencia de bytes comienza con el prefijo especificado.
- `strip()`: Elimina los espacios en blanco al principio y al final de la secuencia de bytes.
- `swapcase()`: Intercambia mayúsculas y minúsculas.
- `title()`: Convierte la primera letra de cada palabra en mayúscula.
- `translate()`: Devuelve una secuencia de bytes donde algunos caracteres especificados se reemplazan con el carácter descrito en un diccionario.
- `upper()`: Convierte la secuencia de bytes a mayúsculas.
- `zfill()`: Rellena la secuencia de bytes con ceros a la izquierda.

Métodos de los objetos `bytearray`:

- `append()`: Agrega un byte al final del objeto `bytearray`.
- `capitalize()`: Convierte el primer carácter en mayúscula.
- `center()`: Devuelve una secuencia de bytes centrada.
- `clear()`: Elimina todos los elementos del objeto `bytearray`.
- `copy()`: Devuelve una copia superficial del objeto `bytearray`.
- `count()`: Devuelve el número de ocurrencias de un byte en el objeto `bytearray`.
- `decode()`: Decodifica el objeto `bytearray` usando el codec especificado.
- `endswith()`: Devuelve `True` si el objeto `bytearray` termina con el sufijo especificado.
- `expandtabs()`: Establece el tamaño de la tabulación del objeto `bytearray`.
- `extend()`: Agrega los bytes de otro objeto `bytearray` al final del objeto `bytearray`.
- `find()`: Devuelve el índice de la primera aparición de una secuencia de bytes en el objeto `bytearray`.
- `fromhex()`: Crea un objeto `bytearray` desde una cadena hexadecimal.

- `hex()`: Convierte cada byte en dos caracteres hexadecimales.
- `index()`: Devuelve el índice de la primera aparición de una secuencia de bytes en el objeto `bytearray`, genera un error si no se encuentra.
- `insert()`: Inserta un byte en la posición especificada.
- `isalnum()`: Devuelve `True` si todos los bytes son alfanuméricos.
- `isalpha()`: Devuelve `True` si todos los bytes son alfabéticos.
- `isdigit()`: Devuelve `True` si todos los bytes son dígitos.
- `islower()`: Devuelve `True` si todos los bytes están en minúsculas.
- `isspace()`: Devuelve `True` si todos los bytes son espacios en blanco.
- `istitle()`: Devuelve `True` si el objeto `

bytearray` es una cadena con título.

- `isupper()`: Devuelve `True` si todos los bytes están en mayúsculas.
- `join()`: Une los elementos de una lista usando el objeto `bytearray` como separador.
- `ljust()`: Devuelve una versión justificada a la izquierda del objeto `bytearray`.
- `lower()`: Convierte el objeto `bytearray` a minúsculas.
- `lstrip()`: Elimina los espacios en blanco a la izquierda del objeto `bytearray`.
- `maketrans()`: Devuelve una tabla de traducción que se puede usar en `translate()`.
- `partition()`: Devuelve una tupla donde el objeto `bytearray` se divide en tres partes.
- `pop()`: Elimina un byte en la posición especificada y lo devuelve.
- `remove()`: Elimina el primer byte con el valor especificado.
- `replace()`: Reemplaza un byte con otro.
- `reverse()`: Invierte el orden de los bytes en el objeto `bytearray`.
- `rfind()`: Devuelve el índice de la última aparición de una secuencia de bytes en el objeto `bytearray`.
- `rindex()`: Devuelve el índice de la última aparición de una secuencia de bytes en el objeto `bytearray`, genera un error si no se encuentra.
- `rjust()`: Devuelve una versión justificada a la derecha del objeto `bytearray`.
- `rpartition()`: Devuelve una tupla donde el objeto `bytearray` se divide en tres partes, comenzando por la derecha.
- `rsplit()`: Divide el objeto `bytearray` en una lista, comenzando por la derecha.
- `rstrip()`: Elimina los espacios en blanco a la derecha del objeto `bytearray`.
- `split()`: Divide el objeto `bytearray` en una lista.
- `splitlines()`: Divide el objeto `bytearray` en líneas.
- `startswith()`: Devuelve `True` si el objeto `bytearray` comienza con el prefijo especificado.
- `strip()`: Elimina los espacios en blanco al principio y al final del objeto `bytearray`.
- `swapcase()`: Intercambia mayúsculas y minúsculas.
- `title()`: Convierte la primera letra de cada palabra en mayúscula.
- `translate()`: Devuelve un objeto `bytearray` donde algunos bytes especificados se reemplazan con el byte descrito en un diccionario.
- `upper()`: Convierte el objeto `bytearray` a mayúsculas.
- `zfill()`: Rellena el objeto `bytearray` con ceros a la izquierda.

Métodos de los objetos `byte`:

Los objetos `byte` son inmutables, por lo que solo tienen un conjunto limitado de métodos:

- `capitalize()`: Convierte el primer carácter en mayúscula.
- `center()`: Devuelve una secuencia de bytes centrada.
- `count()`: Devuelve el número de ocurrencias de un byte en la secuencia de bytes.
- `decode()`: Decodifica la secuencia de bytes usando el codec especificado.
- `endswith()`: Devuelve `True` si la secuencia de bytes termina con el sufijo especificado.
- `expandtabs()`: Establece el tamaño de la tabulación de la secuencia de bytes.
- `find()`: Devuelve el índice de la primera aparición de una secuencia de bytes.
- `fromhex()`: Crea un objeto `byte` desde una cadena hexadecimal.
- `hex()`: Convierte el byte en dos caracteres hexadecimales.
- `index()`: Devuelve el índice de la primera aparición de una secuencia de bytes, genera un error si no se encuentra.
- `isalnum()`: Devuelve `True` si el byte es alfanumérico.
- `isalpha()`: Devuelve `True` si el byte es alfabético.
- `isdigit()`: Devuelve `True` si el byte es un dígito.
- `islower()`: Devuelve `True` si el byte está en minúsculas.
- `isspace()`: Devuelve `True` si el byte es un espacio en blanco.
- `istitle()`: Devuelve `True` si el byte es una letra con título.
- `isupper()`: Devuelve `True` si el byte está en mayúsculas.
- `join()`: Une los elementos de una lista usando el byte como separador.
- `ljust()`: Devuelve una versión justificada a la izquierda del byte.
- `lower()`: Convierte el byte a minúsculas.
- `lstrip()`: Elimina los espacios en blanco a la izquierda del byte.
- `maketrans()`: Devuelve una tabla de traducción que se puede usar en `translate()`.
- `partition()`: Devuelve una tupla donde el byte se divide en tres partes.
- `replace()`: Reemplaza un byte con otro.
- `rfind()`: Devuelve el índice de la última aparición de una secuencia de bytes.
- `rindex()`: Devuelve el índice de la última aparición de una secuencia de bytes, genera un error si no se encuentra.
- `rjust()`: Devuelve una versión justificada a la derecha del byte.
- `rpartition()`: Devuelve una tupla donde el byte se divide en tres partes, comenzando por la derecha.
- `rsplit()`: Divide la secuencia de bytes en una lista, comenzando por la derecha.
- `rstrip()`: Elimina los espacios en blanco a la derecha del byte.
- `split()`: Divide la secuencia de bytes en una lista.
- `splitlines()`: Divide la secuencia de bytes en líneas.
- `startswith()`: Devuelve `True` si la secuencia de bytes comienza con el prefijo especificado.
- `strip()`: Elimina los espacios en blanco al principio y al final del byte.
- `swapcase()`: Intercambia mayúsculas y minúsculas.
- `title()`: Convierte la primera letra de cada palabra en mayúscula.
- `translate()`: Devuelve una secuencia de bytes donde algunos bytes especificados se reemplazan con el byte descrito en un diccionario.
- `upper()`: Convierte el byte a mayúsculas.
- `zfill()`: Rellena el byte con ceros a la izquierda.

Métodos de los objetos `set`:

- `add()`: Agrega un elemento al conjunto.

- `clear()`: Elimina todos los elementos del conjunto.
- `copy()`: Devuelve una copia superficial del conjunto.
- `difference()`: Devuelve un conjunto que contiene la diferencia entre dos conjuntos.
- `difference_update()`: Elimina los elementos en común con otro conjunto.
- `discard()`: Elimina el

  elemento especificado del conjunto.
- `intersection()`: Devuelve un conjunto que contiene la intersección de dos conjuntos.
- `intersection_update()`: Actualiza el conjunto con la intersección de sí mismo y otro(s) conjunto(s).
- `isdisjoint()`: Devuelve `True` si dos conjuntos no tienen elementos en común.
- `issubset()`: Devuelve `True` si todos los elementos de un conjunto están presentes en otro conjunto.
- `issuperset()`: Devuelve `True` si un conjunto contiene todos los elementos de otro conjunto.
- `pop()`: Elimina un elemento del conjunto y lo devuelve.
- `remove()`: Elimina el elemento especificado del conjunto.
- `symmetric_difference()`: Devuelve un conjunto que contiene todos los elementos que no son comunes entre dos conjuntos.
- `symmetric_difference_update()`: Actualiza el conjunto con la diferencia simétrica de sí mismo y otro conjunto.
- `union()`: Devuelve un conjunto que contiene la unión de conjuntos.
- `update()`: Actualiza el conjunto con la unión de sí mismo y otro(s) conjunto(s).

Métodos de los objetos `frozenset`:

Los objetos `frozenset` son inmutables, por lo que solo tienen un conjunto limitado de métodos, que son los mismos que los métodos de los objetos `set`, pero no tienen métodos que modifiquen el conjunto.

Métodos de los objetos `range`:

- `start`: Devuelve el inicio del rango.
- `stop`: Devuelve el final del rango.
- `step`: Devuelve el tamaño del paso del rango.
- `__contains__`: Verifica si un valor está dentro del rango.
- `__getitem__`: Devuelve el elemento en el índice especificado.

Métodos de los objetos `dict`:

- `clear()`: Elimina todos los elementos del diccionario.
- `copy()`: Devuelve una copia superficial del diccionario.

- `fromkeys()`: Devuelve un diccionario con las claves y el valor especificados.
- `get()`: Devuelve el valor de la clave especificada.
- `items()`: Devuelve una lista que contiene una tupla para cada par clave-valor.
- `keys()`: Devuelve una lista que contiene las claves del diccionario.
- `pop()`: Elimina el elemento con la clave especificada y devuelve su valor.
- `popitem()`: Elimina el último par clave-valor insertado.
- `setdefault()`: Devuelve el valor de la clave especificada. Si la clave no existe, inserta la clave con el valor especificado.
- `update()`: Actualiza el diccionario con los pares clave-valor especificados.
- `values()`: Devuelve una lista que contiene los valores del diccionario.

Métodos de los objetos `tuple`:

Los objetos `tuple` son inmutables, por lo que solo tienen un conjunto limitado de métodos:

- `count()`: Devuelve el número de veces que aparece un elemento en la tupla.
- `index()`: Devuelve el índice del primer elemento con el valor especificado.

Métodos de los objetos `bool`:

Los objetos booleanos en Python son simples (solo `True` o `False`) y no tienen métodos específicos.

<<--- EJEMPLOS DE USO DE ESTOS METODOS --->>

Ejemplos de Métodos de Listas (`list`):

```
lista = [1, 2, 3]
lista.append(4) # Agrega 4 al final: [1, 2, 3, 4]
lista.clear() # Limpia la lista: []
copia_lista = lista.copy() # Crea una copia: []
print(lista.count(2)) # Cuenta cuántas veces está el 2: 0
lista.extend([4, 5]) # Extiende con [4, 5]: [4, 5]
print(lista.index(4)) # Índice de 4: 0
lista.insert(0, 'a') # Inserta 'a' en índice 0: ['a', 4, 5]
print(lista.pop()) # Elimina y devuelve el último: 5
lista.remove('a') # Elimina 'a': [4]
lista.reverse() # Invierte la lista: [4]
lista.sort() # Ordena la lista: [4]
```

Ejemplos de Métodos de Cadenas (`str`):

```
cadena = "hola mundo"
print(cadena.capitalize()) # 'Hola mundo'
print(cadena.casefold()) # 'hola mundo'
print(cadena.center(20)) # '    hola mundo     '
print(cadena.count('o')) # 2
print(cadena.encode()) # b'hola mundo'
print(cadena.endswith('do')) # True
```

```
print(cadena.expandtabs(2)) # 'hola mundo'
print(cadena.find('mu')) # 5
print(cadena.format()) # 'hola mundo'
print(cadena.index('m')) # 5
print(cadena.isalnum()) # False
print(cadena.isalpha()) # False
print(cadena.isdigit()) # False
print(cadena.islower()) # True
print(cadena.isspace()) # False
print(cadena.istitle()) # False
print(cadena.isupper()) # False
print("-".join(['hola', 'mundo'])) # 'hola-mundo'
print(cadena.ljust(20)) # 'hola mundo         '
print(cadena.lower()) # 'hola mundo'
print(cadena.lstrip()) # 'hola mundo'
print(cadena.partition(' ')) # ('hola', ' ', 'mundo')
print(cadena.replace('o', '0')) # 'h0la mund0'
print(cadena.rfind('o')) # 7
print(cadena.rindex('o')) # 7
print(cadena.rjust(20)) # '         hola mundo'
print(cadena.rpartition(' ')) # ('hola', ' ', 'mundo')
print(cadena.rsplit()) # ['hola', 'mundo']
print(cadena.rstrip()) # 'hola mundo'
print(cadena.split()) # ['hola', 'mundo']
print(cadena.splitlines()) # ['hola mundo']
print(cadena.startswith('ho')) # True
print(cadena.strip()) # 'hola mundo'
print(cadena.swapcase()) # 'HOLA MUNDO'
print(cadena.title()) # 'Hola Mundo'
print(cadena.translate({111: 48})) # 'h0la mund0'
print(cadena.upper()) # 'HOLA MUNDO'
print(cadena.zfill(20)) # '0000000000hola mundo'
```

Ejemplos de Métodos de Diccionarios (`dict`):

```
diccionario = {'a': 1, 'b': 2}
diccionario.clear() # {}
copia_diccionario = diccionario.copy() # {}
print(diccionario.fromkeys(['a', 'b'], 0)) # {'a': 0, 'b': 0}
print(diccionario.get('a')) # None
print(diccionario.items()) # dict_items([])
print(diccionario.keys()) # dict_keys([])
print(diccionario.pop('a', 'No encontrado')) # 'No encontrado'
print(diccionario.popitem()) # Error, diccionario vacío
diccionario.setdefault('c', 3) # 3
diccionario.update({'d': 4}) # {'c': 3, 'd': 4}
print(diccionario.values()) # dict_values([3, 4])
```

Ejemplos de Métodos de Conjuntos (`set`):

```
conjunto = {1, 2, 3}
conjunto.add(4) # {1, 2, 3, 4}
conjunto.clear() # set()
copia_conjunto = conjunto.copy() # set()
print(conjunto.difference({2, 3})) # set()
conjunto.difference_update({2, 3}) # set()
conjunto.discard(2) # set()
print(conjunto.intersection({1, 4})) # set()
conjunto.intersection_update({1, 4}) # set()
print(conjunto.isdisjoint({4, 5})) # True
print(conjunto.issubset({1, 2, 3, 4})) # False
print(conjunto.issuperset({1, 2})) # False
print(conjunto.pop()) # Error, conjunto vacío
conjunto.remove(3) # Error, conjunto vacío
print(conjunto.symmetric_difference({2, 4})) # set()
conjunto.symmetric_difference_update({2, 4}) # set()
print(conjunto.union({5, 6})) # {5, 6}
conjunto.update({7, 8}) # {5, 6, 7, 8}
```

Ejemplos de métodos de `range`:

```
rango = range(1, 10, 2)

# start
print(rango.start)  # Salida: 1

# stop
print(rango.stop)  # Salida: 10

# step
print(rango.step)  # Salida: 2

# __contains__
print(5 in rango)  # Salida: True

# __getitem__
print(rango[2])  # Salida: 5
```

Ejemplos de métodos de `tuple`:

```
tupla = (1, 2, 3, 2, 4, 2)

# count()
print(tupla.count(2))  # Salida: 3

# index()
print(tupla.index(3))  # Salida: 2
```

Ejemplos de métodos de `bytes`:

```python
bytes_obj = b"ejemplo de bytes"

# capitalize()
print(bytes_obj.capitalize())  # Salida: b'Ejemplo de bytes'

# center()
print(bytes_obj.center(20))  # Salida: b'  ejemplo de bytes  '

# count()
print(bytes_obj.count(b"e"))  # Salida: 2

# decode()
print(bytes_obj.decode("utf-8"))  # Salida: 'ejemplo de bytes'

# endswith()
print(bytes_obj.endswith(b"bytes"))  # Salida: True

# expandtabs()
print(b"ejemplo\tde\tbytes".expandtabs(4))  # Salida: b'ejemplo  de  bytes'

# find()
print(bytes_obj.find(b"de"))  # Salida: 8

# fromhex()
print(bytes.fromhex('02 0A'))  # Salida: b'\x02\n'

# hex()
print(bytes_obj.hex())  # Salida: '656a656d706c6f206465206279746573'

# index()
print(bytes_obj.index(b"de"))  # Salida: 8

# isalnum()
print(bytes_obj.isalnum())  # Salida: False

# isalpha()
print(bytes_obj.isalpha())  # Salida: False

# isdigit()
print(b"12345".isdigit())  # Salida: True

# islower()
print(bytes_obj.islower())  # Salida: True

# isspace()
print(b"   ".isspace())  # Salida: True

# istitle()
print(b"Ejemplo De Bytes".istitle())  # Salida: True

# isupper()
print(b"EJEMPLO DE BYTES".isupper())  # Salida: True
```

```
# join()
print(b"-".join([b"uno", b"dos", b"tres"]))  # Salida: b'uno-dos-tres'
```