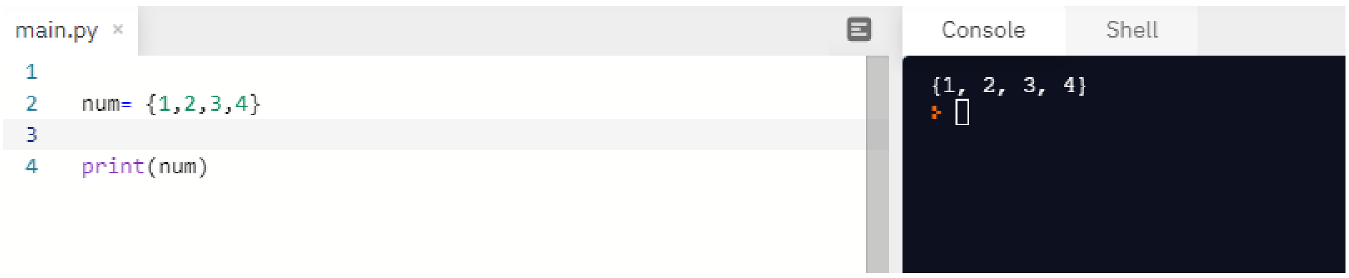
Programming Task - Sets and Tuples

Sets

Look up and write about the following methods that can be used on **sets** in Python. Give examples of each.

1. **add() - The set add( ) method adds a specified value to a set.**

For example in this program I have created the variable 'num' and assigned the set - '1','2','3' and '4' to the variable 'num'.

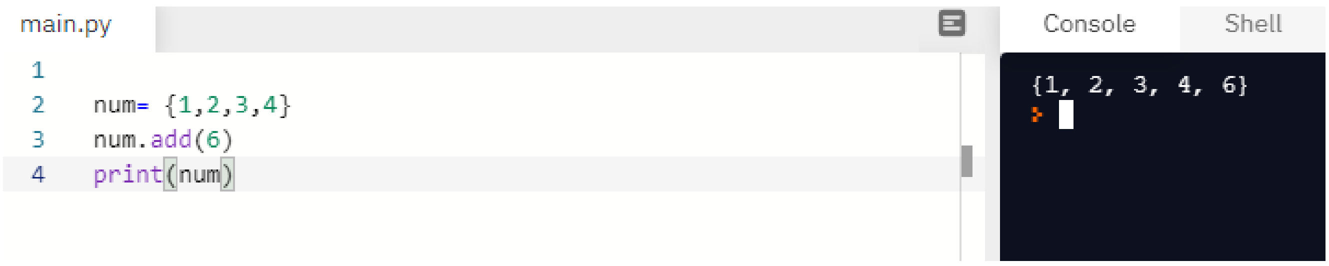


```
num= {1,2,3,4}

print(num)
```

```
{1, 2, 3, 4}
```

I will now include the 'add' method and add '6' to the set:

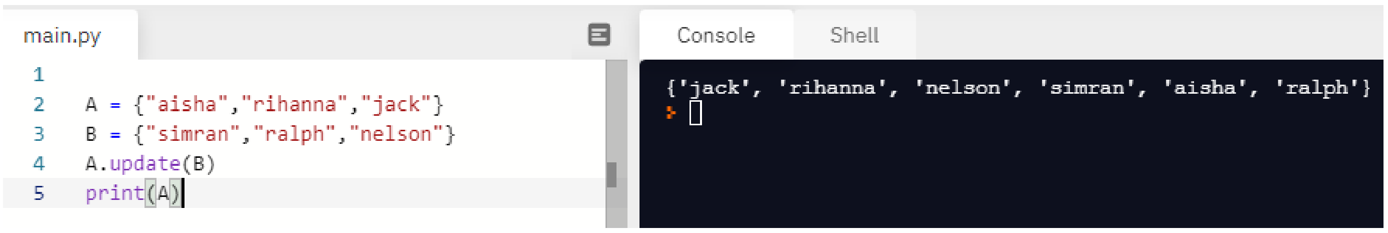


```
num= {1,2,3,4}
num.add(6)
print(num)
```

```
{1, 2, 3, 4, 6}
```

This added '6' to the set.

2. **update() - The set update( ) method updates a set, by adding items from another specified set.**



```
A = {"aisha","rihanna","jack"}
B = {"simran","ralph","nelson"}
A.update(B)
print(A)
```

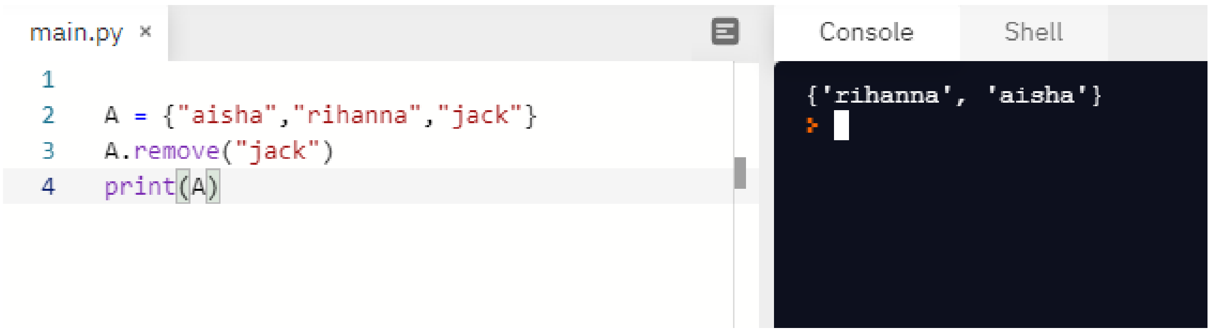
```
{'jack', 'rihanna', 'nelson', 'simran', 'aisha', 'ralph'}
```

In the program above I have created two lists - 'A' and 'B'. These both include variables, which are names. I then used the set 'update' method to add the variables from 'B' into 'A'. This resulted in the update method combining the two and making one set, instead of two.

Q. What is the difference between add() and update() methods? Explain:

The add() method adds a specified value into a set. However the update() method adds values from one set to another and combines the two sets, to make one.

3. **remove()- This method removes the specified value from the set.**

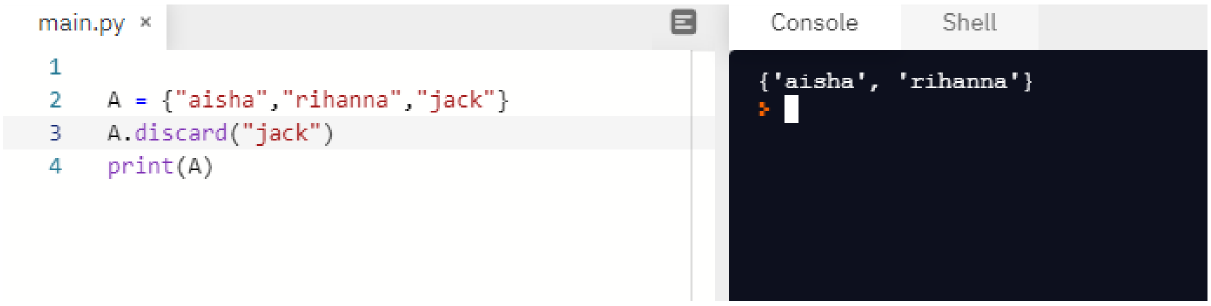


```python
A = {"aisha","rihanna","jack"}
A.remove("jack")
print(A)
```

```
{'rihanna', 'aisha'}
```

The remove() method, removed 'jack' from the set.

4. **discard()- This method also removes the specified item from the set.**



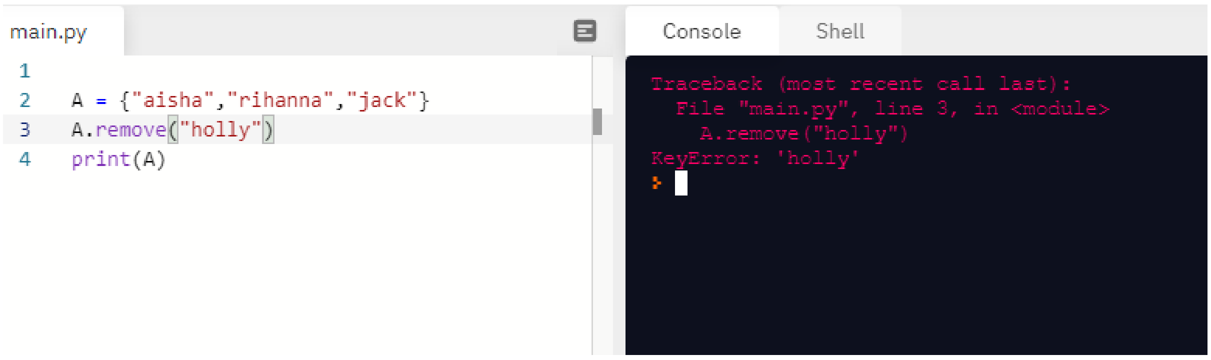
```python
A = {"aisha","rihanna","jack"}
A.discard("jack")
print(A)
```

```
{'aisha', 'rihanna'}
```

Q. What is the difference between remove() and discard() methods? Explain:

The difference between remove() and discard() is, the remove() method will raise an error message if the specified value does not exist within the set; however, the discard method will not.

Here is an example program of the difference between the two methods:

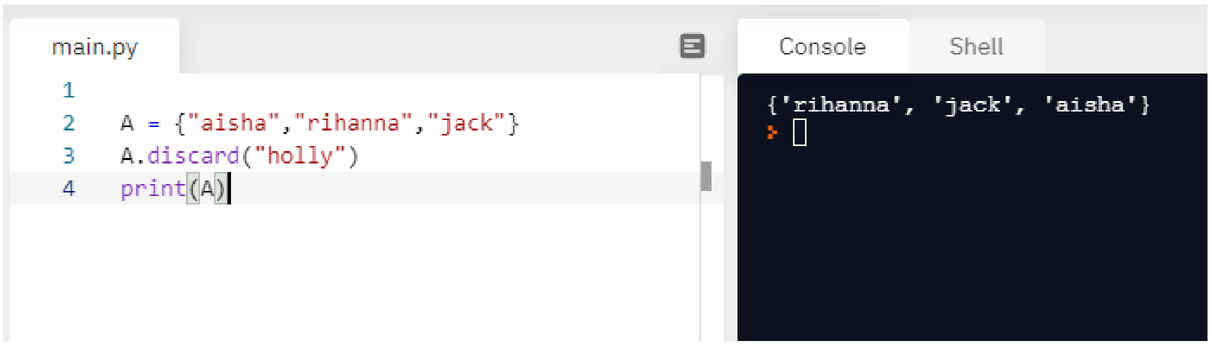


```
A = {"aisha","rihanna","jack"}
A.remove("holly")
print(A)
```

```
Traceback (most recent call last):
  File "main.py", line 3, in <module>
    A.remove("holly")
KeyError: 'holly'
```

In the above example I have used the remove() method and included a variable not in set 'A', this raised an error.

However, with the discard() method this was the result:

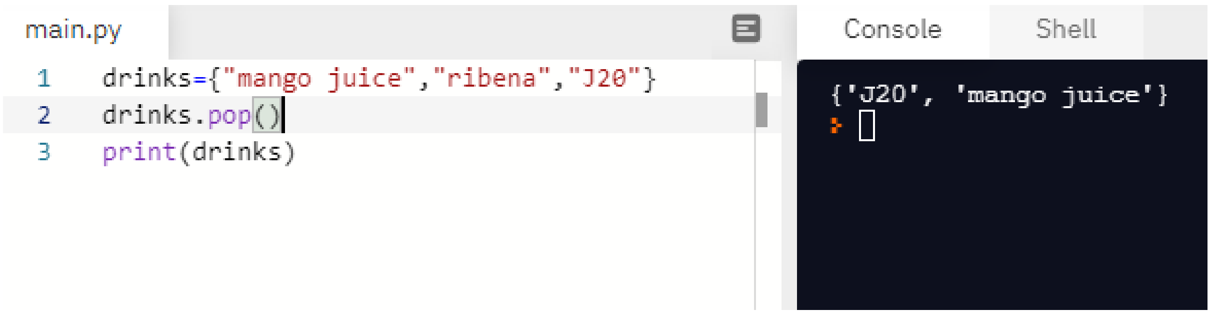


```
A = {"aisha","rihanna","jack"}
A.discard("holly")
print(A)
```

```
{'rihanna', 'jack', 'aisha'}
```

As shown in the example, using the discard() method - no error message appeared, the 3 variables in the set appeared in the output.

5. pop()- Removes a random item from the set.



```
drinks={"mango juice","ribena","J20"}
drinks.pop()
print(drinks)
```

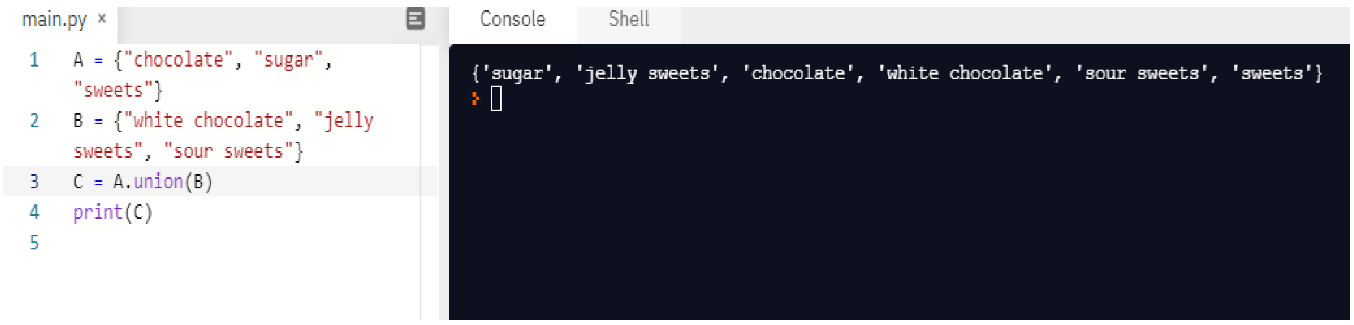
```
{'J20', 'mango juice'}
```

The above example shows the variable "ribena" was removed from the set.

Q. What is the drawback of using pop() with sets? Discuss:

A disadvantage of using the pop() method on sets is, a random item from the list will be removed - an item which may not want to be deleted by the user. In addition, once the item is removed for the set, the set loses the item and the set is updated, without the item.

6. **union()- The union() method returns a set which includes all items from the original set and all the values from the specified sets.**

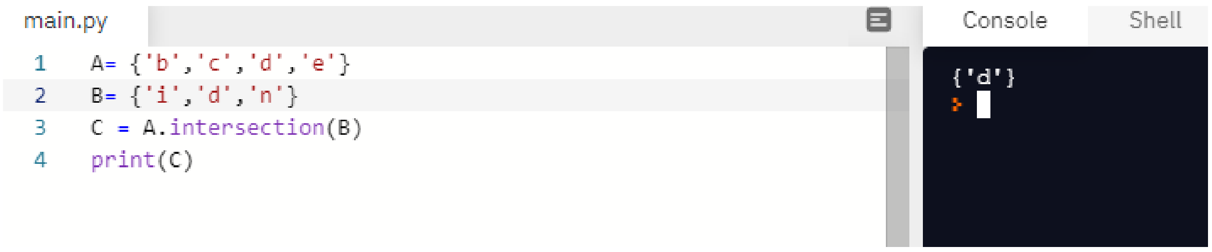


```python
A = {"chocolate", "sugar", "sweets"}
B = {"white chocolate", "jelly sweets", "sour sweets"}
C = A.union(B)
print(C)
```

```
{'sugar', 'jelly sweets', 'chocolate', 'white chocolate', 'sour sweets', 'sweets'}
```

The union() method combined 'A' and 'B' to make a new set - with values from both sets.

7. **intersection()- This method returns a set which incorporates the items, which exist in both sets(or in all sets - if comparison is between more than two sets).**

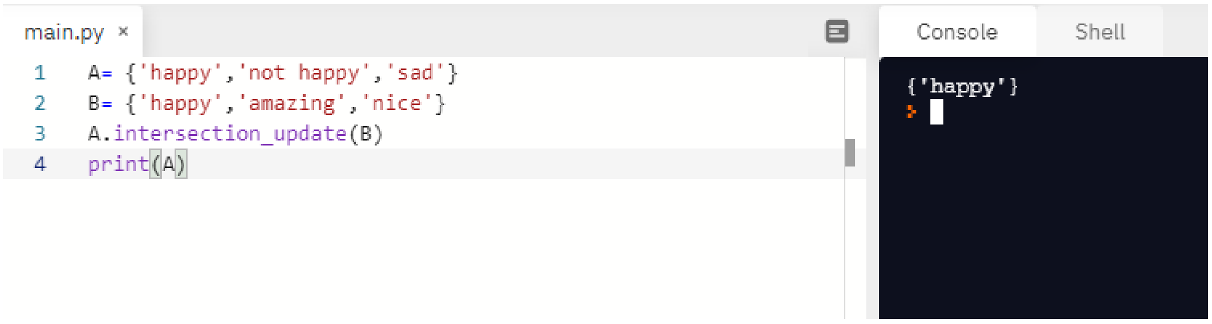


```python
A= {'b','c','d','e'}
B= {'i','d','n'}
C = A.intersection(B)
print(C)
```

```
{'d'}
```

The intersection() method returns a set which includes the variable which appears in both sets, which in this instance is 'd'.

8. **intersection_update()- Removes the items that are not present in both sets.**



```python
A= {'happy','not happy','sad'}
B= {'happy','amazing','nice'}
A.intersection_update(B)
print(A)
```

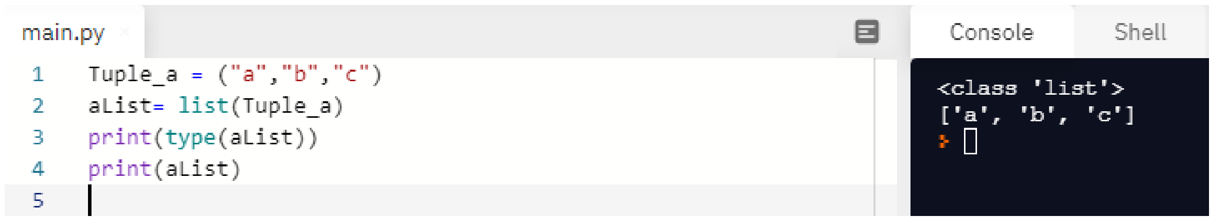
```
{'happy'}
```

The intersection_update() method removes the items which are not present in both sets, in this example this would be all items except for 'happy'.

Tuples

1. How can you convert a tuple to a list and vice versa. Give code examples to demonstrate.

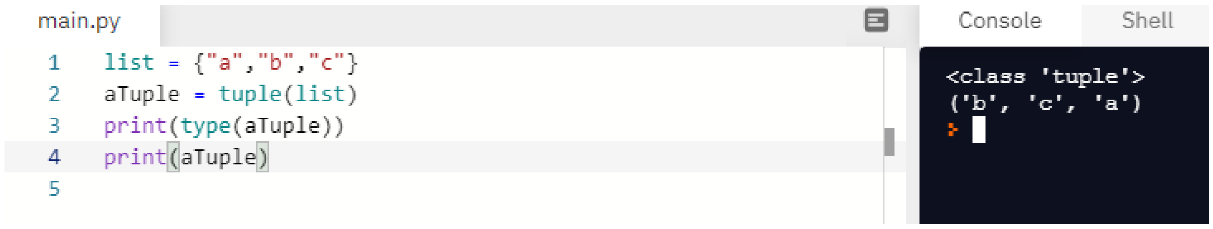
   Tuple to a list:



```
main.py
Tuple_a = ("a","b","c")
aList= list(Tuple_a)
print(type(aList))
print(aList)
```

```
<class 'list'>
['a', 'b', 'c']
```

   List to a Tuple:

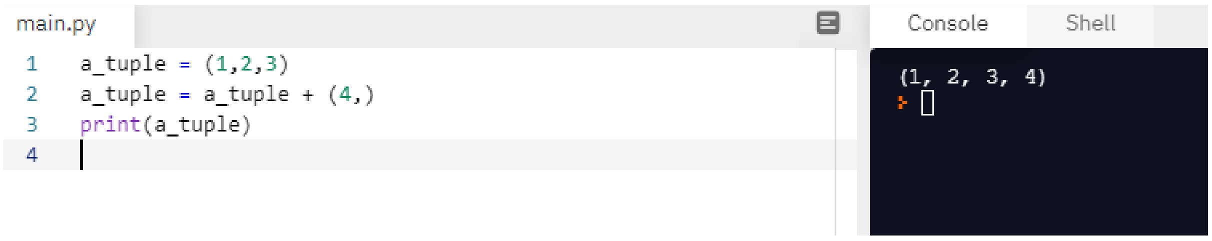


```
main.py
list = {"a","b","c"}
aTuple = tuple(list)
print(type(aTuple))
print(aTuple)
```

```
<class 'tuple'>
('b', 'c', 'a')
```

2. Tuples are immutable, that is they cannot be changed. Discuss a few ways of how you can get around adding / removing items in a tuple. Support your discussion with programs.

   1. Add: using +, creates a new tuple



```
main.py
a_tuple = (1,2,3)
a_tuple = a_tuple + (4,)
print(a_tuple)
```

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

   2. Remove:

3. Write a Python program to replace the last value of tuples in a list.
   Sample list: [(10, 20, 40), (40, 50, 60), (70, 80, 90)]
   Expected Output: [(10, 20, 100), (40, 50, 100), (70, 80, 100)]

4. Write a Python program to remove an empty tuple(s) from a list of tuples.
   Sample data: [(), (), ('',), ('a', 'b'), ('a', 'b', 'c'), ('d')]

   Expected output: [('',), ('a', 'b'), ('a', 'b', 'c'), 'd']