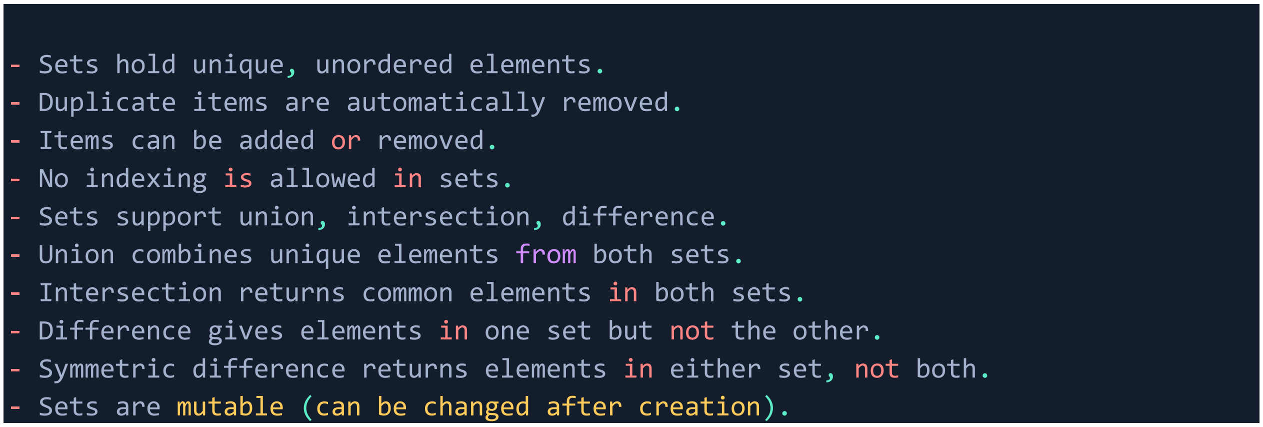
Lab Assignment 15

Student Name : Aryan Verma

Student Id: AF0417100

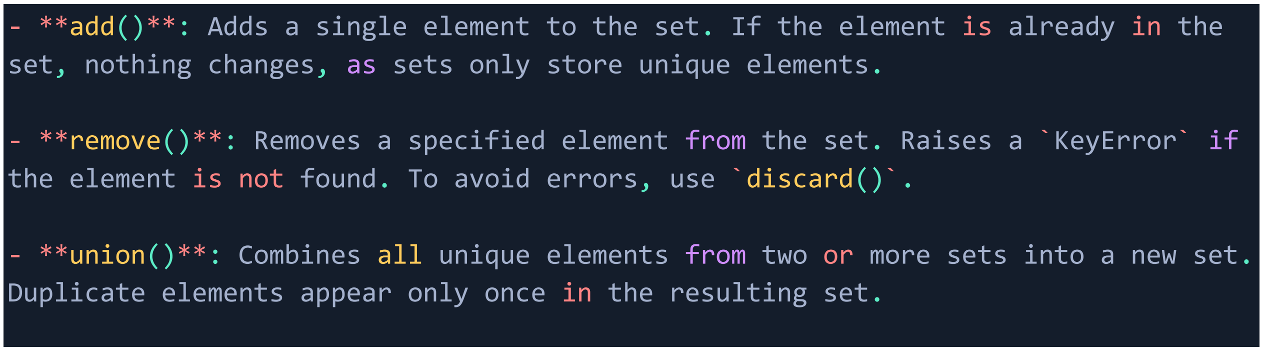
Topic: Sets

KEY POINTS OF SETS:

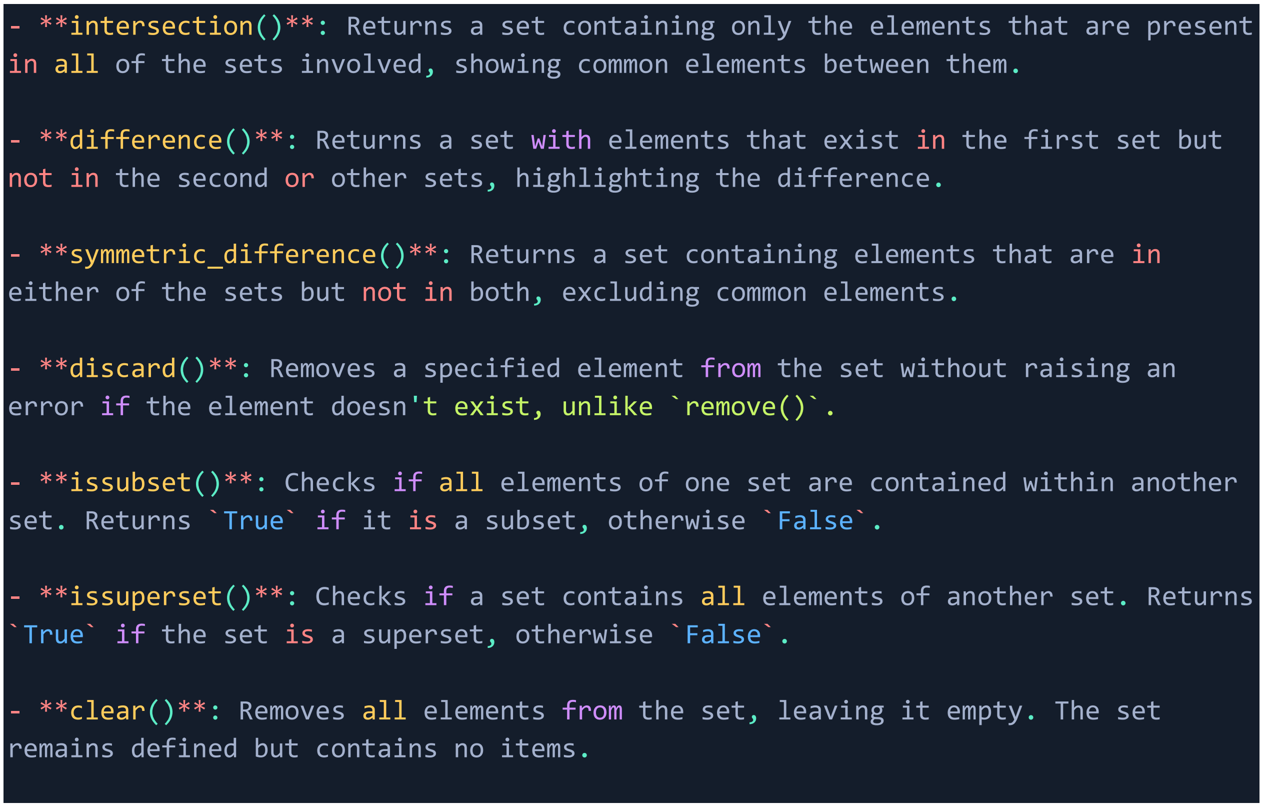


```
- Sets hold unique, unordered elements.
- Duplicate items are automatically removed.
- Items can be added or removed.
- No indexing is allowed in sets.
- Sets support union, intersection, difference.
- Union combines unique elements from both sets.
- Intersection returns common elements in both sets.
- Difference gives elements in one set but not the other.
- Symmetric difference returns elements in either set, not both.
- Sets are mutable (can be changed after creation).
```

SOME FUNCTIONS USED IN SETS:



```
- **add()**: Adds a single element to the set. If the element is already in the
set, nothing changes, as sets only store unique elements.

- **remove()**: Removes a specified element from the set. Raises a `KeyError` if
the element is not found. To avoid errors, use `discard()`.

- **union()**: Combines all unique elements from two or more sets into a new set.
Duplicate elements appear only once in the resulting set.
```

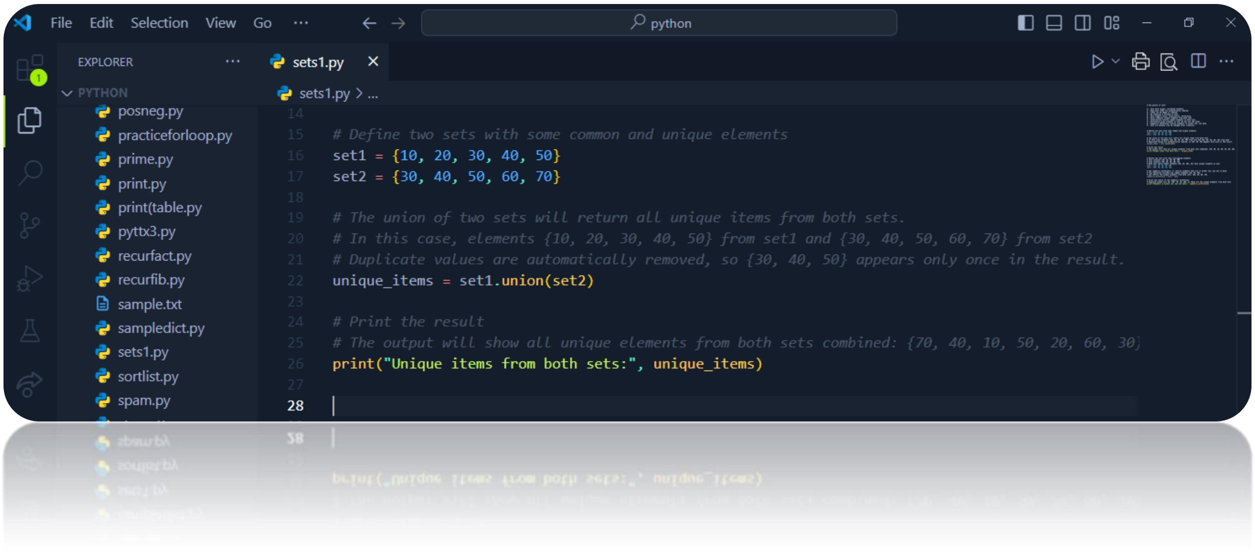


```
- **intersection()**: Returns a set containing only the elements that are present
in all of the sets involved, showing common elements between them.

- **difference()**: Returns a set with elements that exist in the first set but
not in the second or other sets, highlighting the difference.

- **symmetric_difference()**: Returns a set containing elements that are in
either of the sets but not in both, excluding common elements.

- **discard()**: Removes a specified element from the set without raising an
error if the element doesn't exist, unlike `remove()`.

- **issubset()**: Checks if all elements of one set are contained within another
set. Returns `True` if it is a subset, otherwise `False`.

- **issuperset()**: Checks if a set contains all elements of another set. Returns
`True` if the set is a superset, otherwise `False`.

- **clear()**: Removes all elements from the set, leaving it empty. The set
remains defined but contains no items.
```

1. Write a Python program to Get Only unique items from two sets.

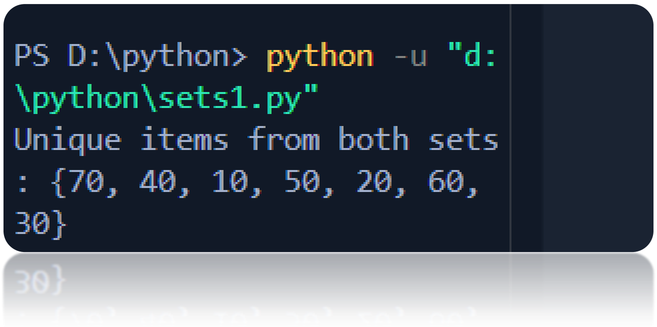
Input: set1 = {10, 20, 30, 40, 50} set2 = {30, 40, 50, 60, 70}
Output: {70, 40, 10, 50, 20, 60, 30}

Ans:



```python

# Define two sets with some common and unique elements
set1 = {10, 20, 30, 40, 50}
set2 = {30, 40, 50, 60, 70}

# The union of two sets will return all unique items from both sets.
# In this case, elements {10, 20, 30, 40, 50} from set1 and {30, 40, 50, 60, 70} from set2
# Duplicate values are automatically removed, so {30, 40, 50} appears only once in the result.
unique_items = set1.union(set2)

# Print the result
# The output will show all unique elements from both sets combined: {70, 40, 10, 50, 20, 60, 30]
print("Unique items from both sets:", unique_items)

```

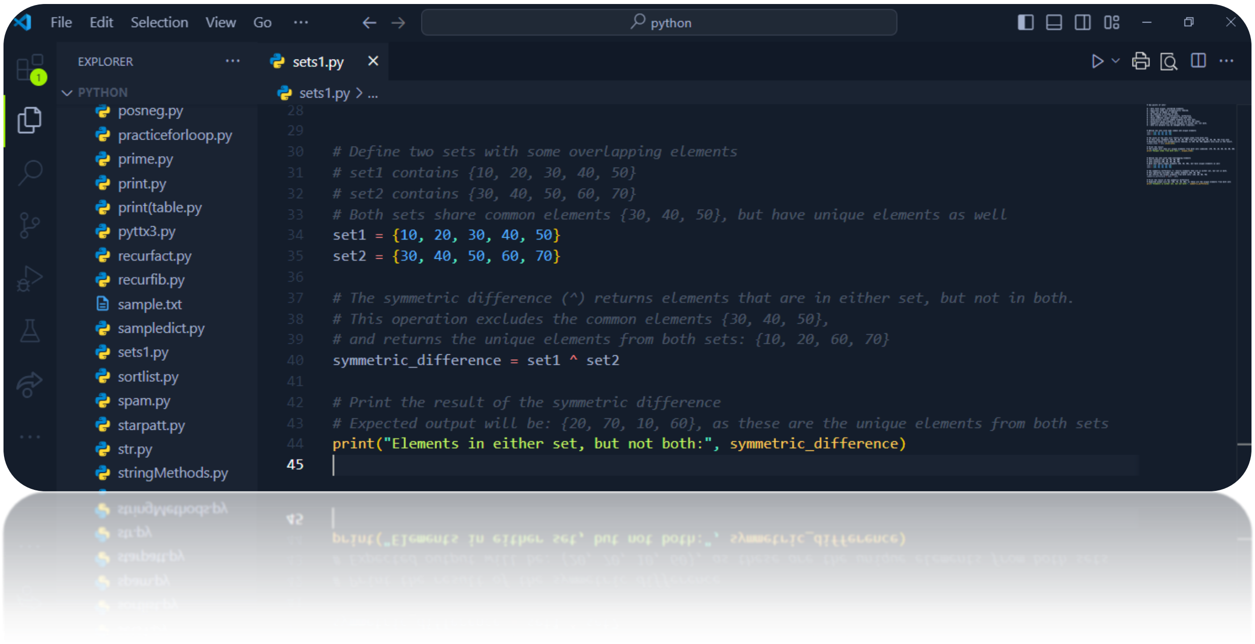
Output:



```
PS D:\python> python -u "d:
\python\sets1.py"
Unique items from both sets
: {70, 40, 10, 50, 20, 60,
30}
```

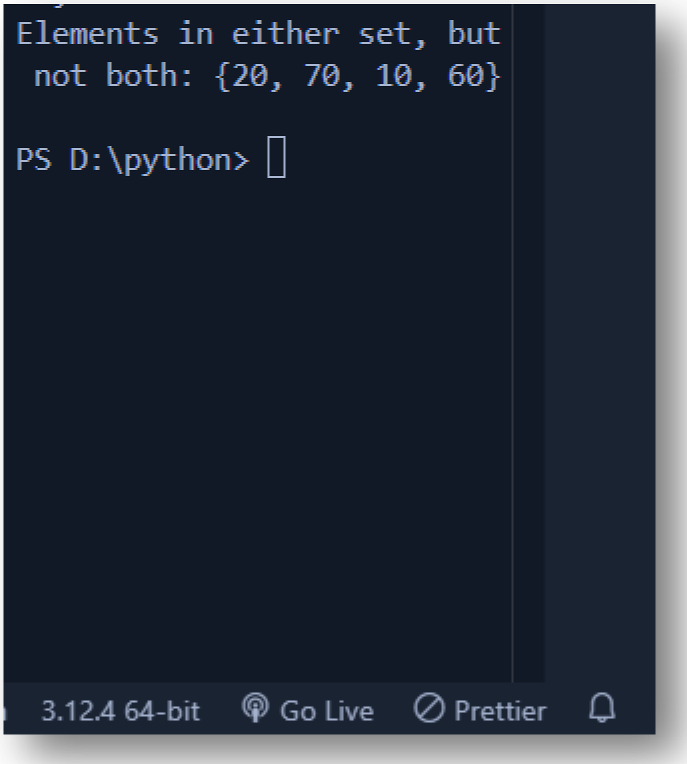
2. Write a Python program to Return a set of elements present in Set A or B, but not both. Input: set1 = {10, 20, 30, 40, 50} set2 = {30, 40, 50, 60, 70} Output: {20, 70, 10, 60}

Ans:



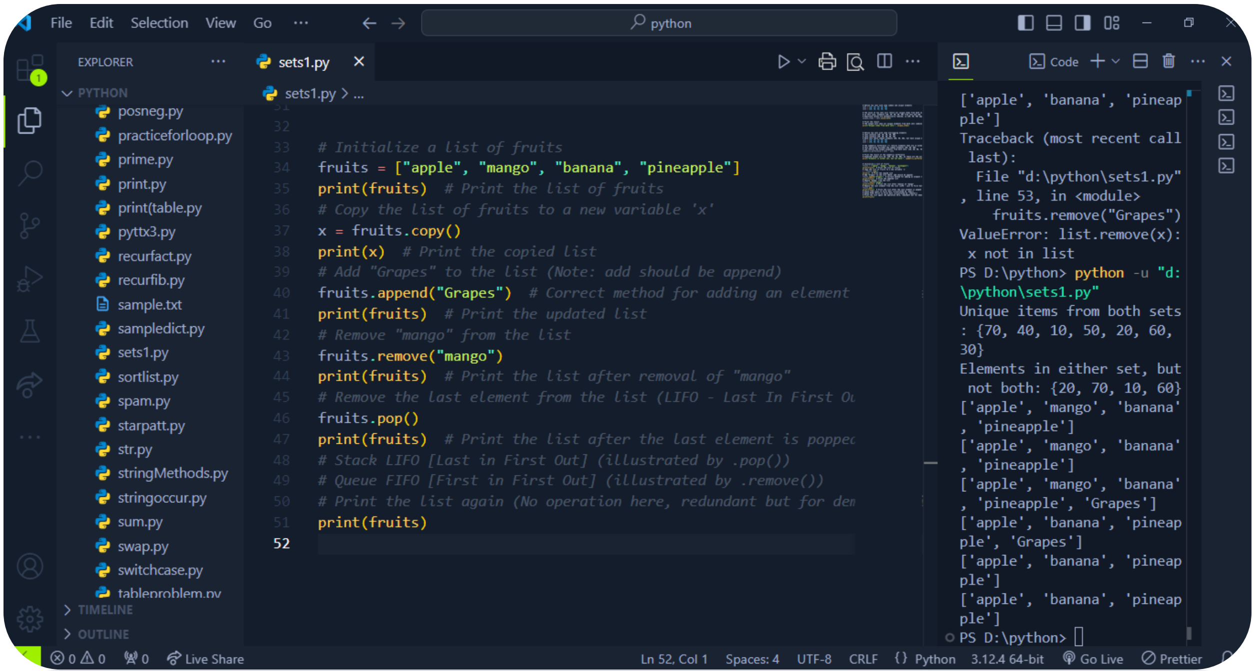
```python


# Define two sets with some overlapping elements
# set1 contains {10, 20, 30, 40, 50}
# set2 contains {30, 40, 50, 60, 70}
# Both sets share common elements {30, 40, 50}, but have unique elements as well
set1 = {10, 20, 30, 40, 50}
set2 = {30, 40, 50, 60, 70}

# The symmetric difference (^) returns elements that are in either set, but not in both.
# This operation excludes the common elements {30, 40, 50},
# and returns the unique elements from both sets: {10, 20, 60, 70}
symmetric_difference = set1 ^ set2

# Print the result of the symmetric difference
# Expected output will be: {20, 70, 10, 60}, as these are the unique elements from both sets
print("Elements in either set, but not both:", symmetric_difference)

```

output:



```
Elements in either set, but
 not both: {20, 70, 10, 60}

PS D:\python> 
```

Sample program with output :



```python

# Initialize a list of fruits
fruits = ["apple", "mango", "banana", "pineapple"]
print(fruits)  # Print the list of fruits
# Copy the list of fruits to a new variable 'x'
x = fruits.copy()
print(x)  # Print the copied list
# Add "Grapes" to the list (Note: add should be append)
fruits.append("Grapes")  # Correct method for adding an element
print(fruits)  # Print the updated list
# Remove "mango" from the list
fruits.remove("mango")
print(fruits)  # Print the list after removal of "mango"
# Remove the last element from the list (LIFO - Last In First Ou
fruits.pop()
print(fruits)  # Print the list after the last element is poppec
# Stack LIFO [Last in First Out] (illustrated by .pop())
# Queue FIFO [First in First Out] (illustrated by .remove())
# Print the list again (No operation here, redundant but for den
print(fruits)

```

```
['apple', 'banana', 'pineap
ple']
Traceback (most recent call
 last):
  File "d:\python\sets1.py"
, line 53, in <module>
    fruits.remove("Grapes")
ValueError: list.remove(x):
 x not in list
PS D:\python> python -u "d:
\python\sets1.py"
Unique items from both sets
: {70, 40, 10, 50, 20, 60,
30}
Elements in either set, but
 not both: {20, 70, 10, 60}
['apple', 'mango', 'banana'
, 'pineapple']
['apple', 'mango', 'banana'
, 'pineapple']
['apple', 'mango', 'banana'
, 'pineapple', 'Grapes']
['apple', 'banana', 'pineap
ple', 'Grapes']
['apple', 'banana', 'pineap
ple']
['apple', 'banana', 'pineap
ple']
PS D:\python> 
```