Day 19

Sets in Python:

What is a Set?

A set is a collection of unique values.

This means:

- No duplicate elements
- Unordered (no fixed index)
- Written using { }

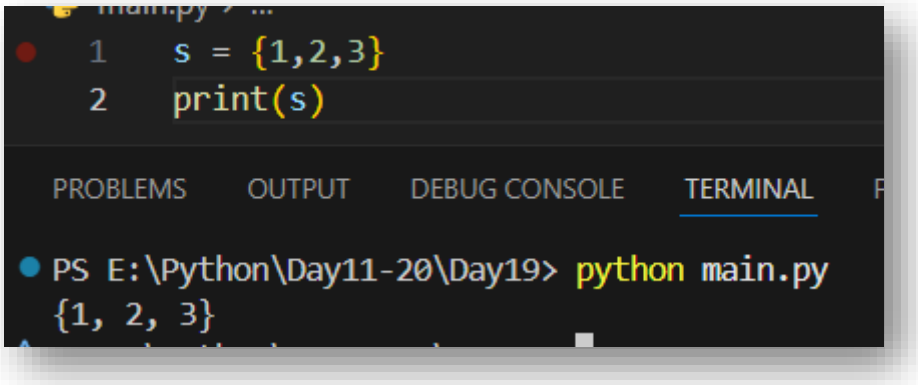
Why Do We Use Sets?

- To remove duplicates
- To perform mathematical set operations
- Fast checking of values (in)

Creating a Set:

numbers = {1, 2, 3, 4}
print(numbers)

Example: a basic example of set.



Example: when there be duplicate items in the set, they automatically get removed.

```
1   s = {1,2,3,4,4}
2   print(s)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  POR

PS E:\Python\Day11-20\Day19> python main.py
{1, 2, 3, 4}
```

Example: they are unordered.

```
1   s = {"Aditya", "Utsav", 4,5,6,6}
2   print(s)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS E:\Python\Day11-20\Day19> python main.py
{4, 5, 6, 'Aditya', 'Utsav'}
```

Example: type(set)

```
1   s = {"Aditya", "Utsav", 4,5,6,6}
2   print(type(s))
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS E:\Python\Day11-20\Day19> python main.py
<class 'set'>
```

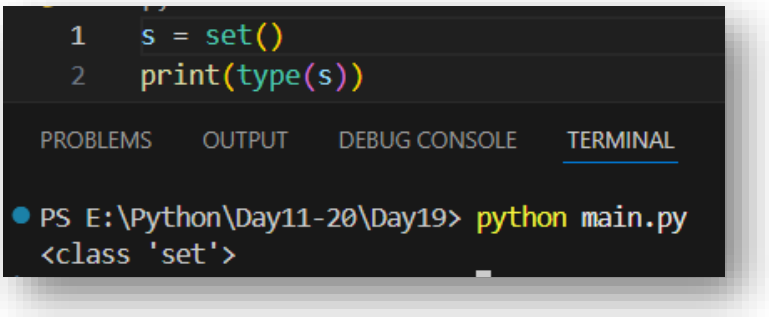
Example: what be the type of the set which has no element? Dictionary.

```
1   s = {}
2   print(type(s))
```

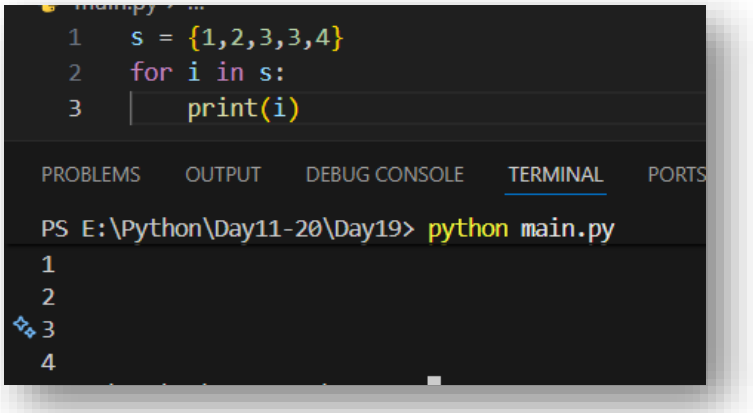
```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS E:\Python\Day11-20\Day19> python main.py
<class 'dict'>
```

Example: thus to create an empty set we uses set()



Example: accessing the elements of the set.

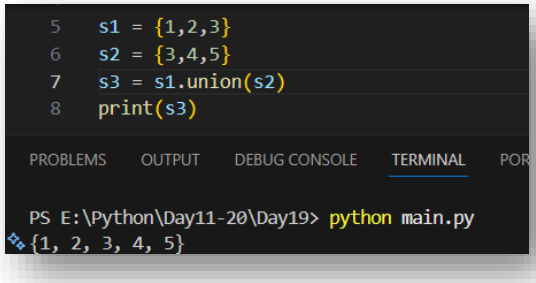


Summary:

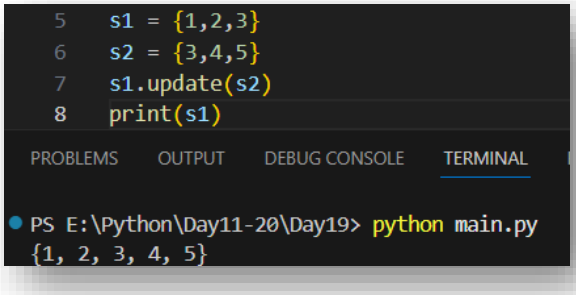
- Set stores unique values
- No order, no index
- Very fast operations
- Used to remove duplicates
- Supports union, intersection, difference

Set Methods in Python:

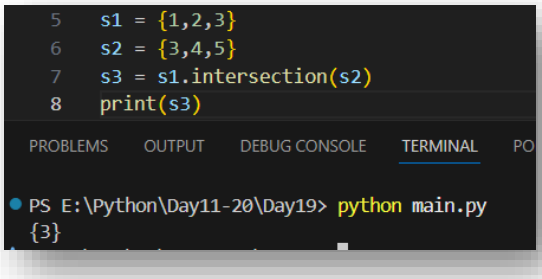
Example: union() -> it will not change the original set.



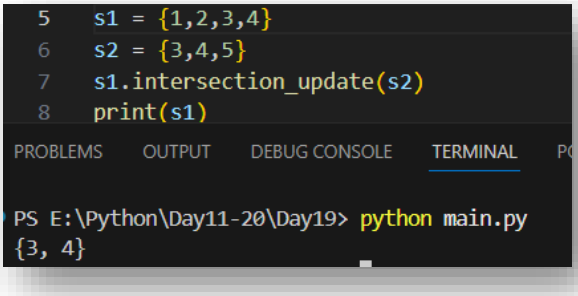
Example: update() -> it will change the original set.



Example: intersect()



Example: intersection_update()



Example: what if there are no common elements for intersection? Then an empty set will be returned.

```
5   s1 = {1,2}
6   s2 = {3,4,5}
7   s3 = s1.intersection(s2)
8   print(s3)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS E:\Python\Day11-20\Day19> python main.py
set()
```

Example: symmetric difference

```
5   s1 = {1,2}
6   s2 = {3,4,5}
7   s3 = s1.symmetric_difference(s2)
8   print(s3)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS E:\Python\Day11-20\Day19> python main.py
{1, 2, 3, 4, 5}
```

Example: symmetric difference update.

```
5   s1 = {1,2}
6   s2 = {3,4,5}
7   s1.symmetric_difference_update(s2)
8   print(s1)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS E:\Python\Day11-20\Day19> python main.py
{1, 2, 3, 4, 5}
```

Example: difference()

```
5   s1 = {1,2, 3,4,5,56,7}
6   s2 = {3,4,5}
7   s3 = s1.difference(s2)
8   print(s3)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS E:\Python\Day11-20\Day19> python main.py
{56, 1, 2, 7}
```

Example: difference_update()

```
5   s1 = {1,2, 3,4,5,56,7}
6   s2 = {3,4,5}
7   s1.difference_update(s2)
8   print(s1)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS E:\Python\Day11-20\Day19> python main.py
{1, 2, 7, 56}
```

Example: isdisjoint()

```
5   s1 = {1,2, 3,4,5,56,7}
6   s2 = {3,4,5}
7   print(s1.isdisjoint(s2))
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS E:\Python\Day11-20\Day19> python main.py
False
```

Example: issuperset()

```
5   s1 = {1,2, 3,4,5,56,7}
6   s2 = {3,4,5}
7   print(s1.issuperset(s2))
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS E:\Python\Day11-20\Day19> python main.py
True
```

Example: issubset()

```
5   s1 = {1,2, 3,4,5,56,7}
6   s2 = {3,4,5}
7   print(s1.issubset(s2))
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS E:\Python\Day11-20\Day19> python main.py
False
```

Example: add()

```
5   s1 = {1,2, 3,4,5,56,7}
6   s2 = {3,4,5}
7   s1.add("Aditya")
8   print(s1)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS E:\Python\Day11-20\Day19> python main.py
{1, 2, 3, 4, 5, 'Aditya', 7, 56}
```

Example: remove()

```
5   s1 = {1,2, 3,4,5,56,7}
6   s2 = {3,4,5}
7   s1.remove(7)
8   print(s1)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS E:\Python\Day11-20\Day19> python main.py
{1, 2, 3, 4, 5, 56}
```

Example: discard()

```
5   s1 = {1,2, 3,4,5,56,7}
6   s2 = {3,4,5}
7   s1.discard(7)
8   print(s1)
```

```
PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL

PS E:\Python\Day11-20\Day19> python main.py
{1, 2, 3, 4, 5, 56}
```

Example: pop()

```
5   s1 = {1,2, 3,4,5,56,7}
6   s2 = {3,4,5}
7   s1.pop()
8   print(s1)

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  POR

PS E:\Python\Day11-20\Day19> python main.py
{2, 3, 4, 5, 7, 56}
```

Example: del -> delete entire set.

```
5   s1 = {1,2, 3,4,5,56,7}
6   del s1
7   print(s1)

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  PORTS

PS E:\Python\Day11-20\Day19> python main.py
Traceback (most recent call last):
  File "E:\Python\Day11-20\Day19\main.py", line 7, in <module>
    print(s1)
          ^^
NameError: name 's1' is not defined
```

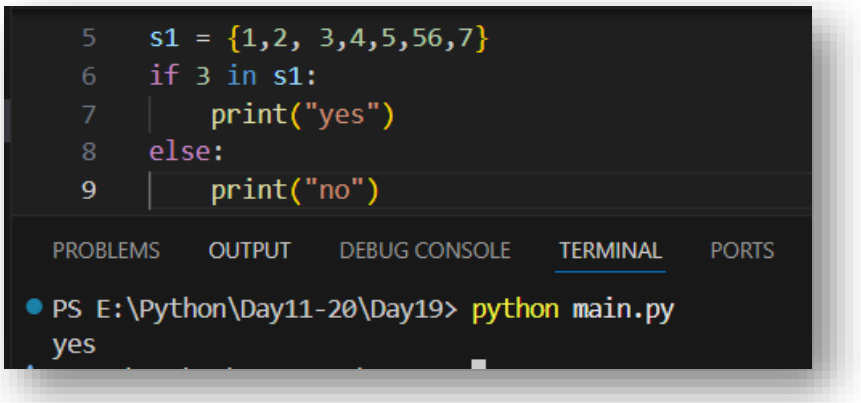
Example: clear()

```
5   s1 = {1,2, 3,4,5,56,7}
6   s1.clear()
7   print(s1)

PROBLEMS  OUTPUT  DEBUG CONSOLE  TERMINAL  P

PS E:\Python\Day11-20\Day19> python main.py
set()
```

Example: check if item exists.



Summary:

- Set methods modify or compare sets
- add, update → add elements
- remove, discard, pop, clear → remove elements
- union, intersection, difference → set operations
- issubset, issuperset, isdisjoint → relationships

--The End--