

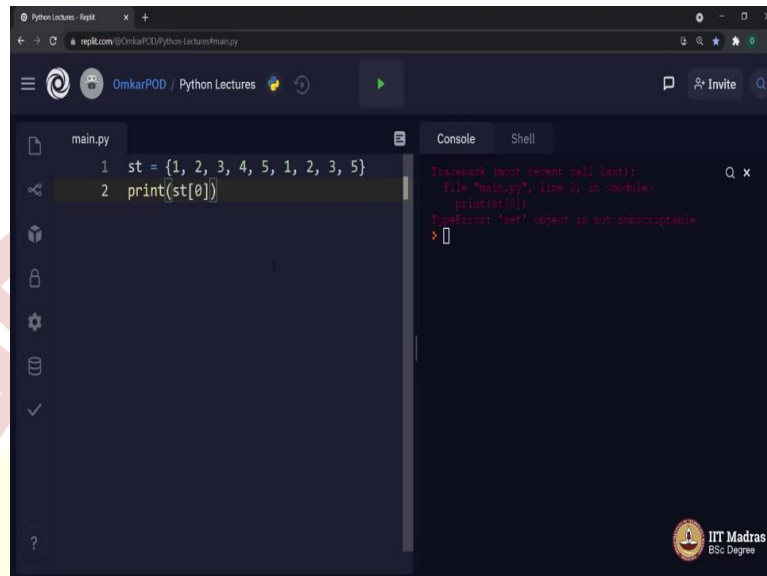


IIT Madras

ONLINE DEGREE

Programing in Python
Professor. Sudarshan Iyengar
Department of Computer Science and Engineering
Indian Institute of Technology Ropar
More on Sets

(Refer Slide Time: 00:16)

A screenshot of a Python REPL interface. The left pane shows a file named 'main.py' with two lines of code: '1 st = {1, 2, 3, 4, 5, 1, 2, 3, 5}' and '2 print(st[0])'. The right pane shows the console output, which includes a traceback error: 'Traceback (most recent call last): File "main.py", line 2, in module: print(st[0]) TypeError: 'set' object is not subscriptable'. The IIT Madras logo is visible in the bottom right corner of the interface.

```
main.py
1 st = {1, 2, 3, 4, 5, 1, 2, 3, 5}
2 print(st[0])

Console
Traceback (most recent call last):
  File "main.py", line 2, in module:
    print(st[0])
TypeError: 'set' object is not subscriptable
> []
```

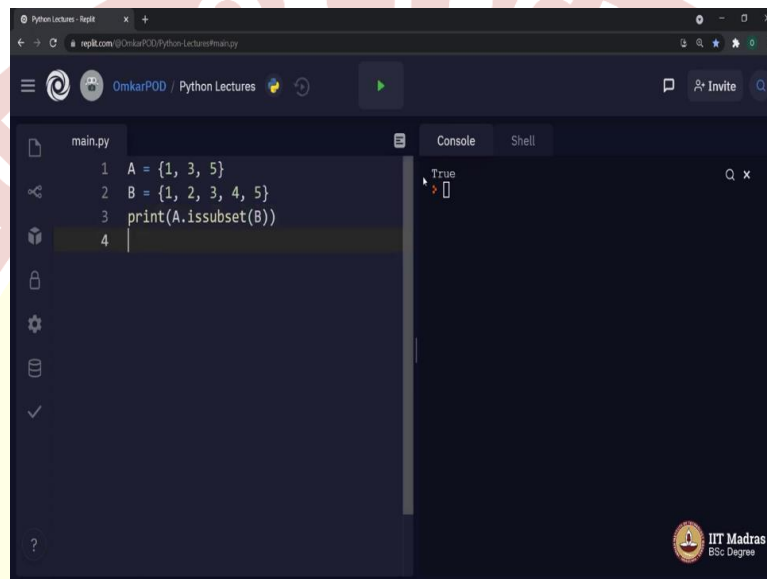
Hello, Python students. In this lecture, we will discuss fourth component of Python which is set. If you look at this example the notation of these brackets, which are used for sets are curly brackets, which are similar to dictionary, whereas values inside follow a pattern, which is like a list or a tuple.

So, the question is what is different about set as compared to other three components? First, set does not allow duplicate values. For example, if we add these duplicate values and try to print the set still, we will get only the unique values in that set and all the duplicates will be discarded. Second, set is an unordered entity, which means we cannot say that this element 1 is at zeroth index or it is the first element in set and because of that, if we try to execute something like this a computer will give us an error, set object is not subscriptable. As I mentioned, sets do not have a notion of order. Set is simply a collection of values.

Third, with respect to mutability set is peculiar. Set itself is mutable, but every value inside set has to be immutable and hashable, which means you can add values into a set, but whatever values which we add, those values has to be immutable. Hence, we cannot add a list or a dictionary or a tuple containing a list or a dictionary to a set, because then it will not be immutable as well as hashable.

Apart from this, we can iterate over a set. Moving to next point, which is set methods. The concept of set in python is derived from mathematical sets. Therefore, Python set also support all those operations, which we usually perform using mathematical sets like subset, superset, union, intersection, difference and so on. All these mathematical operations can be implemented using Python sets in two different ways. Either using a specific operator or using set methods. Let us look at few such examples.

(Refer Slide Time: 03:46)



The screenshot shows a web-based Python REPL interface. The code editor on the left contains the following Python code:

```
1 A = {1, 3, 5}
2 B = {1, 2, 3, 4, 5}
3 print(A.issubset(B))
4
```

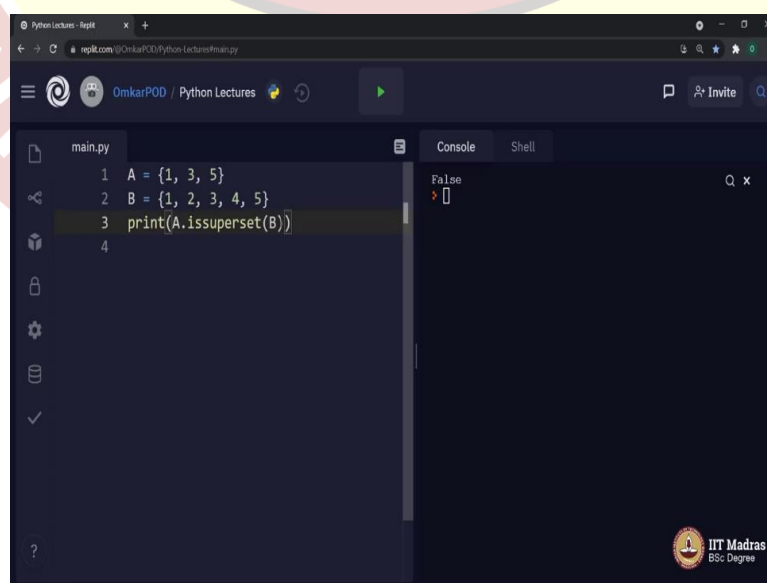
The console on the right shows the output of the execution:

```
True
```

The interface includes a file explorer on the left, a search bar, and a logo for IIT Madras BSc Degree in the bottom right corner.

Look at this code print a dot is subset d. Let us execute, and the answer is true. Because A is a subset of B. Similarly, we can check super set as well.

(Refer Slide Time: 04:08)



The screenshot shows the same web-based Python REPL interface. The code editor on the left contains the following Python code:

```
1 A = {1, 3, 5}
2 B = {1, 2, 3, 4, 5}
3 print(A.issuperset(B))
4
```

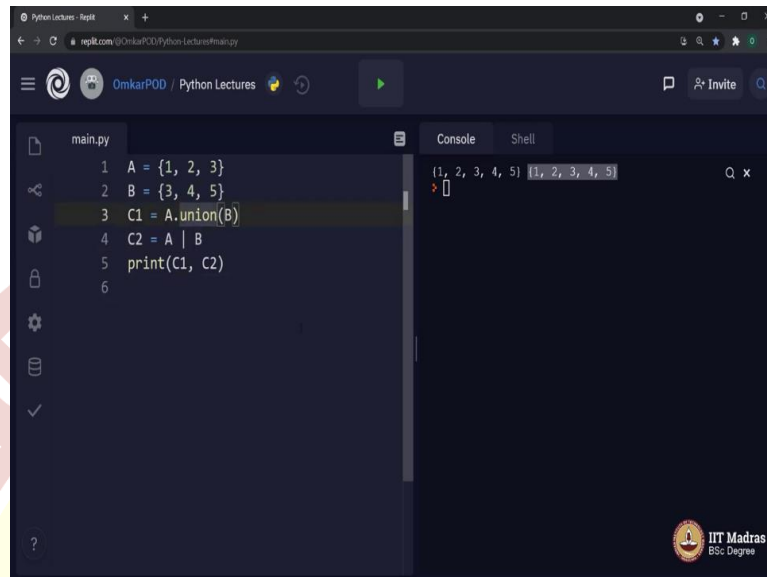
The console on the right shows the output of the execution:

```
False
```

The interface includes a file explorer on the left, a search bar, and a logo for IIT Madras BSc Degree in the bottom right corner.

Now the answer is false A is not a superset of B. In fact, it is other way around. Next mathematical operation is union.

(Refer Slide Time: 04:25)



The screenshot shows a Python REPL window with a file named 'main.py'. The code in the file is as follows:

```
1 A = {1, 2, 3}
2 B = {3, 4, 5}
3 C1 = A.union(B)
4 C2 = A | B
5 print(C1, C2)
6
```

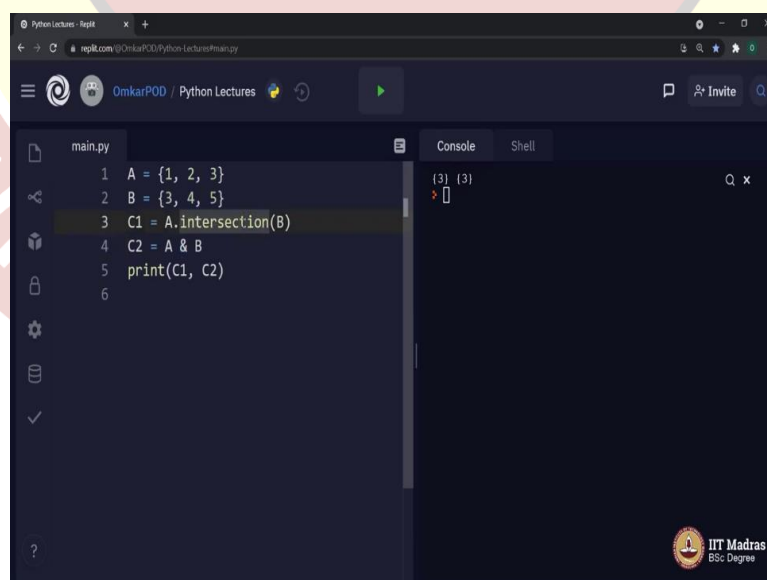
The console output on the right shows the result of the execution:

```
(1, 2, 3, 4, 5) {1, 2, 3, 4, 5}
```

The background features a large, faint watermark of the IIT Madras logo.

As I mentioned earlier Python can perform these mathematical operations in two different ways. First, using a method, second using a specific operator and this is the operator for union. Let us execute. In both cases, we are getting same output as expected. Let us try intersection.

(Refer Slide Time: 04:58)



The screenshot shows a Python REPL window with a file named 'main.py'. The code in the file is as follows:

```
1 A = {1, 2, 3}
2 B = {3, 4, 5}
3 C1 = A.intersection(B)
4 C2 = A & B
5 print(C1, C2)
6
```

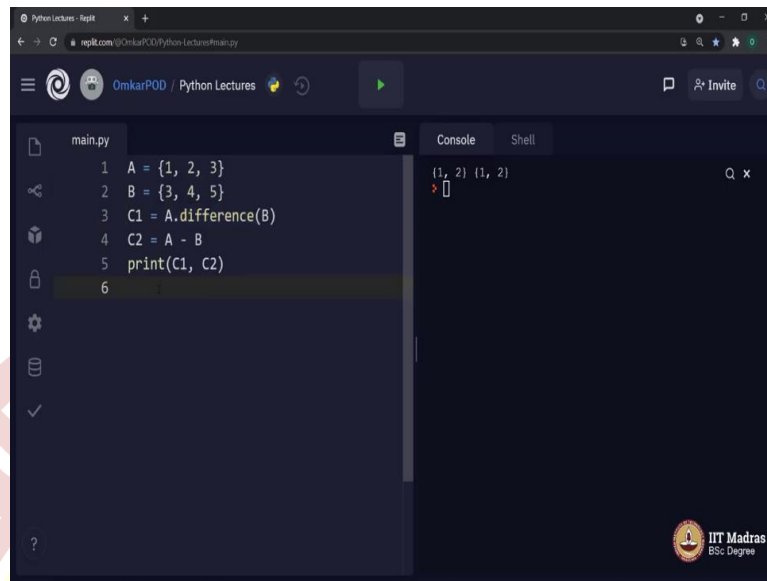
The console output on the right shows the result of the execution:

```
{3} {3}
```

The background features a large, faint watermark of the IIT Madras logo.

Once again, this is the operator for intersection or we can use set method. Let us try one more operation difference.

(Refer Slide Time: 05:14)



The screenshot shows a web-based Python REPL interface. The left pane displays a file named `main.py` with the following code:

```
1 A = {1, 2, 3}
2 B = {3, 4, 5}
3 C1 = A.difference(B)
4 C2 = A - B
5 print(C1, C2)
6
```

The right pane, labeled 'Console', shows the output of the execution:

```
{1, 2} {1, 2}
```

The interface includes a top navigation bar with the 'OmkarPOD / Python Lectures' title and a bottom right corner featuring the 'IIT Madras BSc Degree' logo.

Difference method and difference operator. Let us execute, and we got the result. In this case, it actually performs A minus B. Thank you for watching this lecture. Happy learning.