Python map()

The map() function applies a **given function to each element of an iterable (list, tuple etc.) and returns an iterator containing the results**.

Example

```
numbers = [2, 4, 6, 8, 10]

# returns the square of a number
def square(number):
  return number * number

# apply square() to each item of the numbers list
squared_numbers_iterator = map(square, numbers)


# converting to list
squared_numbers = list(squared_numbers_iterator)
print(squared_numbers)

# Output: [4, 16, 36, 64, 100]
```

map() Syntax

Its syntax is:

```
map(function, iterable, ...)
```

map() Arguments

The map() function takes two arguments:

- **function** - a function
- **iterable** - an iterable like sets, lists, tuples, etc

You can pass more than **one iterable** to the map() function.

map() Return Value

The map() function returns **an object of map class**. The returned value can be passed to functions like

- list() - to convert to list
- set() - to convert to a set, and so on.

Example 1: Working of map()

```
def calculateSquare(n):
    return n*n


numbers = (1, 2, 3, 4)
result = map(calculateSquare, numbers)

print(result)

# converting map object to set
numbersSquare = set(result)
print(numbersSquare)
```

Run Code

Output

```
<map object at 0x7f722da129e8>
{16, 1, 4, 9}
```

In the above example, each item of the tuple is squared.

Since map() expects a function to be passed in, **lambda functions are commonly used while working with map() functions.**

Example 2: How to use lambda function with map()?

```
numbers = (1, 2, 3, 4)
result = map(lambda x: x*x, numbers)

print(result)

# converting map object to set
numbersSquare = set(result)
print(numbersSquare)
```

Example 3: Passing Multiple Iterators to map() Using Lambda

```
num1 = [4, 5, 6]
num2 = [5, 6, 7]

result = map(lambda n1, n2: n1+n2, num1, num2)

print(list(result))
```