|  |
| --- |
| iterable  The object which implements the \_\_iter\_\_() method is called iterable. |

|  |
| --- |
| lst = [10,20,30,40,50] print(dir(lst))  ['\_\_add\_\_', '\_\_class\_\_', '\_\_class\_getitem\_\_', '\_\_contains\_\_', '\_\_delattr\_\_', '\_\_delitem\_\_', '\_\_dir\_\_', '\_\_doc\_\_', '\_\_eq\_\_', '\_\_format\_\_', '\_\_ge\_\_', '\_\_getattribute\_\_', '\_\_getitem\_\_', '\_\_gt\_\_', '\_\_hash\_\_', '\_\_iadd\_\_', '\_\_imul\_\_', '\_\_init\_\_', '\_\_init\_subclass\_\_', '**\_\_iter\_\_**', '\_\_le\_\_', '\_\_len\_\_', '\_\_lt\_\_', '\_\_mul\_\_', '\_\_ne\_\_', '\_\_new\_\_', '\_\_reduce\_\_', '\_\_reduce\_ex\_\_', '\_\_repr\_\_', '\_\_reversed\_\_', '\_\_rmul\_\_', '\_\_setattr\_\_', '\_\_setitem\_\_', '\_\_sizeof\_\_', '\_\_str\_\_', '\_\_subclasshook\_\_', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort'] |

|  |
| --- |
| Get Iterator Object  lst = [10,20,30,40,50] print(dir(lst)) result = lst.\_\_iter\_\_() print(result)  ['\_\_add\_\_', '\_\_class\_\_', '\_\_class\_getitem\_\_', '\_\_contains\_\_', '\_\_delattr\_\_', '\_\_delitem\_\_', '\_\_dir\_\_', '\_\_doc\_\_', '\_\_eq\_\_', '\_\_format\_\_', '\_\_ge\_\_', '\_\_getattribute\_\_', '\_\_getitem\_\_', '\_\_gt\_\_', '\_\_hash\_\_', '\_\_iadd\_\_', '\_\_imul\_\_', '\_\_init\_\_', '\_\_init\_subclass\_\_', '\_\_iter\_\_', '\_\_le\_\_', '\_\_len\_\_', '\_\_lt\_\_', '\_\_mul\_\_', '\_\_ne\_\_', '\_\_new\_\_', '\_\_reduce\_\_', '\_\_reduce\_ex\_\_', '\_\_repr\_\_', '\_\_reversed\_\_', '\_\_rmul\_\_', '\_\_setattr\_\_', '\_\_setitem\_\_', '\_\_sizeof\_\_', '\_\_str\_\_', '\_\_subclasshook\_\_', 'append', 'clear', 'copy', 'count', 'extend', 'index', 'insert', 'pop', 'remove', 'reverse', 'sort']  **<list\_iterator object at 0x000000A7625FB640>** |

|  |
| --- |
| Iterator  Iterator is an object that can return data one at a time while iterating over it  If an object to be an iterator, it must implement two methods  Iter()  next() |

|  |
| --- |
| We can use \_\_iter\_\_() method for iterator object, for next iterations use \_\_next\_\_()  lst = [10,20,30,40,50] result = lst.\_\_iter\_\_() element1 = result.\_\_next\_\_() print(element1) # 10 element2 = result.\_\_next\_\_() print(element2) # 20 element3 = result.\_\_next\_\_() print(element3) # 30 |

|  |
| --- |
| We can use iter() method for iterator object, for next iterations use next(), if there are no iterations we get exception as StopIteration  lst = [10,20,30,40,50] result = iter(lst) element1 = next(result) print(element1) # 10 element2 = next(result) print(element2) # 20 element3 = next(result) print(element3) # 30 element4 = next(result) print(element4) # 40 element5 = next(result) print(element5) # 50 element6 = next(result) print(element6) # StopIteration  Traceback (most recent call last):  File "E:\Python Github\PythonMyWorkSpace\21\_Iterators\Eg3.py", line 13, in <module>  element6 = next(result)  StopIteration |

|  |
| --- |
| Internally how StopIteration works  lst = [10,20,30,40,50] result = iter(lst)  while True:  try:  element = next(result)  print(element, end=' ')  except StopIteration:  break  10 20 30 40 50 |