① L = [10, 20, 30, 40]

```
i = 0
while i < len(L):
    print(i, L[i])
    i = i+1
```

limitation:

① Only applicable to sequential containers.

D = {'a' : 10, 'b' : 20, 'c' : 30, 'd' : 40}

0 1 2 3 ✗

Wrong notion:

```
i = 0
while i < len(D):

    i = i+1
```

D['a'] ⟶ 10

D['b'] ⟶ 20

D['c'] ⟶ 30

D['d'] ⟶ 40

⇒ Items in the dictionary (item = key:value) are indexed.

Correct Notion: Items in the dictionary are NOT indexed.

Therefore, indexing does not play any role in the traversal of dictionary object.

EQUIVALENCE BETWEEN WHILE AND FOR.

```
statement-1;
while(statement-2)
{
    statement-3;
}
```

≡

```
for(stmt-1; stmt-2; stmt-3)
{

}
```

```
while ((line=getline(---))
        != NULL)
{

}
```

$\equiv$

```
for(i=0; i<N; ++i)
{

}
```

In traditional languages such as C/C++/java for loop and while loop statements are equivalent to each other.	`for( ;(line=getline--) != NULL; )`
	`{`
	`}`

In Python, for-statement $\not\equiv$ while statement.

```
for variable_name in iterable:
```
→ Header of for

```
    Body of for statement
```

As in all block statement:

(i) the syntax of the for statement is broken in the header/Body.

(ii) If the header of the for statement is at indentation level (k) then all statements belonging to the body belong to indent level (k+1).

EXPERIMENT SECTION.

```
>>> L = [10, 20, 30, 40]
>>> for x in L:
        print(x)

10
20
30
40
```

```
>>> D = {'a': 10, 'b': 20, 'c': 30, 'd': 40}
>>> for key in D:
        print(key, D[key]).

a 10
b 20
c 30
d 40

>>>
```

Repetition = Iteration = आवर्तन

Iterative statement = आवर्तक कथन

= A statement which helps you to carry out one or more iterations of any block statement is an iterative statement.

New Term & Python Specific meaning

ITERABLE

≡

ITERABLE OBJECT

Definition: An object is iterable if its class contains '__iter__' method.

Method: A function implemented inside class is called as a method.

EXPERT SECTION

CO-ROUTINE

↳ SUB-ROUTINE

≡

↳ PROCEDUERE (procedural)

↳ function (functional)

(O.O.P.)

member function — method

static member function — static method

virtual member function

pure & virtual member function — abstract method

function template (Generic) — generic method.

Experiment-1

```
>>> L1 = [10, 20, 30, 40]
>>> 10 in L1
True
>>> 20 in L1
True
>>> 50 in L1
False
>>> 30 in L1
True
>>> 40 in L1
True
>>> -10 in L1
False
```

Experiment 2.

```
>>> L2 = ['xyz', 'pqr', 'lmn', 'abc']
>>> 'xyz' in L2
    True
>>> 'pqr' in L2
    True
>>> 'lmn' in L2
    True
>>> 'abc' in L2
    True
>>> 'CPA' in L2
    False
>>> 'Yogeshwar' in L2
    False
```

Experiment #3

```
>>> dir(list)
['__add__', '__class__', '__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']
```

```
>>> '__iter__' in dir(list)
    True
>>> '__iter__' in dir(str)
    True
>>> '__iter__' in dir(tuple)
    True
>>> '__iter__' in dir(dict)
    True
>>> '__iter__' in dir(set)
    True
>>> '__iter__' in dir(bool)
    False
>>> '__iter__' in dir(int)
    False
>>> '__iter__' in dir(float)
    False.
```

EXPERT SECTION.

```
def my_function(x):
    if '__iter__' not in dir(type(x)):
        raise TypeError('x is not iterable')
    for ele in x:
        ≡
```

Experiment #4:

```
>>> b = True
>>> type(b)
        <class 'bool'>
>>> n = 10
>>> type(n)
        <class 'int'>
>>> f = 3.14
>>> type(f)
        <class 'float'>
>>> s = 'Hello'
>>> type (s)
        <class 'str'>
>>> L = [10, 20, 30, 40]
>>> type(L)
        <class 'list'>
>>> T = (10, 20, 30, 40)
>>> type(T)
        <class 'tuple'>
>>> D = {'a': 10, 'b': 20, 'c': 30, 'd': 40}
>>> type (D)
        <class 'dict'>
```

```
>>> S = {10, 20, 30, 40}
>>> type(S)
    <class 'set'>.
>>> for x in b:
        print(x)

    TypeError: bool object is not iterable.
>>> for x in n:
        print(x)
    TypeError: int object is not iterable.
>>> for x in f:
        print(x)
    TypeError : float object is not iterable.
>>> for x in D:
        print(x, D[x]).
    a 10
    b 20
    c 30
    d 40
```

VERY ADVANCED LEVEL

```
>>> L = [10, 20, 30, 40]
>>> for x in L:
        print(x)

10
20
30
40

>>> I = L.__iter__()
>>> while True:
        try:
            x = I.__next__()
            print(x)
        except StopIteration:
            break

10
20
30
40
```

0
1
2
2
1
2