You were Last Reading: Transforming Sequences > Don't Mutate A List That You Are Iterating Through Continue Reading

7.7. Traversal and the `for` Loop: By Index

With a for loop, the loop variable is bound, on each iteration, to the next item in a sequence. Sometimes, it is natural to think about iterating through the *positions*, or *indexes* of a sequence, rather than through the items themselves.

For example, consider the list `['apple', 'pear', 'apricot', 'cherry', 'peach']`. 'apple' is at position 0, 'pear' at position 1, and 'peach' at position 4.

Thus, we can iterate through the indexes by generating a sequence of them, using the `range` function.

Save & Run 4/30/2021, 9:16:54 PM - 2 of 2 Show in CodeLens

```
fruits = ['apple', 'pear', 'apricot', 'cherry', 'peach']
for n in range(5):
    print(n, fruits[n])

```

```
0 apple
1 pear
2 apricot
3 cherry
4 peach
```

Activity: 1 -- ActiveCode (ac14_6_5a)

In order to make the iteration more general, we can use the `len` function to provide the bound for `range`. This is a very common pattern for traversing any sequence by position. Make sure you understand why the range function behaves correctly when using `len` of the string as its parameter value.

Save & Run 4/30/2021, 9:16:56 PM - 2 of 2 Show in CodeLens

```
fruits = ['apple', 'pear', 'apricot', 'cherry', 'peach']
for n in range(len(fruits)):
    print(n, fruits[n])

```

```
0 apple
1 pear
2 apricot
3 cherry
4 peach
```

Activity: 2 -- ActiveCode (ac14_6_5)

In some other programming languages, that's the only way to iterate through a sequence, by iterating through the positions and extracting the items at each of the positions. Python code is often easier to read because we don't have to do iteration that way. Compare the iteration above with the more "pythonic" approach below.

Save & Run 4/30/2021, 9:16:58 PM - 2 of 2 Show in CodeLens

```
fruits = ['apple', 'pear', 'apricot', 'cherry', 'peach']
for fruit in fruits:
    print(fruit)

```

```
apple
pear
apricot
cherry
peach
```

Activity: 3 -- ActiveCode (ac14_6_5c)

If we really want to print the indexes (positions) along with the fruit names, then iterating through the indexes as in the previous versions is available to us. Python also provides an `enumerate` function which provides a more “pythonic” way of enumerating the items in a list, but we will delay the explanation of how to use `enumerate` until we cover the notions of tuple packing and unpacking.

Check your understanding

moreiter-6-1: How many times is the letter p printed by the following statements?

```
s = "python"
for idx in range(len(s)):
   print(s[idx % 2])
```

- A. 0
- B. 1
- C. 2
- D. 3
- E. 6

Check me | Compare me

✔ idx % 2 is 0 whenever idx is even

Activity: 4 -- Multiple Choice (question14_6_1)

7.6. The Accumulator Pattern">
« Accumulator Pattern">
✔ Completed. Well Done!
7.8. Nested Iteration: Image Processing">
›

| Back to top