

Python for Engineers Activities - Lists

Contents

Python for Engineers Activities - Lists	1
DRY.....	1
Online Tutorials.....	1
Tutorial 1: Traversing a List with a Loop.....	2
Tutorial 2: List and String Slicing	3
Tutorial 3: List Functions.....	6
in and not in	6
Tutorial 4: List Methods	7
Assignment 1: List Slicing and Iteration	8
Assignment Submission.....	8

Time required: 90 minutes

DRY

Don't Repeat Yourself

Online Tutorials

Go through the following tutorials.

- [LearnPython.org Lists](#)
- [Python Lists](#)
 - [Access List Items](#)
 - [Change List Items](#)
 - [Add List Items](#)
 - [Remove List Items](#)
 - [Loop Lists](#)
 - [Sort Lists](#)

- [Copy Lists](#)
- [Join Lists](#)
- [List Methods](#)

Tutorial 1: Traversing a List with a Loop

A List is a sequence of elements. A String is a sequence of elements. A Tuple is a sequence of elements. The major difference is that a List is mutable, it can be changed. Otherwise, each of these sequences behave in similar ways.

The most common way to iterate or traverse the elements of a list is with a for loop. The syntax is the same as for strings.

```
cheeses = ["cheddar", "colby", "swiss"]
for cheese in cheeses:
    print(cheese)
cheddar
colby
swiss
```

If you want to write or update the elements, you need the indices. A common way to do that is to combine the functions **range** and **len**:

```
# range_len.py
numbers = [1, 3, 5]
for i in range(len(numbers)):
    # Update element in list
    numbers[i] = numbers[i] * 2
    print(f"{i + 1} {numbers[i]}")
```

This loop traverses the list and updates each element. **len** returns the number of elements in the list. **range** returns a list of indices from 0 to $n - 1$, where **n** is the length of the list. Each time through the loop, **i** gets the index of the next element. The assignment statement in the body uses **i** to read the old value of the element and to assign the new value.

Create a Python program named **list_loops.py**

```

1  """
2      Name: list_loops.py
3      Author:
4      Created:
5      Purpose:
6  """
7  # Define a list of numbers: 1, 3, 5
8  numbers = [1, 3, 5]
9
10 # Iterate through each number in the list and print it
11 for number in numbers:
12     print(f"{number}")
13
14 # Iterate through the indices of the list using a for loop
15 for i in range(len(numbers)):
16     # Multiply each number in the list by 2
17     numbers[i] = numbers[i] * 2
18
19     # Print the modified numbers along with their respective indices
20     print(f"{i + 1}. {numbers[i]}")

```

Example run:

```

1
3
5
1. 2
2. 6
3. 10

```

Tutorial 2: List and String Slicing

The Python slice operator, represented by the colon :, allows you to extract a portion of a sequence like a string, list, or tuple by specifying a start, stop, and step size within square brackets [].

The general syntax for slicing is: **sequence[start:stop:step]**

```

new_list = original_list[start:stop:step]
new_string = original_string[start:stop:step]

```

- **start (optional):** The index where the slice begins (inclusive). If omitted, it starts from the beginning of the sequence (index 0).

- **stop (optional):** The index where the slice ends (exclusive). If omitted, it goes until the end of the sequence.
- **step (optional):** The increment between indices while slicing. If omitted, it defaults to 1.

```
t = ["a", "b", "c", "d", "e", "f"]
print(t[1:3])
# Output: ['b', 'c']

print(t[:4])
# Output: ['a', 'b', 'c', 'd']

print(t[3:])
# Output: ['d', 'e', 'f']
```

- `t[1:3]` slices the index 1 and 2 items.
- `t[:4]` slices everything from the beginning of the list to index 3.
- `t[3:]` slices everything after and including index 3.

If you omit the first index, the slice starts at the beginning. If you omit the second, the slice goes to the end. If you omit both, the slice is a copy of the whole list.

```
print(t[:])
# Output: ['a', 'b', 'c', 'd', 'e', 'f']
```

Lists and strings can be sliced in similar ways. Create a Python program named **list_slice.py**

```

1  """
2      Name: list_slice.py
3      Author:
4      Created:
5      Purpose: Demonstrate Python slicing
6  """
7
8  # Example 1: Slicing a range from a list
9  numbers = [1, 2, 3, 4, 5]
10 subset = numbers[1:4]
11 print(subset)
12
13 # Example 1: Slicing a range from a string
14 text = "PythonProgramming"
15 subset = text[0:6]
16 print(subset)

```

Example run:

```

[2, 3, 4]
Python

```

```

18 # Example 2: Slicing with step
19 even_numbers = numbers[::2]
20 print(even_numbers)
21
22 # Example 2: Slicing with step
23 every_second_char = text[::2]
24 print(every_second_char) # Output: PtoPormig
25
26 # Example 3: Slicing from the end
27 last_three = numbers[-3:]
28 print(last_three)
29
30 # Example 4: Modifying elements
31 fruits = ["apple", "banana", "cherry", "date"]
32 fruits[1:3] = ["pear", "grape"]
33 print(fruits)

```

Example run:

```
[1, 3, 5]
PtoPormig
[3, 4, 5]
['apple', 'pear', 'grape', 'date']
```

Tutorial 3: List Functions

There are many built-in functions that operate on lists without writing your own loops.

Function	Description
<code>len()</code>	Returns the number of items in the list
<code>sum()</code>	Returns the sum of the items in the list
<code>min()</code>	Returns the minimum of the items in the list
<code>max()</code>	Returns the maximum of the items in the list
<code>sorted()</code>	Returns a sorted list

in and not in

The `in` operator tells you if a list or string contains something. The `not in` operator tells you if the list or string doesn't contain something.

Create a Python file named **list_functions.py**

```

1  nums = [3, 41, 12, 9, 74, 15]
2
3  print(len(nums))
4
5  print(max(nums))
6
7  print(min(nums))
8
9  print(sum(nums))
10
11 print(sum(nums) / len(nums))
12
13 print(sorted(nums))
14
15 # The in operator tells you if your list or string contains an element
16 if 3 in nums:
17     print("Your list contains the number 3. ")
18
19 # not in tells you if your list or string does not contain an element
20 if 0 not in nums:
21     print("Your list has no zeroes.")

```

Example run:

```

6
74
3
154
25.666666666666668
[3, 9, 12, 15, 41, 74]
Your list contains the number 3.
Your list has no zeroes.

```

Tutorial 4: List Methods

Python provides methods that operate on lists. Methods use the . (dot) operator.

Method	Description
<code>append(x)</code>	add x to the end of the list
<code>sort(x)</code>	sort the list in place
<code>count(x)</code>	return the number of times x occurs in the list
<code>index(x)</code>	return the location of the first occurrence of x

<code>reverse()</code>	reverse the list in place
<code>clear()</code>	remove all the elements from the list
<code>remove(x)</code>	remove the first occurrence of x from the list
<code>pop(p)</code>	remove the item at index p and return its value
<code>insert(p, x)</code>	insert x at index p of the list

Assignment 1: List Slicing and Iteration

Objective: Understand list slicing and iteration.

1. Create a list named numbers with the numbers from 1 to 10.
2. Print the first five numbers using list slicing.
3. Print the last three numbers using list slicing.
4. Use a loop to print each number in the list.
5. Use a loop to print each number in the list that is greater than 5.

Example run:

```
[1, 2, 3, 4, 5]
[8, 9, 10]
1
2
3
4
5
6
7
8
9
10
6
7
8
9
10
```

Assignment Submission

1. In Google Colab → Click the Share button in the upper right hand side.
 - a. Change General Access → Anyone with the link → Click Copy link.

2. Submit in Blackboard.