W7.2 Applied

Preprocessing String

Write a function named `preprocess_string` that takes in a string and a list of delimiters as parameters and returns a list of strings such that the output list splits the input string based on all delimiters. You cannot use the `re` module for this task. You can assume that no delimiter is contained in another delimiter in the list of delimiters.

For example, for the input string `"Python is fun and tssssss; what a language to learn, waow :)"` and the delimiter list `[',', ' ', ';', ':)']`, the function should output `['Python', 'is', 'fun', 'and', 'tssssss', '', 'what', 'a', 'language', 'to', 'learn', '', 'waow', '', '']`.

Modify List

Write a function named `modify_list` that takes a list of strings as a parameter and modifies the input list such that each string is replaced with its all uppercase equivalent if the original string includes the string '`yell`'. The implemented function should not have a `return` statement.

Example

After

```
my_list = ["Hello, world!",
           "Say yello to my friend",
           "Don't yell and be quiet!",
           "I can't believe you are reading this",
           "Why are we yelling?"]
modify_list(my_list)
```

The content of `my_list` should be

```
['Hello, world!', 'SAY YELLO TO MY FRIEND', "DON'T YELL AND BE QUIET!", "I can't believe you are re
```

Modify Table 1

Write a function named `modify_table` that takes a list of lists of numbers (i.e., a table) as a parameter and modifies the input table such that each inner list is extended with the string `'zero'` if the numbers in that list add up to 0. The implemented function should not have a `return` statement.

Example

After running

```
table = [[9,-7,-2], [7,3,1,3], [-34,3,22], [0], [-1,1]]
modify_table(table)
```

The variable `table` should contain

```
[[9, -7, -2, 'zero'], [7, 3, 1, 3], [-34, 3, 22], [0, 'zero'], [-1, 1, 'zero']]
```

Modify Table 2

Write a function named `modify_table` that takes a list of lists of numbers (i.e., a table) as a parameter and modifies the input table such that each inner list is deleted if the numbers in that list add up to 0. The implemented function should not have a `return` statement.

Example

If you run

```
table = [[9,-7,-2], [7,3,1,3], [-34,3,22], [0], [-1,1]]
modify_table(table)
```

the variable `table` should contain

```
[[7, 3, 1, 3], [-34, 3, 22]]
```

Predict the outcome of the following computations by hand

In this activity, please attempt to predict the outcome of the following code *without using the Python interpreter*.

Question 1 *Submitted Apr 12th 2022 at 11:00:19 am*

What is the value of `x`?

```
from copy import copy
x = ['Fellowship', 'Towers', 'King']
my_list = copy(x)
my_list.append('Hobbit')
```

- ['Fellowship', 'Towers', 'Hobbit']
- ['Fellowship', 'Towers', 'King', 'Hobbit']
- ['Fellowship', 'Towers', 'King']
- ['Hobbit', 'Fellowship', 'Towers', 'King']

Question 2 *Submitted Apr 12th 2022 at 11:00:42 am*

What is the value of `x`?

```
from copy import copy
x = [['Menace', 'Clones', 'Sith'], ['Hope', 'Empire', 'Return']]
my_list = copy(x)
my_list[-1][-1] = 'Jedi'
```

- ['Menace', 'Clones', 'Sith']
- [['Menace', 'Clones', 'Sith'], ['Hope', 'Empire', 'Return']]
- [['Menace', 'Clones', 'Sith'], ['Hope', 'Empire', 'Jedi']]

- ['Hope', 'Empire', 'Jedi']

Question 3 *Submitted Apr 12th 2022 at 11:01:05 am*

Which of the following is correct after execution of the following code?

```
from copy import deepcopy
t1 = [['a', 'b'], ['c', 'd']]
t2 = deepcopy(t1)
t2[0] = t1[0]
t2[1] = t1[1]
```

- `t2` is a shallow copy of `t1`.
- `t2` is a deep copy of `t1`.
- None of the statements are true.
- All the statements are true.

Question 4 *Submitted Apr 12th 2022 at 11:01:49 am*

What is the value of `t1[0][1]`?

```
from copy import deepcopy
t1 = [['a', 'b'], ['c', 'd']]
t2 = deepcopy(t1)
t2[0] = t1[0]
t2[0][1] = 'B'
```

- 'b'
- 'B'
- 'bb'
- 'bB'

Question 5 *Submitted Apr 12th 2022 at 11:02:25 am*

What is the value of `x` and `y` after running the following block of code?

```
import copy
table = [[1, 2, 3], [4, 5, 6]]
table_copy = copy.deepcopy(table)
row1 = copy.copy(table[0])
row1[0] = 9
table_copy[1] = row1
x = table
y = table_copy
```

- (●) x=[[1,2,3],[4,5,6]]
  y=[[1,2,3],[9,2,3]]
- ( ) x=[[1,2,3],[9,5,6]]
  y=[[1,2,3],[9,2,3]]
- ( ) x=[[1,2,3],[9,2,3]]
  y=[[1,2,3],[9,2,3]]
- ( ) x=[[1,2,3],[9,5,6]]
  y=[[1,2,3],[9,5,6]]

Implement deepcopy (for depth at most 2)

Design and implement a function named `deepcopy_atmost_2` that accepts a nested list with depth of at most 2, and returns a deep copy of the list. You are **not allowed** to import the module `copy`.

Focus on articulating the necessary steps as dot-points. Once you have designed an algorithm, try implementing it in the file named `deepcopy_atmost_2.py`.

An example of a nested list with depth of at most 2: `['a', [ 'b', 'c'], 'd', ['e'], 'f']`

You may find the Python function `isinstance(object, type)` useful.

Feedback

Question 1

Feedback

What worked best in this lesson?

No response

Question 2

Feedback

What needs improvement most?

No response