

Nested Functions II

The screenshot shows a web-based coding exercise interface. On the left, there's a sidebar with a 'Learn / Courses / Introduction to Functions in Python' breadcrumb, a 'Back to course page' button, and a 'Nested Functions II' section. Below this, there's an 'Instructions' section with a list of tasks and a 'Take Hint (-30 XP)' button. The main area on the right is a code editor with a dark theme, showing a Python script. The script defines an outer function 'echo(n)' which returns an inner function 'inner_echo'. The inner function 'inner_echo' takes a parameter 'word1' and returns 'word1 * n'. The script also includes calls to 'echo(2)' (assigning to 'twice') and 'echo(3)' (assigning to 'thrice'), followed by print statements for 'twice' and 'thrice' with the argument 'hello'. At the bottom right of the code editor are buttons for 'Run Code' and 'Submit Answer'. Below the code editor is a 'Python Shell' section with a prompt 'In [1]:'.

Learn / Courses / Introduction to Functions in Python

Exercise Back to course page

Nested Functions II

Great job, you've just nested a function within another function. One other pretty cool reason for nesting functions is the idea of a closure. This means that the nested or inner function remembers the state of its enclosing scope when called. Thus, anything defined locally in the enclosing scope is available to the inner function even when the outer function has finished execution.

Let's move forward then! In this exercise, you will complete the definition of the inner function `inner_echo()` and then call `echo()` a couple of times, each with a different argument. Complete the exercise and see what the output will be!

Instructions 150 XP

- Complete the function header of the inner function with the function name `inner_echo()` and a single parameter `word1`.
- Complete the function `echo()` so that it returns `inner_echo`.
- We have called `echo()`, passing 2 as an argument, and assigned the resulting function to `twice`. Your job is to call `echo()`, passing 3 as an argument. Assign the resulting function to `thrice`.
- Hit Submit to call `twice()` and `thrice()` and print the results.

Take Hint (-30 XP)

```
1 # Define echo
2 def echo(n):
3     """Return the inner_echo function."""
4
5     # Define inner_echo
6     def ____():
7         """Concatenate n copies of word1."""
8         echo_word = word1 * n
9         return echo_word
10
11     # Return inner_echo
12
13
14 # Call echo: twice
15 twice = echo(2)
16
17 # Call echo: thrice
18
19
20 # Call twice() and thrice() then print
21 print(twice('hello'), thrice('hello'))
```

Run Code Submit Answer

Python Shell Slides

In [1]:

Question:

Great job, you've just nested a function within another function. One other pretty cool reason for nesting functions is the idea of a closure.

This means that the inner function remembers the state of its enclosing scope when called.

Thus, anything defined locally in the enclosing scope is available to the inner function even when the outer function has finished execution.

****Instructions:****

1. Complete the function header of the inner function with the function name `inner_echo()` and a single parameter `word1`.
2. Complete the function `echo()` so that it returns `inner_echo`.
3. We have called `echo()`, passing 2 as an argument, and assigned the resulting function to `twice`. Your job is to call `echo()`, passing 3 as an argument. Assign the resulting function to `thrice`.

4. Hit Submit to call ``twice()`` and ``thrice()`` then print the results.

Answer:

```
# Define echo
def echo(n):
    """Return the inner_echo function."""

    # Define inner_echo
    def inner_echo(word1):
        """Concatenate n copies of word1."""
        echo_word = word1 * n
        return echo_word

    return inner_echo

# Call echo: twice
twice = echo(2)

# Call echo: thrice
thrice = echo(3)

# Call twice() and thrice() then print
print(twice('hello'), thrice('hello'))
```

Explanation:

1. `def echo(n):` - This defines the outer function ``echo``, which takes a single parameter ``n``. It will return a nested function ``inner_echo``.
2. `def inner_echo(word1):` - Defines a nested function ``inner_echo`` that takes a single parameter ``word1``. This function concatenates ``word1`` repeated ``n`` times.
3. `echo_word = word1 * n` - Creates a string by repeating ``word1`` exactly ``n`` times.
4. `return inner_echo` - The outer function ``echo`` returns the inner function ``inner_echo``.
5. `twice = echo(2)` - Calls ``echo`` with ``n=2`` and assigns the resulting function ``inner_echo`` to the variable ``twice``.
6. `thrice = echo(3)` - Calls ``echo`` with ``n=3`` and assigns the resulting function ``inner_echo`` to the variable ``thrice``.
7. `print(twice('hello'), thrice('hello'))` - Calls the functions ``twice`` and ``thrice`` with the argument ``'hello'``, printing the results ``'hellohello'`` and

`'hellohellohello'` respectively.