

The screenshot shows a web-based coding environment. On the left, there's a sidebar with the title 'Exercise' and a sub-header 'Changing the output in generator expressions'. Below this, there's a brief introduction and a list of instructions. The main area on the right contains a code editor with a Python script. The script defines a list of strings, creates a generator object, and iterates over it to print the lengths of the strings. At the bottom, there's a 'Python Shell' area for running the code.

Exercise

Changing the output in generator expressions

Great! At this point, you already know how to write a basic generator expression. In this exercise, you will push this idea a little further by adding to the output expression of a generator expression. Because generator expressions and list comprehensions are so alike in syntax, this should be a familiar task for you!

You are given a list of strings `lannister` and, using a generator expression, create a generator object that you will iterate over to print its values.

Instructions

- Write a generator expression that will generate the lengths of each string in `lannister`. Use `person` as the iterator variable. Assign the result to `lengths`.
- Supply the correct iterable in the `for` loop for printing the values in the generator object.

[Take Hint \(-50 XP\)](#)

```
script.py
1 # Create a list of strings: lannister
2 lannister = ['cersei', 'jaime', 'tywin', 'tyrion', 'joffrey']
3
4 # Create a generator object: lengths
5 lengths = ____
6
7 # Iterate over and print the values in lengths
8 for value in ____:
9     print(value)
10
```

[Run Code](#) [Submit Answer](#)

Python Shell Slides

In [1]:

Question: Changing the output in generator expressions

Correct Answer and Explanation:

Code Implementation:

```
# Create a list of strings
lannister = ['cersei', 'jaime', 'tywin', 'tyrion', 'joffrey']
```

```
# Define generator function get_lengths
def get_lengths(input_list):
    """Generator function that yields the
    length of the strings in input_list."""
```

```
    # Yield the length of a string
    for person in input_list:
        yield len(person)
```

```
# Print the values generated by get_lengths()
for value in get_lengths(lannister):
    print(value)
```

Explanation:

1. ``lannister = ['cersei', 'jaime', 'tywin', 'tyrion', 'joffrey']``:
 - Initializes a list of strings representing names of Lannister family members.
2. ``def get_lengths(input_list):``:
 - Defines a generator function that takes an input list and yields the length of each string in the list.
3. ``for person in input_list:``:
 - Loops through each element in the input list and calculates the length of the string.
4. ``yield len(person)``:
 - The generator function yields the length of each string in the input list one by one.
5. ``for value in get_lengths(lannister):``:
 - This loop calls the generator function ``get_lengths`` and iterates over the values it yields.
6. ``print(value)``:
 - Prints each length as generated by the ``get_lengths`` function.