

The screenshot shows a web-based Python learning environment. On the left, there's a sidebar with a 'Learn / Courses / Python Toolbox' header. Below it, an 'Exercise' section titled 'Write your own generator expressions' provides background information and instructions. The instructions list three steps: creating a generator object, printing the first 5 values, and printing the rest of the values. A 'Take Hint (-30 XP)' button is also present. The main area on the right contains a code editor with a dark theme, showing a partially completed Python script. The script starts with a comment '# Create generator object: result' and an assignment 'result = ____'. It then has a comment '# Print the first 5 values' followed by five 'print(____)' statements. Below that is a comment '# Print the rest of the values' followed by a 'for value in ____:' loop with a 'print(value)' statement. At the bottom right of the code editor are buttons for 'Run Code' and 'Submit Answer'. Below the code editor is an 'IPython Shell' area with a prompt 'In [1]:'.

Learn / Courses / Python Toolbox

Exercise

Write your own generator expressions

You are familiar with what generators and generator expressions are, as well as its difference from list comprehensions. In this exercise, you will practice building generator expressions on your own.

Recall that generator expressions basically have the same syntax as list comprehensions, except that it uses parentheses `()` instead of brackets `[]`; this should make things feel familiar! Furthermore, if you have ever iterated over a dictionary with `.items()`, or used the `range()` function, for example, you have already encountered and used generators before, without knowing it! When you use these functions, Python creates generators for you behind the scenes.

Now, you will start simple by creating a generator object that produces numeric values.

Instructions 100 XP

- Create a generator object that will produce values from 8 to 38. Assign the result to `result` and use `num` as the iterator variable in the generator expression.
- Print the first 5 values by using `next()` appropriately in `print()`.
- Print the rest of the values by using a `for` loop to iterate over the generator object.

Take Hint (-30 XP)

```
script.py
1 # Create generator object: result
2 result = ____
3
4 # Print the first 5 values
5 print(____)
6 print(____)
7 print(____)
8 print(____)
9 print(____)
10
11 # Print the rest of the values
12 for value in ____:
13     print(value)
14
```

Run Code Submit Answer

IPython Shell Slides

In [1]:

Question: Write your own generator expressions

Correct Answer and Explanation:

Code Implementation:

```
# Create generator object: result
result = (num for num in range(31))
```

```
# Print the first 5 values
print(next(result)) # Outputs: 8
print(next(result)) # Outputs: 9
print(next(result)) # Outputs: 10
print(next(result)) # Outputs: 11
print(next(result)) # Outputs: 12
```

```
# Print the rest of the values
```

```
for value in result:  
    print(value)
```

Explanation:

1. ``result = (num for num in range(31))``:

- This creates a generator object that generates numbers from 0 to 30 (inclusive).

2. ``print(next(result))``:

- The ``next()`` function retrieves the next item in the generator object ``result``.

- The first call to ``next()`` starts from the first value in the generator and retrieves subsequent values upon further calls.

3. ``for value in result:``:

- The loop iterates over the remaining items in the generator object ``result``.

- This allows printing all remaining values after the first 5 have been retrieved.