

## Reduce() and Lambda Functions

Learn / Courses / Introduction to Functions in Python

Exercise

**Pop quiz on lambda functions**

In this exercise, you will practice writing a simple lambda function and calling this function. Recall what you know about lambda functions and answer the following questions:

- How would you write a lambda function `add_bangs` that adds three exclamation points `!!!` to the end of a string `s`?
- How would you call `add_bangs` with the argument `'hello'`?

You may use the IPython shell to test your code.

Instructions 80 XP

Possible answers

- ☒ The lambda function definition is: `add_bangs = (s + '!!!')`, and the function call is: `add_bangs('hello')`.
- ☐ The lambda function definition is: `add_bangs = (lambda s: s + '!!!')`, and the function call is: `add_bangs('hello')`.
- ☐ The lambda function definition is: `(lambda s: s + '!!!') * add_bangs`, and the function call is: `add_bangs('hello')`.

Submit Answer

Take Hint (-15 XP)

Python Shell

In [1]:

### Question:

You're getting very good at using lambda functions! Here's one more function to add to your repertoire of skills.

The `reduce()` function is useful for performing some computation on a list and, unlike `map()` and `filter()`, returns a single value as a result. To use `reduce()`, you must import it from the `functools` module.

Remember `gibberish()` from a few exercises back?

```
# Define gibberish()
def gibberish(*args):
    """Concatenate strings in *args together."""
    hodgepodge = ""
    for word in args:
        hodgepodge += word
    return hodgepodge
```

`gibberish()` simply takes a list of strings as an argument and returns, as a

single-value result, the concatenation of all of these strings.  
In this exercise, you will replicate this functionality by using `reduce()` and a lambda function that concatenates strings together.

**\*\*Instructions:\*\***

1. Import the `reduce` function from the `functools` module.
2. In the `reduce()` call, pass a lambda function that takes two string arguments `item1` and `item2` and concatenates them; also pass the list of strings, `stark`.

Assign the result to `result`. The first argument to `reduce()` should be the lambda function and the second argument is the list `stark`.

3. Print the result.

## Answer:

```
# Import reduce from functools
from functools import reduce

# Create a list of strings: stark
stark = ['robb', 'sansa', 'arya', 'brandon', 'rickon']

# Use reduce() to apply a lambda function over stark: result
result = reduce(lambda item1, item2: item1 + item2, stark)

# Print the result
print(result)
```

## Explanation:

1. **\*\*Importing reduce():\*\***

The `reduce()` function is not built-in and must be imported from the `functools` module.

Syntax: `from functools import reduce`.

2. **\*\*List of Strings:\*\***

The list `stark` contains the names of members of House Stark in "Game of Thrones".

Example: `['robb', 'sansa', 'arya', 'brandon', 'rickon']`.

3. **\*\*Lambda Function in reduce():\*\***

The lambda function takes two arguments (`item1` and `item2`) and concatenates them using `+`.

Syntax: ``lambda item1, item2: item1 + item2``.

4. **`**reduce():**`**

The ``reduce()`` function applies the lambda function cumulatively to the elements of the list, reducing the list to a single concatenated string.

Example Output: ``'robbsansaaryabrandonrickon'``.

5. **`**Print the Result:**`**

The concatenated result of all strings in the ``stark`` list is printed to the console.

Expected Output: ``'robbsansaaryabrandonrickon'``.