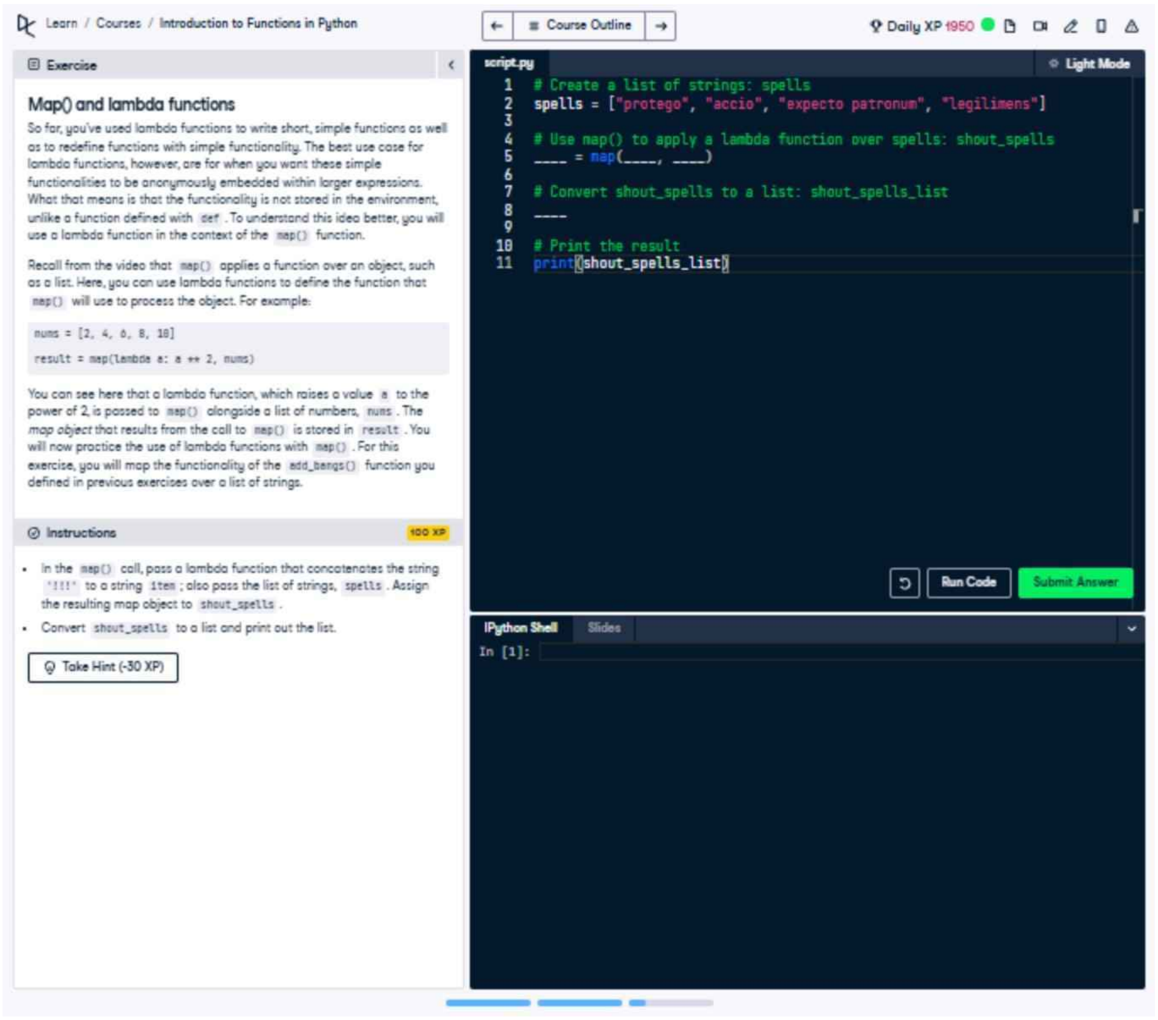
Map() and Lambda Functions



Map() and lambda functions

So far, you've used lambda functions to write short, simple functions as well as to redefine functions with simple functionality. The best use case for lambda functions, however, are for when you want these simple functionalities to be anonymously embedded within larger expressions. What that means is that the functionality is not stored in the environment, unlike a function defined with `def`. To understand this idea better, you will use a lambda function in the context of the `map()` function.

Recall from the video that `map()` applies a function over an object, such as a list. Here, you can use lambda functions to define the function that `map()` will use to process the object. For example:

```
nums = [2, 4, 6, 8, 10]

result = map(lambda a: a ** 2, nums)
```

You can see here that a lambda function, which raises a value `a` to the power of 2, is passed to `map()` alongside a list of numbers, `nums`. The *map object* that results from the call to `map()` is stored in `result`. You will now practice the use of lambda functions with `map()`. For this exercise, you will map the functionality of the `add_bangs()` function you defined in previous exercises over a list of strings.

Instructions

- In the `map()` call, pass a lambda function that concatenates the string `'!!!'` to a string `item`; also pass the list of strings, `spells`. Assign the resulting map object to `shout_spells`.
- Convert `shout_spells` to a list and print out the list.

script.py

```python
# Create a list of strings: spells
spells = ["protego", "accio", "expecto patronum", "legilimens"]

# Use map() to apply a lambda function over spells: shout_spells
____ = map(____, ____)

# Convert shout_spells to a list: shout_spells_list
____

# Print the result
print(shout_spells_list)
```

```
In [1]:
```

Question:

So far, you’ve used lambda functions to write short, simple functions as well as to redefine functions with simple functionality.
The best use case for lambda functions, however, are for when you want these simple functionalities to be anonymously embedded within larger expressions.
What that means is that the functionality is not stored in the environment, unlike a function defined with `def`.
To understand this idea better, you will use a lambda function in the context of the `map()` function.

Recall from the video that `map()` applies a function over an object, such as a list.

Here, you can use lambda functions to define the function that `map()` will use to process the object. For example:

```python
nums = [2, 4, 6, 8, 10]
result = map(lambda a: a ** 2, nums)
```

You can see here that a lambda function, which raises a value to the power of 2, is passed to `map()` alongside a list of numbers, `nums`.
The `map` object that results from the call to `map()` is stored in `result`.
You will now practice the use of lambda functions with `map()`.
For this exercise, you will map the functionality of the `add_bangs()` function you defined in previous exercises over a list of strings.

Instructions:
1. In the `map()` call, pass a lambda function that concatenates the string `'!!!'` to a string `item`; also pass the list of strings, `spells`.
   Assign the resulting map object to `shout_spells`.
2. Convert `shout_spells` to a list and print out the list.

Answer:

```
# Create a list of strings: spells
spells = ["protego", "accio", "expecto patronum", "legilimens"]

# Use map() to apply a lambda function over spells: shout_spells
shout_spells = map(lambda item: item + "!!!", spells)

# Convert shout_spells to a list: shout_spells_list
shout_spells_list = list(shout_spells)

# Print the result
print(shout_spells_list)
```

Explanation:

1. **List of Strings:**
   `spells` is a list of strings containing magical incantations.
   Example: `["protego", "accio", "expecto patronum", "legilimens"]`.

2. **Lambda Function:**

   A lambda function is defined inline with `map()` to append `'!!!'` to each string in `spells`.
   Syntax: `lambda item: item + "!!!"`.

3. **map():**
   The `map()` function applies the lambda function to each element of `spells`.
   Result: A map object containing transformed strings.

4. **Convert to List:**
   Since `map()` returns an object, it is converted to a list using `list()`.
   Example: `["protego!!!", "accio!!!", "expecto patronum!!!", "legilimens!!!"]`.

5. **Print Result:**
   The resulting list is printed to display the transformed strings.
   Example Output: `["protego!!!", "accio!!!", "expecto patronum!!!", "legilimens!!!"]`.