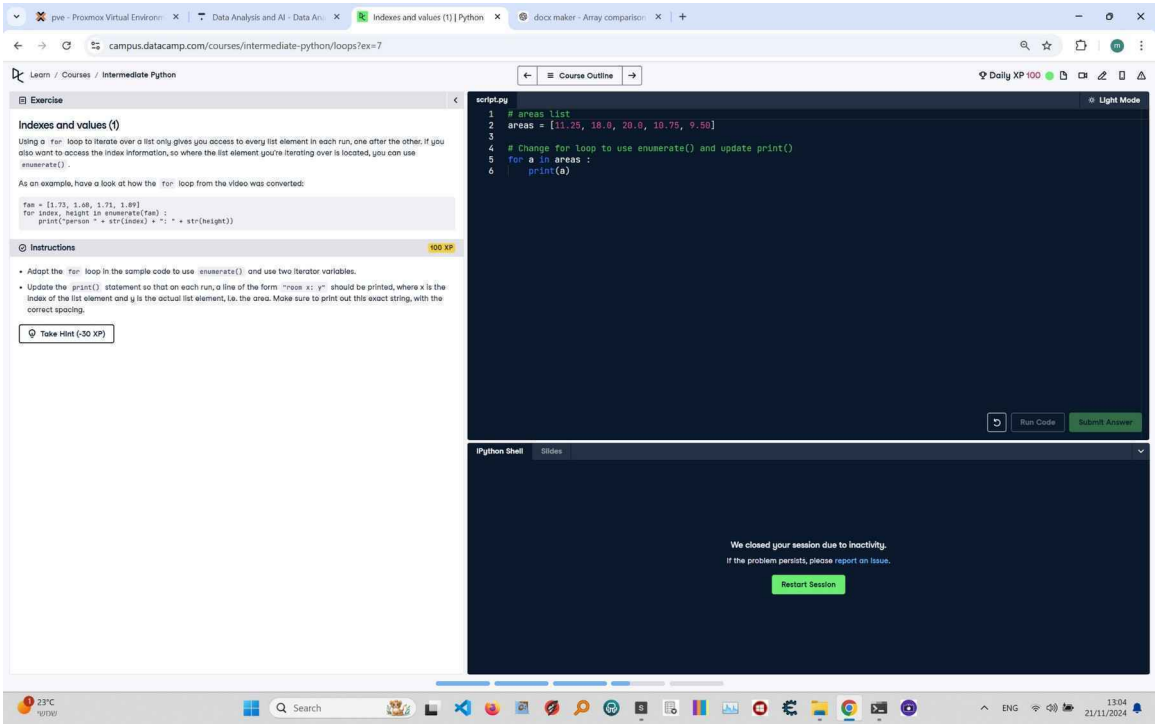
Indexes and Values (1) in Python



Question:

Using a `for` loop to iterate over a list only gives you access to every list element in each run, one after the other. If you also want to access the index information, so where the list element you're iterating over is located, you can use `enumerate()`.

As an example, have a look at how the `for` loop from the video was converted:

```python
fam = [1.73, 1.68, 1.71, 1.89]
for index, height in enumerate(fam):
    print("person " + str(index) + ": " + str(height))
```

Instructions:

1. Adapt the `for` loop in the sample code to use `enumerate()` and two iterator variables.
2. Update the `print()` statement so that on each run, a line of the form `room x: y` should be printed, where `x` is the index of the list element and `y` is the actual list element, i.e., the area. Make sure to print out this exact string, with the correct spacing.

Answer:

Here is the Python code that solves the problem:

```
# areas list
areas = [11.25, 18.0, 20.0, 10.75, 9.50]

# Adapt for loop to use enumerate() and update print()
for index, area in enumerate(areas):
    print(f"room {index}: {area}")
```

Explanation:

1. **Initialization**: The `areas` list is defined, containing the areas of different rooms in a house.
2. **Using enumerate()**: The `enumerate()` function is used to loop over the list and get both the index and the value of each element.
3. **Updating print()**: The `print()` statement is updated to use an f-string, ensuring the output format is `room x: y`, where `x` is the index of the element and `y` is the element value.
4. **Execution**: The loop runs once for each element in the `areas` list, printing the index and value for each room.