

# [220] Copying

Meena Syamkumar  
Mike Doescher

**Do not post > 5 lines on  
Piazza!**

**Reminder: partners don't  
submit project twice!**

**Cheaters caught: 0  
(Through P5)**

**14 suspicious works for P5  
(email Mike to confess)**

# Test yourself!

**A** what do variables contain?

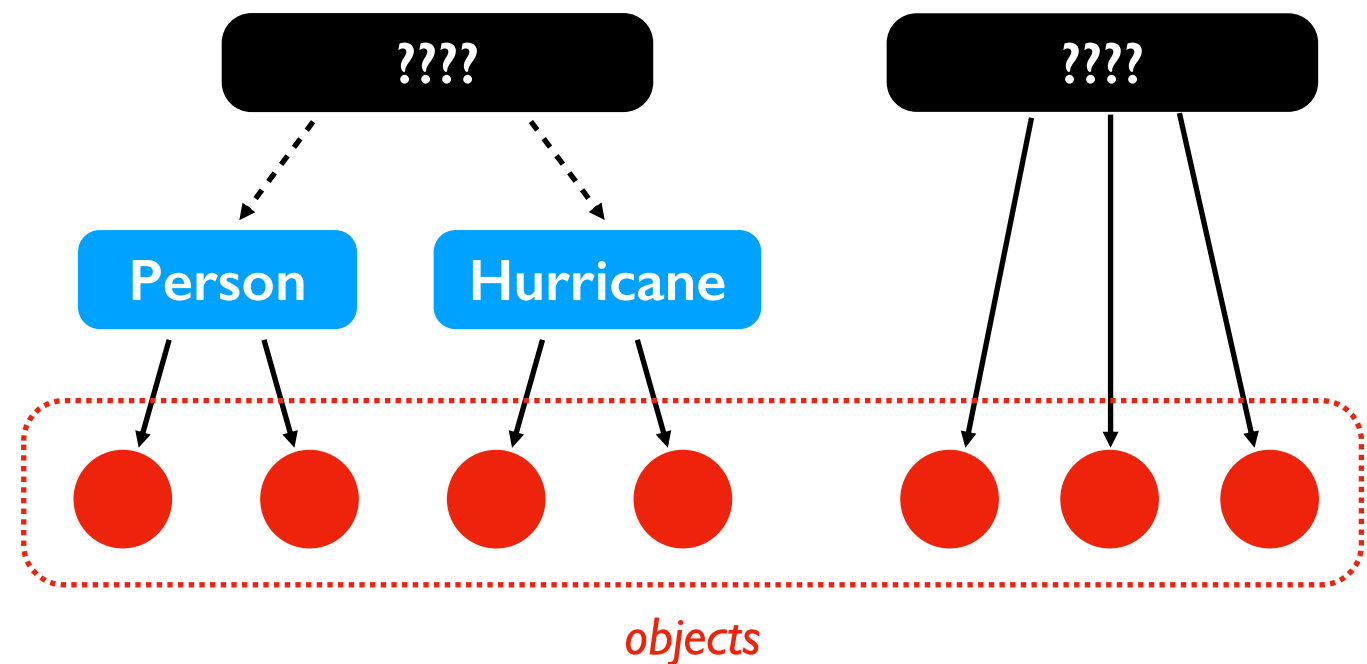
- 1 objects
- 2 references to objects

**B** how should we label the blanks in the hierarchy?

- 1 namedtuple, tuple
- 2 tuple, namedtuple

**C** which of the following live inside frames?

- 1 objects
- 2 variables



# Learning Objectives Today

Practice objects/references!

## Levels of copying

- Making a new reference
- Shallow copy
- Deep copy



<https://www.copymachinesdirect.com/copier-leasing.php>

Read:

- ✦ Sweigart Ch 4 ("References" to the end)  
<https://automatetheboringstuff.com/chapter4/>

# Today's Outline

## Review

More references

## Copying

- reference
- shallow
- deep

Worksheet

# Worksheet Problem I

# What does assignment ACTUALLY do?

```
x = [ "A" , "B" , "C" ]
```

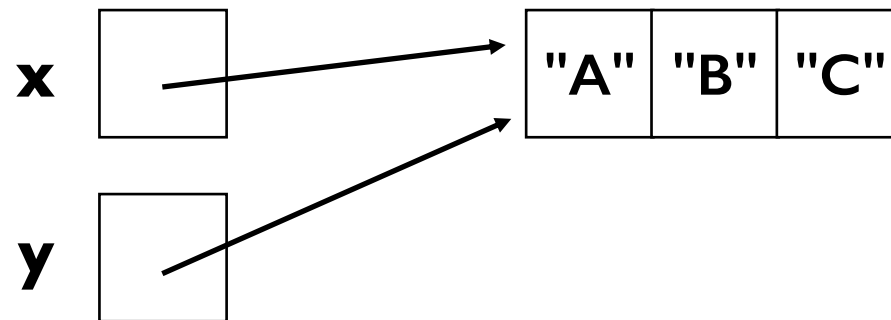
```
y = x
```

# What does assignment ACTUALLY do?

```
x = [ "A" , "B" , "C" ]
```

```
y = x
```

**YES**



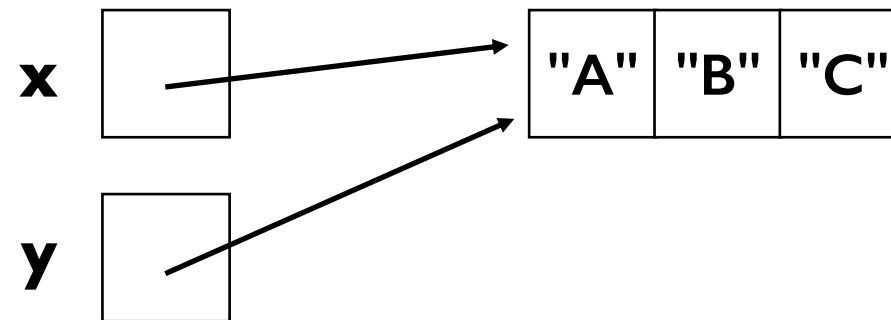
y should reference  
whatever x references

# What does assignment ACTUALLY do?

```
x = [ "A" , "B" , "C" ]
```

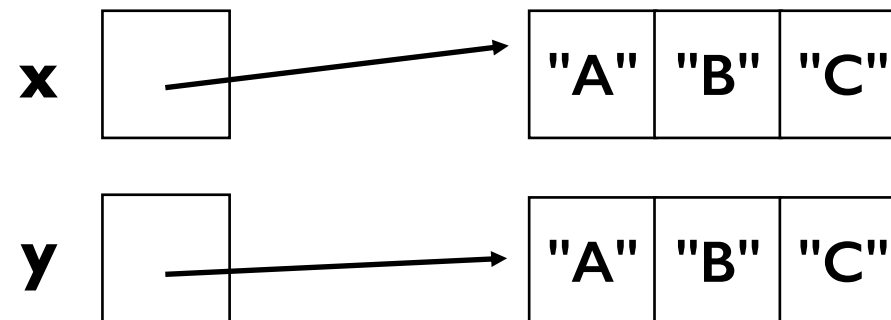
```
y = x
```

**YES**



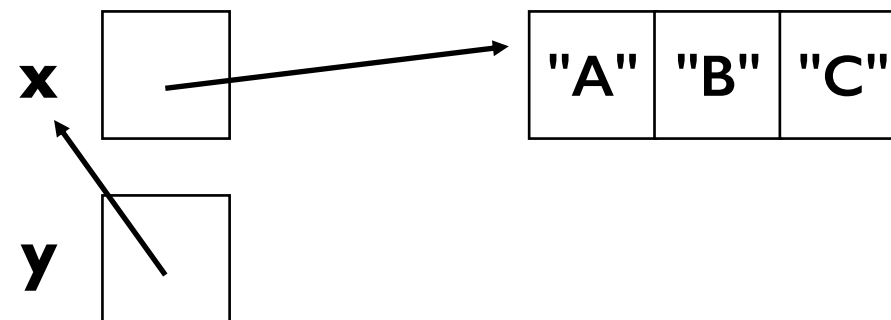
y should reference  
whatever x references

**NO**



different code would  
be needed to do this

**NO**



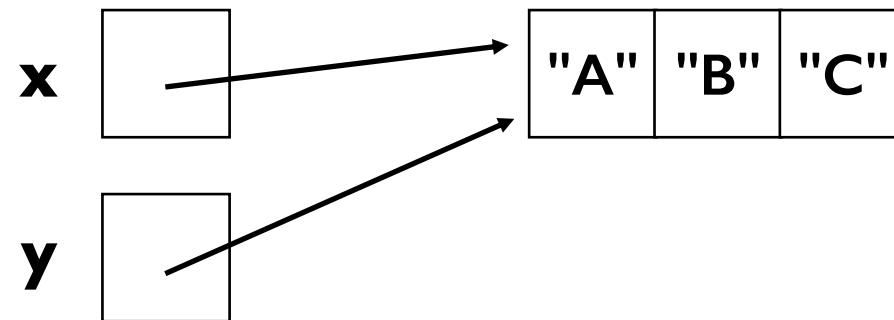
no code could ever  
make this happen



# What does assignment ACTUALLY do?

```
x = [ "A" , "B" , "C" ]
```

```
y = x
```



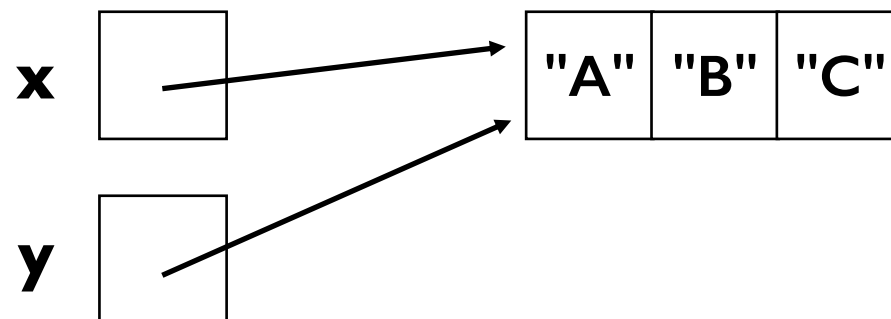
# What does assignment ACTUALLY do?

~~x~~ = [ "A", "B", "C" ]

~~y~~ = ~~x~~

```
def f(y):  
    ➡ pass
```

```
x = [ "A", "B", "C" ]  
f(x)
```



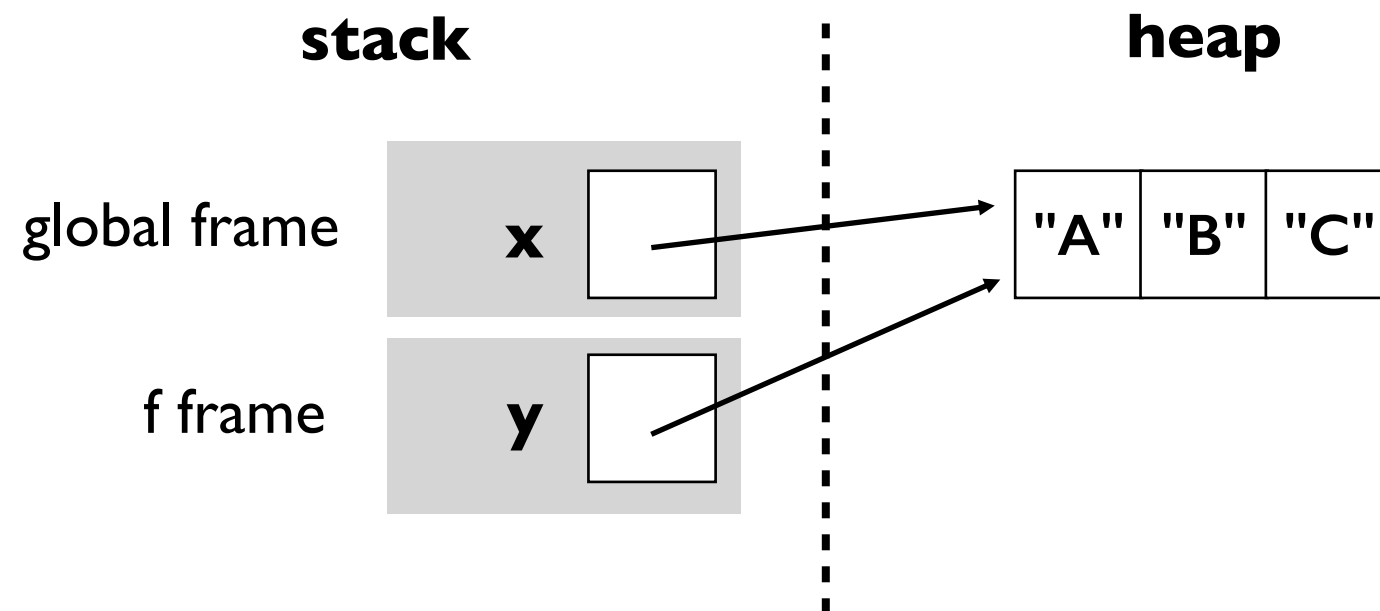
# What does assignment ACTUALLY do?

~~x~~ = [ "A", "B", "C" ]

~~y~~ = ~~x~~

```
def f(y):  
    ➡ pass
```

```
x = [ "A", "B", "C" ]  
f(x)
```



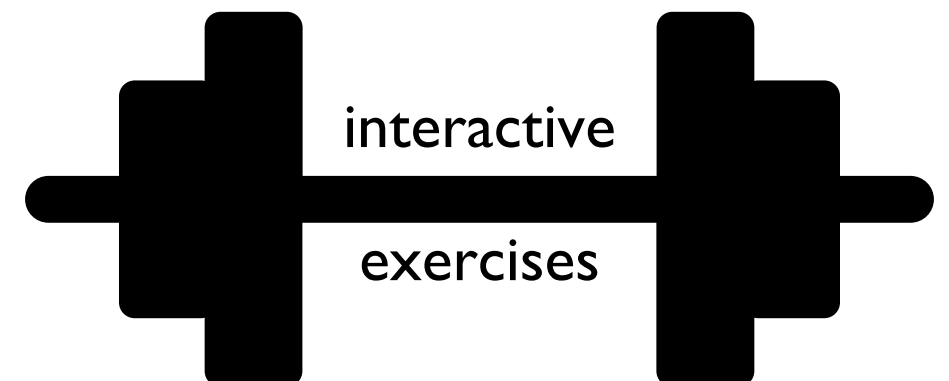
# Example I

```
x = {}
```

```
y = x
```

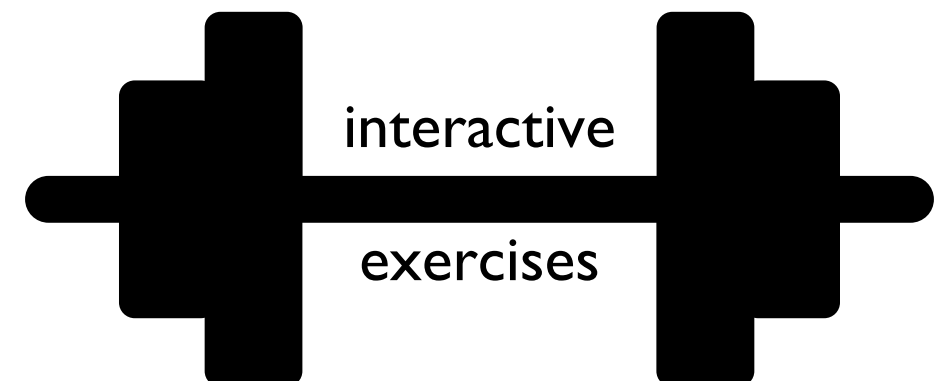
```
y["WI"] = "Madison"
```

```
print(x["WI"])
```



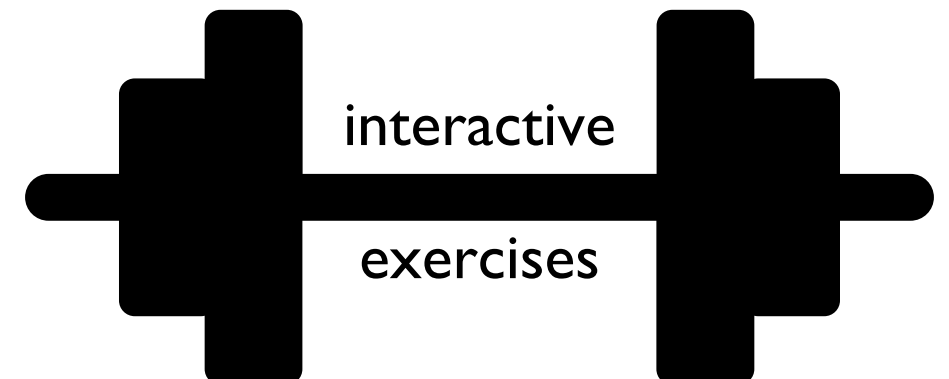
# Example 2

```
def foo(nums):  
    nums.append(3)  
    print(nums)  
items = [1,2]  
numbers = items  
foo(numbers)  
print(items)  
print(numbers)
```



# Example 3

```
x = [ "aaa", "bbb" ]  
y = x[:]  
x.pop(0)  
print(len(y))
```



# Worksheet Problems 2-6

# Today's Outline

Review

More references

Copying

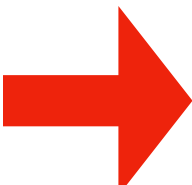
- reference
- shallow
- deep

Worksheet



```
from recordclass import recordclass
```

```
Person = recordclass("Person", ["name", "score", "age"])
```



```
alice = Person(name="Alice", score=10, age=30)  
bob = Person(name="Bob", score=8, age=25)  
team = [alice, bob]  
players = {"A": alice, "B": bob}
```

---

**State:**

*references*

*objects*

```
from recordclass import recordclass
```

```
Person = recordclass("Person", ["name", "score", "age"])
```



```
alice = Person(name="Alice", score=10, age=30)
```

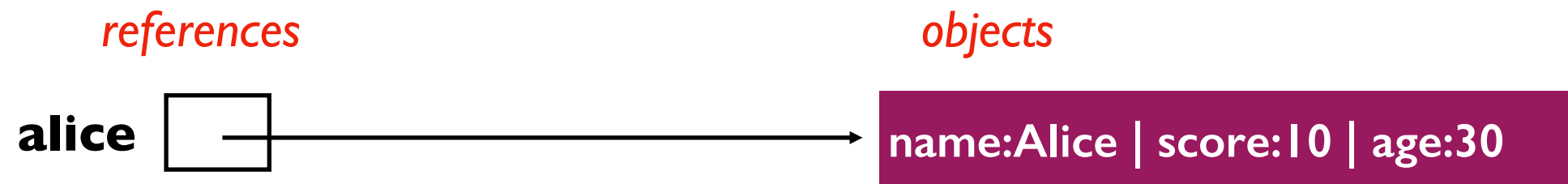
```
bob = Person(name="Bob", score=8, age=25)
```

```
team = [alice, bob]
```

```
players = {"A": alice, "B": bob}
```

---

### State:



```
from recordclass import recordclass
```

```
Person = recordclass("Person", ["name", "score", "age"])
```

```
alice = Person(name="Alice", score=10, age=30)
```

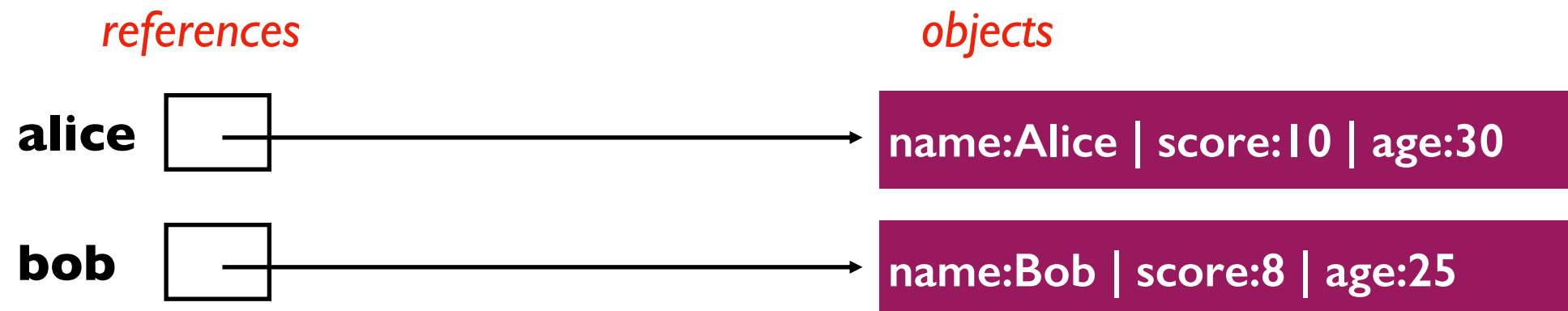
```
bob = Person(name="Bob", score=8, age=25)
```

```
team = [alice, bob]
```

```
players = {"A": alice, "B": bob}
```

---

### State:



```
from recordclass import recordclass
```

```
Person = recordclass("Person", ["name", "score", "age"])
```

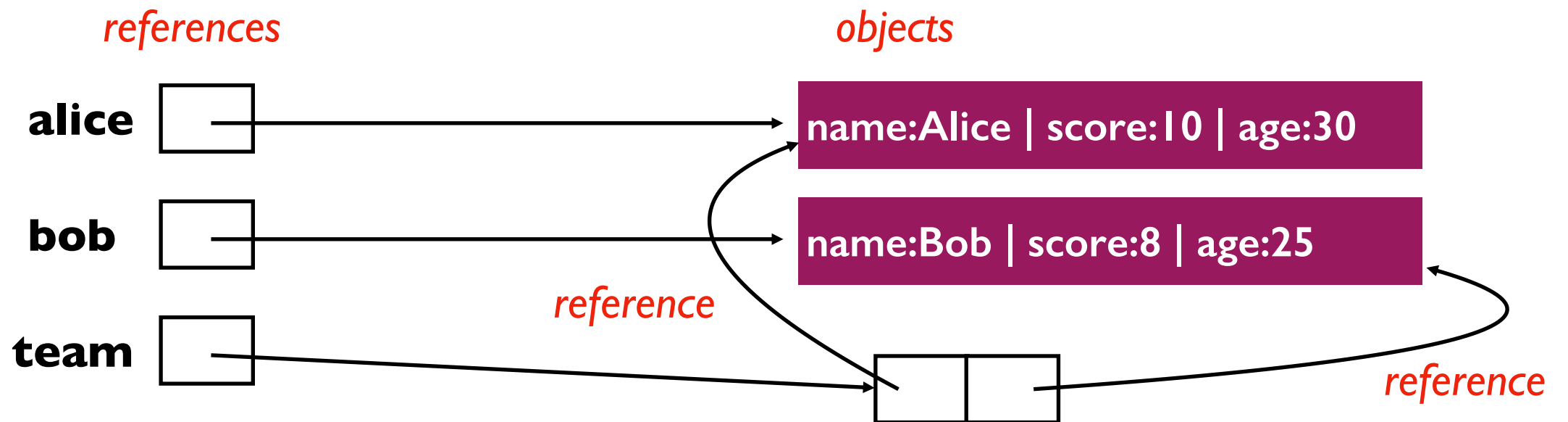
```
alice = Person(name="Alice", score=10, age=30)
```

```
bob = Person(name="Bob", score=8, age=25)
```

```
team = [alice, bob]
```

```
players = {"A": alice, "B": bob}
```

### State:



```
from recordclass import recordclass
```

```
Person = recordclass("Person", ["name", "score", "age"])
```

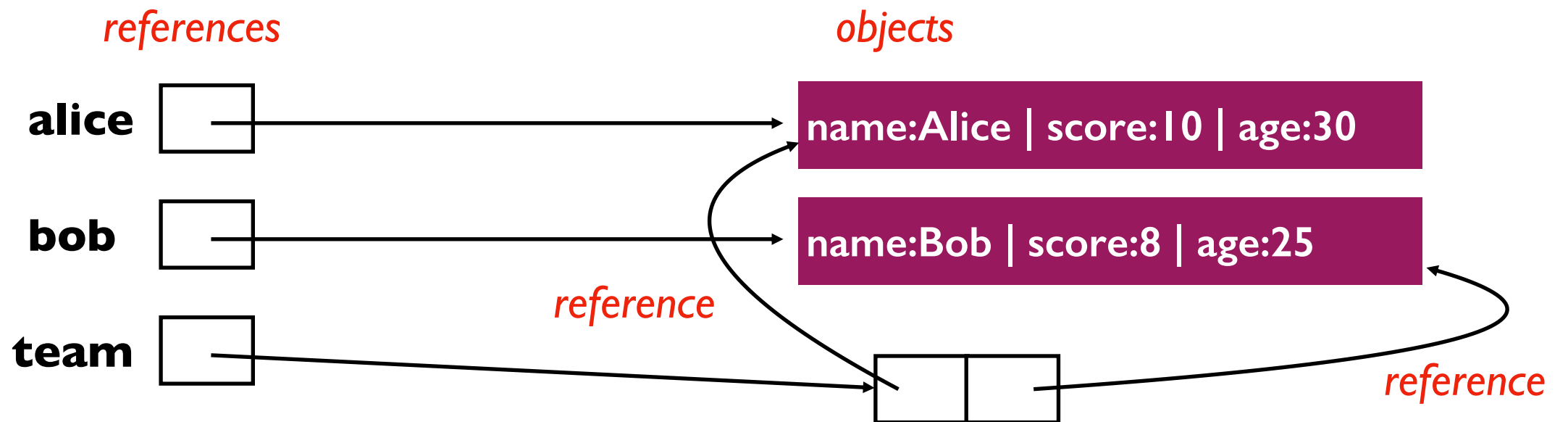
```
alice = Person(name="Alice", score=10, age=30)
```

```
bob = Person(name="Bob", score=8, age=25)
```

```
team = [alice, bob]
```

```
players = {"A": alice, "B": bob}
```

**State:**



**what DID NOT happen:** `team` contains the `alice` and `bob` variables

**what DID happen:** `team` contains references to the objects referenced by `bob` and `alice`

```
from recordclass import recordclass
```

```
Person = recordclass("Person", ["name", "score", "age"])
```

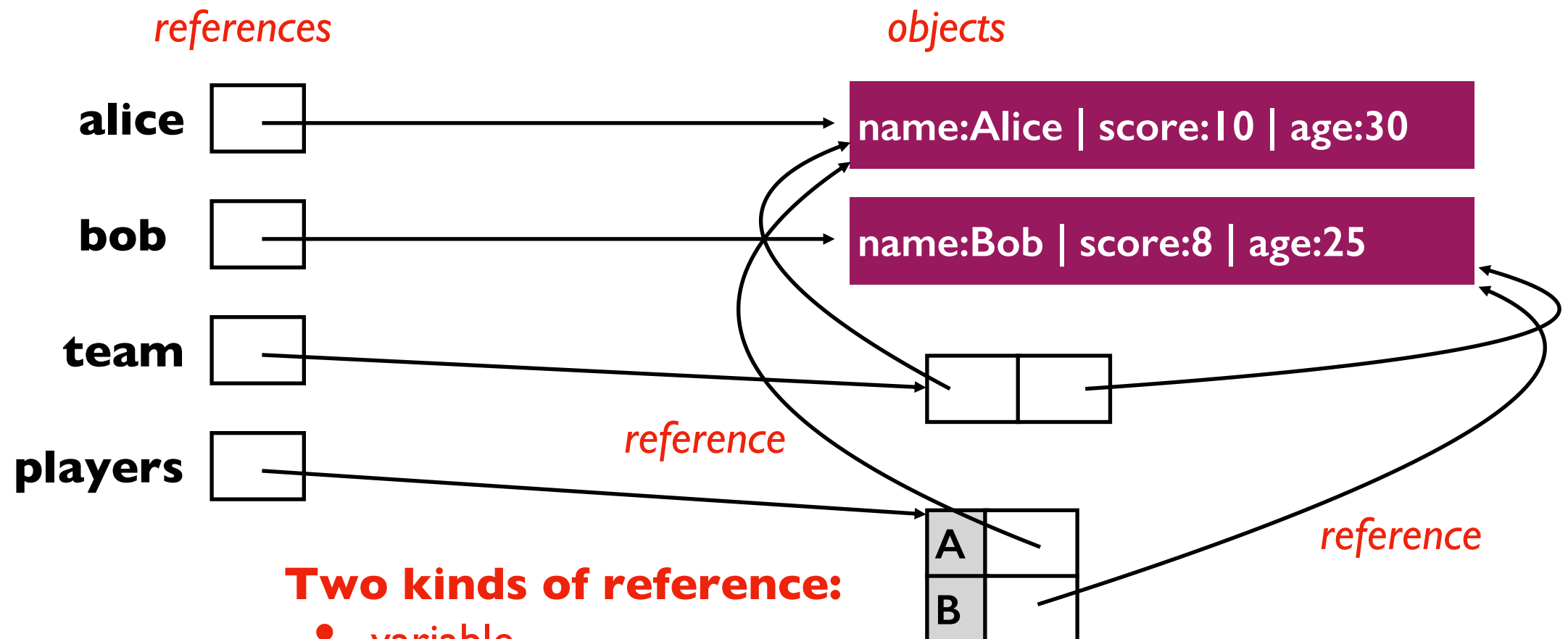
```
alice = Person(name="Alice", score=10, age=30)
```

```
bob = Person(name="Bob", score=8, age=25)
```

```
team = [alice, bob]
```

```
players = {"A": alice, "B": bob}
```

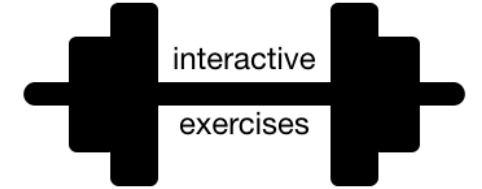
## State:



### Two kinds of reference:

- variable
- item in list, dict, etc

# Deduplication review



```
x = 2**4
y = 2**4
z = y
print(x == z, x is z)
```

```
x = 2**100
y = 2**100
z = y
print(x == z, x is z)
```

# Today's Outline

Review

More references

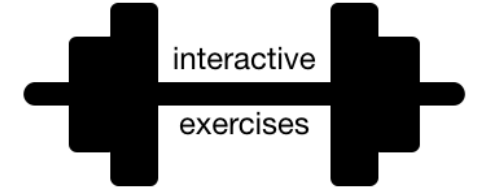
Copying

- reference
- shallow
- deep

Worksheet



# Three Levels of Copy



When should we  
use which one?

```
import copy
x = [
    {"name": "A", "score": 88},
    {"name": "B", "score": 111},
    {"name": "C", "score": 100}]
```

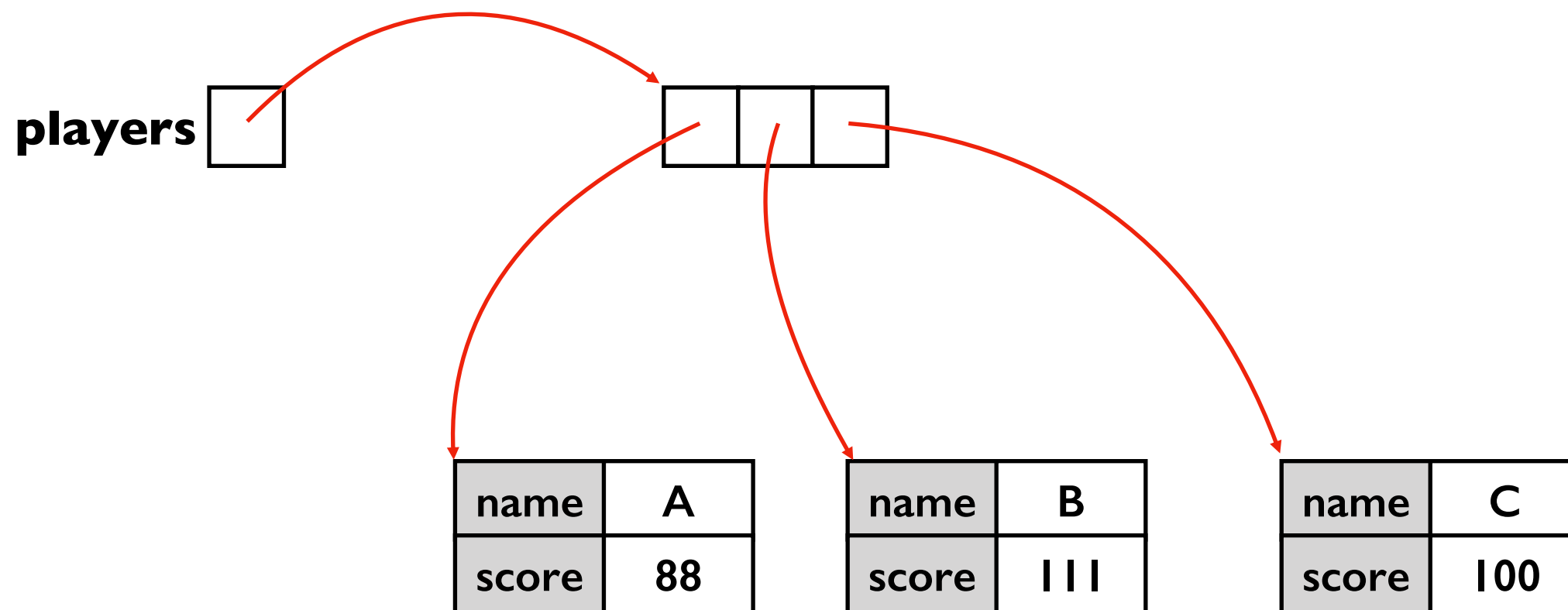
# uncomment one of these

```
#y = x ← reference copy [fastest, most dangerous]
#y = copy.copy(x) ← shallow copy
#y = copy.deepcopy(x) ← deep copy [slowest, safest]
```

# Example: Player Scores

```
players = [  
    {"name": "A", "score": 88},  
    {"name": "B", "score": 111},  
    {"name": "C", "score": 100}  
]
```

Depending on the use case,  
there are **three ways** we might  
"copy" the player's data



# Example: Player Scores

```
players = [  
    {"name": "A", "score": 88},  
    {"name": "B", "score": 111},  
    {"name": "C", "score": 100}  
]
```

## Use Case 1

Get max score  
(reference copy)

## Use Case 2

Get median score  
(shallow copy)

## Use Case 3

Record historical scores  
(deep copy)

name	A
score	88

score	111
-------	-----

score	100
-------	-----

# Example: Player Scores

```
players = [  
    {"name": "A", "score": 88},  
    {"name": "B", "score": 111},  
    {"name": "C", "score": 100}  
]
```

## Use Case 1

Get max score  
(reference copy)

## Use Case 2

Get median score  
(shallow copy)

## Use Case 3

Record historical scores  
(deep copy)

name	A
score	88

score	111
-------	-----

score	100
-------	-----


```
def max_score(people):  
    highest = None  
    for p in people:  
        if highest == None or p["score"] > highest:  
            highest = p["score"]  
    return highest
```

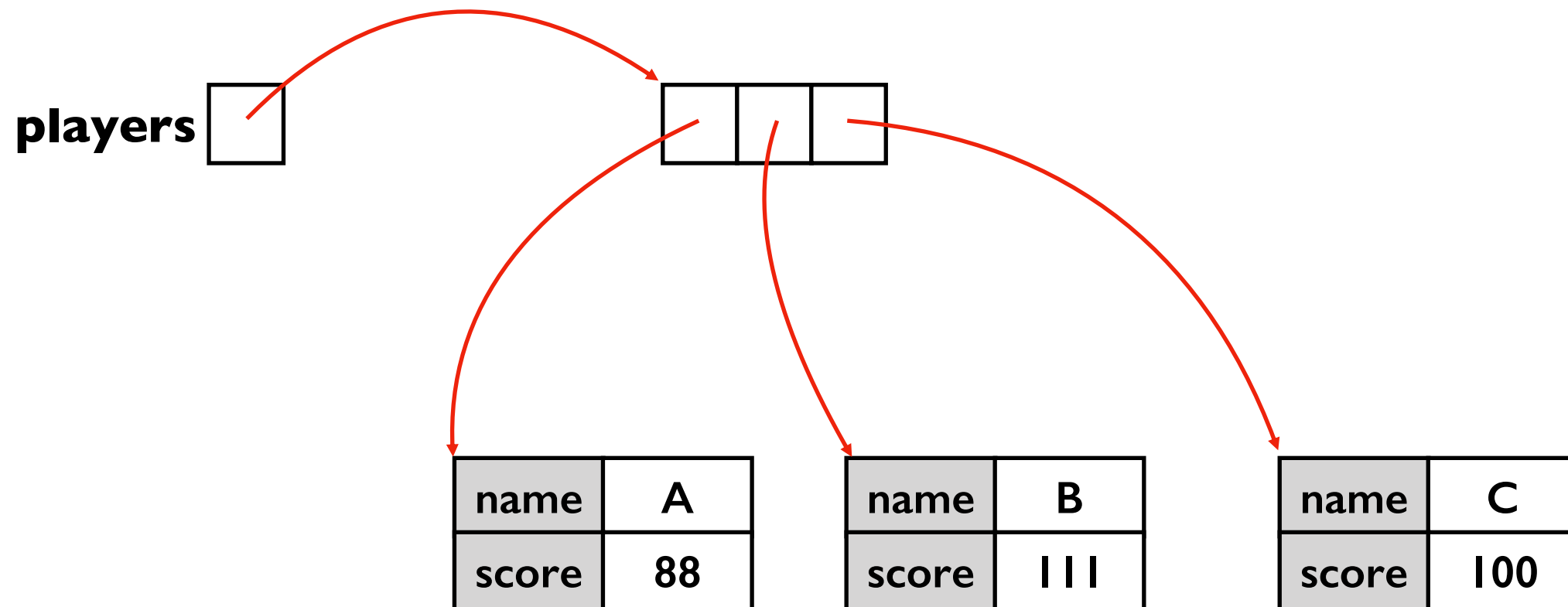


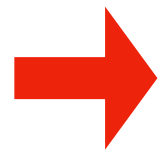
```
players = ...  
m = max_score(players)
```

---

```
def max_score(people):  
    highest = None  
    for p in people:  
        if highest == None or p["score"] > highest:  
            highest = p["score"]  
    return highest
```

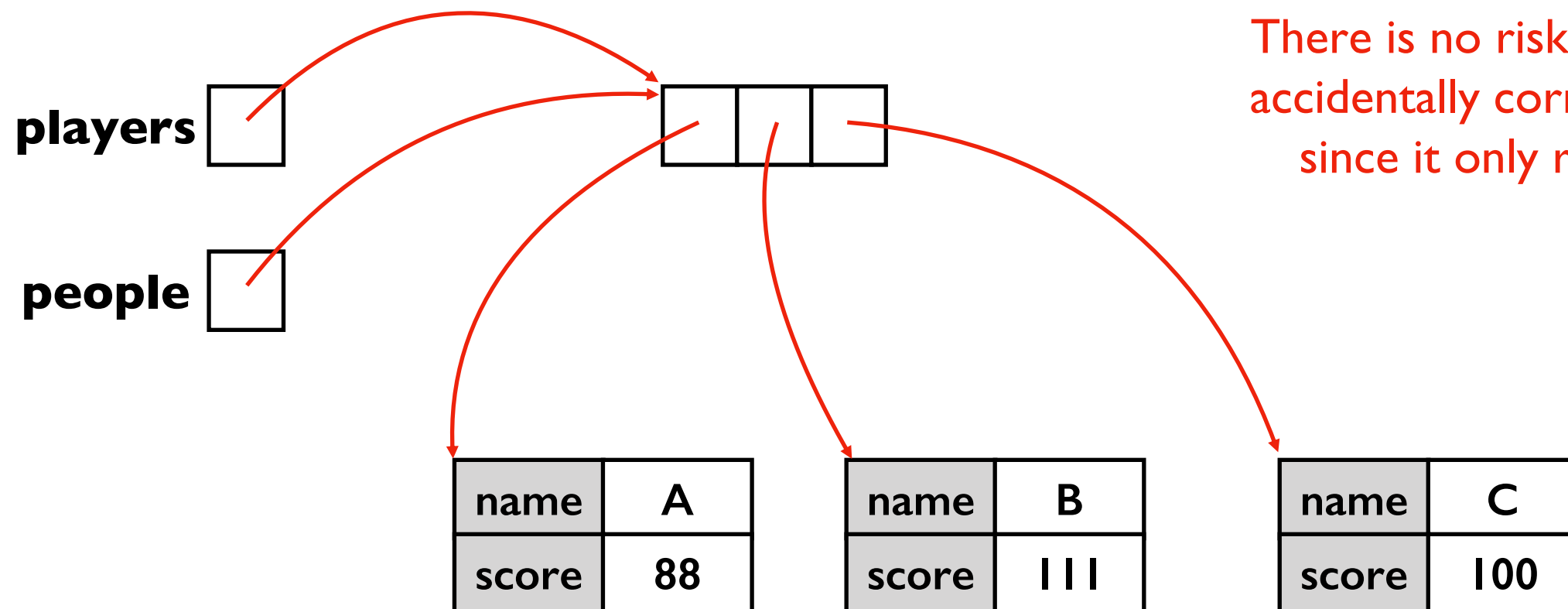
 **players** = ...  
m = max\_score(**players**)



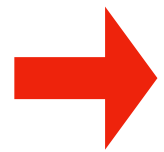


```
def max_score(people):  
    highest = None  
    for p in people:  
        if highest == None or p["score"] > highest:  
            highest = p["score"]  
    return highest
```

```
players = ...  
m = max_score(players)
```

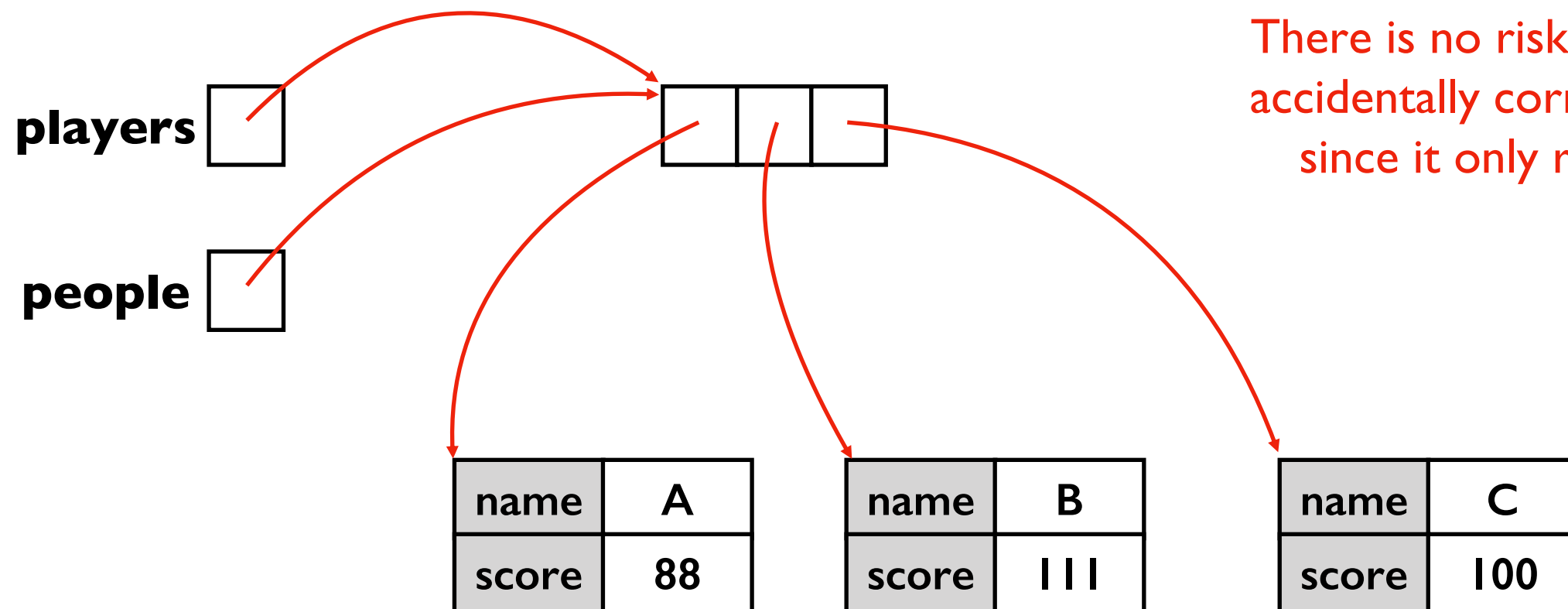


There is no risk of `max_score` accidentally corrupting `players` since it only reads `people`



```
def max_score(people):  
    highest = None  
    for p in people:  
        if highest == None or p["score"] > highest:  
            highest = p["score"]  
    return highest
```

```
players = ...  
m = max_score(players)
```



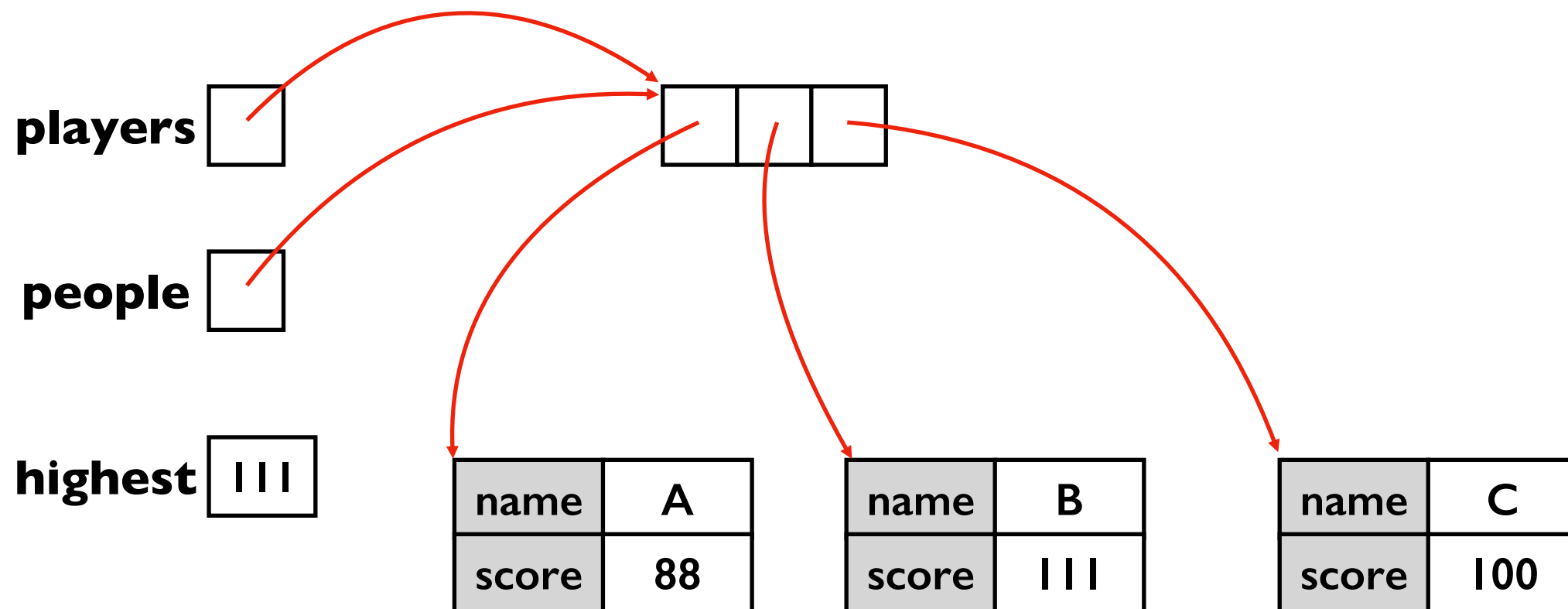
There is no risk of `max_score` accidentally corrupting `players` since it only reads `people`



```
def max_score(people):  
    highest = None  
    for p in people:  
        if highest == None or p["score"] > highest:  
            highest = p["score"]  
    return highest
```



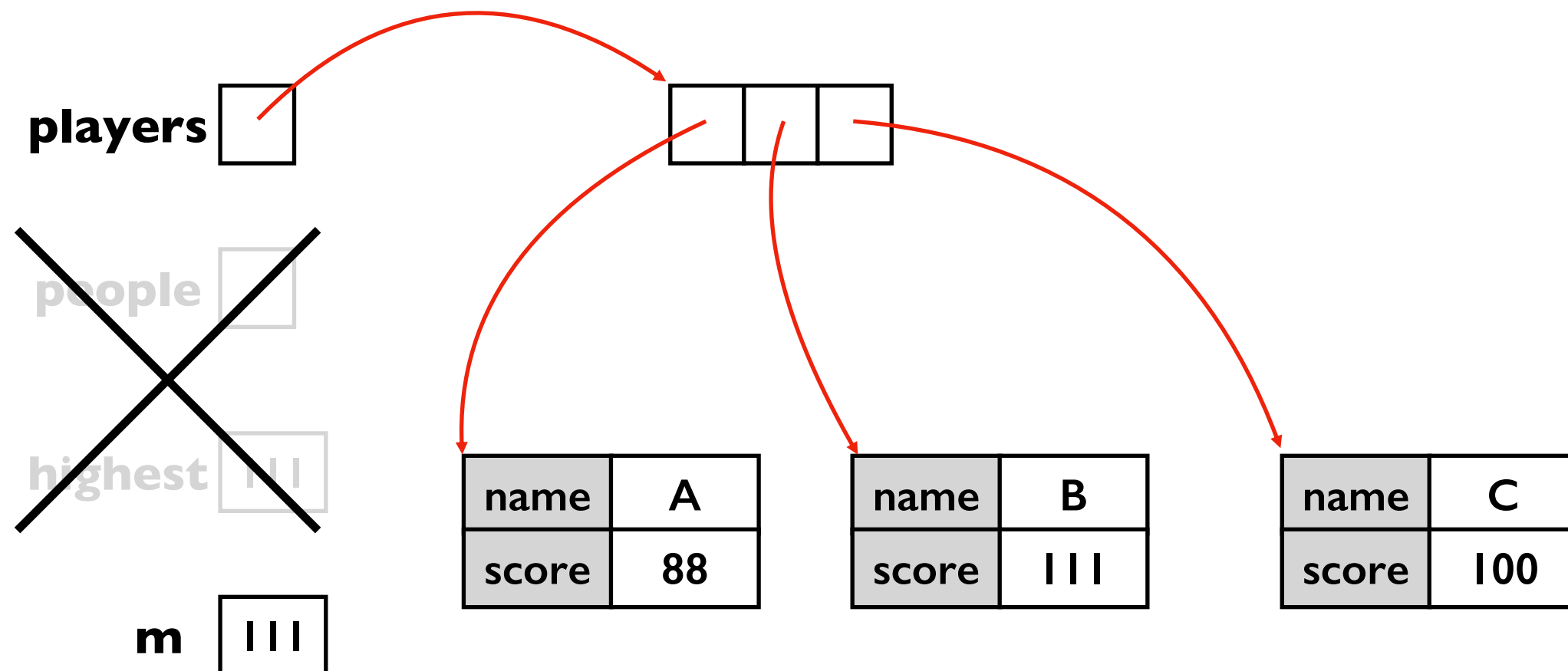
```
players = ...  
m = max_score(players)
```



```
def max_score(people):  
    highest = None  
    for p in people:  
        if highest == None or p["score"] > highest:  
            highest = p["score"]  
    return highest
```

**players** = ...

m = max\_score(**players**)



# Example: Player Scores

```
players = [  
    {"name": "A", "score": 88},  
    {"name": "B", "score": 111},  
    {"name": "C", "score": 100}  
]
```

## Use Case 1

Get max score  
(reference copy)

## Use Case 2

Get median score  
(shallow copy)

## Use Case 3

Record historical scores  
(deep copy)

name	A
score	88

score	111
-------	-----

score	100
-------	-----

```
def median_score(people):  
    people = copy.copy(people)  
    people.sort(...)  
    # TODO: return score for middle of people
```



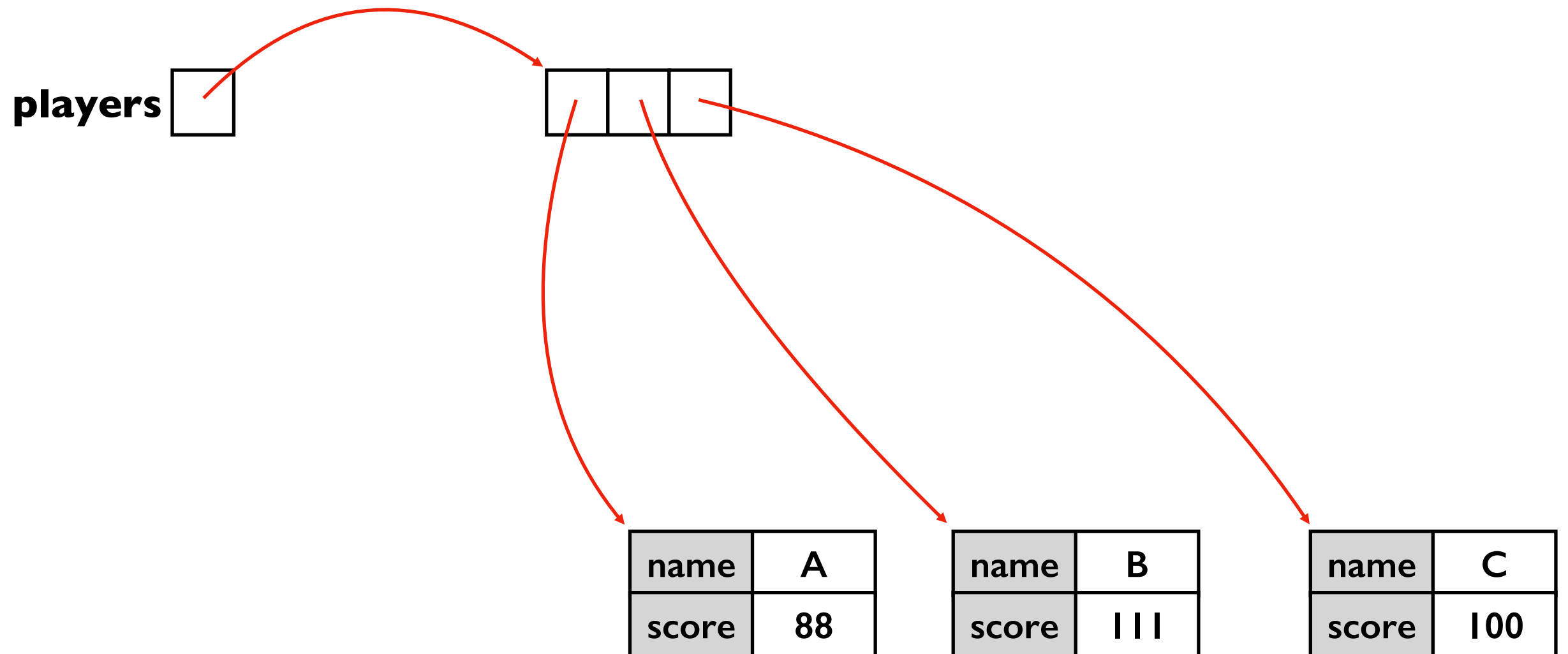
```
players = ...  
m = median_score(players)
```

---

```
def median_score(people):  
    people = copy.copy(people)  
    people.sort(...)  
    # TODO: return score for middle of people
```

➔ **players** = ...  
m = median\_score(**players**)

---

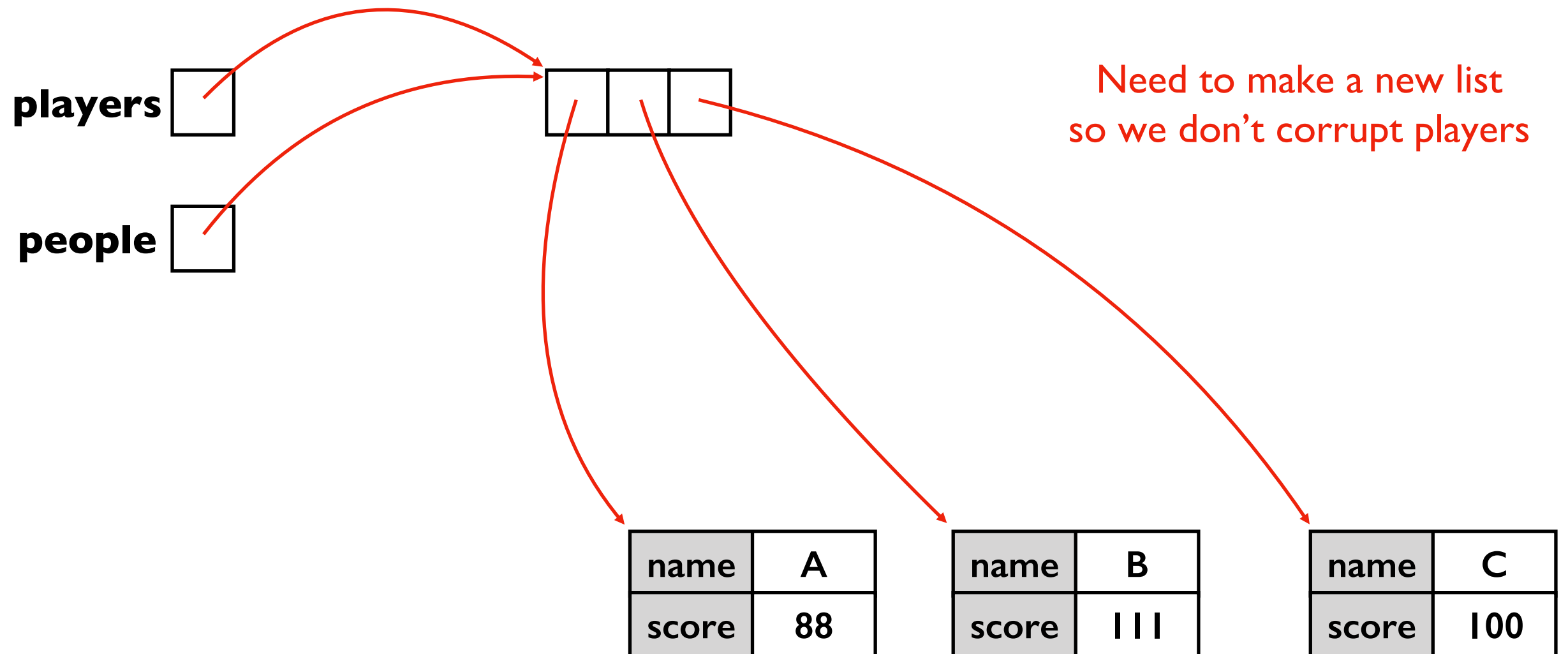




```
def median_score(people):  
    people = copy.copy(people)  
    people.sort(...)  
    # TODO: return score for middle of people
```

```
players = ...  
m = median_score(players)
```

---

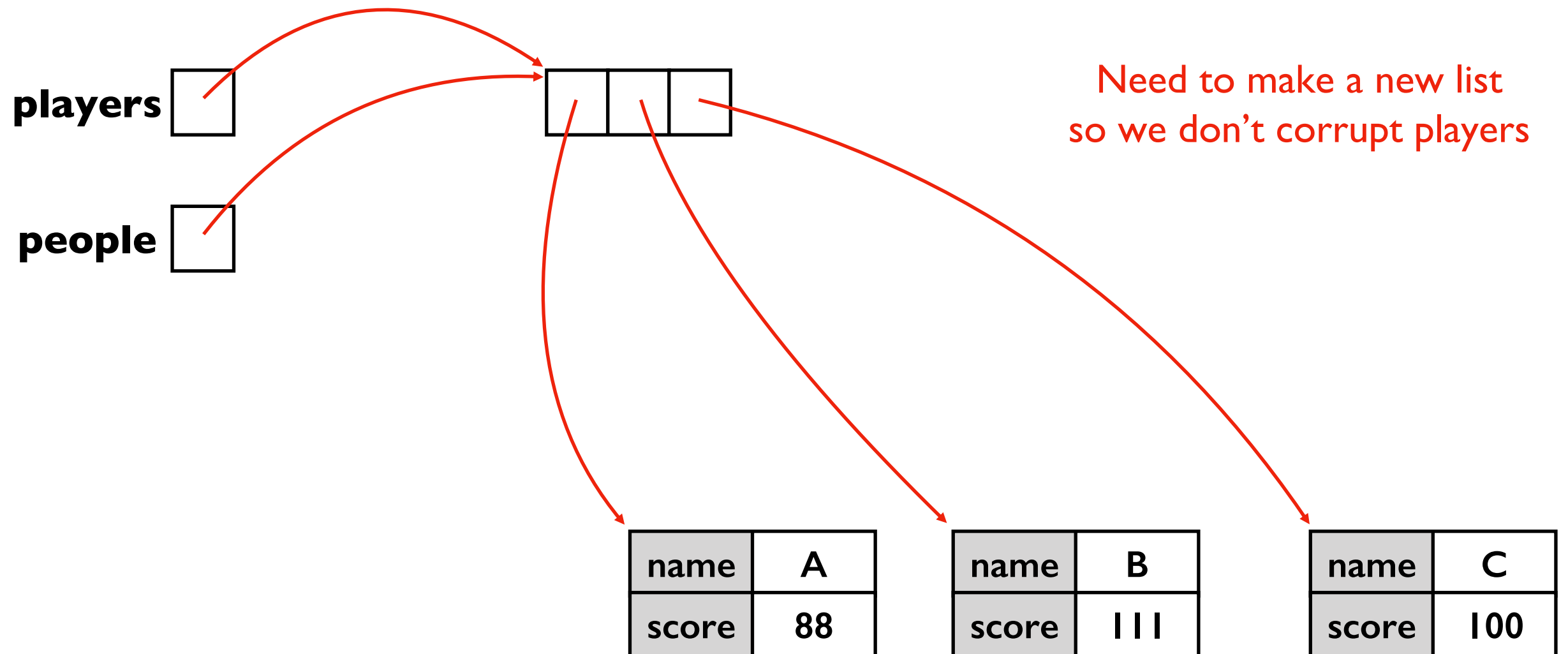




```
def median_score(people):  
    people = copy.copy(people)  
    people.sort(...)  
    # TODO: return score for middle of people
```

```
players = ...  
m = median_score(players)
```

---

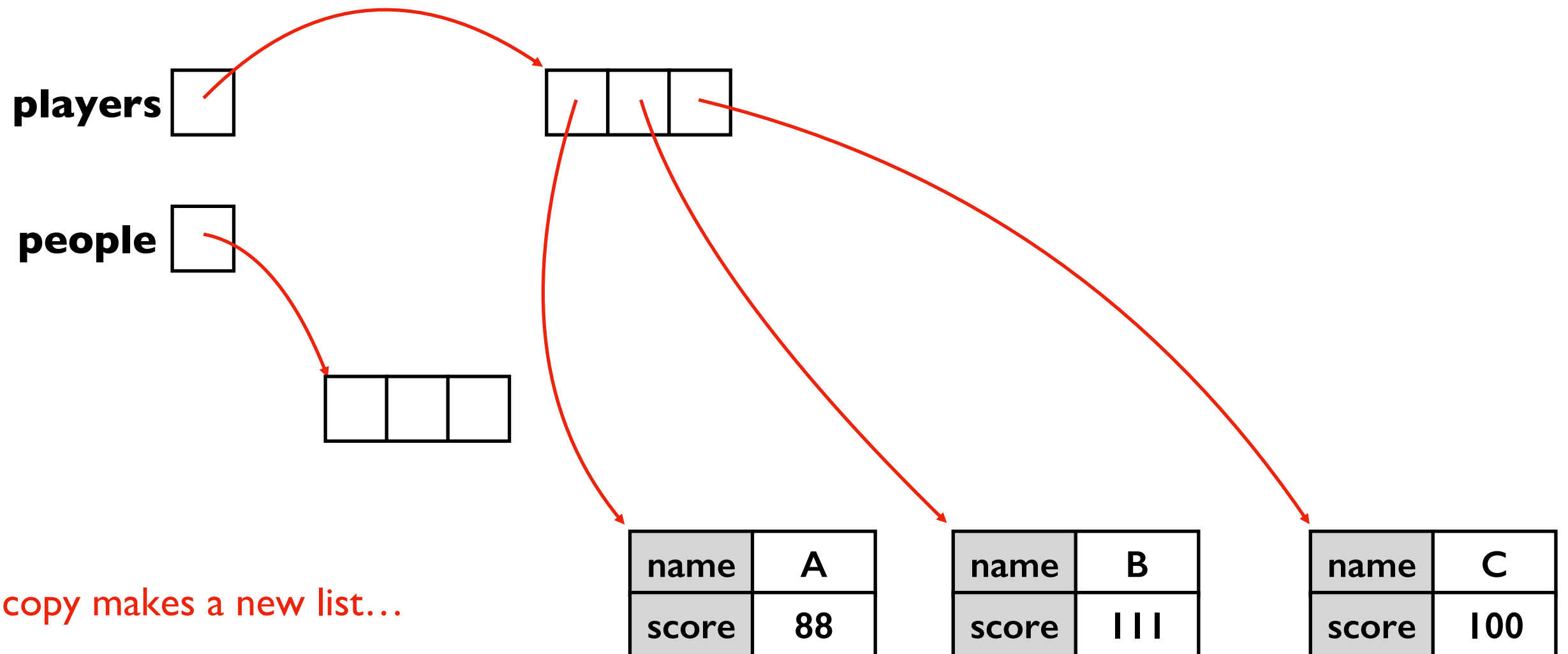




```
def median_score(people):  
    people = copy.copy(people)  
    people.sort(...)  
    # TODO: return score for middle of people
```

```
players = ...  
m = median_score(players)
```

---



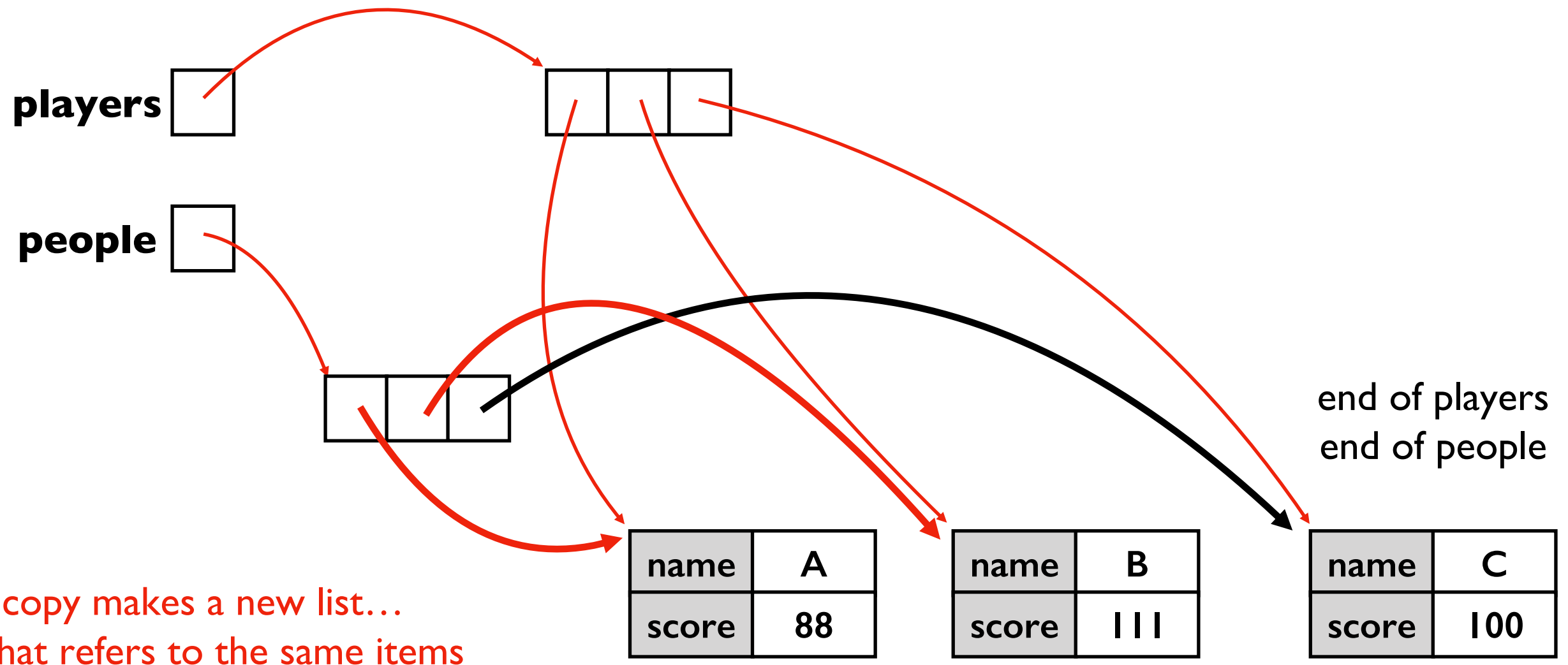




```
def median_score(people):  
    people = copy.copy(people)  
    people.sort(...)  
    # TODO: return score for middle of people
```

```
players = ...  
m = median_score(players)
```

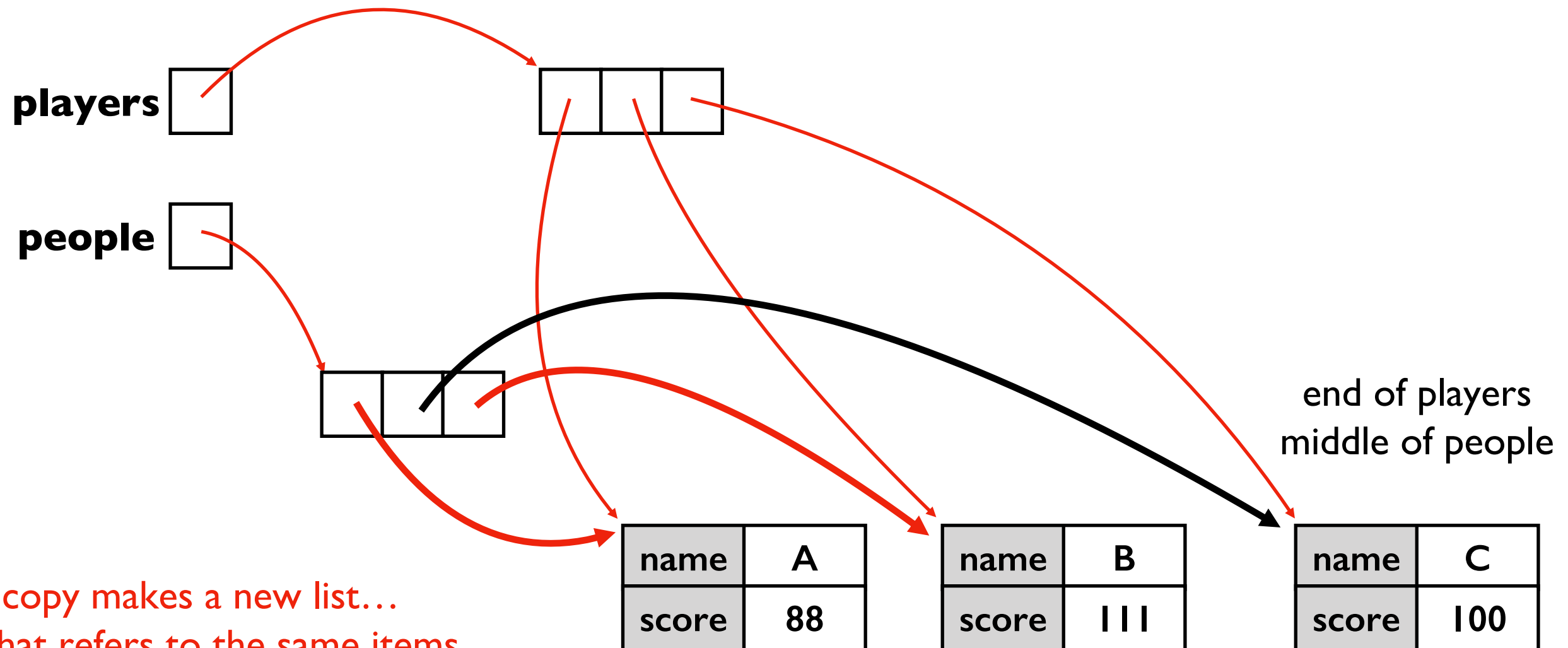
---



```
def median_score(people):  
    people = copy.copy(people)  
    people.sort(...)  
    # TODO: return score for middle of people
```



```
players = ...  
m = median_score(players)
```

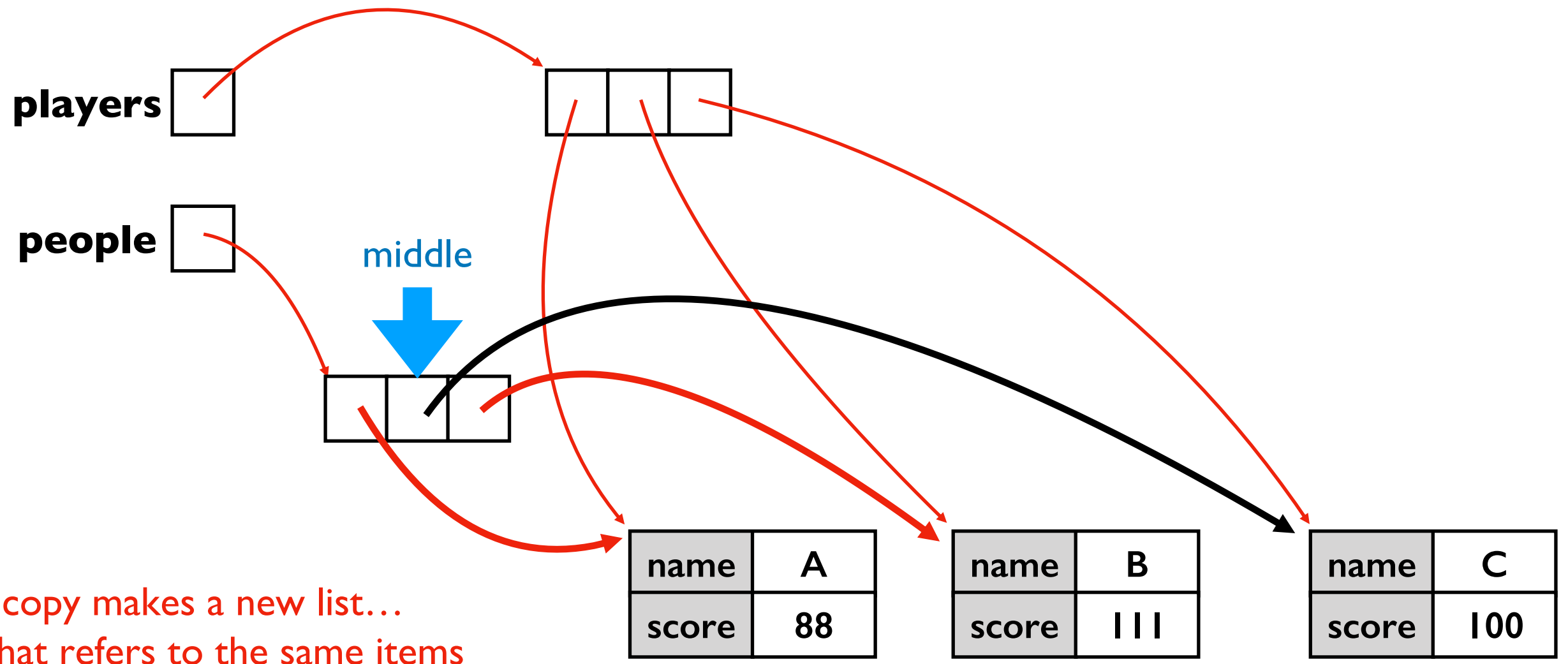




```
def median_score(people):  
    people = copy.copy(people)  
    people.sort(...)  
    # TODO: return score for middle of people
```

```
players = ...  
m = median_score(players)
```

---



# Example: Player Scores

```
players = [  
    {"name": "A", "score": 88},  
    {"name": "B", "score": 111},  
    {"name": "C", "score": 100}  
]
```

## Use Case 1

Get max score  
(reference copy)

## Use Case 2

Get median score  
(shallow copy)


## Use Case 3

Record historical scores  
(deep copy)

name	A
score	88

score	111
-------	-----

score	100
-------	-----



```
players = ...
players_before = copy.deepcopy(players)

# make changes to players
players[0]["score"] += 10

print("score change:",
      players[0]["score"] - players_before[0]["score"])
```

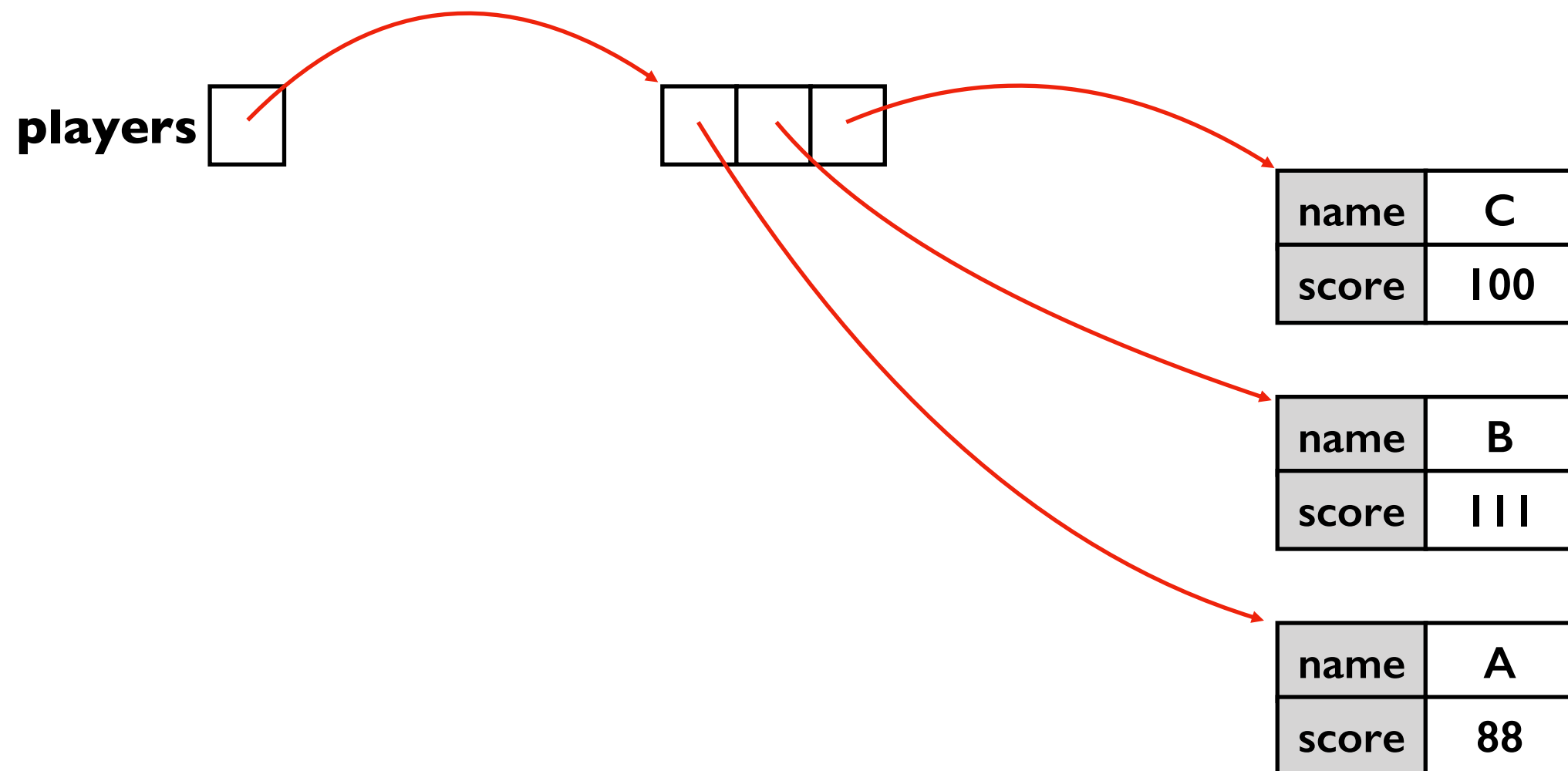
---

➔ **players** = ...  
**players\_before** = `copy.deepcopy(players)`

# make changes to **players**  
**players**[0]["score"] += 10

`print("score change:",  
 players[0]["score"] - players_before[0]["score"])`

---

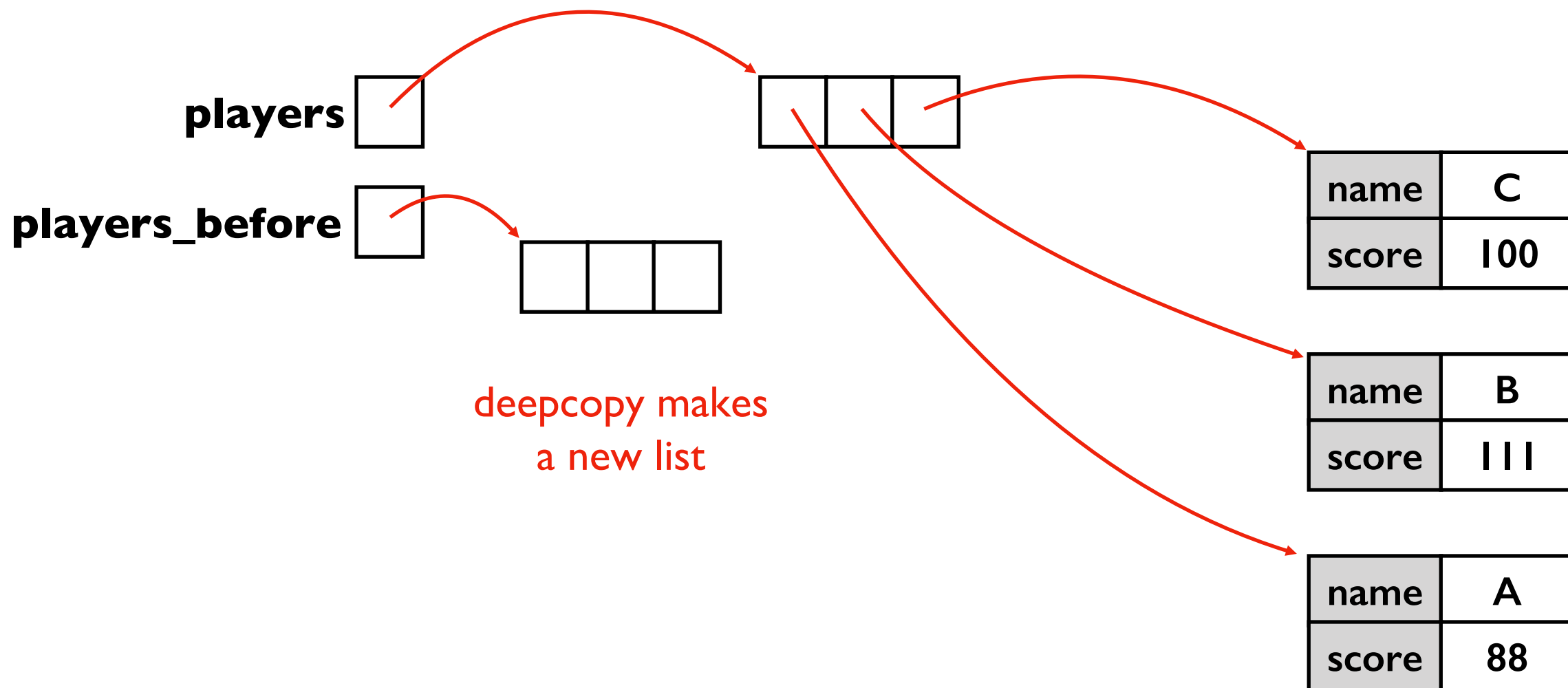


➔ `players = ...`  
`players_before = copy.deepcopy(players)`

`# make changes to players`  
`players[0]["score"] += 10`

`print("score change:",`  
`players[0]["score"] - players_before[0]["score"])`

---



```
players = ...
```

```
players_before = copy.deepcopy(players)
```

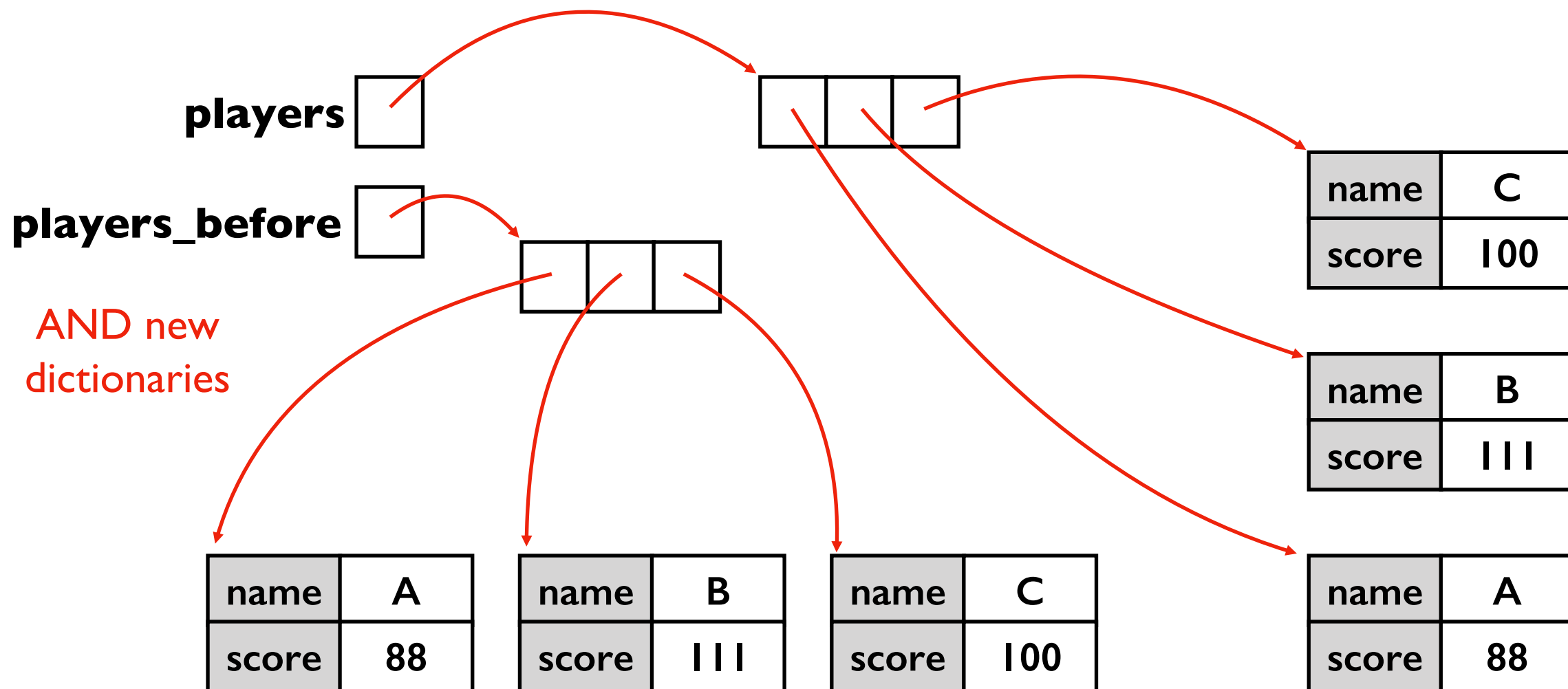


```
# make changes to players
```

```
players[0]["score"] += 10
```

```
print("score change:",
```

```
      players[0]["score"] - players_before[0]["score"])
```





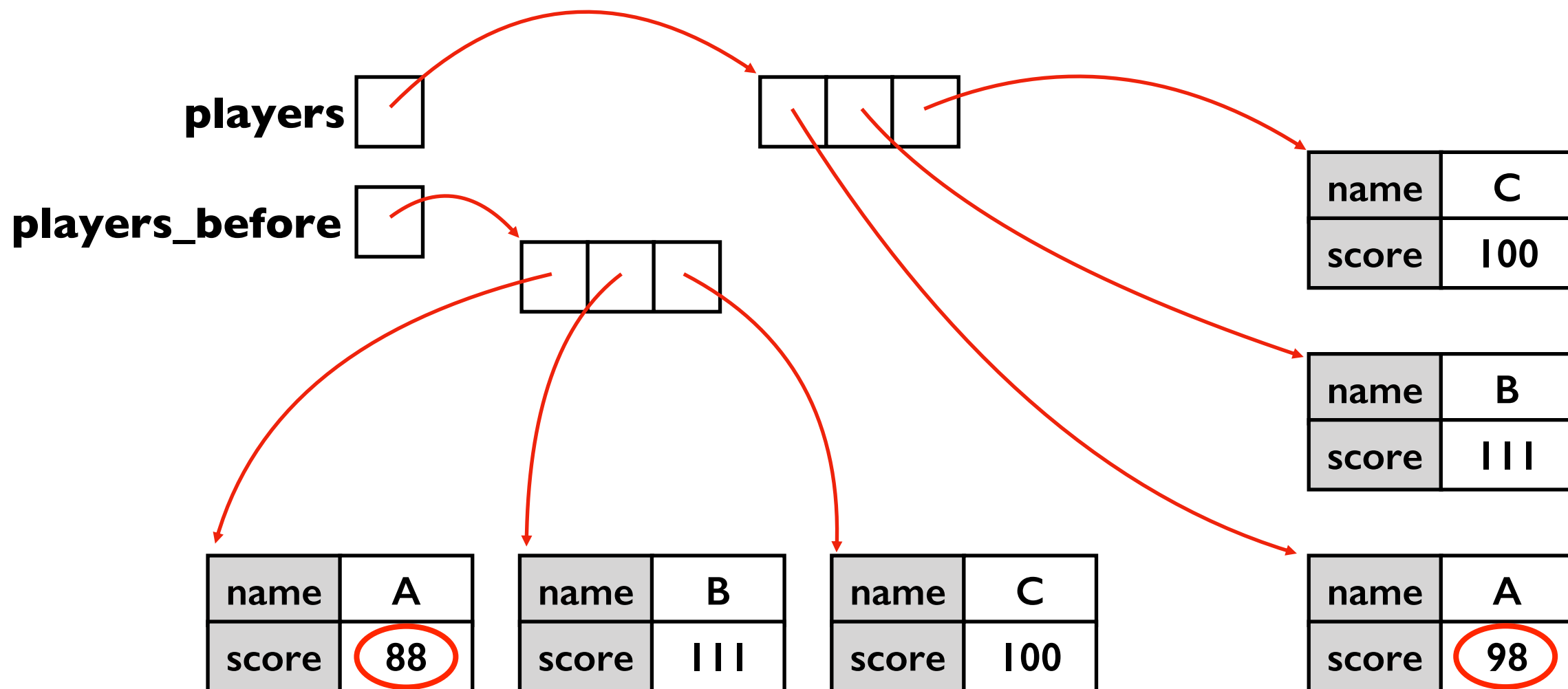
```
players = ...  
players_before = copy.deepcopy(players)
```

```
# make changes to players  
players[0]["score"] += 10
```

➔ 

```
print("score change:",  
      players[0]["score"] - players_before[0]["score"])
```

---

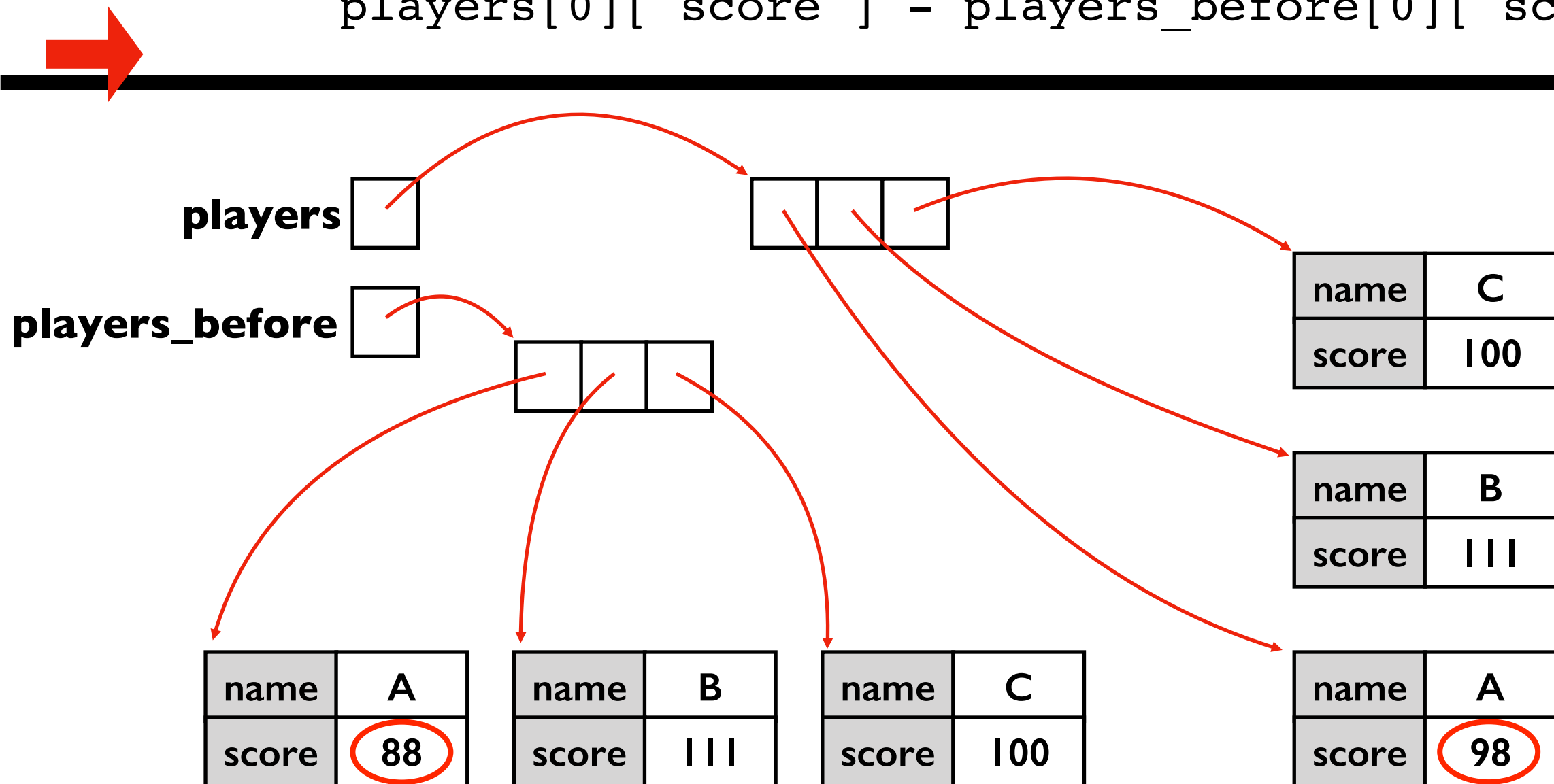


```
players = ...
players_before = copy.deepcopy(players)

# make changes to players
players[0]["score"] += 10

print("score change:",      prints 10
      players[0]["score"] - players_before[0]["score"])
```

---



# Today's Outline

Review

More references

Copying

- reference
- shallow
- deep

Worksheet

# Worksheet Problems 7-11