11.5. Aliasing and copying

Because dictionaries are mutable, you need to be aware of aliasing (as we saw with lists). Whenever two variables refer to the same dictionary object, changes to one affect the other. For example, `opposites` is a dictionary that contains pairs of opposites.

Save & Run | Original - 1 of 1 | Show in CodeLens

```
opposites = {'up': 'down', 'right': 'wrong', 'true': 'false'}
alias = opposites

print(alias is opposites)

alias['right'] = 'left'
print(opposites['right'])

```

```
True
left
```

Activity: 1 -- ActiveCode (ac10_4_1)

As you can see from the `is` operator, `alias` and `opposites` refer to the same object.

If you want to modify a dictionary and keep a copy of the original, use the dictionary `copy` method. Since *acopy* is a copy of the dictionary, changes to it will not effect the original.

```
acopy = opposites.copy()
acopy['right'] = 'left'    # does not change opposites
```

Check your understanding

dictionaries-4-1: What is printed by the following statements?

```
mydict = {"cat":12, "dog":6, "elephant":23, "bear":20}
yourdict = mydict
yourdict["elephant"] = 999
print(mydict["elephant"])
```

- A. 23
- B. None
- C. 999
- D. Error, there are two different keys named elephant.

Check me | Compare me

✔ Yes, since yourdict is an alias for mydict, the value for the key elephant has been changed.

Activity: 2 -- Multiple Choice (question10_4_1)

You have attempted 3 of 2 activities on this page

11.4. Dictionary methods">
ctionary methods">
11.6. Introduction: Accumulating Multiple Results In a Dictionary">Next Section - 11.6. Introduction: Accumulating Multiple Results In a Dictionary

| Back to top