More on Dictionaries
Sets

Example usage of dictionaries

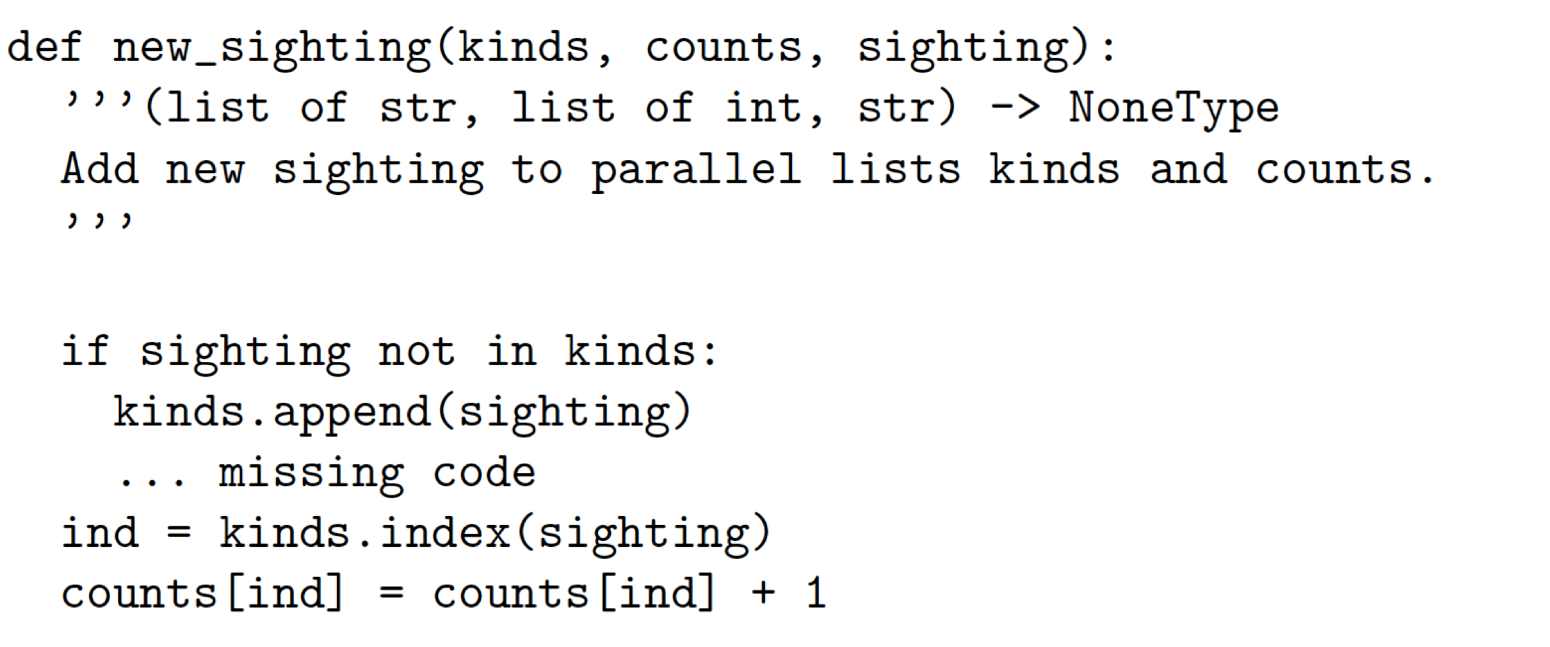
- Let's say we're bird-watching, and we want to keep track of the number of each type of bird we've seen

kind	count
`falcon`	`1`
`owl`	`5`
`hawk`	`2`
`eagle`	`11`

- One approach: parallel lists
- The element `kinds[i]` corresponds with `counts[i]`

```
kinds = ['falcon', 'owl', 'hawk', 'eagle’]
counts = [1, 5, 2, 11]
```

Concep Test:



```
def new_sighting(kinds, counts, sighting):
  '''(list of str, list of int, str) -> NoneType
  Add new sighting to parallel lists kinds and counts.
  '''

  if sighting not in kinds:
    kinds.append(sighting)
    ... missing code
  ind = kinds.index(sighting)
  counts[ind] = counts[ind] + 1
```

What code should go in place of the missing code?

A. `counts.append(0)`

B. `counts.append(1)`

C. `counts.append(kind)`

D. No code necessary there

Dictionaries vs. Parallel Lists

```
bird_dict=
{'falcon': 1, 'owl': 5, 'hawk': 2, 'eagle': 11}
```

- Rewrite the new_sighting function
- Compared to parallel lists:
  - Only one dict (not two)
  - No call to index that might search the whole list

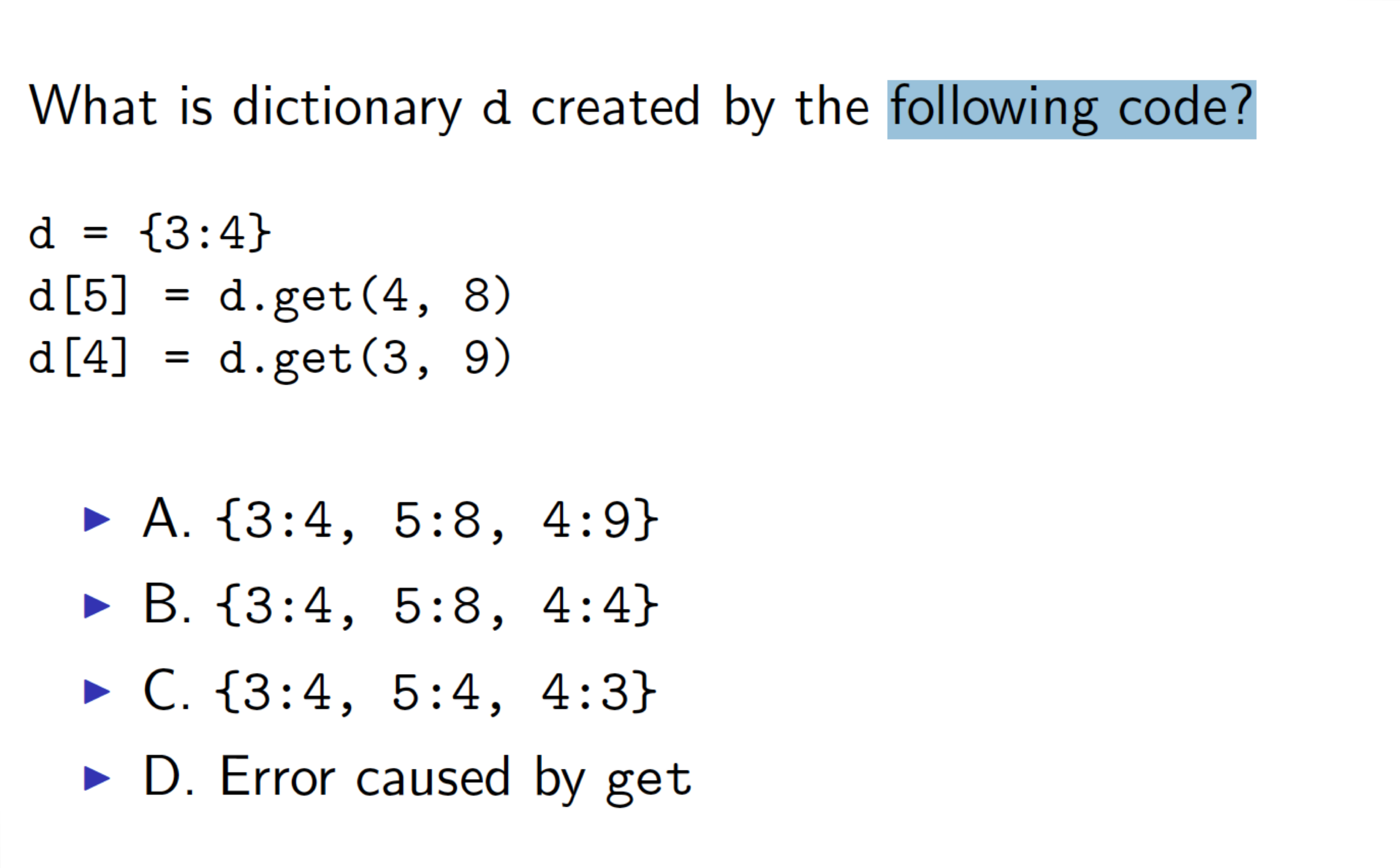
Adding to dictionaries

- Keys must be immutable
- Values can be mutable or immutable
- Use d[k] = v to add key k with value v to dictionary d
    - If k is already present, its value is overwritten
- To copy all key/value pairs from another dictionary, use the `update` method

Getting Values from Dictionaries

- Use d[k] to obtain the value associated with key k of dictionary d
- If k does not exist, this causes an error
- The get method is similar, except it returns None instead of giving an error when the key does not exist
- If a second parameter v is provided, get returns v instead of None when the key is not found

Concept Test

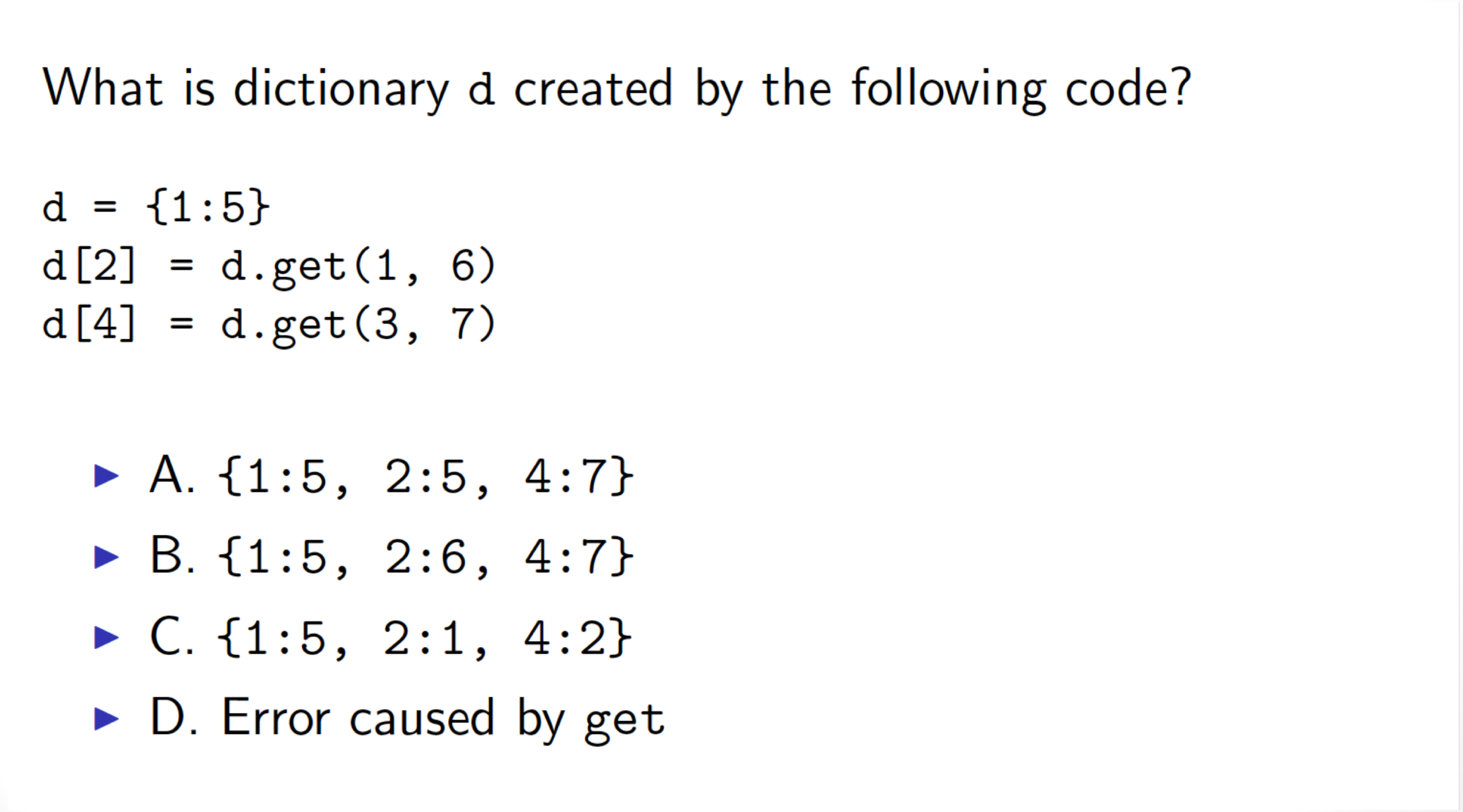


What is dictionary `d` created by the following code?

```
d = {3:4}
d[5] = d.get(4, 8)
d[4] = d.get(3, 9)
```

- A. `{3:4, 5:8, 4:9}`
- B. `{3:4, 5:8, 4:4}`
- C. `{3:4, 5:4, 4:3}`
- D. Error caused by `get`

Concept Test



What is dictionary `d` created by the following code?

```
d = {1:5}
d[2] = d.get(1, 6)
d[4] = d.get(3, 7)
```

- A. `{1:5, 2:5, 4:7}`
- B. `{1:5, 2:6, 4:7}`
- C. `{1:5, 2:1, 4:2}`
- D. Error caused by `get`

More practice

```
def count_occurrences(L):
'''return a dictionary in which the keys are
the elements in L and their associated values
are integers denoting the number of times the
element is contained in L.
>>> count_occurrences([8, 9, 8, 8, 9])
{8:3, 9:2}
'''
```

Python Sets

- Similar to sets in math
- A collection of items with:
    - no duplicates
    - order and position does not matter
- Keep track of unique items (active IDs, SSN, Driver's License)
- Efficient lookup (is something there or not)

Syntax:
{<value1>,<value2>,...,<valuen>}

Python Set Operators & Methods

Assume s1 and s2 are two sets

- Common operators: in, not in
- Union: s1 |s2
- Intersection: s1 & s2
- Difference: s1- s2
- Unique items: s1^s2
- Comparisons: ==, != , <, > , <=, >=

Set methods

- add()
- remove()
- discard()