Dictionaries, Part 2

CMSC 201 Section 60
October 16, 2024

Administrative Notes

Homework 5 due next week - due October 21

Project 1 is out - due November 1

Summarizing what we learned about Python Dictionaries on Monday

Dictionaries are a Python data structure

They are mutable, like lists

They are multi-valued, like lists

They are unordered, UNLIKE lists

When would you use a dictionary instead of a list in your Python program

- Use a list whenever you have large amounts of ordered & related data
  - You need it to be easy to manage and to access
  - What does this mean? You never have to use a list - you can always write your program using:
    - `state1 = “Alabama”`
    - `state2 = “Alaska”`
    - And so on, but that’s definitely not easy to organize or access
- Use a dictionary whenever you have data related by ***key*** and ***value*** pairs

Some examples

Suppose I wanted to store all of the starting quarterbacks of AFC North football teams. I wanted to keep the team, and the QB.

As a list:

```
qbs = [‘Baltimore’, ‘Jackson’, ‘Pittsburgh’,’Fields’, ‘Cincinnati’, ‘Burrough’,
‘Cleveland’, ‘Watson’]
```

But that’s awkward; I have to remember on my own that I alternate team, then player. Easier to use the team as a key and the player as a value

```
qbs = {‘Baltimore’:’Jackson’,
         ‘Pittsburgh’:’Fields’,
         ‘Cincinnati’:’Burrough’,
         ‘Cleveland’:’Watson’
     }
```

Or, I wanted to keep a data structure with the current Governors of each state

List

```
governors = [
     ‘MD’,’Moore’,
     ‘VA’,’Youngkin’,
     ‘PA’,’Shapiro’,
     ‘MN’,’Walz’
}
```

Dictionary

```
governors = {
     ‘MD’:’Moore’,
     ‘VA’:’Youngkin’,
     ‘PA’:’Shapiro’,
     ‘MN’:’Walz’
}
```

Now, how would I find the current Governor of Minnesota using each?

```
for i in range(len(governors)):
     if governors[i] == “MN”:
          print(governors[i+1])
```

```
print(governors[“MN”])
```

Adding a new key/value pair to a dictionary

Use an assignment statement

```
governors[“CA”] = “Newsom”
```

Modifying the value associated with an existing key

This is a little bit tricky - you have to know what type the value is

If the value is an immutable type - int, float, string, bool - just change the value using an assignment statement

```
governors['MD'] = 'Miller"
```

If the value is a mutable type - a list or dictionary - you can perform any legal operation for that type to modify the value

- `append` or `insert` a new value to a list
- `remove` or `pop` a value from a list, or `del` a whole bunch of values
- Reassign an entire new value

Defensive programming - keep your program from crashing

What if you try to access a key/value pair, and the key doesn’t exist?

```
print(governors[‘DC’])
```

The program would crash. So you should use the get method instead

```
print(governors.get(‘DC’))
```

By default, this returns the special value None. But you can make it return some other value if you want

`print(governors.get(‘DC’),-1)` or `print(governors.get(‘DC’),’DC is not a state - sorry’)`

If the key/value pair exists in the dictionary, what you put as the second argument in the get method is ignored

‘Keys’ and ‘values’

Pre-defined methods that operate on dictionaries and return “view objects” containing, respectively, the list of keys in the dictionary and the list of values in the dictionary

```
dogs.keys()
```

```
dogs.values()
```

You can use these methods in your programming - to do some error checking, for example

New Stuff

All that was in the slides for Monday. Now stuff we didn’t get to

Can you have a key with no value?

No, you can't

- But, you CAN have a key with the value None
- Our old friend None, of type NoneType

Removing a key:value pair from the dictionary

Simple but possibly unsafe way: `del`

```
del governors['MN']
```

Works fine as long as there is a key:value pair for Minnesota in the dictionary

- If there's not, your program crashes

Safer but slightly more complex way: `pop`

```
deleted_value =governors.pop('MN')
```

If there is a key:value pair for Minnesota, this removes it from the dictionary and puts the value (***not the key***) in the variable name

Removing key:value pair (continued)

As with get, pop returns None by default if the key:value pair does not exist

- But you can change the return value to whatever you want by putting a second argument in the call

```
deleted_value =governors.pop(‘MN’, -1)
```

or

```
deleted_value =governors.pop(‘MN’, ‘Error - that state does not exist’)
```

Of course you could always code around this yourself:

```
I

if ‘MN’ in governors.keys():

       del governors[‘MN’]

else:

       print(“error - that state does not exist”)
```

Creating a dictionary from two lists

zip - a pre-defined function that takes two lists and pairs the elements in order

- The result is a special type of object called a ‘zip object’

dict() - a pre-defined function that converts pairs of elements into dictionary key:value pairs

An example

```
l1 = ['a','b','c']

l2 = [1,2,3]

print(zip(l1,l2))

d = dict(zip(l1,l2))

print(d)
```