Dictionaries

Thursday, November 12, 2020 10:41 AM

Graded P8.

Quite a few of you are having trouble with the following:

1. Understanding indexing into a list properly. (confusing an index with the contents of the list at that index)
2. Understanding how parameters are passed between functions.
3. How to do a menu driven program - some people forgot to call the menu function at the end of the loop, other people forgot to store the answer the menu function returned in a variable!!!

If you got less than 17 out of 20, then it probably makes sense for you to go back and try to redo the assignments after looking back at the list and function material. Also, try to do a couple practice problems.

A dictionary is like a more flexible version of a list…
It's also, basically just a dictionary!!!

In a list, we have

3	2	4	1	5	5	2	
0	1	2	3	4	5	6	index

In this example, apscores[4] = 5
So if I want to access a data point, I access it through its index, which is always an integer from 0 to len(apscores)-1. (Or negative for negative indexing.)

So imagine the array above was really storing students' AP test scores…at let's say Shania is student #4 and she got a 5!

Wouldn't it be nicer if we could do:

apscores["Shania"] instead of apscores[4]?

Alternatively

Bob	Evan	Lydia	Sharon	Shania	Liz	Anya

I could do two lists, but this seems annoying.

A dictionary is basically a list, where you get to choose the indexes!!!

In a regular dictionary we have what we call keys (these are the words that are being defined), and each key maps to a value (the definition).

In a real dictionary, we have multiple definitions for one word, but in a Python dictionary, each "word" is allowed only 1 definition.

Also, like a real dictionary, two different words can be mapped to the same definition, there is no problem with that.

Example:

In a reg dictionary we have

Key	Value
cat	animal kept as pet
apple	fruit with seeds
run	action where you move your feet fast

In python, you can choose what to store in a dictionary

our first example apscores:

Key	Value
Bob	3
Evan	2
Lydia	4
Sharon	1
Shania	5
Liz	5
Anya	2

Other examples:

phone numbers

Key	Value
Alice	4072223333
Brenda	3216678888
Cam	3052121111

Age

Key	Value
Doug	33
Ellie	19
Frank	79

Votes for school president

Key	Value

Items in stock in the grocery store

Key	Value
apple	10
bananas	40

age = {} # Defines a empty dictionary called age.

Here is the IDLE session:

```
>>> age = {}
>>> age["Arup"] = 45
>>> age["Anya"] = 7
>>> age["Simi"] = 16
>>> age["Anita"] = 45
>>> age
{'Arup': 45, 'Anya': 7, 'Simi': 16, 'Anita': 45}
>>> name = input("who do you want to ask about?\n")
who do you want to ask about?
Simi
>>> print(name,"is",age[name],"years old")
```

```
Simi is 16 years old
>>> name = input("Who's birthday is next?\n")
Who's birthday is next?
Anya
>>> age[name] = age[name] + 1
>>> print("Now",name,"is",age[name],"years old")
Now Anya is 8 years old
>>> name = input("Whose age do you want?\n")
Whose age do you want?
Nalini
>>> print(name,"is",age[name],"years old.")
Traceback (most recent call last):
  File "<pyshell#12>", line 1, in <module>
    print(name,"is",age[name],"years old.")
KeyError: 'Nalini'
>>>
```

We add to a dictionary by doing

age[key] = value

if we try to access a key that doesn't exist, we get a KeyError and our program crashes (see above).

How do I avoid a key error?

Answer: use the in operator to check if something is in the dictionary first!

We use the in operator to check if a key is in the dictionary or not:

```
>>> if name in age:
      print(name,"is",age[name],"years old.")
else:
      print("Sorry I have no idea how old",name,"is")
```

Sorry I have no idea how old Nalini is

Here is how we find the # of entries in a dictionary

```
>>> len(age)
4
```

Here is how we loop through each key in a dictionary

```
>>> for x in age:
      print(x,age[x])

Arup 45
Anya 8
Simi 16
Anita 45
```

To remove an entry, just do

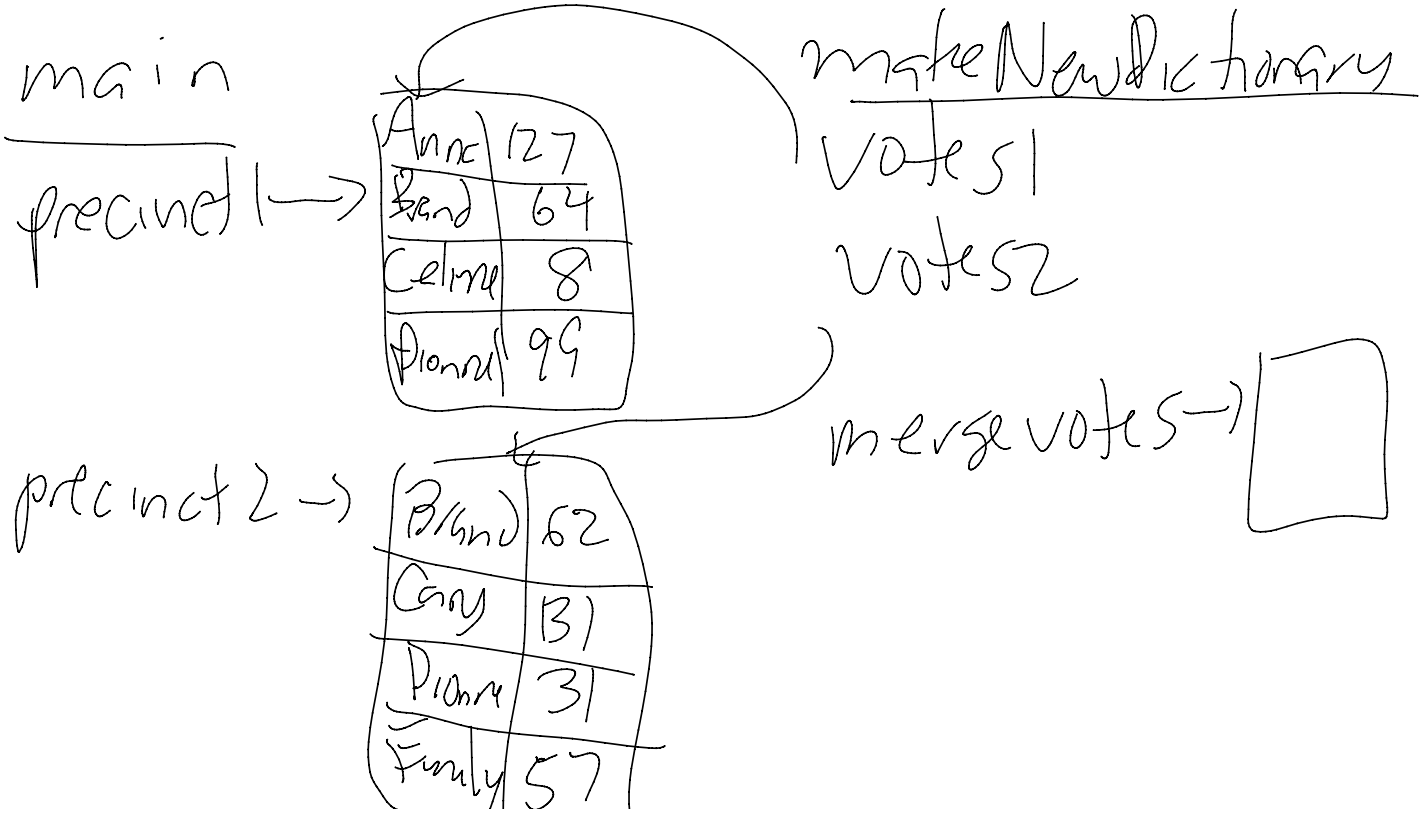
del age["Arup"]

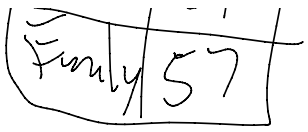
```
>>> del age["Arup"]
>>> age
{'Anya': 8, 'Simi': 16, 'Anita': 45}
```

My new function - combining votes
---

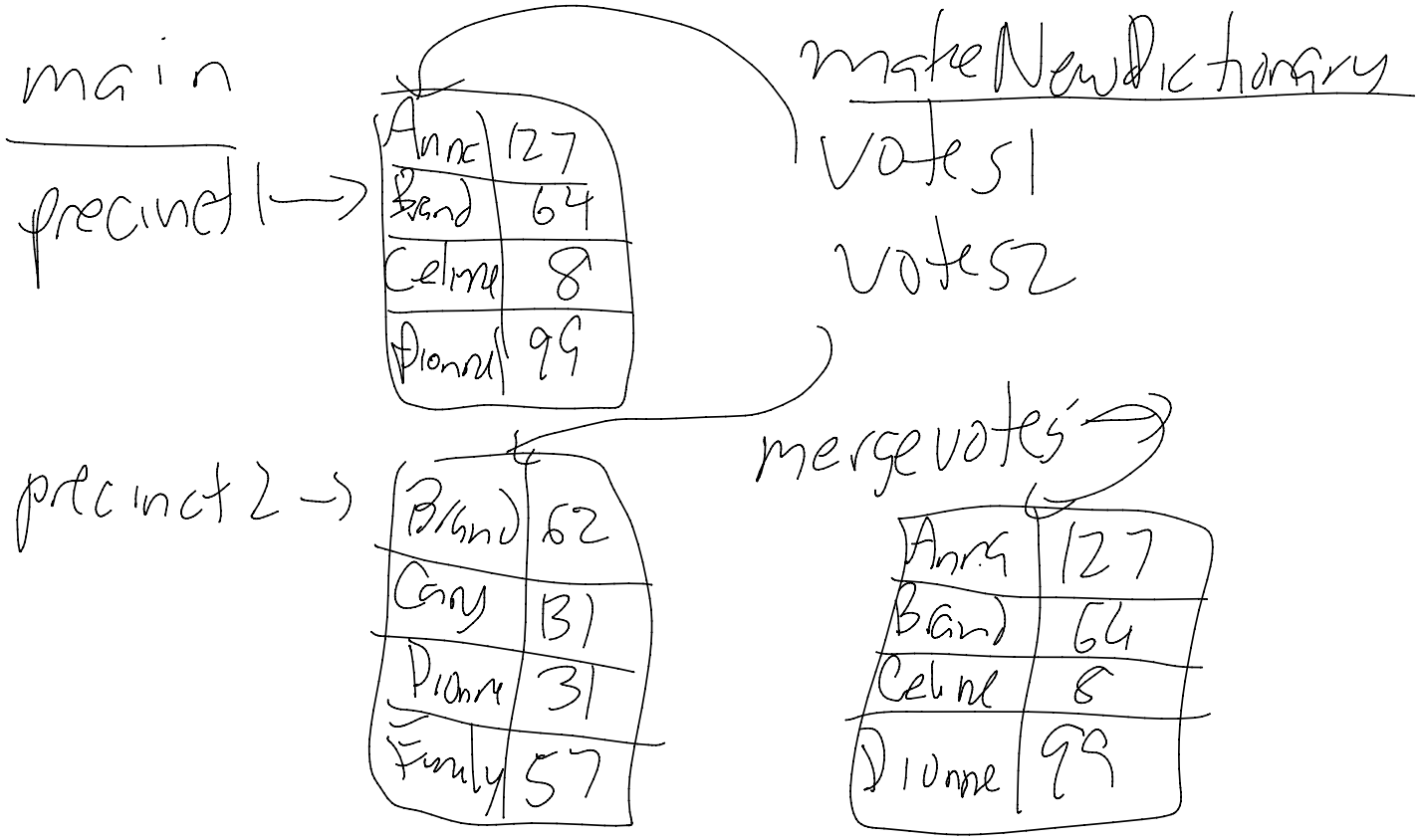
votes1 is a dictionary of # of votes each person has
votes2 is also.
Think of two precincts reporting.
We will create new dictionary that merges both of these.

Picture of votemerge function

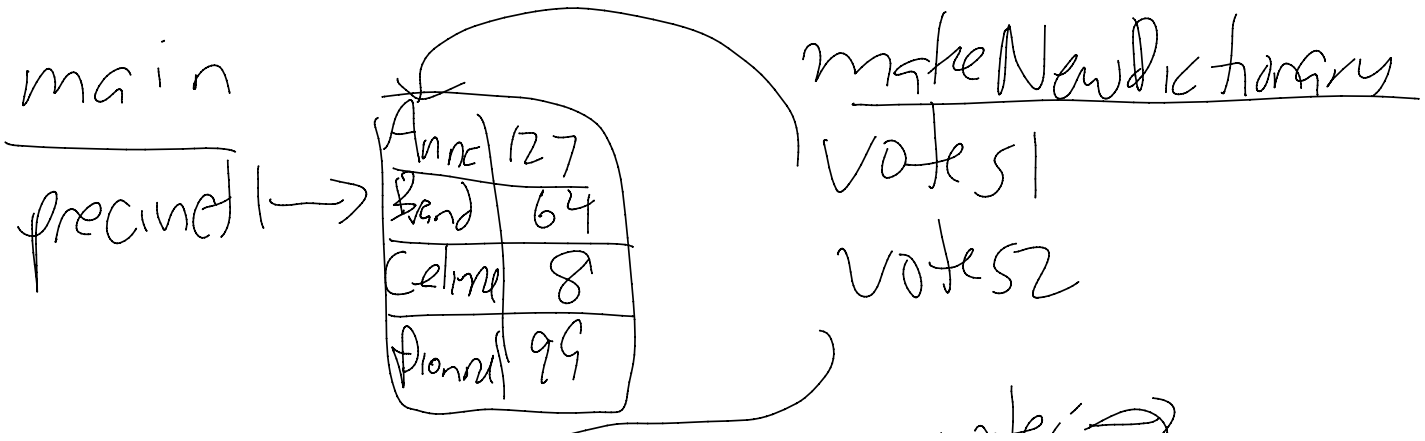


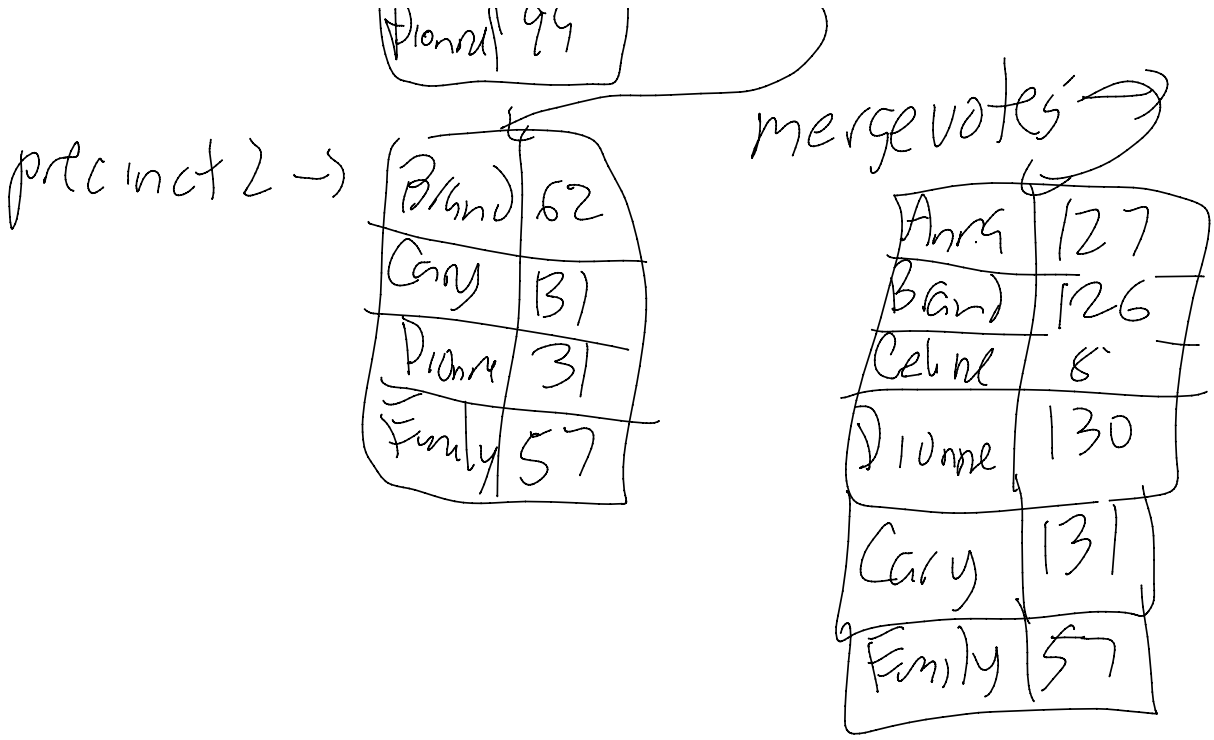


Picture after for loop



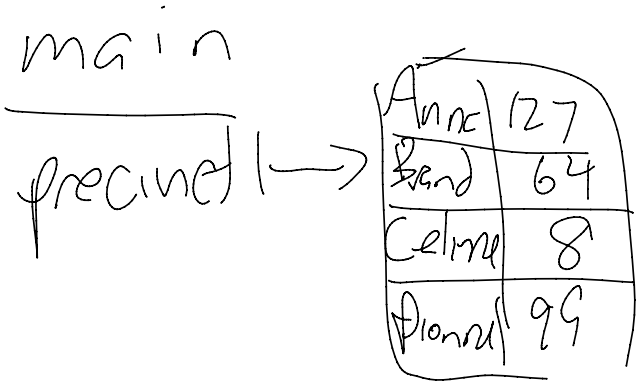
Now, when the second for loop runs. for Brandon and Dionne, we see the keys are already in the dictionary, so no new key is made and the votes are just added:

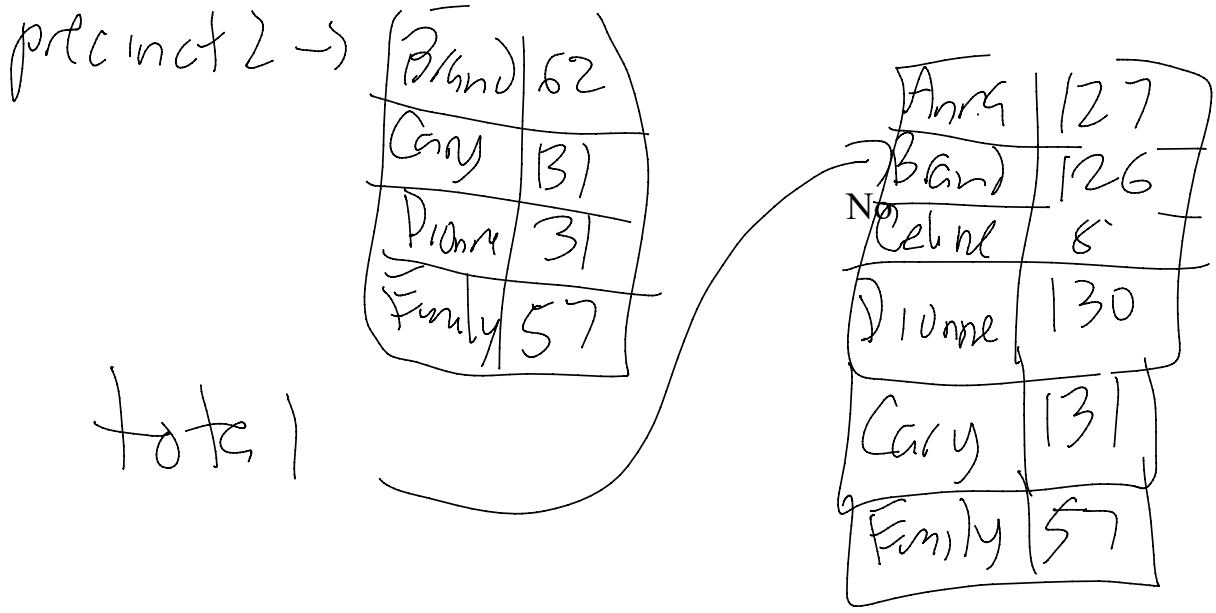




Finally, the makeNewDictionary function returns a pointer to the newly created dictionary, mergevotes and stores this in a variable called total in main:

Now, when the second for loop runs. for Brandon and Dionne, we see the keys are already in the dictionary, so no new key is made and the votes are just added:





Notice that when the function returns, the memory for it (the variables that were pointers) are now gone, but the dictionary persists and total points to it.