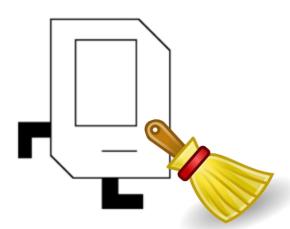


Housekeeping



- Midterm will be on Tuesday, May 3rd from 7pm-9pm
 - Location by first letter of last name:
 - Last name: A-H in room: 420-040
 - Last name: I-Z in room: CEMEX Aud
 - Midterm covers material through class on April 27
 - Open book/notes, but will need to bring <u>printed</u> reference material
- Will be using BlueBook software to take the exam
 - BlueBook will not allow you to access other applications/files on your computer during the exam



Review of String

```
PUNCTUATION = '.!?,-:;'
def delete punctuation(s):
    ** ** **
    Removes punctuation characters from a string and
    returns the resulting string.
    result = ''
    for char in s:
        # Check char is not a punctuation mark
        if char not in PUNCTUATION:
            result += char # append non-punctuation chars
    return result
Calling: delete punctuation('REMOVE -the- punctuation!!!')
Returns: 'REMOVE the punctuation'
```

Reading Lines from a File

```
def count words(filename):
  count = 0
  with open(filename, 'r') as file: # Open file to read
      for line in file:
        line = line.strip() # Remove newline
        word list = line.split() # Create list of words
        for word in word list: # Print words
           print("#" + str(count) + ": " + word)
           count += 1
  print(filename + " contains " + str(count) + " words")
testfile.txt
Very few
words here.
```

Console:

```
#0: Very
#1: few
#2: words
#3: here.
testfile.txt contains 4 words
```

Learning Goals

- 1. Learning about dictionaries
- 2. Building programs using dictionaries



Dictionaries

submission; one's authority: ere which is out of my writ and compewrite-off > noun 1 Brit. a vehicle or other obje is too badly damaged to be repaired. ody of writing. a worthless or ineffectual person or thing: the m as a general term denoting writzine was a write-off, its credibility rating below zero Germanic base of WRITE. 2 Finance a cancellation from an account of a bad or worthless asset. t participle of write. write-once > adjective Computing denoting a mem ar and obvious: the unspoken or storage device, typically an optical one, on wh on Rose's face. In a stark or data, once written, cannot be modified people by way of tax allowwrite-protect ▶ verb [with obj.] Computing protect writ large. disk) from accidental writing or erasure. participle written) [with obj.] writer noun 1 a person who has written somethin her symbols) on a suror who writes in a particular way: the writer of the en, pencil, or similar he paper | Alice wrote a person who writes books, stories, or articles as a job or occupation: Dickens was a prolific writer | a writer ery neatly in blue ink. of short stories. erent letters or 2 Computing a device that writes data to a storage e. a filloutor medium: a CD writer. this way: he 3 historical a scribe African take archaic a clerk, especially in the navy or in govern e English ment offices. -PHRASES Writer's block the condit unable to think of what to with writing was Band

What are Dictionaries?

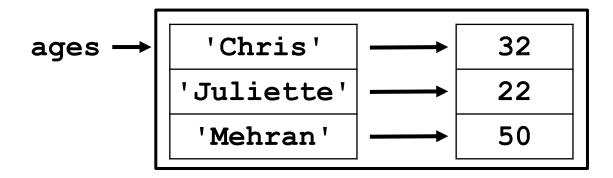
- Dictionaries associate a <u>key</u> with a <u>value</u>
 - Key is a unique identifier
 - Value is something we associate with that key
- Examples in the real world:
 - Phonebook
 - Keys: names
 - Values: phone numbers
 - Dictionary
 - Keys: words
 - Values: word definitions
 - US Government
 - Keys: Social Security number
 - Values: Information about an individual's employment



Dictionaries in Python

- Creating dictionaries
 - Dictionary start/end with braces
 - Key:Value pairs separated by colon
 - Each pair is separated by a comma

```
ages = {'Chris': 32, 'Juliette': 22, 'Mehran': 50}
squares = {2: 4, 3: 9, 4: 16, 5: 25}
phone = {'Pat': '555-1212', 'Jenny': '867-5309'}
empty_dict = {}
```

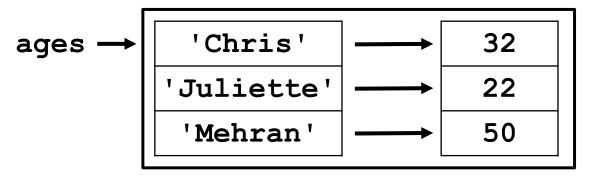




Consider the following dictionary:

```
ages = {'Chris': 32, 'Juliette': 22, 'Mehran': 50}
```

Like a set of variables that are indexed by <u>keys</u>



Use <u>key</u> to access associated <u>value</u>:

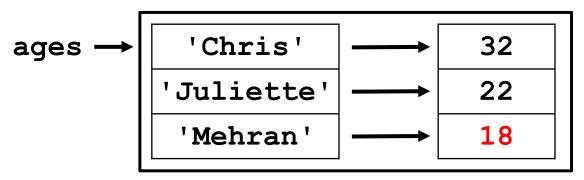
```
ages['Chris'] is 32
ages['Mehran'] is 50
```



Consider the following dictionary:

```
ages = {'Chris': 32, 'Juliette': 22, 'Mehran': 50}
```

Like a set of variables that are indexed by <u>keys</u>



• Use key to access associated value:

```
ages['Chris'] is 32
ages['Mehran'] is 50
```

Can set <u>values</u> like regular variable:

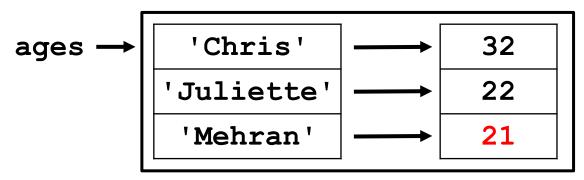
```
ages['Mehran'] = 18
```



Consider the following dictionary:

```
ages = {'Chris': 32, 'Juliette': 22, 'Mehran': 50}
```

Like a set of variables that are indexed by keys



• Use key to access associated value:

```
ages['Chris'] is 32
ages['Mehran'] is 50
```

Can set <u>values</u> like regular variable:

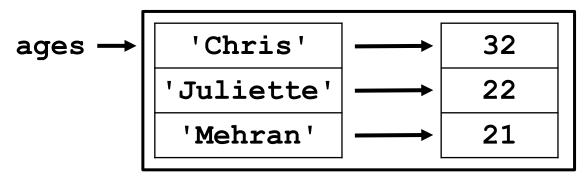
```
ages['Mehran'] = 18
ages['Mehran'] += 3
```



Consider the following dictionary:

```
ages = {'Chris': 32, 'Juliette': 22, 'Mehran': 50}
```

Like a set of variables that are indexed by <u>keys</u>



Good and bad times with accessing pairs:

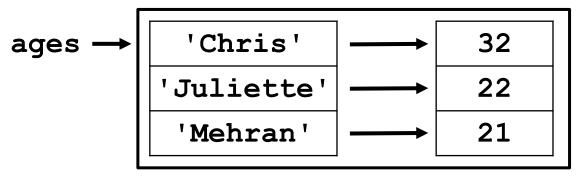
```
>>> juliettes_age = ages['Juliette']
>>> juliettes_age
22
>>> santas_age = ages['Santa Claus']
KeyError: 'Santa Claus'
```



Consider the following dictionary:

```
ages = {'Chris': 32, 'Juliette': 22, 'Mehran': 50}
```

Like a set of variables that are indexed by <u>keys</u>



Checking membership

```
>>> 'Juliette' in ages
True
>>> 'Santa Claus' not in ages
True
```



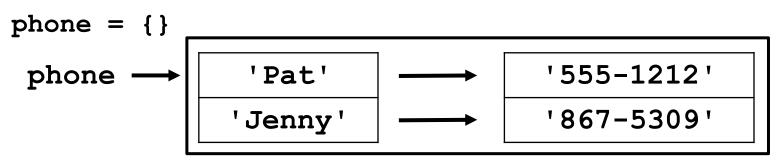
```
phone = {}

phone \rightarrow Empty dictionary
```



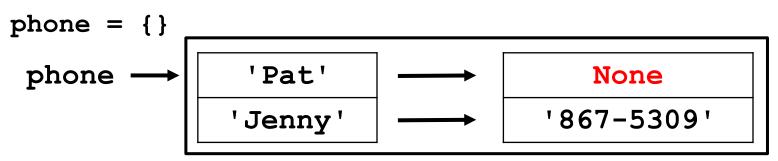
```
phone['Pat'] = '555-1212'
```





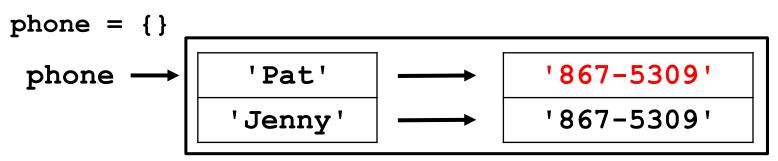
```
phone['Pat'] = '555-1212'
phone['Jenny'] = '867-5309'
```





```
phone['Pat'] = '555-1212'
phone['Jenny'] = '867-5309'
phone['Pat'] = None
```





```
phone['Pat'] = '555-1212'
phone['Jenny'] = '867-5309'
phone['Pat'] = None
phone['Pat'] = '867-5309'
```



A Word About Keys/Values

- Keys must be <u>immutable</u> types
 - E.g., int, float, string
 - Keys <u>cannot</u> be changed in place
 - If you want to change a key, need to remove key/value pair from dictionary and then add key/value pair with new key.
- Values can be <u>mutable</u> or <u>immutable</u> types
 - E.g., int, float, string, <u>lists</u>, <u>dictionaries</u>
 - Values <u>can</u> be changed in place
- Dictionaries are <u>mutable</u>
 - Changes made to a dictionary in a function persist after the function is done.

Changing List in a Function

```
def have_birthday(dict, name):
    print("You're one year older, " + name + "!")
    dict[name] += 1

def main():
    ages = {'Chris': 32, 'Juliette': 22, 'Mehran': 50}
    print(ages)
    have_birthday(ages, 'Chris')
    print(ages)
    have_birthday(ages, 'Mehran')
    print(ages)
```

Terminal:

```
{'Chris': 32, 'Juliette': 22, 'Mehran': 50}
You're one year older, Chris!
{'Chris': 33, 'Juliette': 22, 'Mehran': 50}
You're one year older, Mehran!
{'Chris': 33, 'Juliette': 22, 'Mehran': 51}
```

Dicta-palooza! (Part 1)

```
ages = {'Chris': 32, 'Juliette': 22, 'Mehran': 50}
```

- Function: <u>dict</u>.get(key)
 - Returns value associated with key in dictionary. Returns None if key doesn't exist.

```
>>> print(ages.get('Chris'))
32
>>> print(ages.get('Santa Claus'))
None
```

- Function: <u>dict</u>.get(key, default)
 - Returns value associated with key in dictionary. Returns <u>default</u> if key doesn't exist.

```
>>> print(ages.get('Chris', 100))
32
>>> print(ages.get('Santa Claus', 100))
100
```



Dicta-palooza! (Part 2)

```
ages = {'Chris': 32, 'Juliette': 22, 'Mehran': 50}
```

- Function: <u>dict</u>.keys()
 - Returns something similar to a range of the <u>keys</u> in dictionary
 - Can use that to loop over all keys in a dictionary:

```
for key in ages.keys():
    print(str(key) + " -> " + str(ages[key]))
```

Terminal:

```
Chris -> 32
Juliette -> 22
Mehran -> 50
```

Can turn keys() into a list, using the list function

```
>>> list(ages.keys())
['Chris', 'Juliette', 'Mehran']
```



Dicta-palooza! (Part 3)

```
ages = {'Chris': 32, 'Juliette': 22, 'Mehran': 50}
```

 Can also loop over a dictionary using for-each loop just using name of dictionary:

```
for key in ages:
    print(str(key) + " -> " + str(ages[key]))
```

Terminal:

```
Chris -> 32
Juliette -> 22
Mehran -> 50
```



Dicta-palooza! (Part 4)

```
ages = {'Chris': 32, 'Juliette': 22, 'Mehran': 50}
```

- Function: <u>dict</u>.values()
 - Returns something similar to a range of the <u>values</u> in dictionary
 - Can use that to loop over all keys in a dictionary:

```
for value in ages.values():
    print(value)
```

Terminal:

```
32
22
50
```

- Can turn values() into a list, using the list function
>>> list(ages.values())
[32, 22, 50]



Dicta-palooza! (Part 5)

```
ages = {'Chris': 32, 'Juliette': 22, 'Mehran': 50}
Function: <u>dict</u>.pop (key)

    Removes key/value pair with the given key. Returns value from

     that key/value pair.
   >>> ages
   >>> {'Chris': 32, 'Juliette': 22, 'Mehran': 50}
   >>> ages.pop('Mehran')
   50
   >>> ages
   {'Chris': 32, 'Juliette': 22}
Function: <u>dict</u>.clear()

    Removes all key/value pairs in the dictionary.

   >>> ages.clear()
   >>> ages
```

{ }

Functions You Can Apply

```
ages = {'Chris': 32, 'Juliette': 22, 'Mehran': 50}
• Function: len(<u>dict</u>)

    Returns number of key/value pairs in the dictionary

   >>> ages
   { 'Chris': 32, 'Juliette': 22, 'Mehran': 50}
   >>> len(ages)
• Function: del <u>dict</u>[key]

    Removes key/value pairs in the dictionary.

    Similar to pop, but doesn't return anything.

   >>> ages
   { 'Chris': 32, 'Juliette': 22, 'Mehran': 50}
   >>> del ages['Mehran']
   >>> ages
   {'Chris': 32, 'Juliette': 22}
```

Putting it all together: count_each_word.py

(And we'll also throw in files as a bonus concept!)

Bonus fun: phonebook.py

Learning Goals

- 1. Learning about dictionaries
- 2. Building programs using dictionaries



{ 'breakfast': ,
 'lunch': ,
 'dinner': }

