

Chris Gregg
Based on slides by Chris Piech and Mehran Sahami
CS106A, Stanford University

Review of String

```
PUNCTUATION = '.!?,-:;'
def delete punctuation(s):
    11 11 11
    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
Console:
>>> delete punctuation('REMOVE --the-- punctuation!!!!')
```

'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[:-1]
                                      # Remove newline
        word list = line.split()
                                      # Create list of words
         for word in word list: # Print words
            print(f"#{count}: {word}")
            count += 1
  print(f"{filename} contains {count} words")
testfile.txt
Very few
```

Console:

words here.

```
#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: care cast by writin 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 allow-Write-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?

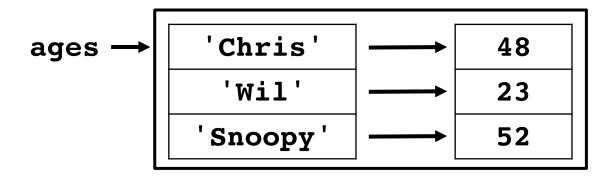
- Dictionaries associate a <u>key</u> with a <u>value</u>
 - Key is a unique identifier
 - Value is something we associate with that key
- Examples in the real world:
 - Phonebook
 - Keys: names
 - Values: phone numbers
 - Dictionary
 - Keys: words
 - Values: word definitions
 - US Government
 - Keys: 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': 48, 'Wil': 23, 'Snoopy': 52}
squares = {2: 4, 3: 9, 4: 16, 5: 25}
phone = {'Pat': '555-1212', 'Jenny': '867-5309'}
empty_dict = {}
```

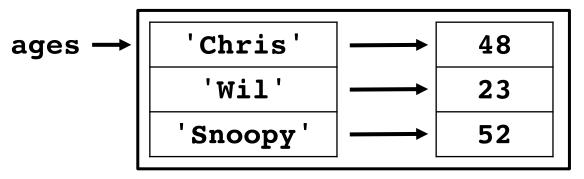




Consider the following dictionary:

```
ages = {'Chris': 48, 'Wil': 23, 'Snoopy': 52}
```

Like a set of variables that are indexed by keys



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

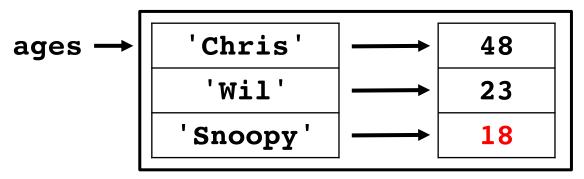
```
ages['Chris'] is 48
ages['Snoopy'] is 52
```



Consider the following dictionary:

```
ages = {'Chris': 48, 'Wil': 23, 'Snoopy': 52}
```

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



• Use key to access associated value:

```
ages['Chris'] is 48
ages['Snoopy'] is 52
```

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

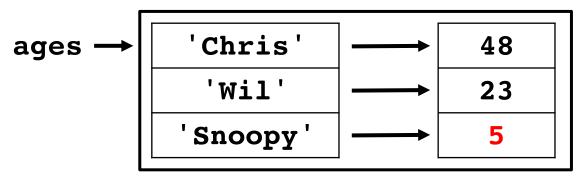
```
ages['Snoopy'] = 18
```



Consider the following dictionary:

```
ages = {'Chris': 48, 'Wil': 23, 'Snoopy': 52}
```

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



• Use key to access associated value:

```
ages['Chris'] is 48
ages['Snoopy'] is 52
```

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

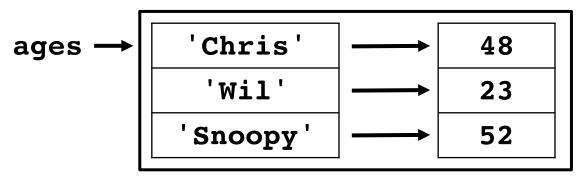
```
ages['Snoopy'] = 18
ages['Snoopy'] -= 13
```



Consider the following dictionary:

```
ages = {'Chris': 48, 'Wil': 23, 'Snoopy': 52}
```

Like a set of variables that are indexed by keys



Good and bad times with accessing pairs:

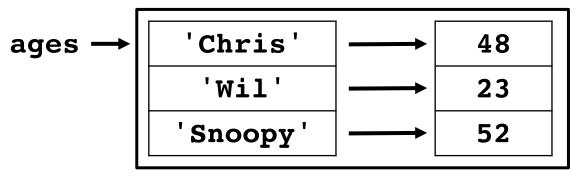
```
>>> wils_age = ages['Wil']
>>> wils_age
23
>>> santas_age = ages['Santa Claus']
KeyError: 'Santa Claus'
```



Consider the following dictionary:

```
ages = {'Chris': 48, 'Wil': 23, 'Snoopy': 52}
```

Like a set of variables that are indexed by keys



Checking membership

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



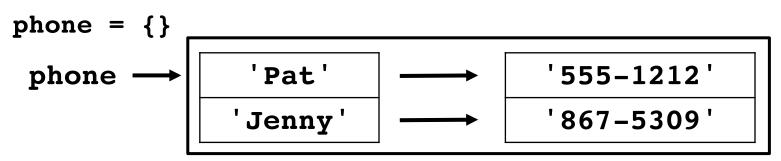
```
phone = {}

phone \leftarrow Empty dictionary
```



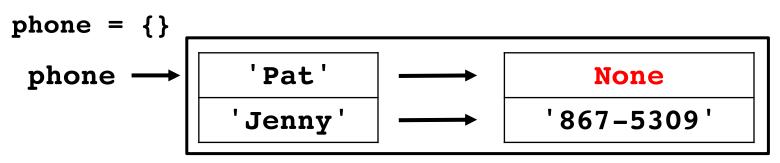
```
phone['Pat'] = '555-1212'
```





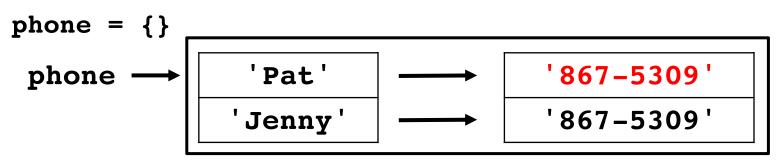
```
phone['Pat'] = '555-1212'
phone['Jenny'] = '867-5309'
```





```
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 can 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(f"You're one year older, {name}!")
    dict[name] += 1

def main():
    ages = {'Chris': 48, 'Wil': 23, 'Snoopy': 52}
    print(ages)
    have_birthday(ages, 'Chris')
    print(ages)
    have_birthday(ages, 'Snoopy')
    print(ages)
```

Terminal:

```
{'Chris': 48, 'Wil': 23, 'Snoopy': 52}
You're one year older, Chris!
{'Chris': 49, 'Wil': 23, 'Snoopy': 52}
You're one year older, Snoopy!
{'Chris': 49, 'Wil': 23, 'Snoopy': 53}
```

Dictiona-palooza! (Part 1)

```
ages = {'Chris': 48, 'Wil': 23, 'Snoopy': 52}
```

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

```
>>> print(ages.get('Chris'))
48
>>> 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))
48
>>> print(ages.get('Santa Claus', 100))
100
```



Dictiona-palooza! (Part 2)

```
ages = {'Chris': 48, 'Wil': 23, 'Snoopy': 52}
```

- 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(f"{key}, {ages[key]}")
```

Terminal:

```
Chris, 48
Wil, 23
Snoopy, 52
```

- Can turn keys() into a list, using the list function
>>> list(ages.keys())
['Chris', 'Wil', 'Snoopy']



Dictiona-palooza! (Part 3)

```
ages = {'Chris': 48, 'Wil': 23, 'Snoopy': 52}
```

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

```
for key in ages:
    print(f"{key}, {ages[key]}")
```

Terminal:

```
Chris, 48
Wil, 23
Snoopy, 52
```



Dictiona-palooza! (Part 4)

```
ages = {'Chris': 48, 'Wil': 23, 'Snoopy': 52}
```

- 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:

```
48
23
52
```

- Can turn values() into a list, using the list function
>>> list(ages.values())
[48, 23, 52]



Dictiona-palooza! (Part 5)

```
ages = {'Chris': 48, 'Wil': 23, 'Snoopy': 52}
```

- Function: <u>dict</u>.pop(key)
 - Removes key/value pair with the given key. Returns value from that key/value pair.

```
>>> ages
>>> {'Chris': 48, 'Wil': 23, 'Snoopy': 52}
>>> ages.pop('Snoopy')
52
>>> ages
{'Chris': 48, 'Wil': 23}
```

- Function: <u>dict</u>.clear()
 - Removes all key/value pairs in the dictionary.

```
>>> ages.clear()
>>> ages
{}
```



Functions You Can Apply

```
ages = {'Chris': 48, 'Wil': 23, 'Snoopy': 52}
• Function: len(<u>dict</u>)

    Returns number of key/value pairs in the dictionary

   >>> ages
   {'Chris': 48, 'Wil': 23, 'Snoopy': 52}
   >>> len(ages)
   3
• Function: del <u>dict[key]</u>

    Removes key/value pairs in the dictionary.

    Similar to pop, but doesn't return anything.

   >>> ages
   {'Chris': 48, 'Wil': 23, 'Snoopy': 52}
   >>> del ages['Snoopy']
   >>> ages
   {'Chris': 48, 'Wil': 23}
```

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': }

