

# **Learning Goals**







## Dictionaries

submission; one's authority: rare 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?

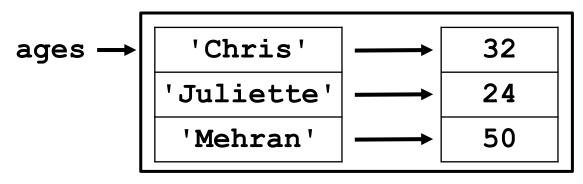
- Dictionaries associate a <u>key</u> with a <u>value</u>
  - Key is a unique identifier
  - Value is something we associate with that key
- Examples in the real world:
  - Phonebook
    - Keys: names
    - Values: phone numbers
  - Dictionary
    - Keys: words
    - Values: word definitions
  - Social Media Platforms
    - Keys: usernames
    - Values: Information like account details, posts, and followers



## 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': 24, 'Mehran': 50}
squares = {2: 4, 3: 9, 4: 16, 5: 25}
phone = {'Pat': '555-1212', 'Jenny': '867-5309'}
empty_dict = {}
```



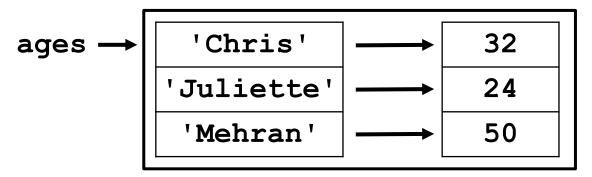


Piech + Sahami, CS106A, Stanford University

Consider the following dictionary:

```
ages = {'Chris': 32, 'Juliette': 24, 'Mehran': 50}
```

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



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

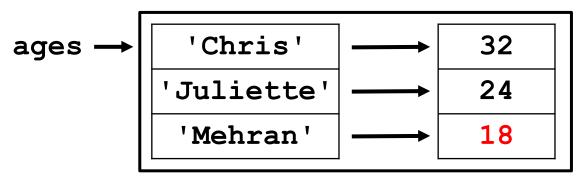
```
ages['Chris'] is 32
ages['Mehran'] is 50
```



Consider the following dictionary:

```
ages = {'Chris': 32, 'Juliette': 24, '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:

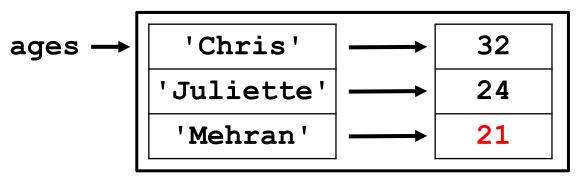
```
ages['Mehran'] = 18
```



Consider the following dictionary:

```
ages = {'Chris': 32, 'Juliette': 24, '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
```

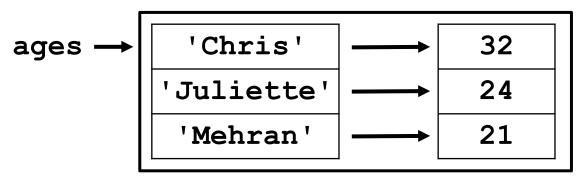
Can set <u>values</u> like regular variable:



Consider the following dictionary:

```
ages = {'Chris': 32, 'Juliette': 24, 'Mehran': 50}
```

Like a set of variables that are indexed by keys



Good and bad times with accessing pairs:

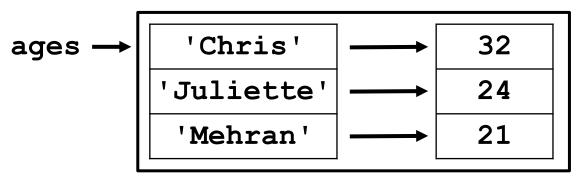
```
>>> chris_age = ages['Chris']
>>> chris_age
32
>>> karels_age = ages['Karel the Robot']
KeyError: 'Karel the Robot'
```



Consider the following dictionary:

```
ages = {'Chris': 32, 'Juliette': 24, 'Mehran': 50}
```

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



Checking membership

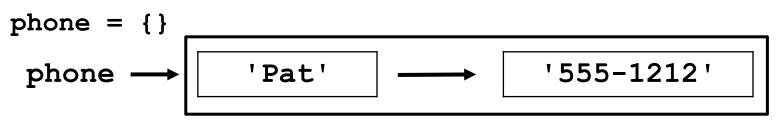
```
>>> 'Mehran' in ages
True
>>> 'Karel the Robot' not in ages
True
```



```
phone = {}

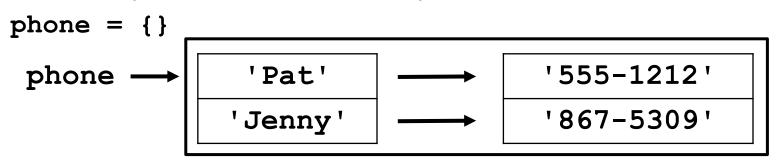
phone \rightarrow Empty dictionary
```





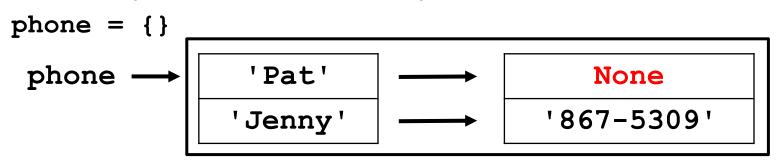
```
phone['Pat'] = '555-1212'
```





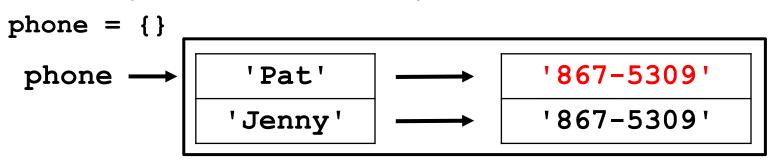
```
phone['Pat'] = '555-1212'
phone['Jenny'] = '867-5309'
```





```
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'
```



## **Looping Over Dictionary**

```
ages = {'Chris': 32, 'Juliette': 24, 'Mehran': 50}
```

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

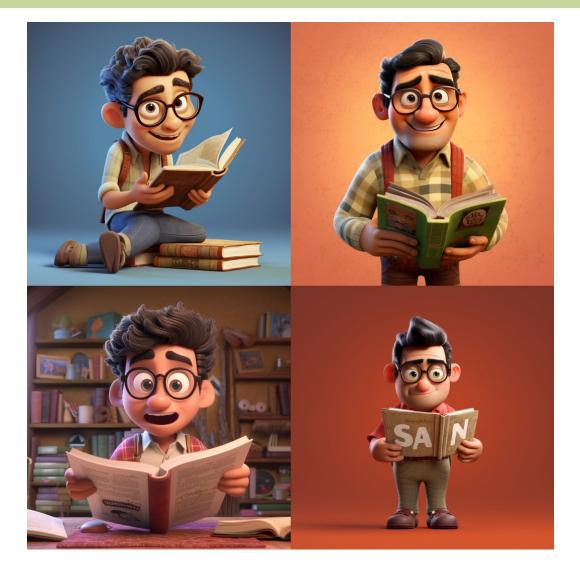
```
for key in ages:
   value = ages[key]
   print(str(key) + ' -> ' + str(value))
```

#### Terminal:

```
Chris -> 32
Juliette -> 24
Mehran -> 50
```



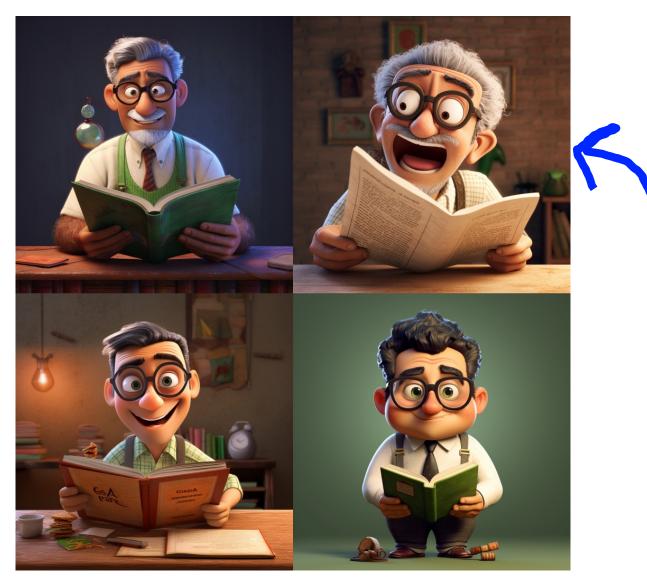
## **Lets Play!**



"Mehran Sahami, as a pixar character, reading a dictionary"



## **Lets Play!**



"Professor Mehran Sahami, as a pixar character, reading a dictionary" Piech + Sahami, CS106A, Stanford University

List index -> value

Dictionary key -> value

## **Dictionary**

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

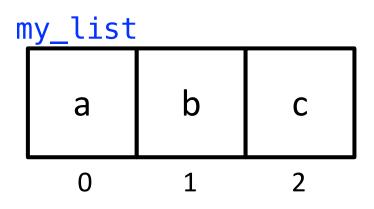
print(my_list[1])

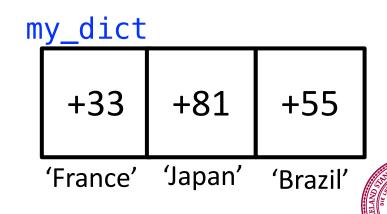
for i in range(len(my_list)):
   value = my_list[i]
   print(i, value)
```

```
my_dict = {
    'France': '+33',
    'Japan': '+81',
    'Brazil':'+55'
]

print(my_list['France'])

for key in my_dict:
    value = my_dict[key]
    print(key, value)
```





indices

## **Dictionary**

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

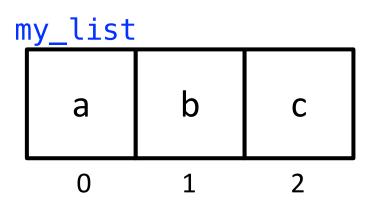
print(my_list[1])

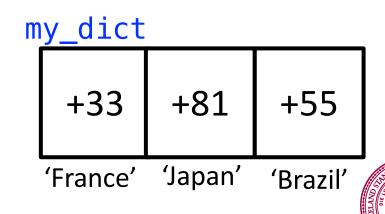
for i in range(len(my_list)):
    value = my_list[i]
    print(i, value)
```

```
my_dict = {
    'France': '+33',
    'Japan': '+81',
    'Brazil':'+55'
]

print(my_list['France'])

for key in my_dict:
    value = my_dict[key]
    print(key, value)
```





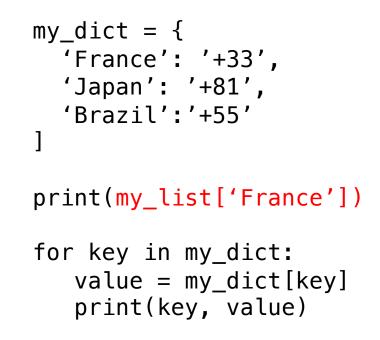
indices

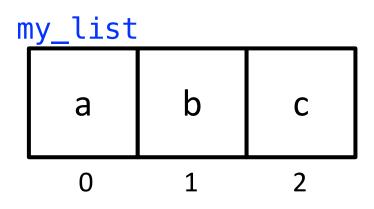
## **Dictionary**

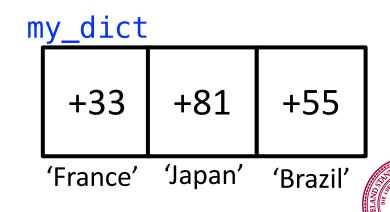
```
my_list = ['a', 'b', 'c']

print(my_list[1])

for i in range(len(my_list)):
    value = my_list[i]
    print(i, value)
```







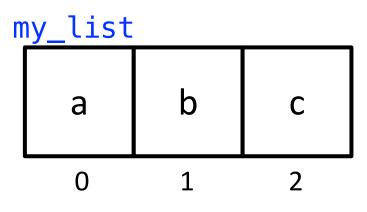
indices

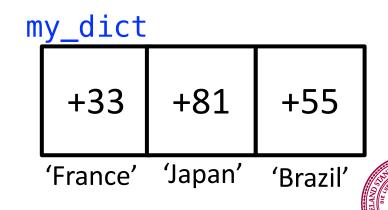
my\_list = ['a', 'b', 'c']

## Dictionary

```
print(my_list[1])

for i in range(len(my_list)):
   value = my_list[i]
   print(i, value)
```





indices

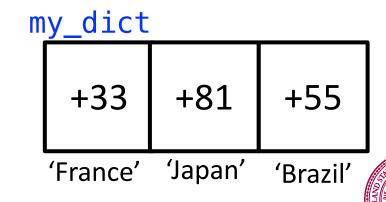
# my\_list = ['a', 'b', 'c']

```
print(my_list[1])

for i in range(len(my_list)):
    value = my_list[i]
    print(i, value)
```

## 

## **Dictionary**



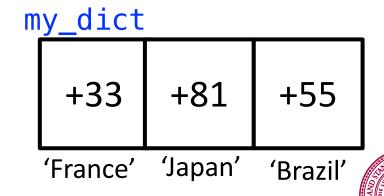
indices

my\_list = ['a', 'b', 'c']

```
print(my_list[1])

for i in range(len(my_list)):
    value = my_list[i]
    print(i, value)
```

# **Dictionary**



indices

## More Dictionary Fun! (Part 1)

```
ages = {'Chris': 32, 'Juliette': 24, '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 -> 24
Mehran -> 50
```

- Can turn keys() into a list, using the list function
>>> list(ages.keys())

```
['Chris', 'Juliette', 'Mehran']
```



## More Dictionary Fun! (Part 2)

```
ages = {'Chris': 32, 'Juliette': 24, '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
24
50
```

- Can turn values() into a list, using the list function
>>> list(ages.values())
[32, 24, 50]



## More Dictionary Fun! (Part 3)

```
ages = {'Chris': 32, 'Juliette': 24, 'Mehran': 50}
Function: dict.items()
   Returns

    Can use that to loop over all key value pairs in a dictionary:

      for key, value in ages.items():
          print(f"{key}, {value}")
      Terminal:
          Chris, 32
          Juliette, 24
```

Mehran, 50



## More Dictionary Fun! (Part 4/5)

```
ages = {'Chris': 32, 'Juliette': 24, 'Mehran': 50}
• Function: dict.pop(key)
```

 Removes key/value pair with the given key. Returns <u>value</u> from that key/value pair.

```
>>> ages
>>> {'Chris': 32, 'Juliette': 24, 'Mehran': 50}
>>> ages.pop('Mehran')
50
>>> ages
{'Chris': 32, 'Juliette': 24}
```

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

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



## Functions You Can Apply

```
ages = {'Chris': 32, 'Juliette': 24, 'Mehran': 50}
• Function: len(<u>dict</u>)

    Returns number of key/value pairs in the dictionary

   >>> ages
   {'Chris': 32, 'Juliette': 24, 'Mehran': 50}
   >>> len(ages)
   3
Function: del dict[key]

    Removes key/value pairs in the dictionary.

    Similar to pop, but doesn't return anything.

   >>> ages
   {'Chris': 32, 'Juliette': 24, 'Mehran': 50}
   >>> del ages['Mehran']
   >>> ages
   {'Chris': 32, 'Juliette': 24}
```

Piech + Sahami, CS106A, Stanford University

Example:
count\_nums.py

Example: phonebook.py

# **Learning Goals**

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

