CS 1112: Introduction To Programming

More on Python Dictionaries

Dr. Nada Basit // basit[*at*]Virginia[*dot*]edu

Spring 2024

Friendly Reminders

- Your ***safety*** and ***comfort*** is important!
  - If you choose to wear a mask you are welcome to do so
  - *We will interpret wearing a mask as being considerate and caring of others in the classroom (not that you are sick), and realize that some may choose to mask to remain distanced*
- Be an ***active*** participant in your learning!
  You're welcome and ***encouraged*** to ask questions during class!
- If you feel ***unwell***, or think you are, please stay home
  - *We will work with you!*
  - Get some rest ☺
  - View the recorded lectures – *please allow 24-48 hours to post*
  - *Contact us!*

Announcements

- **Quiz 6** is due by 11:00pm on 3/25 (***tonight***)!
- **PA05** is due by 11:00pm on 3/27 (***Wednesday***)!
  - Submit on Gradescope
  - Submit the right kinds of files
  - Submit files using the correct names
  - REMEMBER on Gradescope: you can submit an **UNLIMITED** number of times prior to the deadline.

Coming up...

- **Exam 2**: Monday, April 8, 2024 *(SDAC accommodations? Book time slot on April 8!)*
  - *In-class; exam on Sherlock (like last time)*
  - *Closed-book/closed-notes/closed-PyCharm/closed-everything!*
  - *Duration: 1 hour and 15 minutes (like last time)*

Reminder: Dictionaries (*Python keyword:* `dict`)

- Like a list, but with index names that you create (called "KEYS")
- Each key is paired with a "VALUE"
- We can think of a dictionary similar to a list, but instead of indices 0, 1, 2, 3, 4, …, we choose the index (an int, or a string, …)
- Using a dictionary:

```
d = {}  # an empty dictionary named d

d = {4: "San Francisco", 7: "Edinburgh"} # 2 key-value pairs

d[12] = "Tokyo"  # Adding a new key-value pair to a dictionary

x = d[4]        # Retrieving a value from a dictionary.

                # X will be assigned "San Francisco"
```

Reminder: Lists vs. Dictionaries

LIST

- Index to access members
- Indexes start with 0
- Indexes are consecutive ints
- To **add** a new thing:
  `list.append(something)`

DICTIONARY

- Has keys to access members
- Each key must be unique
- Key can be:
  - Strings, ints, floats, booleans, tuples
  - (Not: lists, sets, dictionaries)
- To **add** a new thing: `d[key]=value`
  - `Adding key-value pair`

Reminder: A dictionary contains **Key-Value Pairs**

- Think of **key-value pairs** like safety deposit boxes at a bank
- The ***values*** are stored in safety deposit boxes
- In order to access a value, you need the ***key*** to unlock the box
- Every box has a unique key

Additional ways to work with dictionaries

(Assuming we have a dictionary named: **`d`**`)`

`d[key]`

Return the value of *d* with key *key*. Raises a `KeyError` if *key* is not in the dictionary.

`len(d)`

Return the number of items in the dictionary.

`key in d`

Return `True` if *d* has a key *key*, else `False`.

`list(d)`

Return a list of all the keys in the dictionary.

`del d[key]`

Remove `d[key]` from *d*. Raises a `KeyError` if *key* is not in the dictionary.

`d.popitem()`

Remove and return a `(key, value)` pair from the dictionary. Pairs are returned in LIFO (most recent) order.

`d.pop``(`*`key`*`[, `*`default`*`])` *`# default is optional`*

If *key* is in the dictionary, remove it and return its value, else return *default*. If *default* is not given and *key* is not in the dictionary, a `KeyError` is raised.

`d.get``(`*`key`*`[, `*`default`*`])` *`# default is optional`*

Return the value for *key* if *key* is in the dictionary, else *default*. If *default* is not given, it defaults to `None`, so that this method never raises a `KeyError`.

Practice question 1 - *What is the output?*

```
instructors = {"001": "Lina", "002": "Jasmine", "003": "Kai"}

print(instructors["002"])
```

Practice question 1 - *What is the output?*

```
instructors = {"001": "Lina", "002": "Jasmine", "003": "Kai"}

print(instructors["002"])
```

Provide a kay… Receive associated value.

Jasmine

Practice question 2 - *What is the output?*

```
instructors = {"001": "Lina", "002": "Jasmine", "003": "Kai"}

print(len(instructors))
```

Practice question 2 - *What is the output?*

```
instructors = {"001": "Lina", "002": "Jasmine", "003": "Kai"}

print(len(instructors))
```

How many key-value pairs in the dictionary?

3

Practice question 3 - *What is the output?*

```
instructors = {"001": "Lina", "002": "Jasmine", "003": "Kai"}

print(1 in instructors)
```

Practice question 3 - *What is the output?*

```
instructors = {"001": "Lina", "002": "Jasmine", "003": "Kai"}

print(1 in instructors)
```

Is 1 (a key) in this dictionary?

False

1 is not a key, but “001” is a key.
1 is perhaps part of “001” but we need the whole key

Practice question 4 - *What is the output?*

```
instructors = {"001": "Lina", "002": "Jasmine", "003": "Kai"}

print("002" in instructors)
```

Practice question 4 - *What is the output?*

```
instructors = {"001": "Lina", "002": "Jasmine", "003": "Kai"}

print("002" in instructors)
```

Is “002” (a key) in this dictionary?

True

Practice question 5 - *What is the output?*

```
instructors = {"001": "Lina", "002": "Jasmine", "003": "Kai"}

print("Lina" in instructors)
```

Practice question 5 - *What is the output?*

```
instructors = {"001": "Lina", "002": "Jasmine", "003": "Kai"}

print("Lina" in instructors)
```

Is “Lina” (a key) in this dictionary?

False

No because “Lina” is a value, not a **key**

Practice question 6 - *What is the output?*

```
instructors = {"001": "Lina", "002": "Jasmine", "003": "Kai"}

print("Lina" in instructors.values())
```

Practice question 6 - *What is the output?*

```
instructors = {"001": "Lina", "002": "Jasmine", "003": "Kai"}

print("Lina" in instructors.values())
```

Is “Lina” (a value) in the collection of values of this dictionary?

True

Practice question 7 - *What is the output?*

```
instructors = {"001": "Lina", "002": "Jasmine", "003": "Kai"}

print(list(instructors)[0])
```

Practice question 7 - *What is the output?*

```
instructors = {"001": "Lina", "002": "Jasmine", "003": "Kai"}

print(list(instructors)[0])
```

Print the first (index item 0) item in the list of keys

001

Practice question 8 - *What is the output?*

```
instructors = {"001": "Lina", "002": "Jasmine", "003": "Kai"}

del instructors["001"]
print(instructors["001"])
```

Practice question 8 - *What is the output?*

```
instructors = {"001": "Lina", "002": "Jasmine", "003": "Kai"}

del instructors["001"]
print(instructors["001"])
```

Delete the key-value pair where the key is “001”
Then let’s try to print the value associated with the key “001”

KeyError: '001'

Given we deleted this key-value pair, we get a KeyError
(Key is no longer in the dictionary)

Practice question 9 - *What is the output?*

```
instructors = {"001": "Lina", "002": "Jasmine", "003": "Kai"}

print(instructors.get("003", "NA"))
```

Practice question 9 - *What is the output?*

```
instructors = {"001": "Lina", "002": "Jasmine", "003": "Kai"}

print(instructors.get("003", "NA"))
```

Get the value associated with key “003”

Kai

Given we have a key that matches “003” we get “Kai”
However, if we didn’t, we would get “NA” (non-applicable)
(See next example)

Practice question 10 - *What is the output?*

```
instructors = {"001": "Lina", "002": "Jasmine", "003": "Kai"}

print(instructors.get("004", "NA"))
```

Practice question 10 - *What is the output?*

```
instructors = {"001": "Lina", "002": "Jasmine", "003": "Kai"}

print(instructors.get("004", "NA"))
```

Get the value associated with key “004”

NA

Given we do NOT have a key that matches “004”
we get “NA” because there is no such key (second argument)

Practice question 11 - *What is the output?*

```
instructors = {"001": "Lina", "002": "Jasmine", "003": "Kai"}

print(instructors.get("004"))
```

Practice question 11 - *What is the output?*

```
instructors = {"001": "Lina", "002": "Jasmine", "003": "Kai"}

print(instructors.get("004"))
```

Get the value associated with key “004” – notice we do not have the second argument

None

Given we do NOT have the second argument to the get() function, and given we do NOT have a key that matches “004” we get None. Remember that this method **never** raises a `KeyError`

Reminder: Looping Through a Dictionary

- Typical to use a for loop and use the .keys() function *(a simple way!)*

```
# Add elements to a dictionary
painting_years = {}
painting_years["Mona Lisa"] = 1503  # Leonardo da Vinci
painting_years["Girl With A Pearl Earring"] = 1665  # Johannes
painting_years["Starry Night"] = 1889  # Vincent van Gogh


# Print every painting with the year it was painted
for i in painting_years.keys():
    print(i, "was painted in the year", painting_years[i])
```

Class Activity

Given a **dictionary** containing states and capitals. Write a program that allows users to pick a state, then guess the capital. The beginning of a program is given below:

```
# states and capitals
capitals = {
  "Alabama": "Montgomery"
  , "Alaska": "Juneau"
  … } # assume the rest are included here…

which_state = input("Pick a state: ")
if which_state in capitals: # in - is a key in the dict
  guess = input("What is the capital of " + which_state + "?
")
(... YOUR CODE HERE)
```

```
# states and capitals
capitals = {
  "Alabama": "Montgomery"
  , "Alaska": "Juneau"
  , "Arizona": "Phoenix"
  , "Arkansas": "Little Rock"
  , "California": "Sacramento"
  , "Colorado": "Denver"
  , "Connecticut": "Hartford"
  , "Delaware": "Dover"
  , "Hawaii": "Honolulu"
  , "Florida": "Tallahassee"
  , "Georgia": "Atlanta"
  , "Idaho": "Boise"
  , "Illinois": "Springfield"
  , "Indiana": "Indianapolis"
  , "Iowa": "Des Moines"
  , "Kansas": "Topeka"
  , "Kentucky": "Frankfort"
  , "Louisiana": "Baton Rouge"
  , "Maine": "Augusta"
  , "Maryland": "Annapolis"
  , "Massachusetts": "Boston"
  , "Michigan": "Lansing"
  , "Minnesota": "St. Paul"
  , "Mississippi": "Jackson"
  , "Missouri": "Jefferson City"
  , "Montana": "Helena"
  , "Nebraska": "Lincoln"
  , "Nevada": "Carson City"
  , "New Hampshire": "Concord"
  , "New Jersey": "Trenton"
  , "New Mexico": "Santa Fe"
  , "North Carolina": "Raleigh"
  , "North Dakota": "Bismarck"
  , "New York": "Albany"
  , "Ohio": "Columbus"
  , "Oklahoma": "Oklahoma City"
  , "Oregon": "Salem"
  , "Pennsylvania": "Harrisburg"
  , "Rhode Island": "Providence"
  , "South Carolina": "Columbia"
  , "South Dakota": "Pierre"
  , "Tennessee": "Nashville"
  , "Texas": "Austin"
  , "Utah": "Salt Lake City"
  , "Vermont": "Montpelier"
  , "Virginia": "Richmond"
  , "Washington": "Olympia"
  , "West Virginia": "Charleston"
  , "Wisconsin": "Madison"
  , "Wyoming": "Cheyenne"
}   # this is my dictionary
print(capitals)

which_state = input("Pick a state: ")
if which_state in capitals: # in - is a key in the dict
  guess = input("What is the capital of " + which_state + "? ")
  if guess == capitals[which_state]: # capitals[which_state]: -> value
     print("That's correct")
  else:
     print("Sorry,", capitals[which_state], "is the capital of", which_state)
else:
  print('Hmmm, I don\'t recognize the state, "' + which_state + '"')
```

```python
# records votes for a restaurant
#   each restaurant has a related global variable, initialized to 0
#   the vote function compares the string to related strings and
#      records a vote if it finds a match

chipotle = 0
bodos = 0
mellow = 0

def vote(restaurant, num_votes=1):
  global chipotle, bodos, mellow
  if restaurant.lower() in ['chipotle', 'burrito']:
     chipotle += num_votes
  elif restaurant.lower() in ['bodos', 'bodo\'s', 'bagels']:
     bodos += num_votes
  elif restaurant.lower() in ['mellow', 'mellow mushroom', 'pizza']:
     mellow += num_votes
  else:
     print('I don\'t know about:',restaurant)

def print_totals():
  print(chipotle, 'votes for Chipotle')
  print(bodos, 'votes for Bodos')
  print(mellow, 'votes for Mellow')

  largest = max(chipotle, bodos, mellow)
  if chipotle == largest:
     print('Chipotle wins!')
  if bodos == largest:
     print('Bodos wins!')
  if mellow == largest:
     print('Mellow wins!')
```

```python
vote('Mellow Mushroom')
vote('Bodos')
vote('Pizza')
vote('College Inn')
vote('Chipotle', 3)
print_totals()



# Let's improve this with dictionaries.

totals = {'chipotle':0, 'bodos':0, 'mellow':0}

def vote(restaurant, num_votes=1):
  if restaurant.lower() in ['chipotle', 'burrito']:
     totals['chipotle'] += num_votes
  elif restaurant.lower() in ['bodos', 'bodo\'s', 'bagels']:
     totals['bodos'] += num_votes
  elif restaurant.lower() in ['mellow', 'mellow mushroom', 'pizza']:
     totals['mellow'] += num_votes
  else:
     yesno = input('Do you want to add ' + restaurant + ' to the list? ')
     if yesno.lower() in ['y','yes','yep','ok','sure']:
        totals[restaurant.lower()] = num_votes

def print_totals():
  for k,v in totals.items():
     print(v, 'votes for', k.title())
  #largest = max(totals.values())
  for k,v in totals.items():
     if v == largest:
        print(k.title(), 'wins!')
```

Python Demonstration

Let's jump on PyCharm!

`dictionaries.py` – Examples illustrating the dictionary data structure.

In-Class "lab" Activity!

Activity for Today!

- In pairs or groups up to three work on the following activity.
- **dictionaries_ica2.py**
- *Write an interactive student grade tracker using a dictionary*

Remember to check-in with a TA before leaving class today!

Reminder: CS Laptop Loaner Program

- This course requires students to have a **laptop**
- I realize that not everybody might have one (nor necessarily need one for their desired major / path…)
- If you do not have a laptop for any reason… ***not to worry!***
- The CS department's Systems staff has a notebook / laptop loaner program and will be able to loan you a notebook / laptop computer for the duration of the semester if you don't have one or if you cannot afford one.
  - Also available if your laptop is broken and under repair, we can arrange for you to receive a loaner laptop for a week or two until your own laptop is fixed

Interested? Link: https://www.cs.virginia.edu/wiki/doku.php?id=cs_laptop_loaner

I am happy to be your sponsor. Please let me know.