CMSC201
Computer Science I for Majors

Lecture 19 – Dictionaries

Last Class We Covered

- File I/O
  - How to open a file
    - For reading or writing
  - How to read from a file
  - How to write to a file
  - How to close a file

#TBT

Boolean Expressions

Any Questions from Last Time?

Today's Objectives

- Learn about the dictionary data type
- Construct dictionaries and access entries in those dictionaries
- Use methods to manipulate dictionaries
- Decide whether a list or a dictionary is an appropriate data structure for a given application

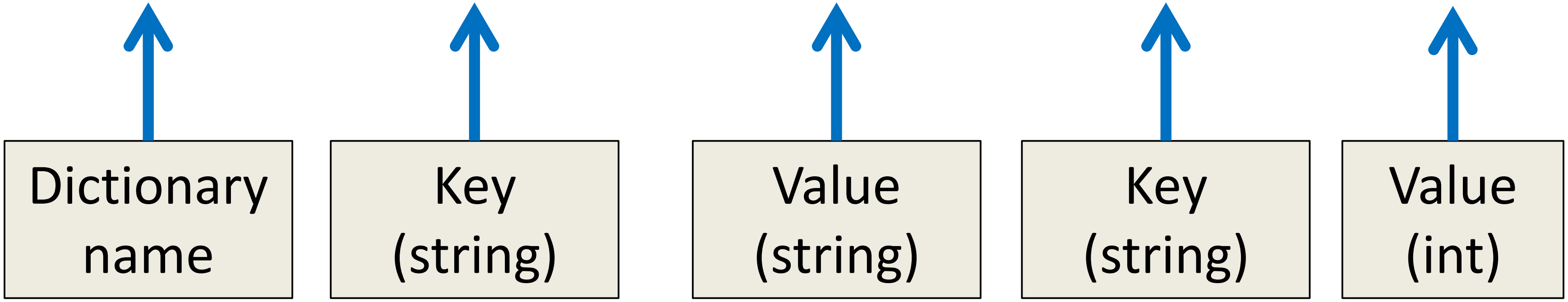
Organization

- Information in a list is organized how?
  - By order
- Information in a dictionary is organized...
  - By ***association***
- Python dictionaries associate a set of ***keys*** with corresponding data ***values***

Keys and Values

- A dictionary is a set of “keys” (terms), each pointing to their own “values” (meanings)

```
course = {"major" : "CMSC", "number" : 201}
```



Dictionary name

Key (string)

Value (string)

Key (string)

Value (int)

Purpose of Dictionaries

- Why use a dictionary instead of a list?

- Dictionaries are ***association*** based
  - It's very easy (and quick!) to find something if you know the key
- No matter how big the dictionary is, it can find any entry almost instantaneously
  - Lists would require iterating over the list until the item is found

Dictionary Keys

- Think of a dictionary as an unordered set of ***key:value*** pairs
- Dictionary keys must be ***unique***
  - A key in a dictionary is like an index in a list
  - Python must know exactly which value you want
- Keys can be of any data type
  - As long as it is ***immutable***

Dictionary Values

- Dictionary keys have many rules, but the values do not have many restrictions
- They do not have to be unique
  - Why?

We can have duplicate values in a list, but indexes must be unique

- They can be mutable or immutable
  - Why?

Since they don’t need to be unique, we can change them without restriction

Dictionary Usage Example

- What if we have a list of every student at UMBC, with all the info represented as a list?
  - The first element of the info list is the UMBC ID #
- How long would it take to find a specific student?
  - If the list is unsorted, a very long time!
  - If it’s sorted, resort every time a student is added
- Finding a student by ID # in a dictionary, on the other hand, is very *very* quick

Hashing

- Why are dictionaries so fast?
  - Hashing!

- Hashing is a way of translating arbitrary data (like strings or large numbers) into a smaller set space for ease of use

Hashing

- Hashing takes in anything (a string, an int, a float, etc.) and generate a number based on it
  - Same result for same input
  - Use a number to tell where to store in memory

- Given the same input, you get the same number, and can find it again very quickly

Hash Functions

- A function that, given a value, returns a value that tells us where it is stored in memory
  - If it’s in that location, it’s in the dictionary
  - If it’s not in that location, it’s not in the dictionary
- The hashing function has no other purpose
  - If we look at the function’s inputs and outputs, they probably won’t “make sense”
  - This function is called a hash function because it “makes hash” of its inputs

Hash Usage Example

- The **`AB12345`** UMBC student ID number
  - Gives 67,600,000 possible combinations
  - Making a list of that size wastes a lot of space
    - Wouldn’t use even 1% of the list
  - Making a dictionary allows us to better store the thousands of students without requiring a massive waste of space

Creating Dictionaries

Creating Dictionaries (Curly Braces)

- The empty dictionary is written as two curly braces containing nothing

```
dict1 = {}
```

- To create a dictionary, use curly braces and a colon (:) to separate keys from their value

```
dict2 = {"name" : "Maya", "age" : 7}
```

Creating Dictionaries (From a List)

- To cast a list as a dictionary, you use `dict()`

```
myPantry = [['candy', 5],
 ['cookies', 16],
 ['ice cream', 2]]
```

Must be key, value pairs

```
# cast to a dictionary
myDict = dict(myPantry)
```

Dictionary Operations

Dictionary Operations

- Dictionaries are probably most similar to a list

- You can do a number of operations:
  - Access a key’s value
  - Update a key’s value
  - Add new key:value pairs
  - Delete key:value pairs

Accessing Values

- To access dictionary elements, you use the square brackets and the key to obtain its value

```
dogBreeds = {"A" : "Akita", "B" : "Basenji",
             "C" : "Chesapeake Bay Retriever"}
print("dogBreeds at C:", dogBreeds["C"])
print("dogBreeds at B:", dogBreeds["B"])
```

```
Output:
dogBreeds at C: Chesapeake Bay Retriever
dogBreeds at B: Basenji
```

Updating Values

- To update dictionary elements, you use the square brackets and the key to indicate which value you would like to update

```
dogBreeds["B"] = "Beagle"
print(dogBreeds)
```

Why are these out of order?

```
Output:
{'C': 'Chesapeake Bay Retriever',
'B': 'Beagle', 'A': 'Akita'}
```

Dictionaries organize by ***association***, not by order

Adding New Key:Value Pairs

- To add new values, we don't need to use **append()** – we simply state the key and value we want to use, with square brackets

```
dogBreeds["D"] = "Dunker"
dogBreeds["E"] = "Eurasier"
print(dogBreeds)

Output:
{'C': 'Chesapeake Bay Retriever', 'B': 'Beagle',
'A': 'Akita', 'E': 'Eurasier', 'D': 'Dunker'}
```

Deleting Key:Value Pairs

- Key:value pairs must be deleted together; you can’t have a key with no value
- To delete a key:value, use the **`del`** keyword and specify the key you want to delete

```
del dogBreeds["D"]
print(dogBreeds)

Output:
{'C': 'Chesapeake Bay Retriever', 'B': 'Beagle',
'A': 'Akita', 'E': 'Eurasier'}
```

Time for…

LIVECODING!!!

Creating Dictionaries (From Two Lists)

- Here we have two lists
  - Of the same length
  - Contents of each index match up
    - (Pratik is Social Work, Amber is Pre-Med, etc.)

```
names = ["Pratik", "Amber", "Sam"]
major = ["Social Work", "Pre-Med", "Art"]
```

- Write the code to create a dictionary from these

Dictionary Methods

Methods

- Methods are functions that are specific to a data type (like **`append()`** or **`lower()`**, etc.)
- **`theDictionary.get(theKey)`**
  - For a key **`theKey`**, returns the associated value
  - If **`theKey`** doesn't exist, returns **`None`**
  - Optionally use a second parameter to return something other than **`None`** if not found
    - **`theDictionary.get(theKey, -1)`**

Methods

- **`theDictionary.values()`**
  - Returns a “view” of the **`theDictionary`**’s values
  - Need to cast to a list
- **`theDictionary.keys()`**
  - Returns a “view” of the **`theDictionary`**’s keys
  - Need to cast to a list
- The two lists returned are in the same order
  - (Value at index 0 matches key at index 0, etc.)

When to Use Dictionaries

- Dictionaries are very useful if you have...
  - Data whose order doesn’t matter
  - A set of unique keys
    - Key is a word, value is the definition (or translation)
    - Key is a postal abbreviation, value is the full state name
    - Key is a name, value is a list of their game scores
  - A need to find things easily and quickly
  - A need to easily add and remove elements

Daily CS History

- Charles Babbage
  - – Invented the Analytical Engine
    - Was never built, but would have used punched cards to control a mechanical calculator
  - – Work fell into obscurity, and computer builders in the 30s and 40s re-invented many of his architectural innovations
  - – Also invented the cow catcher for trains

More Daily CS History

- Ada Lovelace
  - – Wrote the first ever computer algorithm
  - – Realized the potential of the Analytical Engine
    - If numbers could be used to represent other things (like music notes), the “engine might compose elaborate and scientific pieces of music of any degree of complexity or extent”

Announcements

- Homework 6 is due this Friday at 11:59:59 PM

- Final exam is going to be on:
  - Friday, May 17th from 6 to 8 PM
  - Rooms will be assigned closer to the date
  - If you can’t take the exam at that time, you need to let Dr. Gibson know via email NOW, not later

Image Sources

- Charles Babbage (adapted from):
  - https://commons.wikimedia.org/wiki/File:Charles_Babbage_1860.jpg
- Ada Lovelace (adapted from):
  - https://commons.wikimedia.org/wiki/File:Ada_Lovelace.jpg