More Dictionaries

CSE 160 Summer 2026

Questions?

Announcements

- **Written Check-in 4** due tonight, July 17th at 11:59pm
- **Programming Practice 3** due Sunday, July 19th at 11:59pm
- **Homework Assignment 3** due Monday, July 20th at 11:59pm
- **Resubmission 1** due Wednesday, July 22th at 11:59pm
    - If you do not submit a form, we cannot accept your resubmission
- **Midterm on Monday, July 20th during class (9:40AM-10:40AM)**
    - Please get a good night's rest!
    - Best of luck studying!

Midterm Exam

Logistics

- Held in-class on Monday, July 20th (in HRC 155 - this room!)
- You **may** bring **one sheet of 8.5 x 11 inch paper as your own cheat sheet**
- You will be provided with an additional reference sheet as part of the exam packet

Resources

- **Past midterm exams** (+ answer keys) on course website midterm page
- Office Hours + Ed :)

Questions?

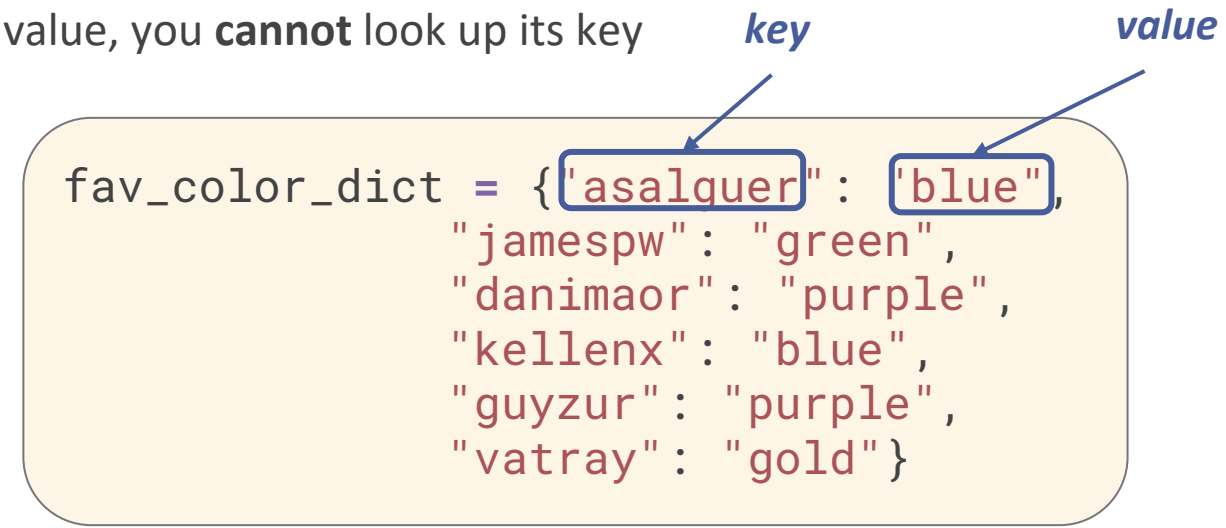
Today's Roadmap

1. Dictionaries
2. Dictionary Methods
3. Nested Dictionaries

Jupyter Hub

What is a Dictionary?

- A dictionary is a data structure that maps **keys** to **values**
- Unlike lists, order does not matter in dictionaries.
  - You could think of a dictionary as a big bag of key -> value pairs
- Given a key, you can look up its associated value
  - Given a value, you **cannot** look up its key

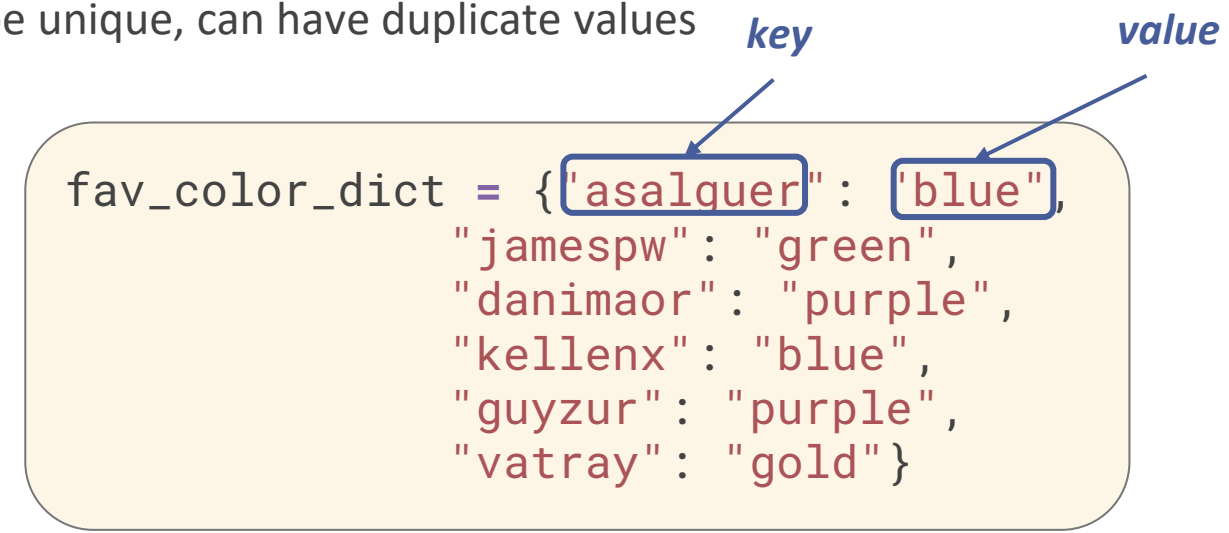


Dictionary Syntax

- **Curly braces** begin and end the dictionary
- **Colon** separates each key from its associated value
- **Comma** between each key: value pair

A Dictionary maps Keys to Values

- **Keys**:
  - Must be unique! No duplicate keys!
  - Must be an *immutable* type (e.g. integer, float, string)
- **Values**:
  - Do not need to be unique, can have duplicate values
  - Can be any type



Creating Dictionaries

Python Tutor

```
fav_color = {"asalguer": "blue", "jamespw": "green",
             "danimaor": "purple", "kellenx": "blue",
             "guyzur": "purple", "vatray": "gold"}

squares = {2: 4, 3: 9, 5: 25, -5: 25, -2: 4, -3: 9}

atomic_number = {"H": 1, "Fe": 26, "Au": 79}

food_price = {"Taco": 3.25, "Burrito": 7.5,
                            "Chips": 2, "Guac": 4.75}
```

Is a Key in a dictionary?

`x in dict_name` is a boolean expression that evaluates to **True** if **`x`** is *one of the keys* in **`dict_name`**.

`x not in dict_name` is a boolean expression that evaluates to **True** if **`x`** is ***not*** *one of the keys* in **`dict_name`**.

```
fav_color = {"asalguer": "blue", "jamespw": "green",
             "danimaor": "purple", "kellenx": "blue",
             "guyzur": "purple", "vatray": "gold"}

print(fav_color["asalguer"])        # "blue"
print("maria" not in fav_color)     # True
print("jamespw" in fav_color)       # True
print("blue" in fav_color)          # False
```

Python Tutor

More Dictionary Methods

- `dict_name.keys()`
  - Retrieve all the keys in the dictionary
  - Returned in the form: `dict_keys([key1, key2,...])`
- `dict_name.values()`
  - Retrieve all the values in the dictionary
  - Return in the form: `dict_values([value1, value2,...])`
- `dict_name.items()`
  - Retrieve all the key-value pairs in the dictionary
  - Returned in the form: `dict_items([(key1, value1), (key2, value2), …])`

These are not lists, but a special dictionary object!
However, you can iterate through them using a for-loop just like a list!

Iterating through a dictionary

- Usually we want to look up what value is a associated with a single key:
  ```
  print("Adrian's fav color is ", fav_color["asalguer"])
  ```
- Sometimes it is useful to iterate over an entire dictionary
  - Note: It is not necessary to iterate over a dictionary to find a key, just look it up!
    ```
    print("The person's fav color is ", fav_color[person])
    ```

```
fav_color = {"asalguer": "blue", "jamespw": "green",
             "danimaor": "purple", "kellenx": "blue",
             "guyzur": "purple", "vatray": "gold"}

for uwnetid in fav_color.keys():
    print(uwnetid, " fav color is ", fav_color[uwnetid])
```

Python Tutor

Usually you should not iterate over a dictionary

- A very common mistake is to iterate over a dictionary with a loop, when all that is needed is to do a look up of a single value
- Iterate over a dictionary with a loop when:
  - You need to **modify** the values associated with every key in the dictionary
  - You need to **print** the values associated with every key in the dictionary
- Do NOT use a loop when you just need to look up the value associated with one key!

```
fav_color = {"asalguer": "blue", "jamespw": "green",
             "danimaor": "purple", "kellenx": "blue",
             "guyzur": "purple", "vatray": "gold"}

for uwnetid in fav_color.keys():
    print(uwnetid, " fav color is ", fav_color[uwnetid])
```

Counting words

Write code that uses a dictionary to count how many times each unique word appears in a phrase.

```
phrase = "how much wood could a woodchuck chuck if a
woodchuck could chuck wood"
```

Python Tutor

Jupyter Hub

Dictionary Values can be Lists

- Values do not need to be unique, can have duplicate values
- Values can be any type, including:
  - **Lists**

Python Tutor

```
class_rolls = {"CSE160": ["John", "Maria"],
               "CSE123": ["Anna", "Lin", "Ray"]}
print(class_rolls["CSE160"][1]) # prints "Maria"

students = {"John Doe": [1234567, "jdoe@uw.edu","B"],
            "Maria Vera": [3456789, "mvera@uw.edu","A"]}
print(students["John Doe"][0]) # prints 1234567
```

Dictionary Values can be Dictionaries

- Values do not need to be unique, can have duplicate values
- Values can be any type, including:
  - **Another Dictionary**
    - Sometimes called a "nested dictionary"

Python Tutor

```
students = {"John Doe": {"ID": 1234567,
                         "email": "jdoe@uw.edu",
                         "grade": "B"},
            "Maria Vera": {"ID": 3456789,
                           "email": "mvera@uw.edu",
                           "grade": "A"}}
print(students["John Doe"]["ID"]) # prints 1234567
```

Think Pair Share

Select **all** of the statements that will result in an error (or indicate that none do):

A. `print(students[2])`

B. `students["Lisa"] = 3`

C. `print(students["Maria Vera"][1])`

A. `students["John Doe"][ID] = 4567734`

B. None of these cause an error

```
students =
{"John Doe": {"ID": 1234567,
              "email": "jdoe@uw.edu",
              "grade": "B"},
 "Maria Vera": {"ID": 3456789,
                "email": "mvera@uw.edu",
                "grade": "A"}}
```

Sli.do #cse160

Nested Dictionary Example

Jupyter Hub

```
students = {"John Doe":     {"ID": 1234567,
                             "email": "jdoe@uw.edu",
                             "grade": "B"},
            "Maria Vera": {"ID": 3456789,
                             "email": "mvera@uw.edu",
                             "grade": "A"},
            "Jun Li": {"ID": 5678912,
                             "email": "jli@uw.edu",
                             "grade": "B"}}
# Determine if Maria Vera is in this course
# Find Jun Li's email address
# Modify John Doe's grade
# Add a new student to the dictionary
# Create a list of all student email addresses
```

Practice problem: Create a nested dictionary

Create a dictionary that maps state names to a dict that maps city names to populations:

population.txt

```
Washington Seattle 800000
Oregon Portland 635000
Michigan Detroit 645700
Washington Olympia 56000
Oregon Salem 180400
```

Expected Dictionary:

```
{'Washington': {'Seattle': 800000,
                'Olympia': 56000},
 'Oregon': {'Portland': 635000,
            'Salem': 180400},
 'Michigan': {'Detroit': 645700}}
```

Bonus: Write this as a function that takes filename as a parameter and returns a dictionary

Bonus bonus: Given this dictionary, create a new (non-nested) dictionary that maps cities to their populations.

Bonus bonus bonus: Using this new dictionary, find the average population over all cities.

Jupyter Hub