CS 5001

LECTURE 7

DICTIONARIES & SETS

PLUS EXAM 1 REVIEW

KEITH BAGLEY

FALL 2023

Northeastern University
Khoury College of Computer Sciences

440 Huntington Avenue ▪ 202 West Village H ▪ Boston, MA 02115 ▪ T 617.373.2462 ▪ khoury.northeastern.edu

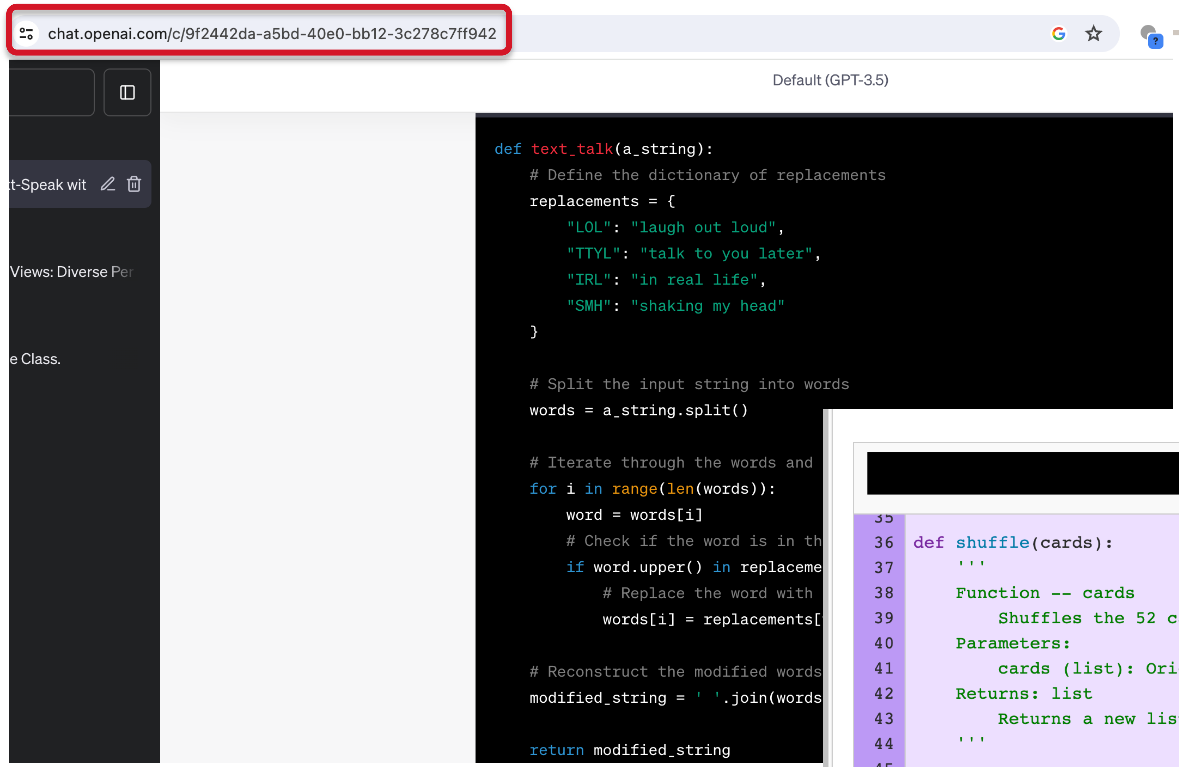
AGENDA

- Course Check-in
- Midterm Review
- Files & Exception Handling Review
- Tracing Code
- Dictionaries
- Sets
- Launch into Lab
- Q&A

QUOTE OF THE WEEK

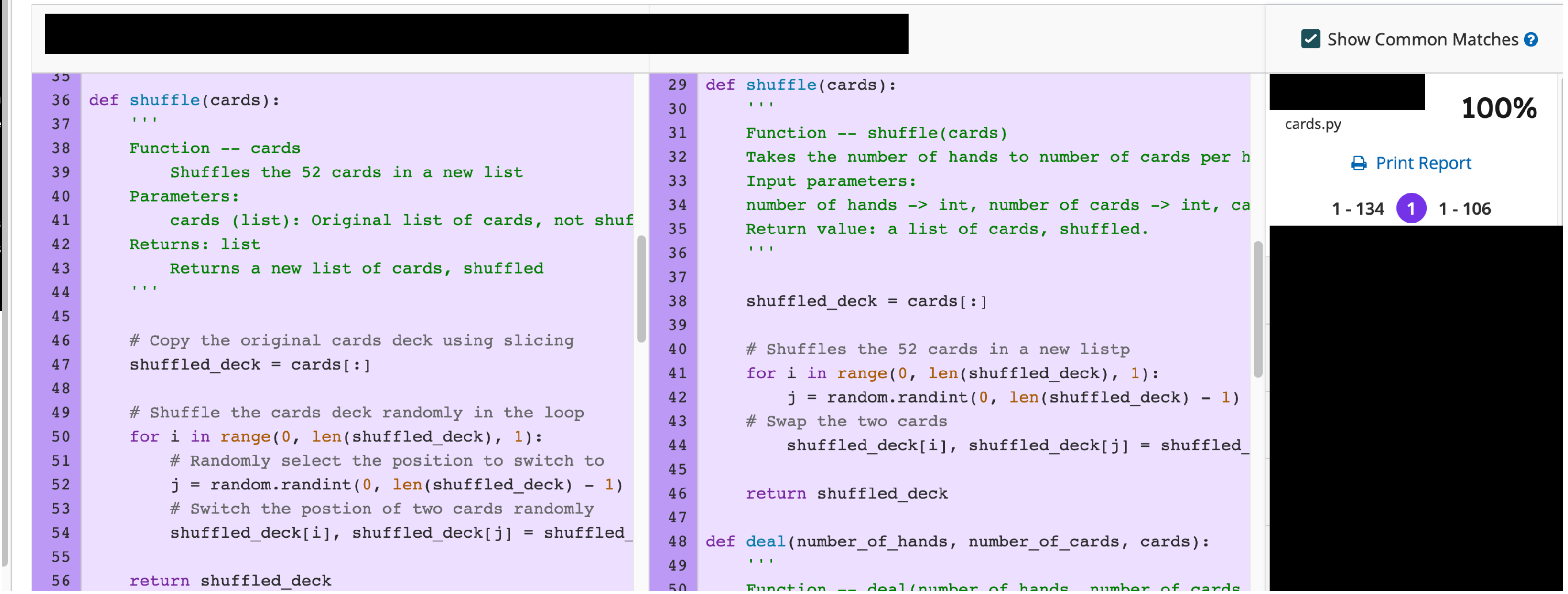
“As soon as possible, experts hungrily seek feedback on how they did. Necessarily, much of that feedback is negative. This means that experts are more interested in what they did wrong—so they can fix it—than what they did right. The active processing of this feedback is as essential as its immediacy.”
— Angela Duckworth, *Grit : The Power of Passion and Perseverance*

CHECK-IN AND PSA



```
def text_talk(a_string):
    # Define the dictionary of replacements
    replacements = {
        "LOL": "laugh out loud",
        "TTYL": "talk to you later",
        "IRL": "in real life",
        "SMH": "shaking my head"
    }

    # Split the input string into words
    words = a_string.split()

    # Iterate through the words and
    for i in range(len(words)):
        word = words[i]
        # Check if the word is in th
        if word.upper() in replaceme
            # Replace the word with
            words[i] = replacements[

    # Reconstruct the modified words
    modified_string = ' '.join(words

    return modified_string
```

Don't let this be you.
Maintain your integrity
Sharing/Copying work or using chatGPT will be penalized



[redacted]

```
def shuffle(cards):
    '''
    Function -- cards
        Shuffles the 52 cards in a new list
    Parameters:
        cards (list): Original list of cards, not shuf
    Returns: list
        Returns a new list of cards, shuffled
    '''

    # Copy the original cards deck using slicing
    shuffled_deck = cards[:]

    # Shuffle the cards deck randomly in the loop
    for i in range(0, len(shuffled_deck), 1):
        # Randomly select the position to switch to
        j = random.randint(0, len(shuffled_deck) - 1)
        # Switch the postion of two cards randomly
        shuffled_deck[i], shuffled_deck[j] = shuffled_

    return shuffled_deck
```

```
def shuffle(cards):
    '''
    Function -- shuffle(cards)
    Takes the number of hands to number of cards per h
    Input parameters:
    number of hands -> int, number of cards -> int, ca
    Return value: a list of cards, shuffled.
    '''

    shuffled_deck = cards[:]

    # Shuffles the 52 cards in a new listp
    for i in range(0, len(shuffled_deck), 1):
        j = random.randint(0, len(shuffled_deck) - 1)
    # Swap the two cards
        shuffled_deck[i], shuffled_deck[j] = shuffled_

    return shuffled_deck

def deal(number_of_hands, number_of_cards, cards):
    '''
    Function -- deal(number of hands, number of cards
```

[redacted]

[redacted]

MIDTERM REFLECTIONS

- Strong performance by the class on the exam
- Goal: Assessment to ensure you are tracking with the concepts and topics
- Confidence that key concepts are “sticking”
- Some people are not good “test takers”. Only one data point

REVIEW: FILES ARE…

- Files are a form of "persistent" data
  - "Non-volatile" memory. The data remains even if you turn off the power on your computer
  - Often stored to disk
    - But…tape, USB, non-volatile RAM, etc. also options

REVIEW: PERSISTENCE IS…

- Persistence is more than just files
- Persistence => the data “persists” over time/space/devices
  - Could be files, databases, non-volatile memory, Cloud storage, continuous transmission
  - For this course, we’ll focus on files

REVIEW: TYPES OF FILES

- There are different kinds of files, and different file formats
  - Text files
  - Music files
  - Video
  - Word Processing
  - Etc.
- Broadly speaking, files can be segmented into 2 major types
  - Text files
    - Sequence of text characters
  - Binary files
    - Sequence of bytes

SUGGESTION: USE THE WITH-AS FORMAT

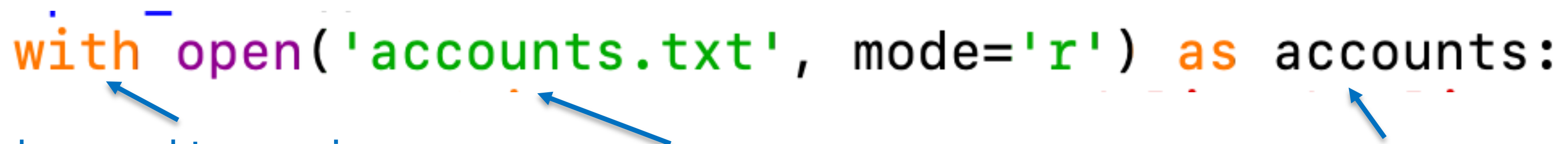
- If you use the plain/basic open function, you are required to close the file
  - Otherwise errors may occur
  - Resource sink
    - Sort of like leaving the door open in the winter with the heat blasting
- ”Modern” Python better practice to use the “with open” approach

```python
def open_with():
    with open('accounts.txt', mode='r') as accounts:
        for record in accounts: # read line by line
            acct_num, name, balance = record.split()
            print(acct_num, name, balance)
```

REVIEW: WITH KEYWORD (WITH FILES)

- Many applications acquire resources
  - files, network connections, database connections and more
  - Should release resources as soon as they're no longer needed
    - Ensures that other applications can use the resources
- with statement
  - Acquires a resource and assigns its corresponding object to a variable
  - Allows the application to use the resource via that variable
  - Calls the resource object's close method to release the resource

No explicit close() needed!



```
with open('accounts.txt', mode='r') as accounts:
```

With keyword to acquire resources (file in this case)

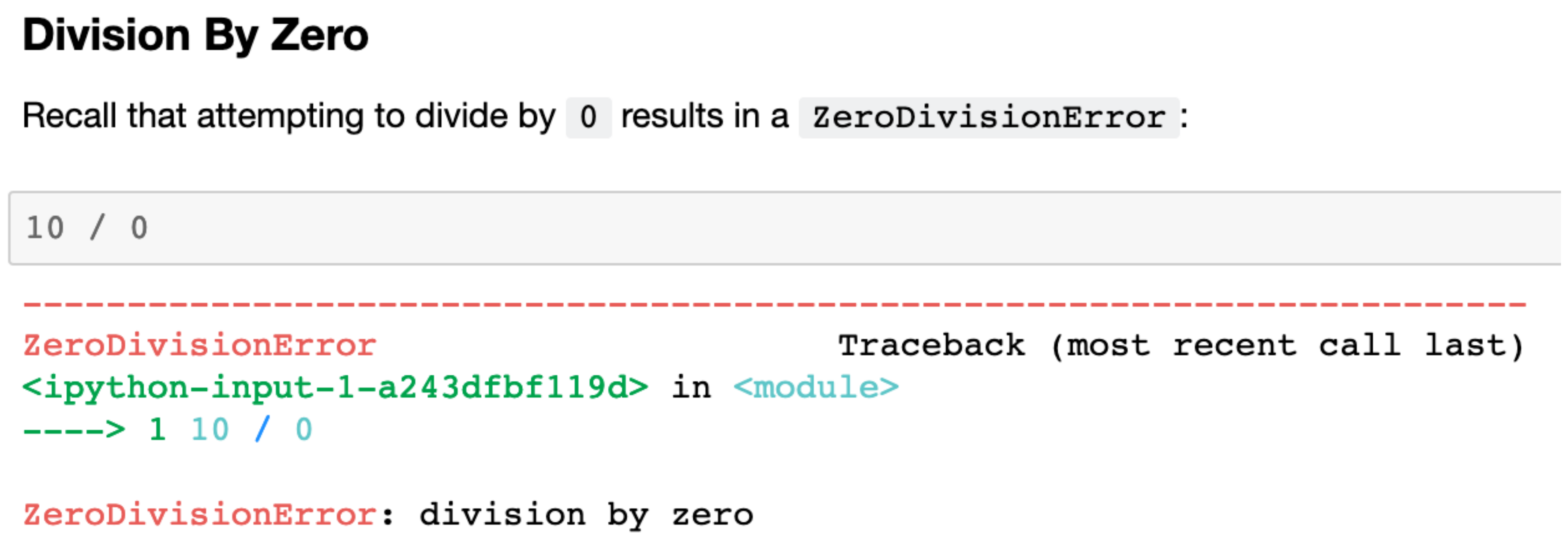
Open file

Variable associated

REVIEW: EXCEPTION HANDLING

- Python’s try/except block to handles errors in general and are useful for file access error.
- An exception is something unusual that happens; the System doesn’t know what to do next, and Python responds by giving us an error message on terminal.
- A try/except block allows us, the programmer, to find these kinds of exceptions, and write code to “handle” them -- give them a better response than the one here, especially a user-facing response

REVIEW: EXCEPTION HANDLING



Division By Zero

Recall that attempting to divide by `0` results in a `ZeroDivisionError`:

```
10 / 0
```

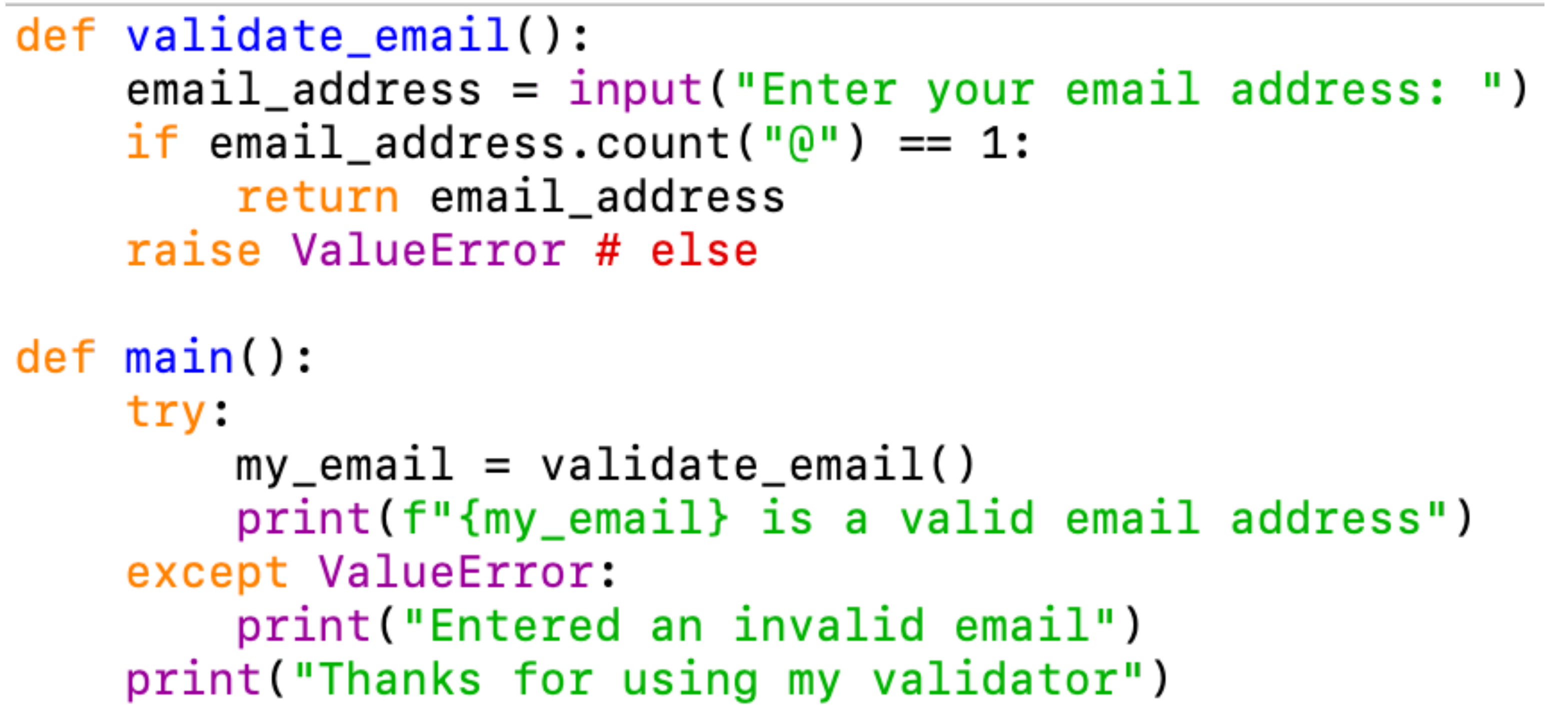
```
---------------------------------------------------------------------------
ZeroDivisionError                         Traceback (most recent call last)
<ipython-input-1-a243dfbf119d> in <module>
----> 1 10 / 0

ZeroDivisionError: division by zero
```

REVIEW: EXCEPTION HANDLING

```
def try_it(value):
    try:
        x = int(value)
    except ValueError:
        print('{} could not be converted to an integer'.format(value))
    else:
        print('int({}) is {}'.format(value, x))

def main():
    try_it(10.7)

    try_it('Python')

main()
```

REVIEW: RAISING EXCEPTIONS

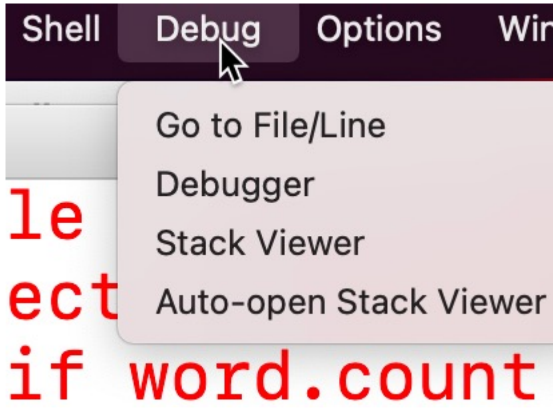
- Using raise
- A more realistic example:

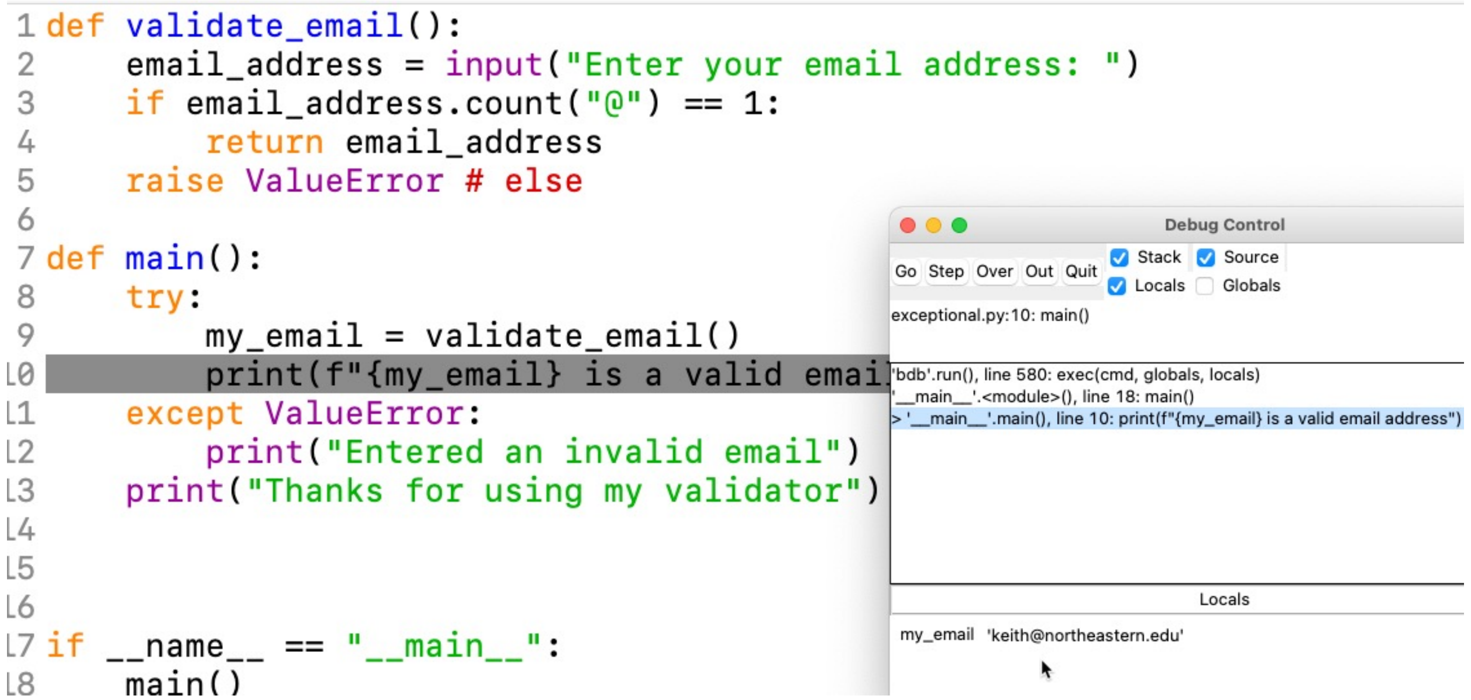


```
def validate_email():
    email_address = input("Enter your email address: ")
    if email_address.count("@") == 1:
        return email_address
    raise ValueError # else

def main():
    try:
        my_email = validate_email()
        print(f"{my_email} is a valid email address")
    except ValueError:
        print("Entered an invalid email")
    print("Thanks for using my validator")
```

REVIEW: DEBUGGING WITH DEBUGGERS

- Modern IDEs have debuggers that allow you to step in/out of functions, set "breakpoints" and watch variables change as the program executes
  - Even IDLE has a simple debugger





```
 1 def validate_email():
 2     email_address = input("Enter your email address: ")
 3     if email_address.count("@") == 1:
 4         return email_address
 5     raise ValueError # else
 6
 7 def main():
 8     try:
 9         my_email = validate_email()
10         print(f"{my_email} is a valid emai
11     except ValueError:
12         print("Entered an invalid email")
13     print("Thanks for using my validator")
14
15
16
17 if __name__ == "__main__":
18     main()
```

Stack [✓] Source [✓] Locals [✓] Globals [ ]

```
exceptional.py:10: main()

'bdb'.run(), line 580: exec(cmd, globals, locals)
'__main__'.<module>(), line 18: main()
> '__main__'.main(), line 10: print(f"{my_email} is a valid email address")
```

Locals

```
my_email  'keith@northeastern.edu'
```

HAND-DEBUGGING/TRACING

- You had a "trace the flowchart" on your midterm
- Knowing how to debug and "hand trace" code is also a useful skill
- Let's work through a quick example

AGENDA

- Course Check-in
- Midterm Review
- Files & Exception Handling Review
- Tracing Code
- Dictionaries
- Sets
- Launch into Lab
- Q&A

DICTIONARIES

- Dictionaries are like Lists
  - Variable with multiple values
  - Mutable
  - Each value in it’s own….~~index~~?
    - Named slot (KEY)

DICTIONARIES

- Dictionaries are NOT like Lists
  - Each value in it’s own Key
  - Key/Value pairs

LIST

- Lists have position & values

value

A list of movies

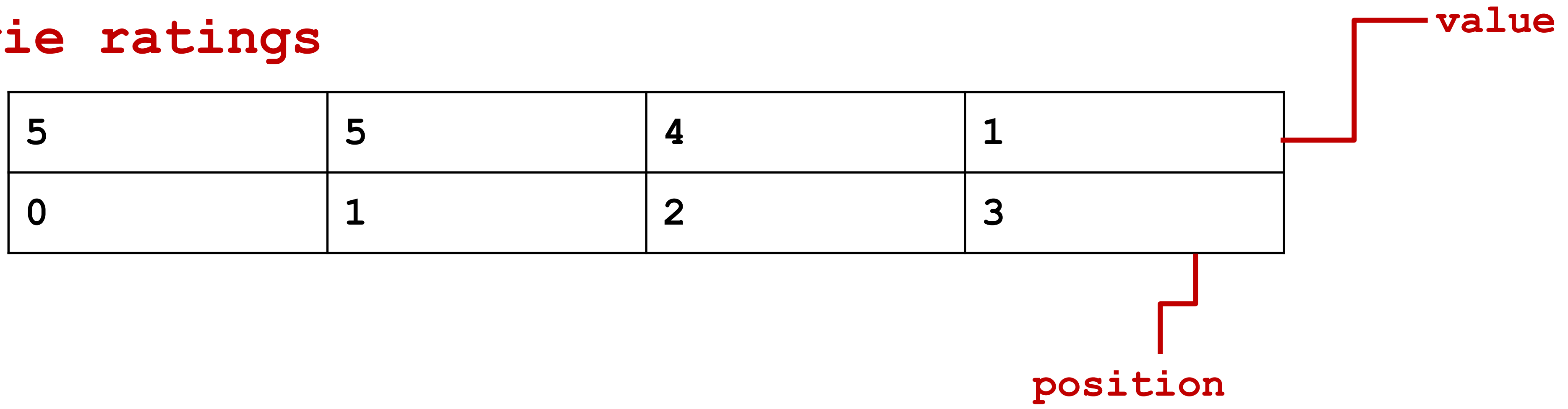
'Aquaman'	'Black Panther'	'Greenbook'	'Mary Poppins'
0	1	2	3

position

LIST

- Lists have position & values

A list of movie ratings



5	5	4	1
0	1	2	3

LIST

- Two lists can correspond (Associative Arrays)

A list of movies

'Aquaman'	'Black Panther'	'Greenbook'	'Mary Poppins'
0	1	2	3

A list of movie ratings

5	5	4	1
0	1	2	3

LIST

- Two lists can correspond (Associative Arrays)

movies[i]

'Aquaman'	'Black Panther'	'Greenbook'	'Mary Poppins'
0	1	2	3

ratings[i]

5	5	4	1
0	1	2	3

LIST

- Two lists can correspond (Associative Arrays)

`movies[i]`

'Aquaman'	'Black Panther'	'Greenbook'	'Mary Poppins'
0	1	2	3

`ratings[i]`

5	5	4	1
0	1	2	3

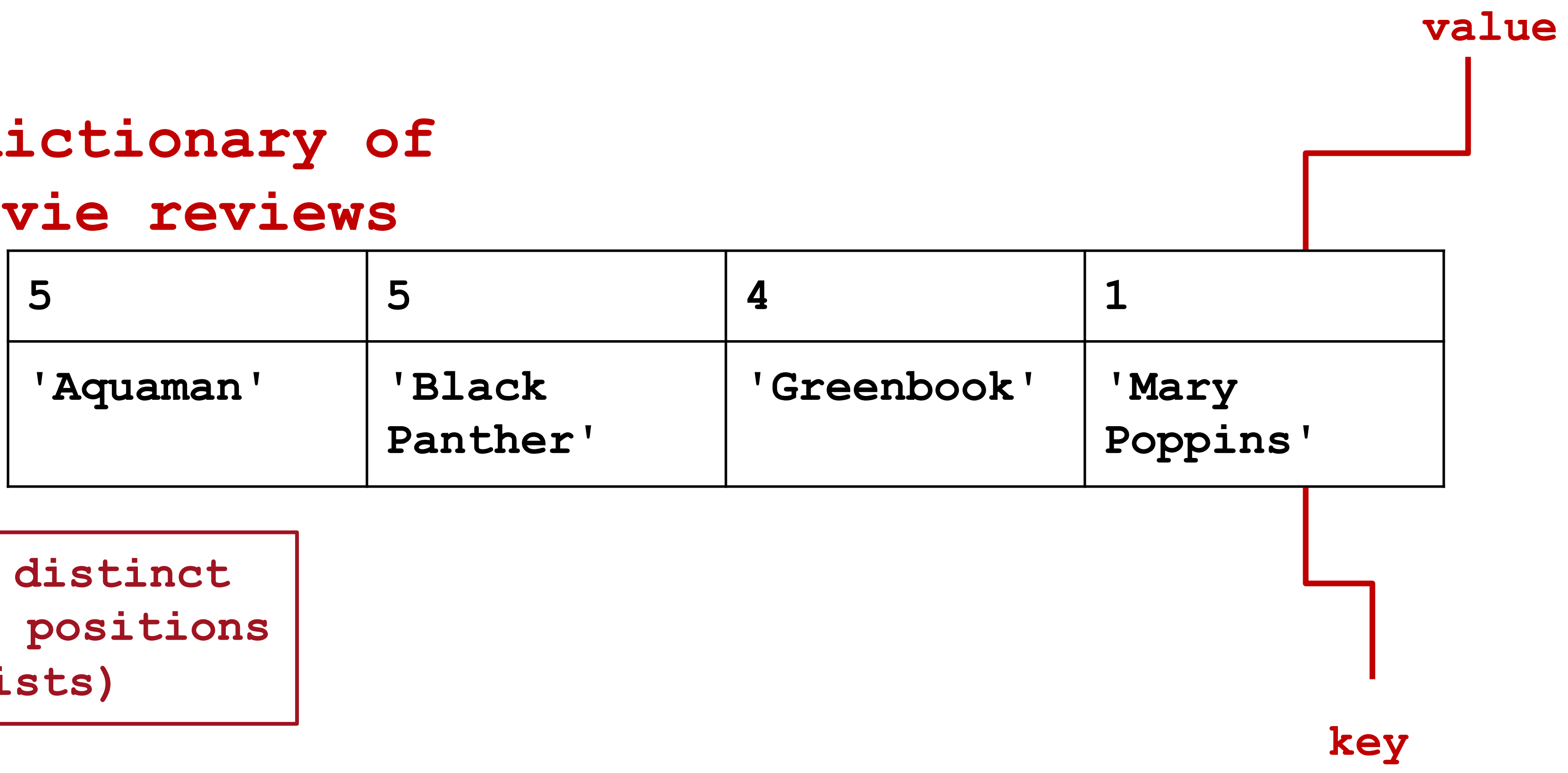
2 LISTS (ASSOCIATIVE ARRAYS)

- Associative Arrays work okay…
- But can be a maintenance/management problem
  - How did Greenbook do?
    - Find in 1st list
    - Get the same position in 2nd list
  - Add a new movie
    - Don’t forget to add to the second list too, or else!
  - What if we want to track more characteristics
    - Add more lists to track those qualities OR add a list with multi-faceted information that we need to remember index/positions for

DICTIONARIES FIT BETTER (SOMETIMES)

- Key instead of index/position

A dictionary of movie reviews



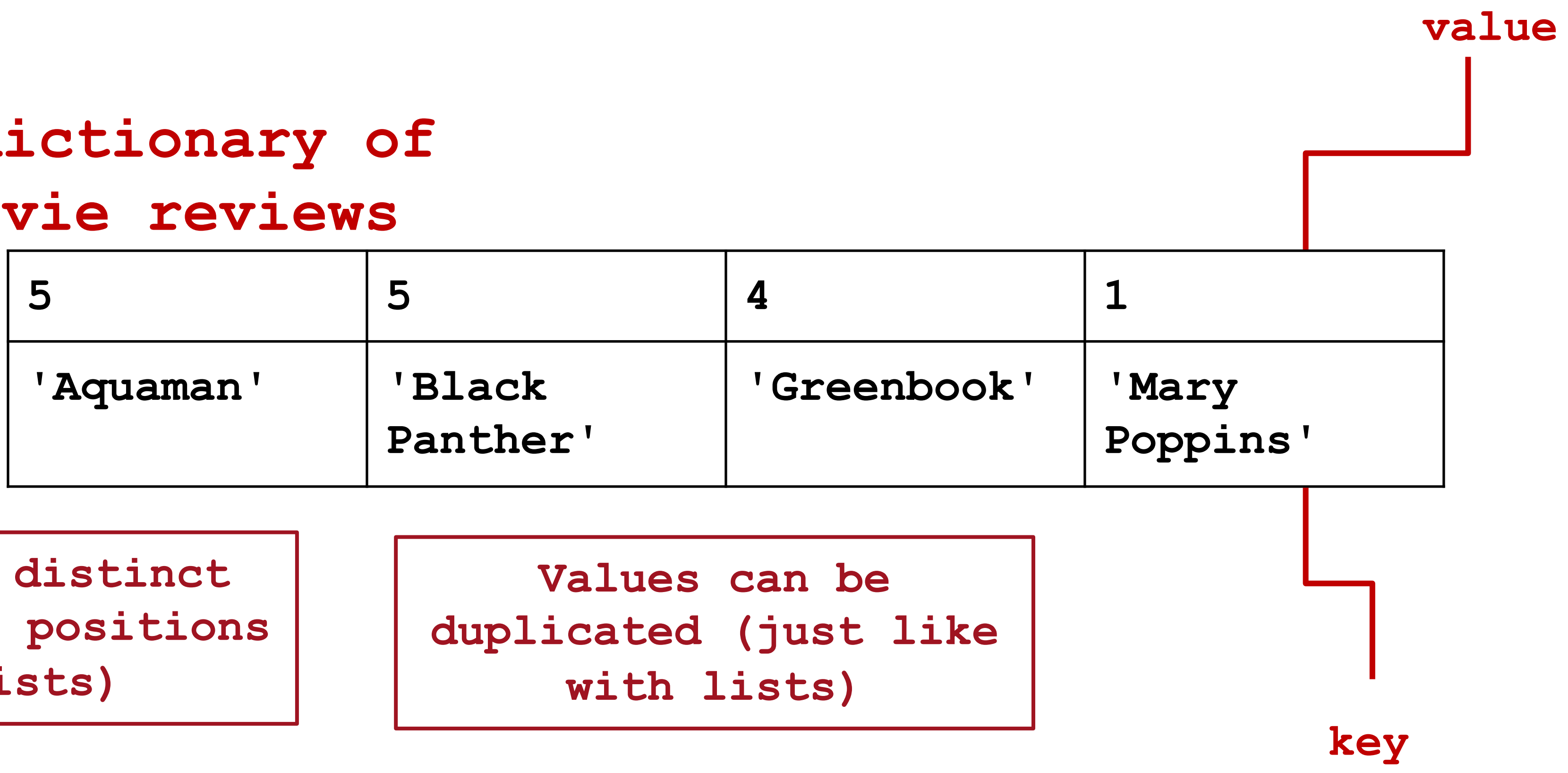
5	5	4	1
'Aquaman'	'Black Panther'	'Greenbook'	'Mary Poppins'

Keys are distinct (just like positions in lists)

DICTIONARIES FIT BETTER (SOMETIMES)

- Key instead of index/position

A dictionary of movie reviews



5	5	4	1
'Aquaman'	'Black Panther'	'Greenbook'	'Mary Poppins'

Keys are distinct (just like positions in lists)

Values can be duplicated (just like with lists)

DICTIONARIES WYNTK (WHAT YOU NEED TO KNOW)

- Use the help() command in IDLE to learn about all dictionary features

```
>>> help(dict)
```

- Create an empty dictionary

```
reviews = {} # could also do: reviews = dict()
```

- Add a Key/Value Pair

```
reviews = {} # OR reviews = {'Aquaman': 5}
reviews['Aquaman'] = 5
reviews['Black Panther'] = 5
```

DICTIONARIES ACCESSING DATA

- Bracket operator [] gives access for setting and getting data

```
reviews = {}
reviews['Aquaman'] = 5
reviews['Black Panther'] = 5
reviews['Greenbook'] = 4
reviews['Mary Poppins'] = 1
```

- Keys must be unique, otherwise you are overwriting existing information

```
reviews = {}
reviews['Aquaman'] = 5
reviews['Black Panther'] = 5
reviews['Greenbook'] = 4
reviews['Mary Poppins'] = 1
reviews['Aquaman'] = 4
```

```
overwrites the
old value!
```

DICTIONARIES ACCESSING DATA

- Access one element

```
reviews = {}
reviews['Aquaman'] = 5
reviews['Black Panther'] = 5
reviews['Greenbook'] = 4
reviews['Mary Poppins'] = 1
reviews['Aquaman'] = 4
print(reviews['Greenbook'])
```

```
prints
4
```

DICTIONARIES: ITERATION

- Can traverse over a Dictionary’s elements, one at a time with for-in

```
reviews = {}
reviews['Aquaman'] = 5
reviews['Black Panther'] = 5
reviews['Greenbook'] = 4
reviews['Mary Poppins'] = 1
reviews['Aquaman'] = 4
print(reviews['Greenbook'])
for key, value in reviews.items():
    print('Review of', key, 'is', value)
```

DICTIONARIES: ERRORS

- Similar to Lists and Index Errors, Dictionaries can have a KeyError

```
reviews = {}
reviews['Aquaman'] = 5
reviews['Black Panther'] = 5
reviews['Greenbook'] = 4
reviews['Mary Poppins'] = 1
reviews['Aquaman'] = 4
print(reviews['Greenbook'])
for key, value in reviews.items():
    print('Review of', key, 'is', value)
print(reviews['Princess Bride'])
```

```
KeyError
```

DICTIONARIES: ERRORS

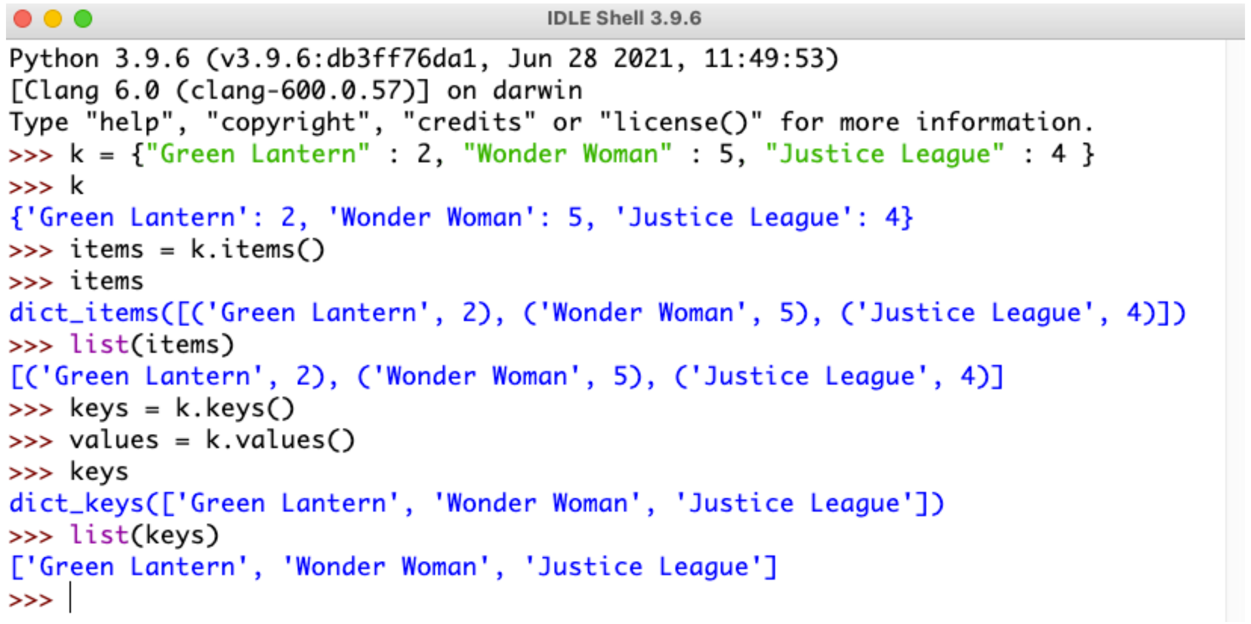
- Similar to Lists and Index Errors, Dictionaries can have a KeyError
  - Can handle this by using try/except clauses, OR
  - Use the synonym method get()
    - Works just like [] but allows you to specify a default value IF the Key does not exist

```
reviews = {}
reviews['Aquaman'] = 5
reviews['Black Panther'] = 5
print(reviews.get('Princess Bride', 3))
```

```
No KeyError Now!
```

CONVERTING DICTIONARIES TO LISTS

- Shouldn't need to do this often, but on occasion can be useful to gather keys or values as a list
- list() function can take a dictionary view and convert it easily



IDLE Shell 3.9.6

```
Python 3.9.6 (v3.9.6:db3ff76da1, Jun 28 2021, 11:49:53)
[Clang 6.0 (clang-600.0.57)] on darwin
Type "help", "copyright", "credits" or "license()" for more information.
>>> k = {"Green Lantern" : 2, "Wonder Woman" : 5, "Justice League" : 4 }
>>> k
{'Green Lantern': 2, 'Wonder Woman': 5, 'Justice League': 4}
>>> items = k.items()
>>> items
dict_items([('Green Lantern', 2), ('Wonder Woman', 5), ('Justice League', 4)])
>>> list(items)
[('Green Lantern', 2), ('Wonder Woman', 5), ('Justice League', 4)]
>>> keys = k.keys()
>>> values = k.values()
>>> keys
dict_keys(['Green Lantern', 'Wonder Woman', 'Justice League'])
>>> list(keys)
['Green Lantern', 'Wonder Woman', 'Justice League']
>>> 
```

WHEN TO USE WHAT?

Lists: We care about the positions of our values.
Or, maybe we just care about the values, and iterating over them

Dictionaries: We care about the names tied to our values.

WHEN TO USE WHAT?

2 lists

(1) movie names
(2) ratings

'Aquaman'	'Black Panther'	'Greenbook'	'Mary Poppins'
0	1	2	3

5	5	4	1
0	1	2	3

One dictionary

value	5	5	4	1
key	'Aquaman'	'Black Panther'	'Greenbook'	'Mary Poppins'

SOMETIMES USE BOTH

(1) temperature on a day
(2) precipitation on the same day

'Sunday'	'Sunday'	'Monday'	'Monday'
0	1	2	3

95.5	5	40	8
0	1	2	3

Hmmm. Doesn't seem quite right. Is this temp or precip?

SOMETIMES USE BOTH

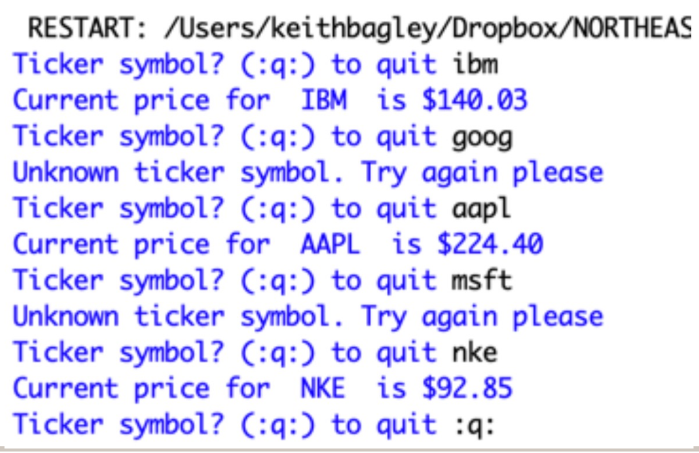
(1) temperature on a day
(2) precipitation on the same day

```
>>> weather = dict()
>>> weather[95.5] = 2.2
Or even
>>> weather2 = dict()
>>> weather2['Sunday'] = [95.5, 2.2]
>>> weather2
{'Sunday': [95.5, 2.2]}
```

YOU TRY! DICTIONARIES

I have a friend who used to be an obsessive stock watcher. The Dow Jones or NASDAQ couldn't move one point without him knowing about it and selling or buying shares.
We're not going to obsess over stocks this week, but we will use the market to practice with dictionaries.
Given the following (almost) fictitious stocks & prices:
IBM = $140.03, AAPL = $224.40 and NKE = $92.85

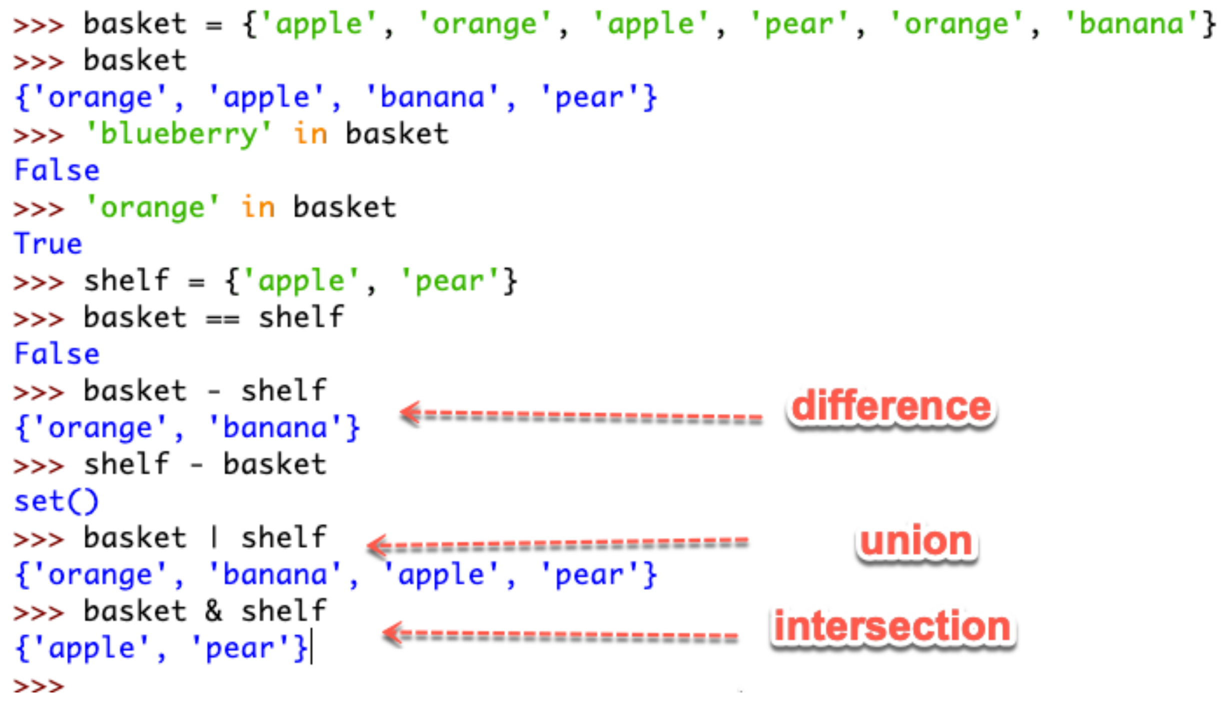
Open a new Python file. Create a new dictionary using the ticker symbol as the key and the current stock value as the value. Then, write a function that allows you to interactively ask for the stock price by ticker symbol and presents the price. If the stock doesn't exist in your dictionary, print an informative message to the user.



```
 RESTART: /Users/keithbagley/Dropbox/NORTHEAS
Ticker symbol? (:q:) to quit ibm
Current price for  IBM  is $140.03
Ticker symbol? (:q:) to quit goog
Unknown ticker symbol. Try again please
Ticker symbol? (:q:) to quit aapl
Current price for  AAPL  is $224.40
Ticker symbol? (:q:) to quit msft
Unknown ticker symbol. Try again please
Ticker symbol? (:q:) to quit nke
Current price for  NKE  is $92.85
Ticker symbol? (:q:) to quit :q:
```

SETS

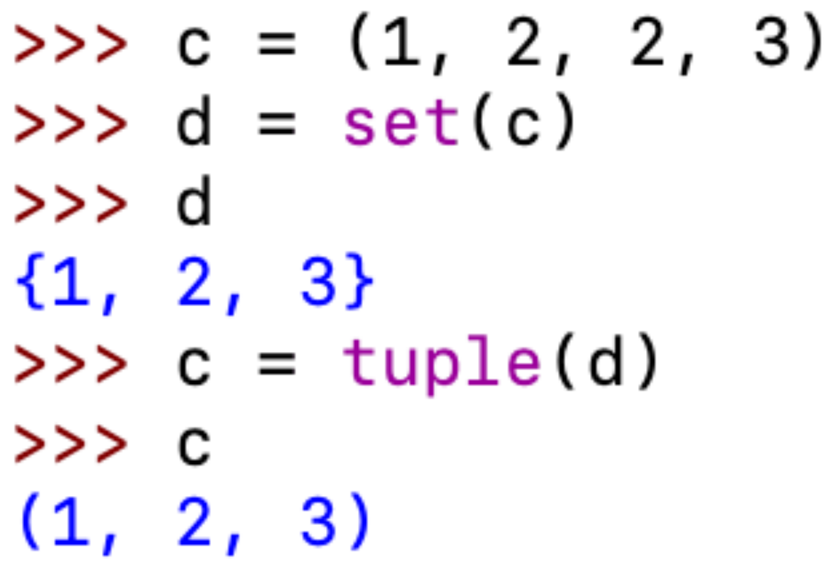
- A set is an unordered collection of unique values
    - They are not sequences so we cannot use [ ]
- Can contain only immutable elements
    - Strings, ints, floats, tuples that contain immutables
- Do not support slicing



```
>>> basket = {'apple', 'orange', 'apple', 'pear', 'orange', 'banana'}
>>> basket
{'orange', 'apple', 'banana', 'pear'}
>>> 'blueberry' in basket
False
>>> 'orange' in basket
True
>>> shelf = {'apple', 'pear'}
>>> basket == shelf
False
>>> basket - shelf
{'orange', 'banana'}
>>> shelf - basket
set()
>>> basket | shelf
{'orange', 'banana', 'apple', 'pear'}
>>> basket & shelf
{'apple', 'pear'}
>>>
```

SETS

- We can convert from Sets to Lists to Tuples and back
- Useful for when we have a collection containing duplicates that we want to remove quickly and easily



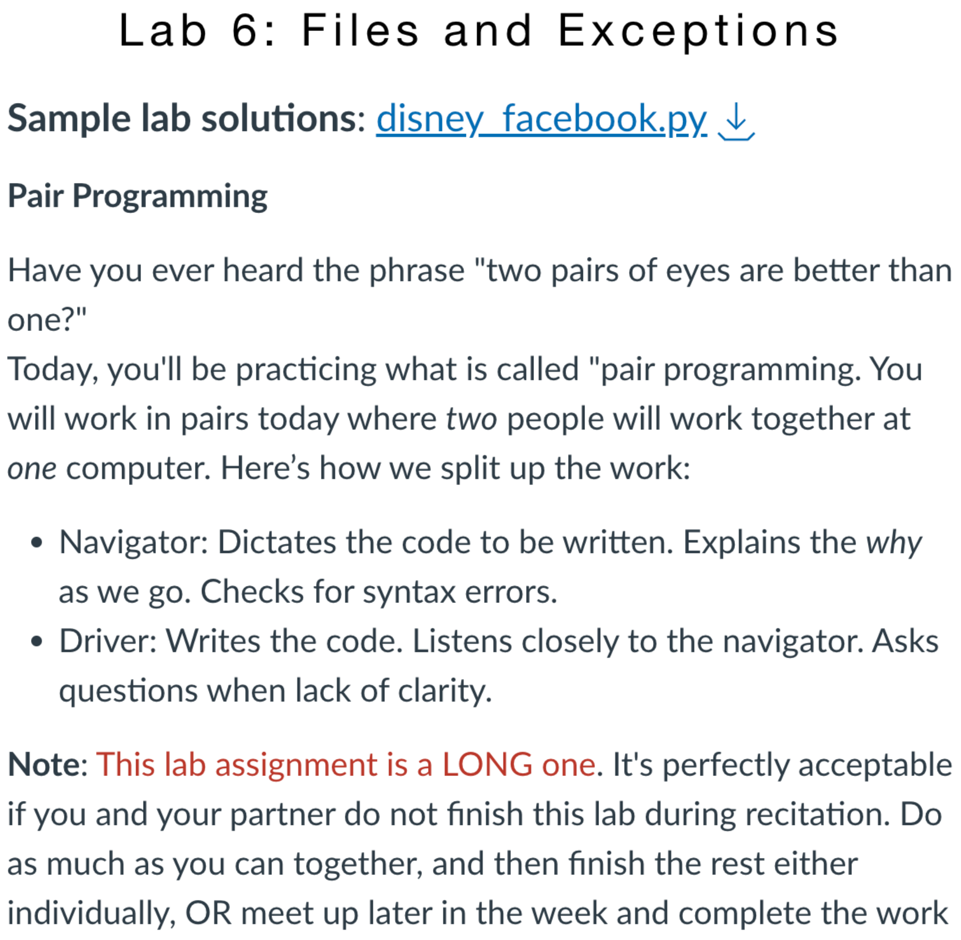
```
>>> c = (1, 2, 2, 3)
>>> d = set(c)
>>> d
{1, 2, 3}
>>> c = tuple(d)
>>> c
(1, 2, 3)
```

LAUNCH INTO LAB

- This week's lab focuses on Files & Exception Handling
- It's a LONG lab – it is perfectly fine if you do not finish it in one sitting during recitation. Do what you can, and continue working on it at home
- Disney Facebook

What do you need to know & remember?

What do you need to do?



Lab 6: Files and Exceptions

Sample lab solutions: disney_facebook.py

Pair Programming

Have you ever heard the phrase "two pairs of eyes are better than one?"
Today, you'll be practicing what is called "pair programming. You will work in pairs today where *two* people will work together at *one* computer. Here's how we split up the work:

- Navigator: Dictates the code to be written. Explains the *why* as we go. Checks for syntax errors.
- Driver: Writes the code. Listens closely to the navigator. Asks questions when lack of clarity.

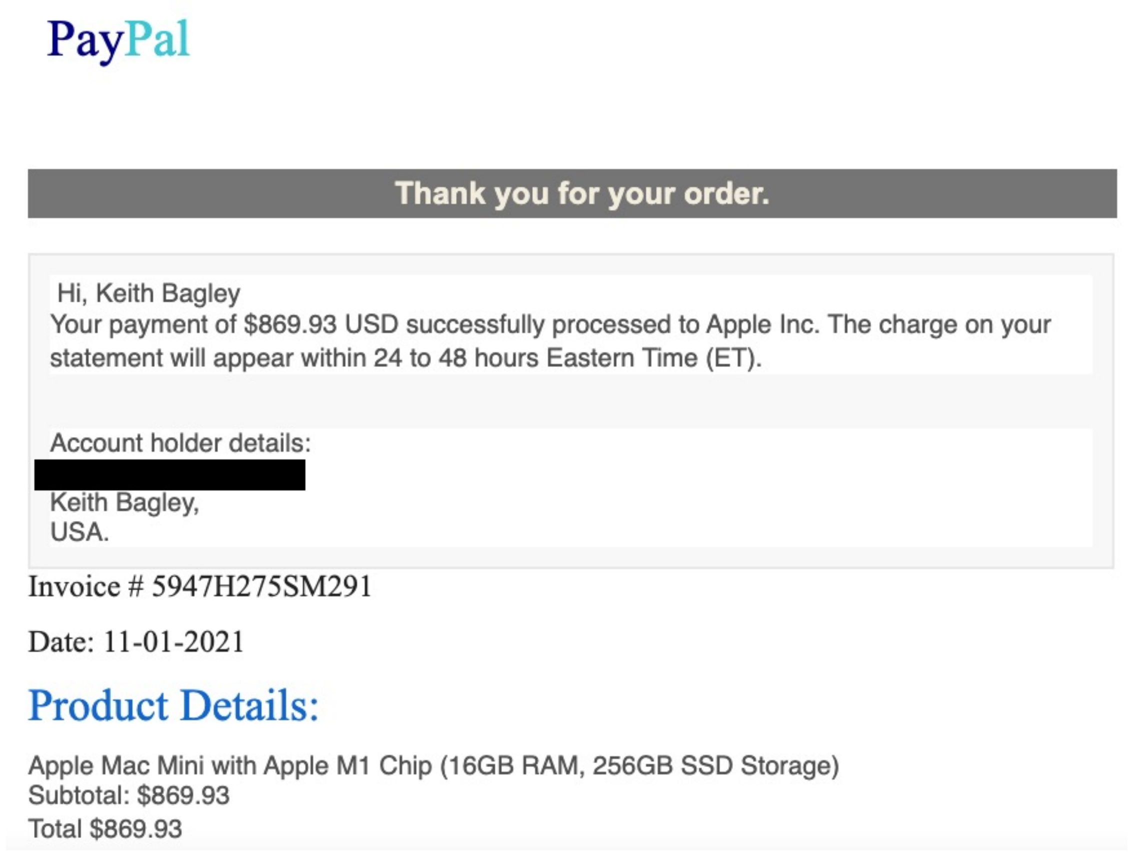
Note: This lab assignment is a LONG one. It's perfectly acceptable if you and your partner do not finish this lab during recitation. Do as much as you can together, and then finish the rest either individually, OR meet up later in the week and complete the work

FACES OF C.S.: WINDOW SNYDER

Mwende Window Snyder is an American-Kenyan technologist who is one of the first computer scientist to specialize in cyber security, and to proactively try to bridge the gap between corporations and the security researchers often termed ”hackers”.
Window has worked for various companies (Microsoft, Square, etc)as a senior security strategist and was a contributor to the Security Design Lifecycle (SDL). She co-developed a new methodology for threat modeling software. She has founded a few companies: Matasano Security, a security services and Thistle Technologies

https://en.wikipedia.org/wiki/Window_Snyder

PROVOKING THOUGHTS FOR THE WEEK: SEE SOMETHING, SAY SOMETHING?



PayPal

Thank you for your order.

Hi, Keith Bagley
Your payment of $869.93 USD successfully processed to Apple Inc. The charge on your statement will appear within 24 to 48 hours Eastern Time (ET).

Account holder details:
[redacted]
Keith Bagley,
USA.

Invoice # 5947H275SM291

Date: 11-01-2021

Product Details:

Apple Mac Mini with Apple M1 Chip (16GB RAM, 256GB SSD Storage)
Subtotal: $869.93
Total $869.93

It's easy & safe to delete/ignore spam and Phishing emails like the one I received this week.

Are we – as astute technologists ethically bound to do more? Report & escalate? Help warn?

Q & A

- Questions?

THANKS!

- Stay safe, be encouraged, & see you next week!