Strings

- **Video:** Basic Structures Introduction — 1 min
- **Video:** What is a string? — 2 min
- **Video:** The Parts of a String — 4 min
- **Reading:** String Indexing and Slicing — 10 min
- **Video:** Creating New Strings — 6 min
- **Reading:** Basic String Methods — 10 min
- **Video:** More String Methods — 4 min
- **Reading:** Advanced String Methods — 10 min
- **Video:** Formatting Strings — 5 min
- **Reading:** String Formatting — 10 min
- **Reading:** String Reference Cheat Sheet — 10 min
- **Reading:** Formatting Strings Cheat Sheet — 10 min
- **Practice Quiz:** Practice Quiz: Strings — 5 questions

Lists

- **Video:** What is a list? — 4 min
- **Reading:** Lists Defined — 10 min
- **Video:** Modifying the Contents of a List — 5 min
- **Reading:** Modifying Lists — 10 min
- **Video:** Lists and Tuples — 3 min
- **Reading:** Tuples — 10 min
- **Video:** Iterating over Lists and Tuples — 7 min
- **Reading:** Iterating Over Lists Using Enumerate — 10 min
- **Video:** List Comprehensions — 4 min
- **Reading:** List Comprehensions — 10 min
- **Reading:** Lists and Tuples Operations Cheat Sheet — 10 min
- **Practice Quiz:** Practice Quiz: Lists — 6 questions

Dictionaries

- **Video:** What is a dictionary? — 5 min
- **Reading:** Dictionaries Defined — 10 min
- **Video:** Iterating over the Contents of a Dictionary — 4 min
- **Reading:** Iterating Over Dictionaries — 10 min
- **Video:** Dictionaries vs. Lists — 3 min
- **Reading:** Dictionary Methods Cheat Sheet — 10 min
- **Practice Quiz:** Practice Quiz: Dictionaries — 5 questions

Module Review

- **Video:** Basic Structures Wrap Up — 1 min
- **Video:** In Marga's Words: My Most Challenging Script — 1 min
- **Quiz:** Module 4 Graded Assessment — 10 questions
- **Discussion Prompt:** Discussion Prompt — 5 min

PRACTICE QUIZ • 25 MIN

Congratulations! You passed!

TO PASS 80% or higher

Keep Learning

GRADE 100%

Practice Quiz: Dictionaries

Practice Quiz: Dictionaries

TOTAL POINTS 5

Submit your assignment

Receive grade

TO PASS 80% or higher

Try again

Grade 100%

View Feedback

We keep your highest score

1. The email_list function receives a dictionary, which contains domain names as keys, and a list of users as values. Fill in the blanks to generate a list that contains complete email addresses (e.g. diana.prince@gmail.com). **1 / 1 point**

```
def email_list(domains):
    emails = []
    for domain,users in domains.items():
        for user in users:
                emails.append(user+'@'+domain)
    return(emails)
print(email_list({"gmail.com": ["clark.kent", "diana.prince", "peter.parker"], "yahoo.com": ["barbara.gordon", "jean.grey"], "hotmail.com": ["bruce.wayne"]}))
```

Run

Reset

Correct

Well done! You've created quite an email list!

2. The groups_per_user function receives a dictionary, which contains group names with the list of users. Users can belong to multiple groups. Fill in the blanks to return a dictionary with the users as keys and a list of their groups as values. **1 / 1 point**

```
def groups_per_user(group_dictionary):
    user_groups = {}
    # Go through group_dictionary
    for group in group_dictionary.keys():
        # Now go through the users in the group
        for users in group_dictionary[group]:
          lst = []
          for group in group_dictionary.keys():
            if users in group_dictionary[group] and users not in lst:
              lst.append(group)
          user_groups[users] = lst
    return user_groups
            # Now add the group to the the list of
# groups for this user, creating the entry
# in the dictionary if necessary


print(groups_per_user({"local": ["admin", "userA"],
        "public":  ["admin", "userB"],
        "administrator": ["admin"] }))
```

Run

Reset

Correct

Well done, you! You're now creating dictionaries out of other dictionaries!

3. The dict.update method updates one dictionary with the items coming from the other dictionary, so that existing entries are replaced and new entries are added. What is the content of the dictionary “wardrobe“ at the end of the following code? **1 / 1 point**

```
wardrobe = {'shirt': ['red', 'blue', 'white'], 'jeans': ['blue', 'black']}
new_items = {'jeans': ['white'], 'scarf': ['yellow'], 'socks': ['black', 'brown']}
wardrobe.update(new_items)
```

- ○ {'jeans': ['white'], 'scarf': ['yellow'], 'socks': ['black', 'brown']}
- ◉ {'shirt': ['red', 'blue', 'white'], 'jeans': ['white'], 'scarf': ['yellow'], 'socks': ['black', 'brown']}
- ○ {'shirt': ['red', 'blue', 'white'], 'jeans': ['blue', 'black', 'white'], 'scarf': ['yellow'], 'socks': ['black', 'brown']}
- ○ {'shirt': ['red', 'blue', 'white'], 'jeans': ['blue', 'black'], 'jeans': ['white'], 'scarf': ['yellow'], 'socks': ['black', 'brown']}

Correct

Correct! The dict.update method updates the dictionary (wardrobe) with the items coming from the other dictionary (new_items), adding new entries and replacing existing entries.

4. What’s a major advantage of using dictionaries over lists? **1 / 1 point**

- ○ Dictionaries are ordered sets
- ○ Dictionaries can be accessed by the index number of the element
- ○ Elements can be removed and inserted into dictionaries
- ◉ It’s quicker and easier to find a specific element in a dictionary

Correct

Right on! Because of their unordered nature and use of key value pairs, searching a dictionary takes the same amount of time no matter how many elements it contains

5. The add_prices function returns the total price of all of the groceries in the dictionary. Fill in the blanks to complete this function. **1 / 1 point**

```
def add_prices(basket):
    # Initialize the variable that will be used for the calculation
    total = 0
    # Iterate through the dictionary items
    for fruits, prices in basket.items():
        # Add each price to the total calculation
        # Hint: how do you access the values of
        # dictionary items?
        total += basket.get(fruits)
    # Limit the return value to 2 decimal places
    return round(total, 2)

groceries = {"bananas": 1.56, "apples": 2.50, "oranges": 0.99, "bread": 4.59,
    "coffee": 6.99, "milk": 3.39, "eggs": 2.98, "cheese": 5.44}
print(add_prices(groceries)) # Should print 28.44

```

Run

Reset

Correct

Nicely done! Dictionaries are a helpful way to store information, and access it easily when it's needed.