Practice Quiz: Dictionaries

Total points 5

Question 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"]}))
```

['clark.kent@gmail.com', 'diana.prince@gmail.com', 'peter.parker@gmail.com', 'barbara.gordon@yahoo.com', 'jean.grey@yahoo.com', 'bruce.wayne@hotmail.com']

Correct

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

Question 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, users in group_dictionary.items():
        # Now go through the users in the group
        for user in users:
            # Now add the group to the the list of
            # groups for this user, creating the entry
            # in the dictionary if necessary
            if user not in user_groups.keys():
                user_groups[user] = []
            user_groups[user].append(group)
    return(user_groups)
print(groups_per_user({"local": ["admin", "userA"],
                       "public": ["admin", "userB"],
                       "administrator": ["admin"] }))
```

{'admin': ['local', 'public', 'administrator'], 'userA': ['local'], 'userB': ['public']}

Correct

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

Question 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.

Question 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

Question 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 item in basket:
        # Add each price to the total calculation
        # Hint: how do you access the values of
        # dictionary items?
        total += basket[item]
    # 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
```

28.44

Correct

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