Practice Quiz: Dictionaries

TOTAL POINTS 5

---

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

```
1   def email_list(domains):
2       emails = []
3       for email,users in domains.items():
4         for user in users:
5             emails.append(user+'@'+email)
6       return(emails)
7
8   print(email_list({"gmail.com": ["clark.kent", "diana.prince", "peter.parker"], "yahoo.com":
```

Run

Reset

```
['clark.kent@gmail.com', 'diana.prince@gmail.com', 'peter.parker@gmail.com', 'barbara.gordon@ya
```
◄ ░░░░░░░░░░░░░░░░░░░░░░░░░░░░   ►

✓ **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

```
1    def groups_per_user(group_dictionary):
2        user_groups = {}
3        # Go through group_dictionary
4        for group, users in group_dictionary.items():
5            # Now go through the users in the group
6            for user in users:
7                if user not in user_groups:
8                    user_groups[user] = []
9                    user_groups[user].append(group)# Now add the group to the the list of
10   # groups for this user, creating the entry
11   # in the dictionary if necessary
12
13       return(user_groups)
14
15   print(groups_per_user({"local": ["admin", "userA"],
16           "public":  ["admin", "userB"],
17           "administrator": ["admin"] }))
```

Run

Reset

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

✓ **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

```
1    wardrobe = {'shirt': ['red', 'blue', 'white'], 'jeans': ['blue', 'black']}
2    new_items = {'jeans': ['white'], 'scarf': ['yellow'], 'socks': ['black', 'brown']}
3    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   **1 / 1 point**
    in the blanks to complete this function.

```
1    def add_prices(basket):
2        # Initialize the variable that will be used for the calculation
3        total = 0
4        # Iterate through the dictionary items
5        for item in basket.values():
6
7            # Add each price to the total calculation
8            # Hint: how do you access the values of
9            # dictionary items?
10           total += item
11       # Limit the return value to 2 decimal places
12       return round(total, 2)
13
14   groceries = {"bananas": 1.56, "apples": 2.50, "oranges": 0.99, "bread": 4.59,
15       "coffee": 6.99, "milk": 3.39, "eggs": 2.98, "cheese": 5.44}
16
17   print(add_prices(groceries)) # Should print 28.44
18
```

Run

Reset

28.44

✓ **Correct**

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