¡Felicitaciones! ¡Aprobaste!

Calificación recibida 100 % **Para Aprobar** 80 % o más

Ir al siguiente elemento

Estás viendo una versión traducida automáticamente de esta evaluación
Puedes volver a ver el contenido en su idioma original si lo prefieres. No perderás el progreso que hayas conseguido si cambias el idioma.
Mostrar la versión en Inglés

Desestimar

1. La función email_list recibe un diccionario, que contiene nombres de dominio como claves, y una lista de usuarios como valores. Rellene los espacios en blanco para generar una lista que contenga direcciones de correo electrónico completas (por ejemplo, diana.prince@gmail.com). **1 / 1 punto**

```
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"
```

Ejecutar
Restablecer

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

Correcto
¡Bien hecho! ¡Ha creado toda una lista de correo electrónico!

2. La función groups_per_user recibe un diccionario, que contiene los nombres de los grupos con la lista de usuarios. Los usuarios pueden pertenecer a varios grupos. Rellena los espacios en blanco para devolver un diccionario con los usuarios como claves y una lista de sus grupos como valores. **1 / 1 punto**

```
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
            if user not in user_groups:
                user_groups[user] = []
# groups for this user, creating the entry
# in the dictionary if necessary
            user_groups[user].append(group)

    return(user_groups)

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

Ejecutar
Restablecer

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

Correcto
¡Bien hecho! Ahora está creando diccionarios a partir de
¡otros diccionarios!

3. El método dict.update actualiza un diccionario con los elementos procedentes del otro diccionario, de forma que se sustituyen las entradas existentes y se añaden otras nuevas. ¿Cuál es el contenido del diccionario "armario" al final del siguiente código? **1 / 1 punto**

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

- ○ {'vaqueros': ['blanco'], 'bufanda': ['amarillo'], 'calcetines': ['negro', 'marrón']}
- ◉ {'camisa': ['rojo', 'azul', 'blanco'], 'vaqueros': ['blanco'], 'bufanda': ['amarillo'], 'calcetines': ['negro', 'marrón']}
- ○ {'camisa': ['rojo', 'azul', 'blanco'], 'vaqueros': ['azul', 'negro', 'blanco'], 'bufanda': ['amarillo'], 'calcetines': ['negro', 'marrón']}
- ○ {'camisa': ['rojo', 'azul', 'blanco'], 'vaqueros': ['azul', 'negro'], 'vaqueros': ['blanco'], 'bufanda': ['amarillo'], 'calcetines': ['negro', 'marrón']}

Correcto
Correcto El método dict.update actualiza el diccionario (armario) con los elementos procedentes del otro diccionario (nuevos_elementos), añadiendo nuevas entradas y sustituyendo las existentes.

4. ¿Cuál es la principal ventaja de utilizar diccionarios frente a listas? **1 / 1 punto**

- ○ los diccionarios son conjuntos ordenados
- ○ se puede acceder a los diccionarios por el número de índice del elemento
- ○ Los elementos pueden eliminarse e insertarse en los diccionarios
- ◉ Es más rápido y fácil encontrar un elemento específico en un diccionario

Correcto
Así es Debido a su naturaleza desordenada y al uso de pares clave-valor, la búsqueda en un diccionario lleva la misma cantidad de tiempo independientemente del número de elementos que contenga

5. La función sumar_precios devuelve el precio total de todos los comestibles del diccionario. Rellene los espacios en blanco para completar esta función. **1 / 1 punto**

```
def add_prices(basket):
    # Initialize the variable that will be used for the calculation
    total = 0
    # Iterate through the dictionary items
    for item, price in basket.items():
        # Add each price to the total calculation
        # Hint: how do you access the values of
        # dictionary items?
        total += price
    # 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
```

Ejecutar
Restablecer

```
28.44
```

Correcto
¡Bien hecho! Los diccionarios son una forma útil de almacenar
información y acceder a ella fácilmente cuando se necesita.