Study Guide: Week 4 Graded Quiz

It is time to prepare for the Week 4 Graded Quiz. Please review the following items from this module before beginning the quiz. If you would like to refresh your memory on these materials, please also revisit the Study Guides located before each practice quiz in Week 4: Study Guide: Strings, Study Guide: Lists Operations and Methods, and Study Guide: Dictionary Methods.

Knowledge

- How to output a list of the keys in a Python dictionary.
- How to determine the output of a string index range used on a string.
- Determine what a list should contain after the .insert() method is used on the list.
- How to replace a specific word in a sentence with the same word in all uppercase letters.
- How to use a dictionary to count the frequency of letters in a string.

Operations, Methods, and Functions

- **String Methods, Operations, and Functions**
  - .upper()
  - .lower()
  - .split()
  - .format()
  - .isnumeric()
  - .isalpha()
  - .replace()
  - string index [ ]
  - len()
- **List Operations and Methods**
  - .reverse()
  - .extend()
  - .insert()
  - .append()
  - .remove()
  - .sort()
  - list comprehensions [ ]
  - list comprehensions [ ] with if condition
- **Dictionary Operations and Methods**
  - .items()
  - .update()
  - .keys()
  - .values()
  - .copy()
  - dictionary[key]
  - dictionary[key] = value

Coding Skills

Skill 1: Using **string** methods

- Separate numerical values from text values in a string using **.split()**.
- Iterate over the elements in a string.
- Test if the element contains letters with **.isalpha()**.
- Assign the elements of the split string to new variables.
- Trim any extra white space using **.strip()**.
- Format a string using **.format()** and **{ }** variable placeholders.

```
def sales_prices(item_and_price):
    # Initialize variables "item" and "price" as strings
    item = ""
    price = ""
    # Create a variable "item_or_price" to hold the result of the split.
    item_or_price = item_and_price.split()

    # For each element "x" in the split variable "item_or_price"
    for x in item_or_price:

        # Check if the element is a number
        if x.isalpha():

            # If true, assign the element to the "item" string variable and add a space
            # for any item names containing multiple words, like "Winter fleece jacket".
            item += x + " "

        # Else, if x is a number (if x.isalpha() is false):
        else:
            # Assign the element to the "price" string variable.
            price = x

    # Strip the extra space to the right of the last "item" word
    item = item.strip()

    # Return the item name and price formatted in a sentence
    return "{} are on sale for ${}".format(item,price)


# Call to the function
print(sales_prices("Winter fleece jackets 49.99"))
# Should print "Winter fleece jackets are on sale for $49.99"
```

Run

Reset

- Use the len() function to measure a string.

```
# This function accepts a string variable "data_field".
def count_words(data_field):

    # Splits the string into individual words.
    split_data = data_field.split()

    # Then returns the number of words in the string using the len()
    # function.
    return len(split_data)

    # Note that it is possible to combine the len() function and the
    # .split() method into the same line of code by inserting the
    # data_field.split() command into the the len() function parameters.

# Call to the function
count_words("Catalog item 3523: Organic raw pumpkin seeds in shell")
# Should print 9
```

Run

Reset

Skill 2: Using **list** methods

- Reverse the order of a list using the **.reverse()** method.
- Combine two lists using the **.extend()** method.

```
# This function accepts two variables, each containing a list of years.
# A current "recent_first" list contains [2022, 2018, 2011, 2006].
# An older "recent_last" list contains [1989, 1992, 1997, 2001].
# The lists need to be combined with the years in chronological order.
def record_profit_years(recent_first, recent_last):

    # Reverse the order of the "recent_first" list so that it is in
    # chronological order.
    recent_first.reverse()

    # Extend the "recent_last" list by appending the newly reversed
    # "recent_first" list.
    recent_last.extend(recent_first)

    # Return the "recent_last", which now contains the two lists
    # combined in chronological order.
    return recent_last

# Assign the two lists to the two variables to be passed to the
# record_profit_years() function.
recent_first = [2022, 2018, 2011, 2006]
recent_last = [1989, 1992, 1997, 2001]



# Call the record_profit_years() function and pass the two lists as
# parameters.
print(record_profit_years(recent_first, recent_last))
# Should print [1989, 1992, 1997, 2001, 2006, 2011, 2018, 2022]
```

Run

Reset

Skill 3: Using a **list comprehension**

- Use a list comprehension [ ] as a shortcut for creating a new list from a range.
- Include a calculation with a **for** loop **in** a **range** with 2 parameters (lower, upper+1).

```
# The function accepts two parameters: a start year and an end year.
def list_years(start, end):

    # It returns a list comprehension that creates a list of years in a for
    # loop using a range from the start year to the end year (inclusive of
    # the upper range year, using end+1).
    return [year for year in range(start, end+1)]


```