Technical Case Study

Coding Skills – Round 1

Solution by : Agrani Mishra

1. Given the list of basket values, do the following:

a. Print out whether each basket is small (basket value < £5), medium (£5 ≤ basket value < £10) or large (basket value ≥ £10). b. Sum and print the value of the medium value baskets. basket_values = [3.43,9.73,7.56,9.52,15.23,2.25,6.44,7.38]

In [73]:
```
basket_values = [3.43, 9.73, 7.56, 9.52, 15.23, 2.25, 6.44, 7.38]

#Define a function to get the list of medium basket values
def basket_size(basket):
    baskets = []  #List to store the medium value baskets
    for value in basket_values:
        if value < 5:
            print(f"Small: £{value}")
        elif value >= 5 and value < 10:
            print(f"Medium: £{value}")
            baskets.append(value)
        else:
            print(f"Large: £{value}")
    return baskets

medium_baskets = basket_size(basket)
total_medium_baskets = sum(medium_baskets) #Total value of medium value basket
s
print(f"\nTotal value of medium value baskets: £{total_medium_baskets}")
```

```
Small: £3.43
Medium: £9.73
Medium: £7.56
Medium: £9.52
Large: £15.23
Small: £2.25
Medium: £6.44
Medium: £7.38

Total value of medium value baskets: £40.63
```

2. You are given the following nested dictionaries, which represent items in a basket. Do the following:

a. Return the product name for item 7527. b. Return the total value of this basket. c. Add another entry for a product that costs £4.95, has ID 7524 and name ‘poppy seeds’. basket = {'2624': {'price': 0.5, 'prod_name': 'salt'}, '2894': {'price': 3.25, 'prod_name': 'yeast'}, '7527': {'price': 2.5, 'prod_name': 'flour'}}

In [70]:
```
basket = {'2624': {'price': 0.5, 'prod_name': 'salt'},
    '2894': {'price': 3.25, 'prod_name': 'yeast'},
    '7527': {'price': 2.5, 'prod_name': 'flour'}
}
# a. Return the product name for item 7527.
item_7527_name = basket['7527']['prod_name'] #Can also be solved using get met
hod
print("Product name for item 7527:", item_7527_name)

# b. Return the total value of this basket.
total_value = 0
for item in basket.values():
    total_value = total_value + item['price']
print("Total value of the basket:", total_value)

# c. Add another entry for a product that costs £4.95, has ID 7524 and name 'p
oppy seeds'.
new_product = {'price': 4.95, 'prod_name': 'poppy seeds'}
basket['7524'] = new_product

# Print the updated basket
print("\nUpdated basket:")
print(basket)
```

```
Product name for item 7527: flour
Total value of the basket: 6.25

Updated basket:
{'2624': {'price': 0.5, 'prod_name': 'salt'}, '2894': {'price': 3.25, 'prod_n
ame': 'yeast'}, '7527': {'price': 2.5, 'prod_name': 'flour'}, '7524': {'pric
e': 4.95, 'prod_name': 'poppy seeds'}}
```

3.Below is the source code for a function called ‘get_sql_string’.

1 def get_sql_string(stores):

2 store_names = [x.split(', ')[0] for x in stores]

3 store*names = [x.replace(' ',* '') for x in store_names]

4 store_regions = [x.split(',')[1] for x in stores]

5 locations = store_names + store_regions

6 columns = ['sales_' + x.lower() for x in locations]

7 return ', '.join(columns)

a. There is a bug in line 4. What should the line be?

b. Assuming this bug was fixed, what would be returned if the following command was executed:

my_stores = ['Fulham Palace Rd, Hammersmith', 'Crown St, Reading', 'Leavesden Green, Watford']

get_sql_string(my_stores)

In [54]:
```
my_stores = ['Fulham Palace Rd, Hammersmith', 'Crown St, Reading',
'Leavesden Green, Watford']

def get_sql_string(stores):
    store_names = [x.split(', ')[0] for x in stores]
    store_names = [x.replace(' ', '_') for x in store_names]
    store_regions = [x.split(',')[1] for x in stores]
    locations = store_names + store_regions
    columns = ['sales_' + x.lower() for x in locations]
    return ', '.join(columns)

print("Output with code Error in Line 4")
get_sql_string(my_stores)
```

```
Output with code Error in Line 4
```

Out[54]:
```
'sales_fulham_palace_rd, sales_crown_st, sales_leavesden_green, sales_ hammer
smith, sales_ reading, sales_ watford'
```

Explanation and code change

As seen from the above Out[49], store_region is getting split without the space which means the split function should use a comma followed by a space ', ' instead of just a comma ',' to correctly split the store names and regions. The corrected line 4 should be:

store_regions = [x.split(', ')[1] for x in stores]

b. Assuming this bug was fixed, what would be returned if the following command was executed:

In [56]:
```
my_stores = ['Fulham Palace Rd, Hammersmith', 'Crown St, Reading',
'Leavesden Green, Watford']

def get_sql_string(stores):
    store_names = [x.split(', ')[0] for x in stores]
    store_names = [x.replace(' ', '_') for x in store_names]
    store_regions = [x.split(', ')[1] for x in stores]
    locations = store_names + store_regions
    columns = ['sales_' + x.lower() for x in locations]
    return ', '.join(columns)

print("Output without code Error in Line 4")
get_sql_string(my_stores)
```

```
Output without code Error in Line 4
```

Out[56]:
```
'sales_fulham_palace_rd, sales_crown_st, sales_leavesden_green, sales_hammers
mith, sales_reading, sales_watford'
```

4. Write a function that:

a. accepts a list of strings as input. b. returns an alphabetically ordered list of unique strings. c. prints the string(s) with maximum length in the console.

In [67]:
```
#a. accepts a list of strings as input.

def unique_strings(input_list):
    unique_strings = list(set(input_list)) #Convert the list to a set to remov
e duplicates

#b. returns an alphabetically ordered list of unique strings
    unique_strings.sort()  #Sort the unique strings

#c. prints the string(s) with maximum length in the console.
    max_str_length = max(len(string) for string in unique_strings) #Find the m
ax length

    longest_strings = []
    for string in unique_strings:
        if len(string) == max_str_length: #Find the strings with the maximum l
ength
            longest_strings.append(string)
    print("String with maximum length:", longest_strings)

    return unique_strings


input_list = ["Selena", "Justin", "Drake", "Dua", "Dua", "Weekend", "Enrique"]
result = unique_strings(input_list)
print("Alphabetically ordered list of unique strings:", result)
```

```
String with maximum length: ['Enrique', 'Weekend']
Alphabetically ordered list of unique strings: ['Drake', 'Dua', 'Enrique', 'J
ustin', 'Selena', 'Weekend']
```