Name : SK Shivaanee
Register ID : 2310257
Section : S02

Assignment 9: Programs using Tuples

Aim: You will be able to apply computational thinking to devise solutions using Tuples, Zip (packing and unpacking sequences), Tuple to list conversions, List of tuples. ● You code Python programs by defining the functions to handle Tuples.

Solve the following problems using Lists/Tuples in Python (CO4, K3):

Question 1: Consider the following list of countries and their capitals. Canada <--> Ottawa China <--> Beijing England <--> London France <--> Paris Germany <--> Berlin India <--> New Delhi Israel <--> Jerusalem Italy <--> Rome Japan <--> Tokyo Mexico <--> Mexico City Russia <--> Moscow United States <--> Washington Store the above data in a tuple data structure. Accept the name of a country as input and display the corresponding capital and accept the name of a capital as input and display the corresponding country. Design the program so that it executes repeatedly, until the word End is entered as input.

Source Code:

```
# Tuple to store countries and their capitals
countries_and_capitals = (
    ("Canada", "Ottawa"),
    ("China", "Beijing"),
    ("England", "London"),
    ("France", "Paris"),
    ("Germany", "Berlin"),
    ("India", "New Delhi"),
    ("Israel", "Jerusalem"),
    ("Italy", "Rome"),
    ("Japan", "Tokyo"),
    ("Mexico", "Mexico City"),
    ("Russia", "Moscow"),
    ("United States", "Washington")
)

# Function to find the corresponding capital or country
def find_corresponding_info(input_text):
    for country, capital in countries_and_capitals:
        if input_text.lower() == country.lower():
            return f"The capital is: {capital}"
        elif input_text.lower() == capital.lower():
            return f"The country is: {country}"
    return "Not found in the data."
```

```
# Main program loop
while True:
    user_input = input("Enter your input (or 'End' to stop): ")

    if user_input.lower() == "end":
        break
    result = find_corresponding_info(user_input)
    print(result)
```

Output:

```
Enter your input (or 'End' to stop): France
The capital is: Paris
Enter your input (or 'End' to stop): End
>
```

Question 2: Some properties of the first few chemical elements in the periodic table are given below. Melting and boiling points are determined at atmospheric pressure.

Element	Symbol	Atomic Number	Melting point (K)	Boiling point (K)
Hydrogen	H	1	14	20
Helium	He	2	1	4
Lithium	Li	3	453	1603
Beryllium	Be	4	1560	2742
Boron	B	5	2349	4200
Carbon	C	6	3915	3915
Nitrogen	N	7	63	77
Oxygen	O	8	54	90
Fluorine	F	9	53	85
Neon	Ne	10	25	27

Create a tuple data structure to store the above information as shown below: Element=(H, He, Li, …..) Melting=(14,1,453,…..) Boiling=(20,4,1603,……) Then write a function element_state(element, temperature) which allows the user to specify an element and a temperature, and which reports whether the element is solid, liquid or gas at that point. Hint: Use Zip(), Zip(*) to pack and unpack the sequences.

Source Code:

```
# Tuple data structure to store information about chemical elements
elements_data = (
    ("H", "He", "Li", "Be", "B", "C", "N", "O", "F", "Ne"),
    (14, 1, 453, 1560, 2349, 3915, 63, 54, 53, 25),
    (20, 4, 1603, 2742, 4200, 3915, 77, 90, 85, 27)
)

def element_state(element, temperature):
    # Unpack the data using zip
    elements, melting_points, boiling_points = elements_data

    # Find the index of the specified element
    try:
        index = elements.index(element)
    except ValueError:
        return "Element not found."

    # Check the state at the given temperature
    melting_point = melting_points[index]
    boiling_point = boiling_points[index]

    if temperature < melting_point:
        return "Solid"
    elif melting_point <= temperature < boiling_point:
        return "Liquid"
    else:
        return "Gas"

# Example usage
element_input = input("Enter the element symbol: ")
temperature_input = float(input("Enter the temperature in Kelvin: "))

result = element_state(element_input, temperature_input)
print(f"At {temperature_input} K, {element_input} is in the {result} state.")
```

Output:

```
Enter the element symbol: C
Enter the temperature in Kelvin: 100
At 100.0 K, C is in the Solid state.
>
```

Question 3: Consider the super market selling fruits. The available fruits for sales and its corresponding unit price are stored in list/tuple sequences. Allow a customer to enter the quantity of purchase for each of fruit and compute the total cost as explained below: a. Create a list of fruits and its unit price / kg in tuples as shown below: Ex: fruits=['Orange', 'Mango',

'Apple', 'Grape', 'Papaya'] price=(60, 80, 220, 80, 90) b. Create a list containing the quantity of purchase for each fruit. (float in quantity represents grams) Ex: purchase_qty=[2, 0, 1, 0, 0.500] c. Compute the total price of each fruit and store it as shown below: Ex: purchase_total=[(2, 120), (1, 220), (0.500,45)] d. Now sum the total amount and store the purchase list as below: Ex: purchase_receipt=[('Orange',60, 2, 120), ('Apple', 220, 1, 220), ('Papaya', 90, 0.500,45), ('Total Amt', 385)] e. Print the purchase receipt as shown below: Fruit Price Qty Total Price Orange 60 2 120 Apple 220 1 220 Papaya 90 0.5 45 Total Amt 385 Hint: Use Zip(), Zip(*) to pack and unpack the sequences.

Source Code:

```
# a. Create a list of fruits and its unit price / kg in tuples
fruits = ['Orange', 'Mango', 'Apple', 'Grape', 'Papaya']
price = (60, 80, 220, 80, 90)

# b. Create a list containing the quantity of purchase for each fruit
purchase_qty = [2, 0, 1, 0, 0.5]

# c. Compute the total price of each fruit
purchase_total = [(qty, qty * unit_price) for qty, unit_price in zip(purchase_qty,
    price)]

# d. Sum the total amount and store the purchase list
total_amount = sum([total_price for _, total_price in purchase_total])
purchase_receipt = list(zip(fruits, price, purchase_qty, [total_price for _,
    total_price in purchase_total]))
purchase_receipt.append(('Total Amt', total_amount))

# e. Print the purchase receipt
print(f"{'Fruit':<10}{'Price':<8}{'Qty':<5}{'Total Price':<12}")
for item in purchase_receipt:
    print(f"{item[0]:<10}{item[1]:<8}{item[2]:<5}{item[3]:<12}")
```

Output:

```
Fruit     Price   Qty  Total Price
Orange    60      2    120
Mango     80      0    0
Apple     220     1    220
Grape     80      0    0           ERROR!
Papaya    90      0.5  45.0
Traceback (most recent call last):
  File "<string>", line 19, in <module>
IndexError: tuple index out of range
```

Additional problems using Lists for practice:

Question 1: **Consider a tuple as T = (1, 3, 2, 4, 6, 5). Write a program to store numbers present at odd index into a new tuple.**

Source Code:

```
# Original tuple
T = (1, 3, 2, 4, 6, 5)

# Extract numbers at odd indices
odd_index_numbers = T[1::2]

# Display the result
print("Original Tuple:", T)
print("Numbers at Odd Indices:", odd_index_numbers)
```

Output:

```
Original Tuple: (1, 3, 2, 4, 6, 5)
Numbers at Odd Indices: (3, 4, 5)
>
```

Question 2: **Consider the mark_list=[(120,55), (121,94), (122,73), (123,88), (124, 62)] which contains the register number and mark of corresponding student as list of tuples. Create a new tuple that assigns a grade based on the following conditions: if Marks >=90 then grade A if Marks >=80 && 65 && < 80 then grade C if Marks > =40 && <=65 then grade D if Marks**

Source code:

```
mark_list = [(120, 55), (121, 94), (122, 73), (123, 88), (124, 62)]

def get_grade(mark):
    if mark >= 90:
        return 'A'
    elif mark >= 80:
        return 'B'
    elif mark >= 65:
        return 'C'
    elif mark >= 40:
        return 'D'
    else:
        return 'F'

# Create a new tuple with register number, mark, and grade
graded_list = [(reg_num, mark, get_grade(mark)) for reg_num, mark in mark_list]

# Display the result
print("Original Mark List:", mark_list)
print("Graded List:", graded_list)
```

Output:

```
Original Mark List: [(120, 55), (121, 94), (122, 73), (123, 88), (124, 62)]
Graded List: [(120, 55, 'D'), (121, 94, 'A'), (122, 73, 'C'), (123, 88, 'B'), (124, 62,
    'D')]
```

Question 3:Given list of tuples, remove all the tuples with length K. Input : test_list = [(4, 5), (4,), (8, 6, 7), (1,), (3, 4, 6, 7)], K = 2 Output : [(4,), (8, 6, 7), (1,), (3, 4, 6, 7)] Input : test_list = [(4, 5), (4,), (8, 6, 7), (1,), (3, 4, 6, 7)], K = 3 Output : [(4, 5), (4,), (1,), (3, 4, 6, 7)]

Source Code:

```python
def remove_tuples_by_length(lst, k):
    result = [tpl for tpl in lst if len(tpl) != k]
    return result

# Example usage
test_list = [(4, 5), (4, ), (8, 6, 7), (1, ), (3, 4, 6, 7)]
k_value = 2

# Remove tuples with length K
result_list = remove_tuples_by_length(test_list, k_value)

# Display the result
print("Original List:", test_list)
print(f"Tuples with length {k_value} removed:", result_list)
```

Output:

```
Original List: [(4, 5), (4,), (8, 6, 7), (1,), (3, 4, 6, 7)]
Tuples with length 2 removed: [(4,), (8, 6, 7), (1,), (3, 4, 6, 7)]
```

Question 4: Given a list, find frequency of each element and save it as list of tuples [(number, frequency)]. Input : test_list = [4, 5, 4, 5, 6, 6, 5] Output : [(4, 2), (5, 3), (6, 2)] Input : test_list = [4, 5, 4, 5, 6, 6, 6] Output : [(4, 2), (5, 3), (6, 3)]

Source code:

```python
def element_frequency(lst):
    frequency_dict = {}

    # Count the frequency of each element
    for element in lst:
        frequency_dict[element] = frequency_dict.get(element, 0) + 1

    # Convert the dictionary into a list of tuples
    result = [(element, frequency) for element, frequency in frequency_dict.items
        ()]

    return result
# Example usage
test_list_1 = [4, 5, 4, 5, 6, 6, 5]
test_list_2 = [4, 5, 4, 5, 6, 6, 6]

# Find frequency of each element
result_1 = element_frequency(test_list_1)
result_2 = element_frequency(test_list_2)

# Display the results
print("Input List 1:", test_list_1)
print("Frequency List 1:", result_1)

print("\nInput List 2:", test_list_2)
print("Frequency List 2:", result_2)
```

Output:

```
Input List 1: [4, 5, 4, 5, 6, 6, 5]
Frequency List 1: [(4, 2), (5, 3), (6, 2)]

Input List 2: [4, 5, 4, 5, 6, 6, 6]
Frequency List 2: [(4, 2), (5, 2), (6, 3)]
```

Learning outcome:

1. Reading inputs / Printing the result
2. Using appropriate datatypes for the given input
3. Variable assignment
4. Converting the formula into python expressions

Result: Thus I learned to implement a simple problems in Python and solve the same using Strings.