National University of Sciences & Technology (NUST)
School of Natural Sciences (SNS)
Department of Mathematics

CS250: Data Structures and Algorithms

Class: BS Mathematics - 2021

MEHTAB AMEEM
(376743)

Lab 01:

Data Structures in Python Lists (Mutable Arrays), Tuple (Immutable Arrays), Sets and Dictionaries in Python

Date: 13 September, 2023

Time: 10:00 am - 01:00pm

Instructor: Fauzia Ehsan

Lab Engineer: Mehwish Kiran

Task 01:
1. In a for loop, print out each river's name!
2. In another for loop, add up and print out the total length of all the rivers!
3. Print out every river's name that begins with the letter M !
4. The length of the rivers is in miles. Print out every river's length in kilometres! (1 mile is roughly 1.6 km **+ -**

Task 01 Py code:

```python
# List of dictionaries of pre-defined rivers
rivers = [
    {"name": "Nile", "length": 4157},
    {"name": "Yangtze", "length": 3434},
    {"name": "Murray-Darling", "length": 2310},
    {"name": "Volga", "length": 2290},
    {"name": "Mississippi", "length": 2540},
    {"name": "Amazon", "length": 3915}
]

# Task 1.
# Display the name of each river
for river in rivers:
    print(river["name"])

# Task 2.
# Display the total length of the rivers
total = 0
for river in rivers:
    total += river["length"]
print(f"Total length is {total} miles")

# Task 3.
for river in rivers:
    # Only keep those names that start with "M"
    if river["name"].startswith("M"):
        print(river["name"])

# Task 4.
# Display the lengths in kilometres
for river in rivers:
    print(f"The length of {river['name']} is {river['length']*1.6:.2f} km")
```

Task 01 Output:

```
In [1]: runfile('D:/DSA/Labs/task01.py', wdir='D:/DSA/Labs')
Nile
Yangtze
Murray-Darling
Volga
Mississippi
Amazon
Total length is 18646 miles
Murray-Darling
Mississippi
The length of Nile is 6651.20 km
The length of Yangtze is 5494.40 km
The length of Murray-Darling is 3696.00 km
The length of Volga is 3664.00 km
The length of Mississippi is 4064.00 km
The length of Amazon is 6264.00 km
```


Task 02:

Write the following functions:

overlap() : Given two lists, find a list of the elements common to both lists and return it.

join(): Given two lists, join them together to be one list without duplicate elements and return that list

Task 02 Py code:

```python
def overlap(u, v):
    '''
    Computes the set-theoretic intersection of two lists.

    Parameters
    ----------
    u : list
    v : list

    Returns
    -------
    w : list
        The list obtained by keeping only the common elements
        of the given lists

    '''
    w = [] # Initialise an empty list for later extensions

    # The common elements must exists in u, so loop through u
    # but keep only those elements which also lie in v, without repeats.
    [w.append(x) for x in u if (x in v) and (x not in w)]
    return w
```

```python
def join(u, v):
    '''
    Computes the set-theoretic union of two lists.

    Parameters
    ----------
    u : list
    v : list

    Returns
    -------
    w : list
        The list obtained by keeping all the elements of u and v,
        without any repeats

    '''
    w = [] # Initialise an empty list for later extensions

    # Add all the elements of u (then v) to w, unless they have already been added
    # Don't collect the resulting list
    [w.append(x) for x in u if x not in w]
    [w.append(x) for x in v if x not in w]
    return w


def main():
    # Call each function with the appropriate input(s) to inspect the output(s)
    print(overlap([1.0, 2.5, 4.5], [2.5, 4.0, 5.0]))
    print(overlap([1.0, 2.0, 2.0, 4.5], [2.0, 4.5, 5.0]))

    print(join([1.0, 4.5], [2.0, 4.5, 5.0]))
    print(join([1.0, 2.0, 1.0, 4.5], [2.0, 4.5, 5.0]))
```

```python
if __name__ == '__main__':
    main()
```

Task 02 Output:

```
In [5]: runfile('D:/DSA/Labs/task02.py', wdir='D:/DSA/Labs')
[2.5]
[2.0, 4.5]
[1.0, 4.5, 2.0, 5.0]
[1.0, 2.0, 4.5, 5.0]
```

Task 03:
Download the data_structures_food.py file from LMS and run it in VS code. Your goal is to practice manipulating sequences with the Python tools. In data_structures_food.py , there is a list of dictionaries representing different spicy foods.
spicy_foods = [{ "name": "Green Curry", "cuisine": "Thai", "heat_level": 9, }, { "name": "Buffalo Wings", "cuisine": "American", "heat_level": 3, }, { "name": "Mapo Tofu", "cuisine": "Sichuan", "heat_level": 6, },]
Practice using loops and Python list comprehensions alongside list and dict methods to solve these deliverables.

Task 03 Py code:

```
spicy_foods = [
    {
        "name": "Green Curry",
        "cuisine": "Thai",
        "heat_level": 9,
    },
    {
        "name": "Buffalo Wings",
        "cuisine": "American",
        "heat_level": 3,
    },
    {
        "name": "Mapo Tofu",
        "cuisine": "Sichuan",
        "heat_level": 6,
    },
]


spicy_food = {'name': 'Griot', 'cuisine': 'Haitian', 'heat_level': 10}

def get_names(spicy_foods):
    '''
    Returns a list containing the names of all the food items in the given list

    Parameters
    ----------
    spicy_foods : list
        A list of dictionaries, with keys "name", "cuisine",
        "heat_level". Each dictionary corresponds to a food item.

    Returns
    -------
    x : list
```

```
        A list of strings

    '''
    # Use list comprehension for brevity. Only keep the names of the foods
    return [food["name"] for food in spicy_foods]


def get_spiciest_foods(spicy_foods):
    '''
    Returns a list containing the details of the food items in the given list
    whose heat_level is above 5

    Parameters
    ----------
    spicy_foods : list
        A list of dictionaries, with keys "name", "cuisine",
        "heat_level". Each dictionary corresponds to a food item.

    Returns
    -------
    x : list
        A list of dictionaries. This is a subset of the given list.

    '''
    # Use list comprehension for brevity.
    # Only keep a food item (dictionary) if its heat_level is above 5
    return [food for food in spicy_foods if food["heat_level"] > 5]


def print_spicy_foods(spicy_foods):
    '''
    Displays the details of all the food items in the given list

    Parameters
    ----------
```

```
    spicy_foods : list
        A list of dictionaries, with keys "name", "cuisine",
        "heat_level". Each dictionary corresponds to a food item.

    Returns
    -------
    None

    '''
    for food in spicy_foods:
        # Using f-strings to format the output as required
        print(f'{food["name"]} ({food["cuisine"]}) | {"🌶️" * int(food["heat_level


def get_spicy_food_by_cuisine(spicy_foods, cuisine):
    '''
    Returns a dictionary containing the details of the (first) food item whose
    cuisine value is specified by the user

    Parameters
    ----------
    spicy_foods : list
        A list of dictionaries, with keys "name", "cuisine",
        "heat_level". Each dictionary corresponds to a food item.
    cuisine : string
        The type of cuisine to keep

    Returns
    -------
    x : dictionary
        This is an element of the given list.

    '''
```

```
    for food in spicy_foods:
        # Only keep a food item (dictionary) if its cuisine-type
        # matches the one specified by the user
        if food["cuisine"] == cuisine:
            return food


def print_spiciest_foods(spicy_foods):
    '''
    Displays the details of the food items in the given list
    whose heat_level is above 5

    Parameters
    ----------
    spicy_foods : list
        A list of dictionaries, with keys "name", "cuisine",
        "heat_level". Each dictionary corresponds to a food item.

    Returns
    -------
    None

    '''
    # Use a previous function to loop through only those foods
    # whose heat_level is high enough
    for food in get_spiciest_foods(spicy_foods):
        # Use f-strings to format the output as specified in the brief
        print(f'{food["name"]} ({food["cuisine"]}) | {"🌶️" * int(food["heat_level"])}')
```

```
def get_average_heat_level(spicy_foods):
    '''
    Computes the average heat level of all the food items
    present in the given list.

    Parameters
    ----------
    spicy_foods : list
        A list of dictionaries, with keys "name", "cuisine",
        "heat_level". Each dictionary corresponds to a food item.

    Returns
    -------
    x : float
        The average of all the heat_level values

    '''
    # Initialise the sum of all the heat levels to zero
    total = 0
    for food in spicy_foods:
        total += food["heat_level"]
    # Divide the sum of the heat levels with the number of food items
    # to get the average value
    return total / len(spicy_foods)
```

```python
def create_spicy_food(spicy_foods, spicy_food):
    '''
    Computes the average heat level of all the food items
    present in the given list.

    Parameters
    ----------
    spicy_foods : list
        A list of dictionaries, with keys "name", "cuisine", and
        "heat_level". Each dictionary corresponds to a food item.
    spicy_food : dictionary
        A dictionary with keys "name", "cuisine", and "heat_level",
        corresponding to a new food item.

    Returns
    -------
    spicy_foods : list
        A list of dictionaries. This is the extension of the original list.

    '''
    # Append the new food item (dictionary) at the end of the original list
    spicy_foods.append(spicy_food)
    return spicy_foods
```

```python
def main():
    # Call each function with the appropriate input(s) to inspect the output(s)
    print("Calling all the functions in main")
    print(get_names(spicy_foods))
    print(get_spiciest_foods(spicy_foods))
    print_spicy_foods(spicy_foods)
    print(get_spicy_food_by_cuisine(spicy_foods, "Thai"))
    print_spiciest_foods(spicy_foods)
    print(get_average_heat_level(spicy_foods))
    print(create_spicy_food(spicy_foods, spicy_food))


# Call the main() function if running this file as a script
if __name__=="__main__":
    main()
```

Task 03 Output:

```
In [8]: runfile('D:/DSA/Labs/task03.py', wdir='D:/DSA/Labs')
Calling all the functions in main
['Green Curry', 'Buffalo Wings', 'Mapo Tofu']
[{'name': 'Green Curry', 'cuisine': 'Thai', 'heat_level': 9}, {'name': 'Mapo Tofu', 'cuisine': 'Sichuan',
'heat_level': 6}]
Green Curry (Thai) | 🌶🌶🌶🌶🌶🌶🌶🌶🌶
Buffalo Wings (American) | 🌶🌶🌶
Mapo Tofu (Sichuan) | 🌶🌶🌶🌶🌶🌶
{'name': 'Green Curry', 'cuisine': 'Thai', 'heat_level': 9}
Green Curry (Thai) | 🌶🌶🌶🌶🌶🌶🌶🌶🌶
Mapo Tofu (Sichuan) | 🌶🌶🌶🌶🌶🌶
6.0
[{'name': 'Green Curry', 'cuisine': 'Thai', 'heat_level': 9}, {'name': 'Buffalo Wings', 'cuisine': 'American',
'heat_level': 3}, {'name': 'Mapo Tofu', 'cuisine': 'Sichuan', 'heat_level': 6}, {'name': 'Griot', 'cuisine':
'Haitian', 'heat_level': 10}]
```