JOYDEEP KARMAKAR
12023006015070

ASSIGNMENT 5

1.The following is a list of 10 students ages:

ages = [19, 22, 19, 24, 20, 25, 26, 24, 25, 24]

I. Sort the list and find the min and max age

II. Add the min age and the max age again to the list

III. Find the median age (one middle item or two middle items divided by two)

IV. Find the average age (sum of all items divided by their number)

V. Find the range of the ages (max minus min)

VI. Compare the value of (min - average) and (max - average), use _abs()_ method

CODE

```
# List of ages
ages = [19, 22, 19, 24, 20, 25, 26, 24, 25, 24]
# I. Sort the list and find the min and max age
ages.sort()
min_age = ages[0]
max_age = ages[-1]
print(f"Sorted ages: {ages}")
print(f"Min age: {min_age}")
print(f"Max age: {max_age}")
# II. Add the min age and the max age again to the list
ages.extend([min_age, max_age])
print(f"Ages after adding min and max again: {ages}")
# III. Find the median age
ages.sort()
length = len(ages)
if length % 2 == 0:
    median_age = (ages[length // 2 - 1] + ages[length // 2]) / 2
else:
    median_age = ages[length // 2]
print(f"Median age: {median_age}")
# IV. Find the average age
average_age = sum(ages) / len(ages)
print(f"Average age: {average_age:.2f}")
# V. Find the range of the ages
age_range = max_age - min_age
print(f"Range of ages: {age_range}")
```

```
# VI. Compare the value of (min - average) and (max - average), using abs() method

min_avg_diff = abs(min_age - average_age)

max_avg_diff = abs(max_age - average_age)

print(f"abs(min - average): {min_avg_diff:.2f}")

print(f"abs(max - average): {max_avg_diff:.2f}")
```

```
n.exe "c:/Users/hp/OneDrive/Desktop/UEM/Joydeep_B_45/SEM 3/Python/A
ssignment 5/q1.py"
Sorted ages: [19, 19, 20, 22, 24, 24, 24, 25, 25, 26]
Min age: 19
Max age: 26
Ages after adding min and max again: [19, 19, 20, 22, 24, 24, 24, 2
5, 25, 26, 19, 26]
Median age: 24.0
Average age: 22.75
Range of ages: 7
abs(min - average): 3.75
abs(max - average): 3.25
```

2.Iterate through the list, ['Python', 'Numpy','Pandas','Django', 'Flask'] using a for loop and print out the items.

CODE

```
# List of items

items = ['Python', 'Numpy', 'Pandas', 'Django', 'Flask']

# Iterate through the list and print each item

for item in items:

    print(item, end=" ")
```

```
e "c:/Users/hp/OneDrive/Desktop/UEM/Joydeep_B_45/SE
t 5/q2.py"
Python Numpy Pandas Django Flask
```

3.Create fruits, vegetables and animal products tuples.

I. Join the three tuples and assign it to a variable called food_stuff_tp.

II. Change the about food_stuff_tp tuple to a food_stuff_lt list

III. Slice out the middle item or items from the food_stuff_tp tuple or food_stuff_lt list.

IV. Slice out the first three items and the last three items from food_staff_lt list

V. Delete the food_staff_tp tuple completely

CODE

```
# Create the tuples

fruits = ('apple', 'banana', 'orange')

vegetables = ('carrot', 'broccoli', 'spinach')

animal_products = ('milk', 'cheese', 'yogurt')

# I. Join the three tuples and assign it to a variable called food_stuff_tp

food_stuff_tp = fruits + vegetables + animal_products

print("Joined tuple (food_stuff_tp):", food_stuff_tp)

# II. Change the food_stuff_tp tuple to a food_stuff_lt list
```

```
food_stuff_lt = list(food_stuff_tp)

print("Converted list (food_stuff_lt):", food_stuff_lt)

# III. Slice out the middle item or items from the food_stuff_tp tuple or food_stuff_lt list

length = len(food_stuff_lt)

if length % 2 == 0:

   middle_items = food_stuff_lt[length // 2 - 1 : length // 2 + 1]

else:

   middle_items = food_stuff_lt[length // 2]

print("Middle item(s):", middle_items)

# IV. Slice out the first three items and the last three items from food_stuff_lt list

first_three_items = food_stuff_lt[:3]

last_three_items = food_stuff_lt[-3:]

print("First three items:", first_three_items)

print("Last three items:", last_three_items)

# V. Delete the food_stuff_tp tuple completely

del food_stuff_tp

try:

   print(food_stuff_tp)

except NameError:

   print("food_stuff_tp has been deleted.")
```

```
:/Users/hp/OneDrive/Desktop/UEM/Joydeep_B_45/SEM 3/Python/Assignment 5/
q3.py"
Joined tuple (food_stuff_tp): ('apple', 'banana', 'orange', 'carrot', '
broccoli', 'spinach', 'milk', 'cheese', 'yogurt')
Converted list (food_stuff_lt): ['apple', 'banana', 'orange', 'carrot',
 'broccoli', 'spinach', 'milk', 'cheese', 'yogurt']
Middle item(s): broccoli
First three items: ['apple', 'banana', 'orange']
Last three items: ['milk', 'cheese', 'yogurt']
food stuff tp has been deleted.
```

4. Create a set given below

it_companies = {'Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle', 'Amazon'}

A = {19, 22, 24, 20, 25, 26}

B = {19, 22, 20, 25, 26, 24, 28, 27}

age = [22, 19, 24, 25, 26, 24, 25, 24]

I. Find the length of the set it_companies

II. Add 'Twitter' to it_companies

III. Insert multiple IT companies at once to the set it_companies

IV. Remove one of the companies from the set it_companies

V. What is the difference between remove and discard

CODE

```
# Create the set of IT companies

it_companies = {'Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle', 'Amazon'}

# I. Find the length of the set it_companies

length_of_set = len(it_companies)

print("Length of it_companies:", length_of_set)
```

```
# II. Add 'Twitter' to it_companies

it_companies.add('Twitter')

print("Set after adding 'Twitter':", it_companies)

# III. Insert multiple IT companies at once to the set it_companies

it_companies.update(['Netflix', 'Tesla', 'Adobe'])

print("Set after adding multiple companies:", it_companies)

# IV. Remove one of the companies from the set it_companies

it_companies.remove('Facebook')  # Removing 'Facebook' as an example

print("Set after removing 'Facebook':", it_companies)

# V. Difference between remove and discard

# - remove: Removes the specified element from the set. Raises a KeyError if the element does not exist.

# - discard: Removes the specified element from the set. Does not raise an error if the element does not exist.

# Example:

it_companies.discard('NonExistentCompany')  # No error

# it_companies.remove('NonExistentCompany')  # This would raise a KeyError
```

```
:/Users/hp/OneDrive/Desktop/UEM/Joydeep_B_45/SEM 3/Python/Assignment 5/
q4.py"
Length of it_companies: 7
Set after adding 'Twitter': {'Oracle', 'Facebook', 'Amazon', 'IBM', 'Go
ogle', 'Apple', 'Twitter', 'Microsoft'}
Set after adding multiple companies: {'Oracle', 'Facebook', 'Netflix',
'Tesla', 'Adobe', 'Google', 'Apple', 'Twitter', 'Amazon', 'Microsoft',
'IBM'}
Set after removing 'Facebook': {'Oracle', 'Netflix', 'Tesla', 'Adobe',
'Google', 'Apple', 'Twitter', 'Amazon', 'Microsoft', 'IBM'}
```

5. From the above sets A and B

I. Join A and B

II. Find A intersection B

III. Is A subset of B

IV. Are A and B disjoint sets

V. Join A with B and B with A

VI. What is the symmetric difference between A and B

VII. Delete the sets completely

CODE

```
# Define the sets A and B

A = {19, 22, 24, 20, 25, 26}

B = {19, 22, 20, 25, 26, 24, 28, 27}

# I. Join A and B (Union of A and B)

joined_AB = A.union(B)

print("Joined A and B (A ∪ B):", joined_AB)

# II. Find A intersection B
```

```
intersection_AB = A.intersection(B)
print("Intersection of A and B (A ∩ B):", intersection_AB)
# III. Is A subset of B
is_A_subset_B = A.issubset(B)
print("Is A a subset of B?:", is_A_subset_B)
# IV. Are A and B disjoint sets
are_A_B_disjoint = A.isdisjoint(B)
print("Are A and B disjoint sets?:", are_A_B_disjoint)
# V. Join A with B and B with A (Union is commutative, so it's the same)
joined_A_with_B = A.union(B)
joined_B_with_A = B.union(A)
print("Join A with B:", joined_A_with_B)
print("Join B with A:", joined_B_with_A)
# VI. What is the symmetric difference between A and B
symmetric_difference_AB = A.symmetric_difference(B)
print("Symmetric difference between A and B:", symmetric_difference_AB)
# VII. Delete the sets completely
del A
del B
try:
    print(A)
    print(B)
except NameError:
    print("Set A & B have been deleted.")
```

```
:/Users/hp/OneDrive/Desktop/UEM/Joydeep_B_45/SEM 3/Python/Assignment 5/
q5.py"
Joined A and B (A U B): {19, 20, 22, 24, 25, 26, 27, 28}
Intersection of A and B (A ∩ B): {19, 20, 22, 24, 25, 26}
Is A a subset of B?: True
Are A and B disjoint sets?: False
Join A with B: {19, 20, 22, 24, 25, 26, 27, 28}
Join B with A: {19, 20, 22, 24, 25, 26, 27, 28}
Symmetric difference between A and B: {27, 28}
Set A & B have been deleted.
```

6. Create an empty dictionary called dog.Add name, color, breed, legs, age to the dog dictionary

CODE

```
# Create an empty dictionary
dog = {}
# Add details to the dog dictionary
dog['name'] = 'Aaron'
dog['color'] = 'Brown'
dog['breed'] = 'Labrador'
dog['legs'] = 4
dog['age'] = 3
# Print the dog dictionary
print(dog)
```

```
:/Users/hp/OneDrive/Desktop/UEM/Joydeep_B_45/SEM 3/Python/Assignment 5/
q6.py"
{'name': 'Aaron', 'color': 'Brown', 'breed': 'Labrador', 'legs': 4, 'ag
e': 3}
```

7. Create a student dictionary and add first_name, last_name, gender, age, marital status, skills, country, city and address as keys for the dictionary

I. Get the length of the student dictionary

II. Get the value of skills and check the data type, it should be a list

III. Modify the skills values by adding one or two skills

IV. Get the dictionary keys as a list

V. Get the dictionary values as a list

VI. Change the dictionary to a list of tuples using _items()_ method

VII. Delete one of the items in the dictionary

VIII. Delete one of the dictionaries

CODE

```
# Create the student dictionary
student = {
   'first_name': 'JOYDEEP',    'last_name': 'KARMAKAR',
   'gender': 'Male',
   'age': 21,
   'marital_status': 'Single',
   'skills': ['Node Js', 'DevOps'],
   'country': 'India',
   'city': 'KOLKATA',
   'address': 'TARULIA 3RD LANE'
}
# I. Get the length of the student dictionary
length_of_student_dict = len(student)
print("Length of the student dictionary:", length_of_student_dict)
# II. Get the value of skills and check the data type, it should be a list
skills = student['skills']
print("Skills:", skills)
print("Data type of skills:", type(skills))
# III. Modify the skills values by adding one or two skills
student['skills'].extend(['Cloud Computing', 'Backend Development'])
print("Modified skills:", student['skills'])
# IV. Get the dictionary keys as a list
keys_list = list(student.keys())
print("Dictionary keys:", keys_list)
# V. Get the dictionary values as a list
```

```
values_list = list(student.values())

print("Dictionary values:", values_list)

# VI. Change the dictionary to a list of tuples using items() method

student_items = list(student.items())

print("List of tuples (items):", student_items)

# VII. Delete one of the items in the dictionary

del student['marital_status']

print("Dictionary after deleting 'marital_status':", student)

# VIII. Delete the dictionary completely

del student

# Trying to print student will raise an error because it's deleted

try:

   print(student)

except NameError:

   print("The student dictionary has been deleted.")
```

```
PS C:& C:/Users/hp/AppData/Local/Programs/Python/Python39/python.exe "c:/Users/hp/OneDrive/Desktop/UEM/Joydeep_B_45/SEM 3/Python/Assignment 5/q7.py"
Length of the student dictionary: 9
Skills: ['Node Js', 'DevOps']
Data type of skills: <class 'list'>
Modified skills: ['Node Js', 'DevOps', 'Cloud Computing', 'Backend Development']
Dictionary keys: ['first_name', 'last_name', 'gender', 'age', 'marital_status', 'skills', 'country', 'city', 'address']
Dictionary values: ['JOYDEEP', 'KARMAKAR', 'Male', 21, 'Single', ['Node Js', 'DevOps', 'Cloud Computing', 'Backend Development'], 'India', 'KOLKATA', 'TARULIA 3RD LANE']
List of tuples (items): [('first_name', 'JOYDEEP'), ('last_name', 'KARMAKAR'), ('gender', 'Male'), ('age', 21), ('marital_status', 'Single'), ('skills', ['Node Js', 'DevOps', 'Cloud Computing', 'Backend Development']), ('country', 'India'), ('city', 'KOLKATA'), ('address', 'TARULIA 3RD LANE')]
Dictionary after deleting 'marital_status': {'first_name': 'JOYDEEP', 'last_name': 'KARMAKAR', 'gender': 'Male', 'age': 21, 'skills': ['Node Js', 'DevOps', 'Cloud Computing', 'Backend Development'], 'country': 'India', 'city': 'KOLKATA', 'address': 'TARULIA 3RD LANE'}
The student dictionary has been deleted.
```

8.Create a person dictionary.

person={

'first_name': 'Asabeneh',

'last_name': 'Yetayeh',

'age': 250,

'country': 'Finland',

'is_marred': True,

'skills': ['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],

'address': {

'street': 'Space street',

'zipcode': '02210'

}

}

I. Check if the person dictionary has skills key, if so print out the middle skill in the skills list.

II. Check if the person dictionary has skills key, if so check if the person has 'Python' skill and print out the result.

III. If a person skills has only JavaScript and React, print('He is a front end developer'), if the person skills has Node, Python, MongoDB, print('He is a backend developer'), if the person skills has React, Node and MongoDB, Print('He is a fullstack developer'), else print('unknown title') - for more accurate results more conditions can be nested!

IV. If the person is married and if he lives in Finland, print the information in the following format:

```py
Asabeneh Yetayeh lives in Finland. He is married.
```

CODE

```
# Create the person dictionary
person = {
    'first_name': 'Asabeneh',
    'last_name': 'Yetayeh',
    'age': 25,
    'country': 'Finland',
    'is_marred': True,
    'skills': ['JavaScript', 'React', 'Node', 'MongoDB', 'Python'],
    'address': {
        'street': 'Space street',
        'zipcode': '02210'
    }
}
# I. Check if the person dictionary has the 'skills' key, if so print out the middle skill
if 'skills' in person:
    skills = person['skills']
    middle_index = len(skills) // 2
    middle_skill = skills[middle_index]
    print("Middle skill:", middle_skill)
# II. Check if the person dictionary has the 'skills' key, if so check if the person has 'Python' skill
if 'skills' in person:
    has_python = 'Python' in person['skills']
    print("Has Python skill:", has_python)
# III. Determine the developer type based on the skills
```

```
:/Users/hp/OneDrive/Desktop/UEM/Joydeep_B_45/SEM 3/Pyt
q8.py"
Middle skill: Node
Has Python skill: True
He is a backend developer
Asabeneh Yetayeh lives in Finland. He is married.
```

```
if 'skills' in person:
    skills = person['skills']
    if 'JavaScript' in skills and 'React' in skills and len(skills) == 2:
        print('He is a front end developer')
    elif 'Node' in skills and 'Python' in skills and 'MongoDB' in skills:
        print('He is a backend developer')
    elif 'React' in skills and 'Node' in skills and 'MongoDB' in skills:
        print('He is a fullstack developer')
    else:
        print('Unknown title')
# IV. Print information if the person is married and lives in Finland
if person.get('is_marred') and person.get('country') == 'Finland':
    full_name = f"{person['first_name']} {person['last_name']}"
    print(f"{full_name} lives in Finland. He is married.")
```

9. Print the season name of the year based on the month number using a dictionary.

CODE

```
# Dictionary mapping month numbers to season names
seasons = {
    1: 'Winter',  2: 'Winter',  3: 'Spring',
    4: 'Summer',  5: 'Summer',  6: 'Summer',
    7: 'Monsoon',  8: 'Monsoon',  9: 'Autumn',
    10: 'Autumn', 11: 'Autumn', 12: 'Winter'
}
# Get the month number from the user
month = int(input("Enter the month number (1-12): "))
# Get the season based on the month number
season = seasons.get(month, 'Invalid month number')
# Print the season
if month < 1 or month > 12: print("Invalid month number")
else: print("The season is:", season)
```

```
:/Users/hp/OneDrive/Desktop/OEM/Joydeep_B_45/St
q9.py"
Enter the month number (1-12): 6
The season is: Summer
```