**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

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

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

ages\_sorted = sorted(ages)

min\_age = ages\_sorted[0]

max\_age = ages\_sorted[-1]

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

ages\_sorted.append(min\_age)

ages\_sorted.append(max\_age)

# III. Find the median age

n = len(ages\_sorted)

if n % 2 == 1:

median\_age = ages\_sorted[n // 2]

else:

median\_age = (ages\_sorted[n // 2 - 1] + ages\_sorted[n // 2]) / 2

# IV. Find the average age

average\_age = sum(ages\_sorted) / len(ages\_sorted)

# V. Find the range of the ages

age\_range = max\_age - min\_age

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

min\_avg\_diff = abs(min\_age - average\_age)

max\_avg\_diff = abs(max\_age - average\_age)

# Output results

print("Sorted Ages:", ages\_sorted)

print("Min Age:", min\_age)

print("Max Age:", max\_age)

print("Median Age:", median\_age)

print("Average Age:", average\_age)

print("Range of Ages:", age\_range)

print("Absolute Difference (Min - Average):", min\_avg\_diff)

print("Absolute Difference (Max - Average):", max\_avg\_diff)

2.Iterate through the list, [&#39;Python&#39;, &#39;Numpy&#39;,&#39;Pandas&#39;,&#39;Django&#39;, &#39;Flask&#39;] using a for loop and print out

the items.

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

for item in items:

print(item)

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

# Create tuples

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

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

animal\_products = ('milk', 'cheese', 'butter')

# I. Join the tuples

food\_stuff\_tp = fruits + vegetables + animal\_products

# II. Convert the tuple to a list

food\_stuff\_lt = list(food\_stuff\_tp)

# III. Slice out the middle item(s)

mid = len(food\_stuff\_lt) // 2

if len(food\_stuff\_lt) % 2 == 1:

middle\_items = food\_stuff\_lt[mid]

else:

middle\_items = food\_stuff\_lt[mid - 1:mid + 1]

# IV. Slice out the first three and last three items

first\_three = food\_stuff\_lt[:3]

last\_three = food\_stuff\_lt[-3:]

# V. Delete the tuple

del food\_stuff\_tp

# Output results

print("Food Stuff Tuple:", food\_stuff\_tp)

print("Food Stuff List:", food\_stuff\_lt)

print("Middle Item(s):", middle\_items)

print("First Three Items:", first\_three)

print("Last Three Items:", last\_three)

4. 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

# Define sets

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

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

# I. Join A and B

union\_AB = A | B

# II. Find A intersection B

intersection\_AB = A & B

# III. Is A subset of B

is\_subset = A.issubset(B)

# IV. Are A and B disjoint sets

are\_disjoint = A.isdisjoint(B)

# V. Join A with B and B with A

A.update(B)

B.update(A)

# VI. Symmetric difference between A and B

symmetric\_diff = A ^ B

# VII. Delete the sets completely

del A

del B

# Additional Operations with `it\_companies`

it\_companies = {'Facebook', 'Google', 'Microsoft', 'Apple', 'IBM', 'Oracle', 'Amazon'}

# Output results

print("Union of A and B:", union\_AB)

print("Intersection of A and B:", intersection\_AB)

print("Is A a subset of B:", is\_subset)

print("Are A and B disjoint sets:", are\_disjoint)

print("Symmetric difference between A and B:", symmetric\_diff)

print("IT Companies Set:", it\_companies)

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

# Create an empty dictionary

dog = {}

# Add attributes to the dictionary

dog['name'] = 'Rex'

dog['color'] = 'Brown'

dog['breed'] = 'Labrador'

dog['legs'] = 4

dog['age'] = 5

# Output the dictionary

print(dog)

6. 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

# Create a student dictionary

student = {

'first\_name': 'John',

'last\_name': 'Doe',

'gender': 'Male',

'age': 21,

'marital\_status': 'Single',

'skills': ['Python', 'JavaScript'],

'country': 'USA',

'city': 'New York',

'address': '123 Main St'

}

# I. Get the length of the student dictionary

length = len(student)

# II. Get the value of skills and check the data type

skills = student['skills']

skills\_type = type(skills)

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

student['skills'].extend(['HTML', 'CSS'])

# IV. Get the dictionary keys as a list

keys\_list = list(student.keys())

# V. Get the dictionary values as a list

values\_list = list(student.values())

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

items\_list = list(student.items())

# VII. Delete one of the items in the dictionary

del student['address']

# VIII. Delete the dictionary

del student

# Output results

print("Length of dictionary:", length)

print("Skills:", skills)

print("Type of skills:", skills\_type)

print("Updated Skills:", student['skills']) # After adding new skills

print("Dictionary Keys as List:", keys\_list)

print("Dictionary Values as List:", values\_list)

print("Dictionary Items as List of Tuples:", items\_list)

7.Create a person dictionary.

person={

&#39;first\_name&#39;: &#39;Asabeneh&#39;,

&#39;last\_name&#39;: &#39;Yetayeh&#39;,

&#39;age&#39;: 250,

&#39;country&#39;: &#39;Finland&#39;,

&#39;is\_marred&#39;: True,

&#39;skills&#39;: [&#39;JavaScript&#39;, &#39;React&#39;, &#39;Node&#39;, &#39;MongoDB&#39;, &#39;Python&#39;],

&#39;address&#39;: {

&#39;street&#39;: &#39;Space street&#39;,

&#39;zipcode&#39;: &#39;02210&#39;

}

}

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 &#39;Python&#39; skill and

print out the result.

III. If a person skills has only JavaScript and React, print(&#39;He is a front end developer&#39;), if the

person skills has Node, Python, MongoDB, print(&#39;He is a backend developer&#39;), if the person

skills has React, Node and MongoDB, Print(&#39;He is a fullstack developer&#39;), else print(&#39;unknown

title&#39;) - 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.

person = {

'first\_name': 'Asabeneh',

'last\_name': 'Yetayeh',

'age': 250,

'country': 'Finland',

'is\_married': True,

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

'address': {'street': 'Space street', 'zipcode': '02210'}

}

if 'skills' in person:

s = person['skills']

print("Middle skill:", s[len(s) // 2])

print("Has Python skill:", 'Python' in s)

if s == ['JavaScript', 'React']:

print('He is a front end developer')

elif {'Node', 'Python', 'MongoDB'}.issubset(s):

print('He is a backend developer')

elif {'React', 'Node', 'MongoDB'}.issubset(s):

print('He is a fullstack developer')

else:

print('Unknown title')

if person.get('is\_married') and person.get('country') == 'Finland':

print(f"{person['first\_name']} {person['last\_name']} lives in Finland. He is married.")

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

seasons = {

1: 'Winter', 2: 'Winter', 3: 'Spring',

4: 'Spring', 5: 'Spring', 6: 'Summer',

7: 'Summer', 8: 'Summer', 9: 'Fall',

10: 'Fall', 11: 'Fall', 12: 'Winter'

}

month = int(input("Enter month number (1-12): "))

print(seasons.get(month, 'Invalid month'))