IYAS K
iyas2458@gmail.com
Github: https://github.com/iyas311/UST-Assignments/tree/main/Assignment2

LIST

```
# Add an element at the end of a list
    lst = [1, 2, 3]
    lst.append(4)

# Remove an element from a list by its index
    lst = [10, 20, 30, 40]
    lst.pop(2)      # removes element at index 2
    # del lst[2]    # another way

# Output of the given code snippet
    lst = [1, 2, 3, 4, 5]
    lst[1:3] = [10, 20]
    print(lst)      # [1, 10, 20, 4, 5]

# Check if an element exists in a list
    lst = [5, 10, 15]
    exists = 10 in lst

# Function to remove duplicates without using set()
    def remove_duplicates(lst):
    result = []
    for item in lst:
    if item not in result:
        result.append(item)
    return result
```

TUPLE

#Can you modify the elements of a tuple after it has been created? Why or why not?

You cannot modify the elements of a tuple after it is created because tuples are immutable.

```
# Access the second-to-last element in a tuple using negative indexing
    T = (10, 20, 30, 40)
    second_last = t[-2]

# Difference between list and tuple:
# List is mutable (can be changed), tuple is immutable (cannot be changed)
        lst = [1, 2, 3]
```

```
        tpl = (1, 2, 3)

# Change value 3 to 100 in tuple t = (1, 2, 3, 4) by converting to list and back
    t = (1, 2, 3, 4)
    temp = list(t)
    temp[2] = 100
    t = tuple(temp)

# Function to return the sum of all elements in a tuple
    def sum_of_tuple(t):
        total = 0
        for num in t:
            total += num
        return total
```

SET

```
# Q1: How do you remove all elements from a set in Python?
s = {1, 2, 3}
s.clear()

# Q2: What is the output of the following code snippet?
a = {1, 2, 3, 4}
b = {3, 4, 5, 6}
print(a - b)    # Output: {1, 2}

# Q3: How do you check if an element is present in a set?
s = {10, 20, 30}
is_present = 20 in s

# Q4: Write a Python program to find the intersection of two sets.
set1 = {1, 2, 3, 4}
set2 = {3, 4, 5, 6}
intersection = set1 & set2
print(intersection)   # {3, 4}

# Q5: How does a set handle duplicate values when adding them?
s = {1, 2, 3}
s.add(2)    # Duplicate value is ignored, set remains unchanged
```

DICT

```
# Q1: How can you add a new key-value pair to an existing dictionary in Python?
d = {"a": 10, "b": 20}
d["c"] = 30

# Q2: What happens if you try to access a key that does not exist in a dictionary?
```

```
d = {"x": 1}
# print(d["y"])       # Raises KeyError
value = d.get("y")    # Returns None instead of error

# Q3: Write a Python function that takes a dictionary and returns a list of keys that have values greater than 50.
def keys_greater_than_50(d):
    result = []
    for key, value in d.items():
        if value > 50:
            result.append(key)
    return result

# Q4: How would you iterate over both keys and values of a dictionary in Python?
d = {"a": 10, "b": 20, "c": 30}
for key, value in d.items():
    print(key, value)

# Q5: Write a Python function that merges two dictionaries.
def merge_dicts(d1, d2):
    merged = d1.copy()
    merged.update(d2)
    return merged
```

MAP AND REDUCE

```
# Q1: How does the map() function work in Python? Give an example where you square each number in a list.
nums = [1, 2, 3, 4]
squared = list(map(lambda x: x * x, nums))
print(squared)   # [1, 4, 9, 16]

# Q2: What is the output of the following code?
from functools import reduce
result = reduce(lambda x, y: x * y, [1, 2, 3, 4])
print(result)    # Output: 24

# Q3: How would you use the map() function to convert all string elements of a list to uppercase?
words = ["python", "map", "reduce"]
upper_words = list(map(lambda x: x.upper(), words))
print(upper_words)   # ['PYTHON', 'MAP', 'REDUCE']

# Q4: Write a Python program that uses reduce() to find the GCD of a list of numbers.
from functools import reduce
import math
```

```
nums = [24, 36, 60]
gcd_result = reduce(math.gcd, nums)
print(gcd_result)    # 12

# Q5: Compare and contrast the map() and filter() functions in Python.
# map() applies a function to each element and returns transformed values
# filter() applies a condition and returns only elements that satisfy it
nums = [1, 2, 3, 4, 5]
mapped = list(map(lambda x: x * 2, nums))        # [2, 4, 6, 8, 10]
filtered = list(filter(lambda x: x % 2 == 0, nums))  # [2, 4]
```

LIST COMPREHENSION

```
# Q1: Write a list comprehension that returns all even numbers from 0 to 20.
evens = [x for x in range(21) if x % 2 == 0]
print(evens)

# Q2: How would you create a new list of squares from an existing list of numbers using list comprehension?
nums = [1, 2, 3, 4, 5]
squares = [x * x for x in nums]
print(squares)

# Q3: Write a list comprehension to extract all words that are longer than 4 characters from a sentence.
sentence = "Python list comprehension is very powerful"
long_words = [word for word in sentence.split() if len(word) > 4]
print(long_words)

# Q4: How can you use list comprehension to generate a list of the first 10 Fibonacci numbers?
fib = [0, 1]
[fib.append(fib[-1] + fib[-2]) for _ in range(8)]
fib = fib[:10]
print(fib)

# Q5: Can you use an if condition inside a list comprehension? Provide an example where only odd numbers are selected from a list.
nums = [1, 2, 3, 4, 5, 6]
odds = [x for x in nums if x % 2 != 0]
print(odds)
```