Module 12

Sets

- Sets are unique collection of data (NO duplicates)
- Unordered ← Indexing not supported.
- Unindexed
- Mutable

```
lst = [1, 2, 3, 1, 1]
lst
[1, 2, 3, 1, 1]
```

Creating a Set

- ⊙ set()
- ⊙ set(iterable)

```
# empty set
s = {}
type(s)
```
✗
dict

```
s = set()
print(s)
set()
type(s)
set
```
✓

```
# now - empty set
s = [1, 2, 3, 1, 2, 4, 5, 1, 2]
print(s)
{1, 2, 3, 4, 5}
```

```
s1 = set("UVARAJ")
type(s)
set
s1
{'A', 'J', 'R', 'U', 'V'}
s1['1']
TypeError: 'set' object is not subscriptable
```

```
# iteration
for i in s:
    print(i, end = " ")
```

A J R U V

Updating a set

- add : For single element
- update (iterable)

```
s = {1, 2, 3, 3, 5, 4, 6}
s
{1, 2, 3, 4, 5, 6}
s.add(8)
s
{1, 2, 3, 4, 5, 6, 8}
s.add(8)
s.add(8)
s.add(8)
s
{1, 2, 3, 4, 5, 6, 8}
```

```
name = "Uvaraj"
s.update(name)
```

iterable

```
s
{1, 2, 3, 4, 5, 6, 8, 'U', 'v', 'a', 'r', 'a', 'j'}
```

Delete an element

- pop → removes random element. We are not sure what it is
- remove (element) → Removes particular element

```
s.pop()
1
s.pop()
2
s
{3, 4, 5, 6, 8, 'U', 'v', 'a', 'r', 'a', 'j'}
```

```
s.remove("U")
s.remove("V")
s.remove("a")
s
{3, 4, 5, 6, 8, 'r', 'a', j'}
s.remove("2")
KeyError: '2'
```

```
a = s.pop()
a
3
b = s.remove(3)
KeyError: 3
```

Intersection

- Suppose Scaler has two courses available for Students. Python and Java
- You want to find out which students are enrolled in both the Python and Java Courses. Then you can use the intersection method.

```
python = {"IronMan", "Hulk", "spidy", "Uvaraj"}
java = {"IronMan", "HarryPotter", "AntMan"}

python.intersection(java)
{'IronMan'}
java.intersection(python)
{'IronMan'}
```

Union

- Suppose you want to find out which students are enrolled in either python or Java course or in both, then you can use Union method.

```
python.Union(java)
{'AntMan', 'HarryPotter', 'Hulk', 'IronMan', 'spidy', 'Uvaraj'}
```

Difference

- Suppose you want to find out the set of students who have enrolled in the Python course but not in Java course, or vice-versa, then we can use the difference method.

```
python.difference(java)  # Only Python
{'Hulk', 'Spidy', 'Uvaraj'}
java.difference(python)  # only Java
{'AntMan', 'HarryPotter'}
```

Challenge

① Count the number of unique elements in a sentence.

```
sent = "be the change you wish to see in the world"
lst = sent.split()
lst
['be', 'the', 'change', 'you', 'wish', 'to', 'see', 'in', 'the', 'world']
s = set(lst)
```

iterable

```
s
{'be', 'change', 'in', 'see', 'the', 'to', 'wish', 'world', 'you'}
len(s)
9
```

---

② What is the output of the following program?

```
set1 = {1, 2, 4}
set2 = {2, 5, 6}
print(len(set1 + set2))
```

a) 3 b) 5 c) 4 d) Error ✓

③ Frequency of Unique elements.

Write a function to print out the frequency of all the unique elements present in a given tuple.

Input format:

This one input line consists of a tuple

output format:

Unique elements and their frequencies are expected to be printed as follows:

```
unique_element1 : freq1
unique_element2 : freq2
:
:
```

Sample input:

(10, 8, 5, 2, 10, 15, 10, 8, 5, 8, 8, 2)

Sample output:

```
10 : 3
8 : 4
5 : 2
2 : 2
15 : 1
```

Sample input:

(100, -10, -10, "ONE", [22], [22], "ONE")

Sample output:

```
100 : 1
-10 : 2
ONE : 2
[22] : 2
```

```
def unique_count(tup):
    # Create a list to store unique elements and their frequencies
    frequency_list = []

    for element in tup:
        found = False
        # Check if the element already exists in the
        # frequency list
        for item in frequency_list:
            if item[0] == element:
                item[1] += 1
                found = True
                break
        # If the element is not found, add it as a new
        # entry
        if not found:
            frequency_list.append([element, 1])

    # Print each unique element with its frequency
    for item in frequency_list:
        print(f"{item[0]}: {item[1]}")

# Test the function with the given input
unique_count((1, 2, 3, 2, "Hello", [4, 8, 16], "Hello"))
```

```
1: 1
2: 2
3: 1
Hello: 2
[4, 8, 16]: 1
```

④ Complete the code snippet with possible statement from options, so that the expected output is obtained.

```
sets = {3, 4, 5}

---

print (sets)
```

Expected output :

{1, 2, 3, 4, 5}

a) sets.update ([1, 2, 3]) ✓

b) sets.add ([1, 2, 3])

c) sets += [1, 2, 3]

d) sets += {1, 2, 3}

e) sets = sets.union ({1, 2, 3})

⑤ Indexing Sets

Given the name of students in the following set. There was a grammatical mistake in the name "Aurn". It should be "Arun". What will be the output if we try to execute the code given below in order to update this name ?

```
a = {'Aurn', 'Nikhil', 'Seeta'}
a[0] = 'Arun'
print (a)
```

a) {'Arun', 'Nikhil', 'Seeta'}

b) {'Aurn', 'Nikhil', 'Seeta'}

c) {'Aurn', 'Nikhil', 'Arun'}

d) Type Error ✓