Name:- Divyang Bagla

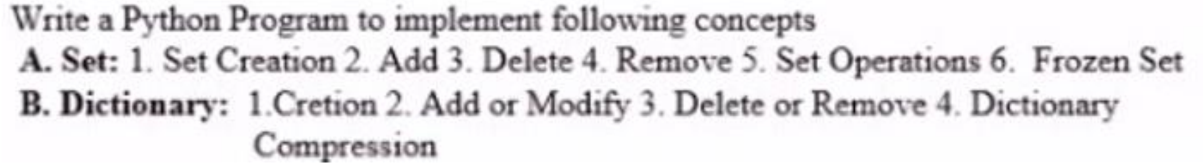
ERP ID:- 1032180739

Batch A2

Roll No. :- PC33

Lab Assignment No. 4

Aim:-



Write a Python Program to implement following concepts

A. Set: 1. Set Creation 2. Add 3. Delete 4. Remove 5. Set Operations 6. Frozen Set

B. Dictionary: 1.Cretion 2. Add or Modify 3. Delete or Remove 4. Dictionary Compression

Theory:-

Sets

Definition: Sets are used to store multiple items in a single variable.

Set is one of 4 built-in data types in Python used to store collections of data, the other 3 are List, Tuple, and Dictionary, all with different qualities and usage.

A set is a collection which is both unordered and unindexed.

Sets are written with curly brackets {}.

Set Features:

1. Unordered
2. Unindexed
3. Mutable
4. Duplicates Allowed

Dictionary

Definition: Dictionary in Python is an ordered collection of data values, used to store data values like a map, which, unlike other Data Types that hold only a single value as an element, Dictionary holds key:value pair.

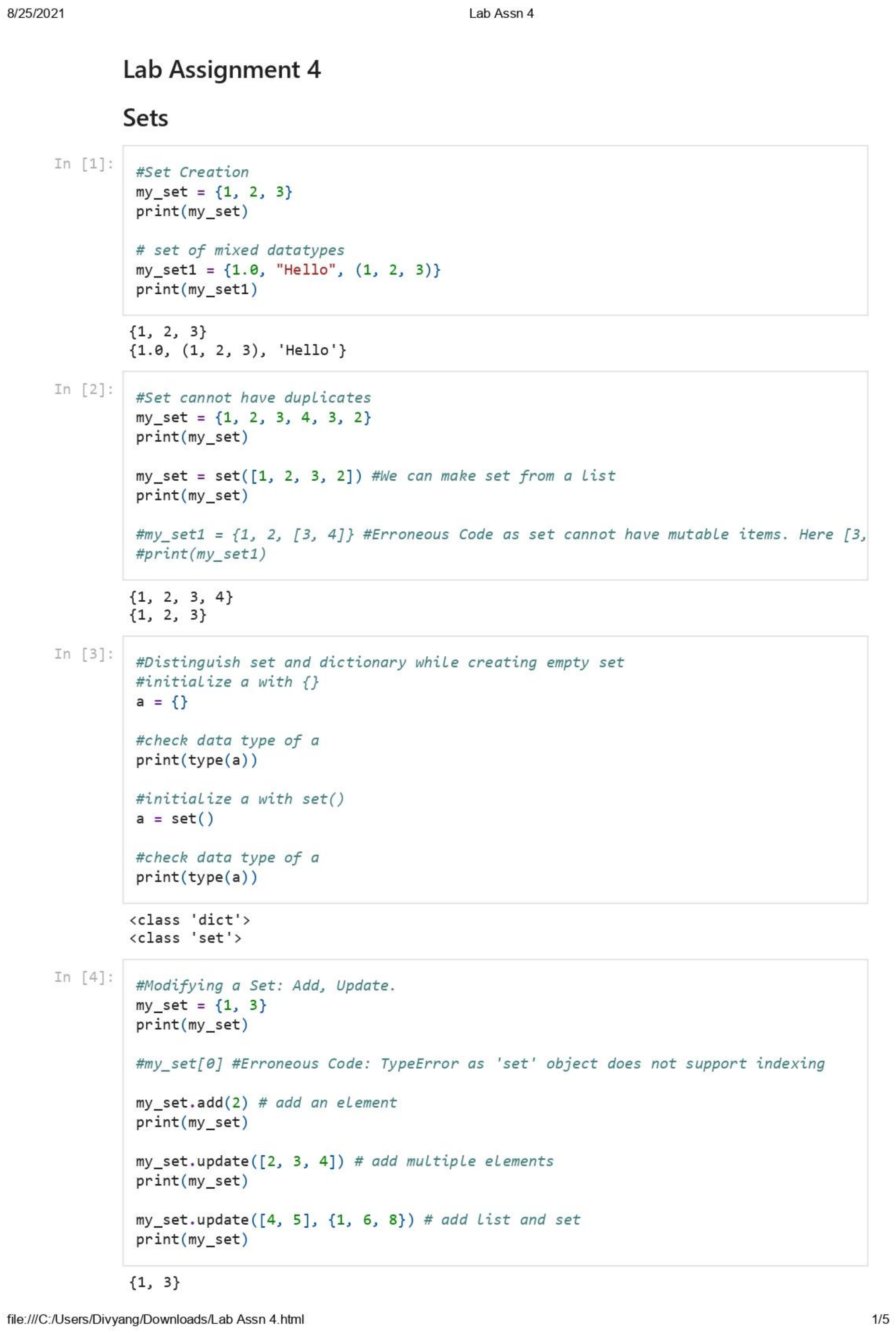
Key-value is provided in the dictionary to make it more optimized.

Dictonary is written with curly brackets {}.

Dictionary Features:

1. Dictionaries are unordered. A dictionary contains key-value pairs but does not possess an order for the pairs.
2. Keys are unique. Dictionary keys must be unique.
3. Keys must be immutable.

Lab Assignment 4 Code



Lab Assignment 4

Sets

In [1]:

```
#Set Creation
my_set = {1, 2, 3}
print(my_set)

# set of mixed datatypes
my_set1 = {1.0, "Hello", (1, 2, 3)}
print(my_set1)
```

```
{1, 2, 3}
{1.0, (1, 2, 3), 'Hello'}
```

In [2]:

```
#Set cannot have duplicates
my_set = {1, 2, 3, 4, 3, 2}
print(my_set)

my_set = set([1, 2, 3, 2]) #We can make set from a list
print(my_set)

#my_set1 = {1, 2, [3, 4]} #Erroneous Code as set cannot have mutable items. Here [3,
#print(my_set1)
```

```
{1, 2, 3, 4}
{1, 2, 3}
```

In [3]:

```
#Distinguish set and dictionary while creating empty set
#initialize a with {}
a = {}

#check data type of a
print(type(a))

#initialize a with set()
a = set()

#check data type of a
print(type(a))
```

```
<class 'dict'>
<class 'set'>
```

In [4]:

```
#Modifying a Set: Add, Update.
my_set = {1, 3}
print(my_set)

#my_set[0] #Erroneous Code: TypeError as 'set' object does not support indexing

my_set.add(2) # add an element
print(my_set)

my_set.update([2, 3, 4]) # add multiple elements
print(my_set)

my_set.update([4, 5], {1, 6, 8}) # add list and set
print(my_set)
```

```
{1, 3}
```

```
{1, 2, 3}
{1, 2, 3, 4}
{1, 2, 3, 4, 5, 6, 8}
```

In [5]:

```
#Discard and Remove

my_set = {1, 3, 4, 5, 6}
print(my_set)

my_set.discard(4) #Discard an element
print(my_set)

my_set.remove(6) #Remove an element
print(my_set)

my_set.discard(2) #Discard an element not present in my_set
print(my_set)

#my_set.remove(2) #Erroneous Code
```

```
{1, 3, 4, 5, 6}
{1, 3, 5, 6}
{1, 3, 5}
{1, 3, 5}
```

In [6]:

```
#Pop and Clear
my_set = set("HelloWorld")
print(my_set)

print(my_set.pop()) # pop an element

my_set.pop() # pop another element
print(my_set)

my_set.clear() # clear my_set
print(my_set)
```

```
{'e', 'd', 'r', 'W', 'H', 'l', 'o'}
e
{'r', 'W', 'H', 'l', 'o'}
set()
```

In [7]:

```
#Set Operations

A = {1, 2, 3, 4, 5}
B = {4, 5, 6, 7, 8}

print(A | B) #Set Union
print(B.union(A))
print(A & B) #Set Intersection
print(A.intersection(B))
print(A-B) #Set Difference
print(A.difference(B))
print(A ^ B) #Set Symmetric Difference

A.isdisjoint(B) #To check for disjoint sets
```

```
{1, 2, 3, 4, 5, 6, 7, 8}
{1, 2, 3, 4, 5, 6, 7, 8}
{4, 5}
{4, 5}
{1, 2, 3}
{1, 2, 3}
{1, 2, 3, 6, 7, 8}
```

Out[7]:
```
False
```

In [8]:
```python
#Frozen Set

vowels = ('a', 'e', 'i', 'o', 'u') #Tuple of vowels

fSet = frozenset(vowels)
print('The frozen set is:', fSet)

#fSet.add('v') #Erroneous Code as frozensets are immutable
```

```
The frozen set is: frozenset({'u', 'e', 'a', 'i', 'o'})
```

Dictionary

In [10]:
```python
#Dictionary Creation
my_dict = {}

my_dict = {1: 'Divyang', 2: 'Bagla'} # dictionary with integer keys

my_dict1 = {'name': 'Python', 1: [2, 4, 3]} # dictionary with mixed keys

my_dict2 = dict({1:'Python', 2:'Lab'}) # using dict()

my_dict3 = dict([(1,'Sets'), (2,'Dicts')]) # from sequence having each item as a pai

print(my_dict)
print(my_dict1)
print(my_dict2)
print(my_dict3)
```

```
{1: 'Divyang', 2: 'Bagla'}
{'name': 'Python', 1: [2, 4, 3]}
{1: 'Python', 2: 'Lab'}
{1: 'Sets', 2: 'Dicts'}
```

In [11]:
```python
#get vs [] for retrieving elements

my_dict = {'firstname': 'Divyang', 'lastname': 'Bagla', 'age': 20}

print(my_dict['firstname'])

print(my_dict.get('age'))

print(my_dict.get('address'))

#print(my_dict['address']) #Erroneous Code: The address key doesn't exist.
```

```
Divyang
20
None
```

In [12]:
```python
#Changing and adding Dictionary Elements
my_dict = {'firstname': 'Divyang', 'lastname': 'Bagla', 'age': 20}

print(my_dict)

my_dict['age'] = 24 #Update age value

print(my_dict)

my_dict['address'] = 'Downtown Los Angeles' #Add item
```

```
print(my_dict)
```

```
{'firstname': 'Divyang', 'lastname': 'Bagla', 'age': 20}
{'firstname': 'Divyang', 'lastname': 'Bagla', 'age': 24}
{'firstname': 'Divyang', 'lastname': 'Bagla', 'age': 24, 'address': 'Downtown Los An
geles'}
```

In [13]:

```
#Removing elements from a dictionary

squares= {1: 1, 2: 4, 3: 9, 4: 16, 5: 25}

print(squares.pop(4)) #Remove a particular item, returns its value

print(squares)

print(squares.popitem()) #Remove an arbitrary item, return (key,value)

print(squares)

squares.clear() #Remove all items

print(squares)

del squares #Delete the entire dictionary

#print(squares) #Erroneous code as dictionary doesnt exist.
```

```
16
{1: 1, 2: 4, 3: 9, 5: 25}
(5, 25)
{1: 1, 2: 4, 3: 9}
{}
```

In [14]:

```
square_dict = dict()
for num in range(1, 11):
    square_dict[num] = num*num
print(square_dict)

# Dictionary Comprehension
squares = {x: x*x for x in range(1, 11)}

print(squares)
```

```
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
{1: 1, 2: 4, 3: 9, 4: 16, 5: 25, 6: 36, 7: 49, 8: 64, 9: 81, 10: 100}
```

In [15]:

```
#2nd Example
#item price in dollars
old_price = {'milk': 1.02, 'coffee': 2.5, 'bread': 2.5}
print(old_price)

dollar_to_pound = 0.76
new_price = {item: value*dollar_to_pound for (item, value) in old_price.items()}
print(new_price)
```

```
{'milk': 1.02, 'coffee': 2.5, 'bread': 2.5}
{'milk': 0.7752, 'coffee': 1.9, 'bread': 1.9}
```

In [16]:

```
#3rd Example: Conditional Dictionary Comprehension

original_dict = {'jack': 38, 'michael': 48, 'guido': 57, 'john': 33}
print(original_dict)
```

```
even_dict = {k: v for (k, v) in original_dict.items() if v % 2 == 0} #only the items
#because of the if clause in the dictionary comprehension.
print(even_dict)

new_dict = {k: v for (k, v) in original_dict.items() if v % 2 != 0 if v < 40} #only
#of less than 40 have been added to the new dictionary.
print(new_dict)

new_dict_1 = {k: ('old' if v > 40 else 'young')
    for (k, v) in original_dict.items()}
print(new_dict_1)
```

```
{'jack': 38, 'michael': 48, 'guido': 57, 'john': 33}
{'jack': 38, 'michael': 48}
{'john': 33}
{'jack': 'young', 'michael': 'old', 'guido': 'old', 'john': 'young'}
```

```
In [ ]:
```