Python_Assignment 2

```
[2]: #Q1 Discuss String Slicing and provide Examples
     #Ans Sub part of String
     # Substring can be extracted using slicing
     # Example
     String1 = "I am a good student"
     String1[3:]
```

```
[2]: 'm a good student'
```

```
[3]: #Q2 Explain the key features of lists in Python
     #Ans List are ordered collection of item
     # List is like a shopping bag which can store everything
     # List are mutable
     []
     type([])
```

```
[3]: list
```

```
[4]: # Indexing you can acess elements by their position
     my_list = [10,20,30]
     print(my_list[1])
```

```
20
```

```
[6]: #Mutable
     #list can be modified after creation
     # You can add or remove
     my_list = [10,2,30]
     my_list[1] = 20
     print(my_list)
```

```
[10, 20, 30]
```

```
[7]: # Dynamic Size
     # List can grow. You can append
     my_list = [10,2,30]
     my_list.append(4)
     print(my_list)
```

```
[10, 2, 30, 4]
```

[9]:
```python
#Q3 Describe how to access ,modify and delete the elements in a lists with
    ↪example
# You access list elements using indexing
s = ["Apple","Mango","Grapes" ]
print(s[2])
```

```
Grapes
```

[11]:
```python
#Slicing
# Retrieve a portion of the list
s = ["Apple","Mango","Grapes" ]
print(s[1:3])
```

```
['Mango', 'Grapes']
```

[12]:
```python
#Modifying Elements
# You can change an elements and assign a new value
s = ["Apple","Mango","Grapes" ]
s[1] = "Blueberry"
print(s)
```

```
['Apple', 'Blueberry', 'Grapes']
```

[15]:
```python
#Deleting Elements
s = ["Apple","Mango","Grapes" ]
del s[0]
print(s)
```

```
['Mango', 'Grapes']
```

[16]:
```python
#Q3 Campare Tuples and Lists with Example
# Lists
#Lists are ordered collection of item
# Lists is like a shopping bag which can store everthing
# Can store any Datatype
#Example
grocery_list = ["Milk","Orange",1,2.2,True,3+5j]
grocery_list
```

[16]: ['Milk', 'Orange', 1, 2.2, True, (3+5j)]

[17]:
```python
type(grocery_list)
```

[17]: list

```python
[18]: #Tuples
      # Tuples are ordered collection of elements, heterogenous
      # Tuples are immutable
      # Tuples have lesss in built methods
      t = ()
      type(t)
```

[18]: tuple

```python
[1]: #Q5 Describe the key features of sets and provide the example of usecase
     # Sets are unordered and unique collection of elements
     # Does not allow duplicate elements
     # set is  mutable
     # Add  festures
     s = {1,2,3,4,5,}
     s
```

[1]: {1, 2, 3, 4, 5}

```python
[3]: s.add(100)
     s
```

[3]: {1, 2, 3, 4, 5, 100}

```python
[7]: # update features
     S = {1,2,3,4,5,}
     S
```

[7]: {1, 2, 3, 4, 5}

```python
[20]: # Union  sets
      # Combine  all  the  elements
      S1 = {1,2,3}
      S2 = {3,4,5}
      S1|S2
```

[20]: {1, 2, 3, 4, 5}

```python
[21]: # Intersection  >>  Common  element
      S1 = {1,2,3}
      S2 = {3,4,5}
      S1&S2
```

[21]: {3}

```python
[22]: # Difference  >>  present  in  first  set  and  not  in  second
      S1 = {1,2,3}
```

```
S2 = {3,4,5}
S1-S2
```

[22]: {1, 2}

[23]:
```
# Symmetric >> return all the element other than common element
S1 = {1,2,3}
S2 = {3,4,5}
S1^S2
```

[23]: {1, 2, 4, 5}

[24]:
```
#Q6 Describe the use case Tuple and Sets in the program
# Tuples are ordered collection of elements, heterogenous
# Tuples are immutable
# Tuples have lesss in built methods
# Immutable Data: Tuples are immutable, meaning once created, their elements
 ↪cannot be changed
# Fixed Collection: Tuples are typically used to store a fixed collection of
 ↪item
# Faster Access: Accessing elements in a tuple is faster compared to accessing
 ↪elements in a list because tuples are implemented as fixed-size arrays.
# Tuples are suitable for representing fixed collections of items like
 ↪coordinates, records from a database

# Sets
# Sets are unordered and unique collection of elements
# Does not allow duplicate elements
# set is mutable
# Unordered Collection of Unique Items: Sets are unordered collections of items
 ↪where each item is unique
# Mathematical Set Operations: Sets support mathematical set operations like
 ↪union, intersection, difference, and symmetric difference.
```

[25]:
```
#Q7 7. Describe how to add, modify, and delete items in a dictionary with
 ↪examples.
# Dictionary is a data strucutre that stores data as key value pair
# Adding item
my_dict = {"name": "John", "age":30}
my_dict["city"] = "New York"
print(my_dict)
```

{'name': 'John', 'age': 30, 'city': 'New York'}

[26]:
```
# Modifying item
my_dict = {"name": "John", "age":30}
my_dict["age"] = 41
```

```
print(my_dict)
```

{'name': 'John', 'age': 41}

[28]:
```
# Deleting  item
my_dict = {"name": "John", "age":30,"city": "new York"}
del my_dict["city"]
print(my_dict)
```

{'name': 'John', 'age': 30}

[1]:
```
#Q7 Discuss the importance of dictionary keys being immutable and provide
 examples.
# Dictionary keys need to be hashable, meaning their hash value remains
 constant over time
# Dictionaries rely on hash tables to store and retrieve data efficiently.
# Example
# Immutable keys
# String
my_dict = {"name": "John", "age":30}
print(my_dict["name"])
```

John

[4]:
```
# Tuples
my_dict = {(1,2): "a" ,(3,4): "b"}
print(my_dict[(1,2)])
```

a

[ ]: