Python Data Structures

1. Lists

Lists are ordered, mutable collections that allow duplicates and different data types.

Example:

```
my_list = [1, 2, 3, 'apple']
print(my_list)
print(my_list[0])
my_list.append('banana')
print(my_list)
my_list.remove(2)
print(my_list)
my_list.sort(key=str)
print(my_list)
```

Output:

```
[1, 2, 3, 'apple']
1
[1, 2, 3, 'apple', 'banana']
[1, 3, 'apple', 'banana']
[1, 3, 'apple', 'banana']
```

Assignments

1. Create a list of 5 numbers. Find the sum, maximum, and minimum value.

Solution:

```
numbers = [5, 10, 15, 20, 25]
print(sum(numbers))
print(max(numbers))
print(min(numbers))
```

2. Create a list of fruits. Replace the second fruit with 'Mango'.

Solution:

```
fruits = ['Apple', 'Banana', 'Cherry']
fruits[1] = 'Mango'
print(fruits)
```

2. Tuples

Tuples are ordered, immutable collections. They are faster than lists and used for fixed data.

Example:

```
my_tuple = (1, 2, 3, 'apple')
print(my_tuple)
print(my_tuple[1])
print(my_tuple.count(2))
print(my_tuple.index('apple'))
```

Output:

```
(1, 2, 3, 'apple')
2
1
3
```

Assignments

1. Create a tuple with 4 elements. Print the last two elements.

Solution:

```
t = (10, 20, 30, 40)
print(t[-2:])
```

2. Count how many times 5 occurs in the tuple (1, 5, 5, 7, 9).

Solution:

```
t = (1, 5, 5, 7, 9)
print(t.count(5))
```

3. Dictionaries

Dictionaries store key-value pairs. Keys must be unique and immutable.

Example:

```
my_dict = {'name': 'John', 'age': 25}
print(my_dict)
print(my_dict['name'])
my_dict['age'] = 26
print(my_dict)
my_dict['city'] = 'Doha'
```

```
print(my_dict)
print(my_dict.keys())
print(my_dict.values())
```

Output:

```
{'name': 'John', 'age': 25}
John
{'name': 'John', 'age': 26}
{'name': 'John', 'age': 26, 'city': 'Doha'}
dict_keys(['name', 'age', 'city'])
dict_values(['John', 26, 'Doha'])
```

Assignments

1. Create a dictionary with 3 student names as keys and marks as values. Print the average marks.

Solution:

```
students = {'A': 85, 'B': 90, 'C': 95}
print(sum(students.values())/len(students))
```

2. Add a new key 'grade' with value 'A' to the dictionary {'name': 'Sara', 'age': 22}.

Solution:

```
student = {'name': 'Sara', 'age': 22}
student['grade'] = 'A'
print(student)
```

4. Sets & Booleans

Sets are unordered collections of unique items. Booleans represent truth values (True/False).

Example:

```
my_set = {1, 2, 3, 3}
print(my_set)
print(my_set.union({4, 5}))
print(my_set.intersection({2, 3, 5}))
print(my_set.difference({3}))

a = 5
b = 3
print(a > b)
print(a == b)
print(a > 2 and b < 4)
```

Output:

```
{1, 2, 3}
{1, 2, 3, 4, 5}
{2, 3}
{1, 2}
True
False
True
```

Assignments

1. Create two sets {1,2,3} and {3,4,5}. Print their union and intersection.

Solution:

```
a = {1,2,3}
b = {3,4,5}
print(a | b)
print(a & b)
```

2. Check if {1,2} is a subset of {1,2,3,4}.

Solution:

```
print({1,2}.issubset({1,2,3,4}))
```

Capstone Project

Project: Student Data Management System

Task: Create a program that manages student data using lists, tuples, sets, and dictionaries.
Requirements:
1. Use a list to store student IDs.
2. Use a dictionary to store student details (name, age, marks).
3. Use a set to keep track of unique courses offered.
4. Use tuples to store immutable student info (ID, Name).

Solution Example:

```
students = {}
student_ids = []
courses = set()

# Adding students
```

```
students[101] = {'name': 'Alice', 'age': 20, 'marks': 85}
students[102] = {'name': 'Bob', 'age': 22, 'marks': 90}
student_ids.extend([101, 102])

# Adding courses
courses.update(['Math', 'Science', 'English'])

# Using tuple for student info
student_info = (101, 'Alice')

print('Student IDs:', student_ids)
print('Students:', students)
print('Courses:', courses)
print('Tuple Info:', student_info)
```