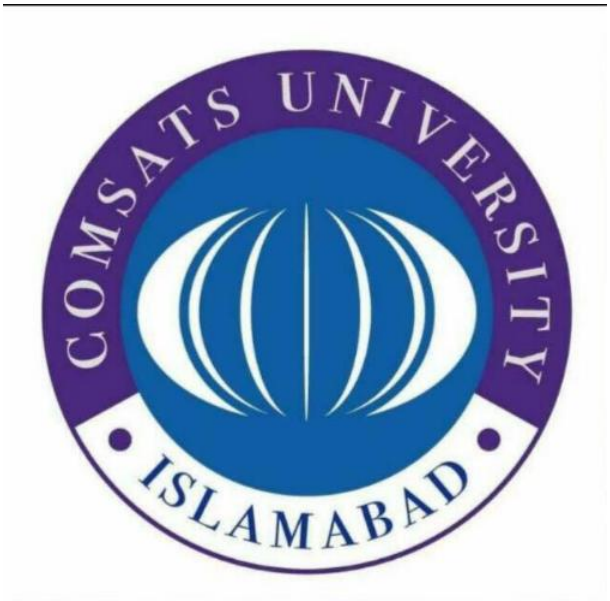
Comsats University Islamabad, Attock Campus

Department of Computer Science

Assignment No (LAB) : 1

Course Name: Machine Learning Fundamentals

Program: Computer Science



Submitted to:	**Sir Qasim Khan**
Submitted by:	**Abdul Basit**
Reg Number:	**SP24-BCS-033**

Graded Lab Task #01:

Questions & Solutions :

i. Write a Python program to create an instance of a specified class and display the namespace of the said instance.

=>

```
class Student:
    def __init__(self, name, age):
        self.name = name
        self.age = age


obj = Student("Ali", 21)
print(obj.__dict__)
```

```
{'name': 'Ali', 'age': 21}
```

ii. Write a Python function student_data() which will print the id of a student (student_id). If the user passes an argument student_name or student_class the function will print the student name and class.

=>

```
def student_data(student_id, student_name=None, student_class=None):
    print("Student ID:", student_id)
    if student_name:
        print("Student Name:", student_name)
```

```
    if student_class:
        print("Student Class:", student_class)


student_data(101)
student_data(102, "Basit", "BSCS")
```

```
Student ID: 101
Student ID: 102
Student Name: Basit
Student Class: BSCS
```

iii. Write a simple Python class named Student and display its type. Also, display the dict attribute keys and the value of the module attribute of the Student class.

=>

```
class Student:
    pass


print(type(Student))
print(Student.__dict__.keys())
print(Student.__module__)
```

```
<class 'type'>
dict_keys(['__module__', '__dict__', '__weakref__', '__doc__'])
__main__
```

iv. Write a Python program to create two empty classes, Student and Marks. Now create some instances and check whether they are instances of the said classes or not. Also, check whether the said classes are subclasses of the built-in object class or not.

=>

```
class Student:
    pass


class Marks:
    pass


s = Student()
m = Marks()


print(isinstance(s, Student))
print(isinstance(m, Marks))
print(issubclass(Student, object))
print(issubclass(Marks, object))
```

```
True
True
True
True
```

v. Write a Python class named Student with two attributes student_name, marks. Modify the attribute values of the said class and print the original and modified values of the said attributes.

=>

```
class Student:
    def __init__(self, student_name, marks):
        self.student_name = student_name
        self.marks = marks


s = Student("Ali", 80)
print("Original:", s.student_name, s.marks)


s.student_name = "Basit"
s.marks = 90
print("Modified:", s.student_name, s.marks)
```

```
Original: Ali 80
Modified: Basit 90
```

vi. Write a Python class named Student with two attributes student_id, student_name. Add a new attribute student_class and display the entire attribute and their values of the said class. Now remove the student_name attribute and display the entire attribute with values.

=>

```
class Student:
    def __init__(self, student_id, student_name):
```

```
        self.student_id = student_id
        self.student_name = student_name

s = Student(101, "Ali")
s.student_class = "BSCS"

print(s.__dict__)

del s.student_name
print(s.__dict__)
```

```
{'student_id': 101, 'student_name': 'Ali', 'student_class': 'BSCS'}
{'student_id': 101, 'student_class': 'BSCS'}
```

vii. Write a Python class named Student with two attributes student_id, student_name. Add a new attribute student_class. Create a function to display the entire attribute and their values in Student class.

=>

```
class Student:
    def __init__(self, student_id, student_name):
        self.student_id = student_id
        self.student_name = student_name

    def display(self):
        for key, value in self.__dict__.items():
            print(key, ":", value)
```

```
s = Student(102, "Basit")
s.student_class = "BSCS"
s.display()
```

```
student_id : 102
student_name : Basit
student_class : BSCS
```

viii. Write a Python class named Student with two instances student1, student2 and assign given values to the said instances attributes. Print all the attributes of student1, student2 instances with their values in the given format.

=>

```
class Student:
    pass


student1 = Student()
student2 = Student()


student1.student_id = 1
student1.student_name = "Ali"


student2.student_id = 2
student2.student_name = "Basit"


print("Student1 ID:", student1.student_id)
```

```
print("Student1 Name:", student1.student_name)
print("Student2 ID:", student2.student_id)
print("Student2 Name:", student2.student_name)
```

```
Student1 ID: 1
Student1 Name: Ali
Student2 ID: 2
Student2 Name: Basit
```

ix. Run Memory Allocation tests on the created Python class.

=>

```
import sys

class Student:
    def __init__(self, name, age):
        self.name = name
        self.age = age

s = Student("Ali", 21)
print(sys.getsizeof(s))
print(sys.getsizeof(s.__dict__))
```

```
48
232
```

x. Try reducing the memory allocation using different techniques.

=>

```
import sys


class Student:
    __slots__ = ['name', 'age']
    def __init__(self, name, age):
        self.name = name
        self.age = age


s = Student("Ali", 21)
print(sys.getsizeof(s))
```

```
48
```

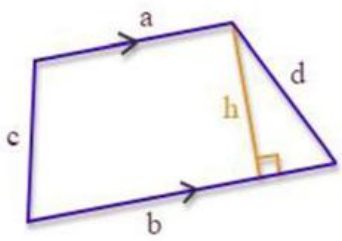
Graded Lab Task #02:

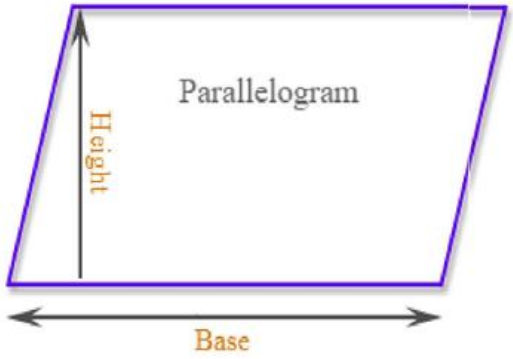
Question :

i. Make two classes for trapezoid and parallelogram and write a code to calculate their areas.
ii. Also write a comparison class to see which shape has the largest area, store results in a dictionary, iterate for different dimensions and use dictionary functions to search the shapes with largest areas.

Area of a Trapezoid

$$A = \frac{1}{2}(a + b)h$$





Area of parallelogram = Base x Height

Solution :

Code :

```
class Shape:
    def area(self):
        raise NotImplementedError
```

```
class Trapezoid(Shape):
    def __init__(self, a, b, h):
        self.a = a
        self.b = b
        self.h = h

    def area(self):
        return 0.5 * (self.a + self.b) * self.h


class Parallelogram(Shape):
    def __init__(self, base, height):
        self.base = base
        self.height = height

    def area(self):
        return self.base * self.height


class ShapeComparison:
    def __init__(self):
        self.results = {}

    def add_shape(self, name, shape):
        self.results[name] = shape.area()
```

```
    def display_all(self):
        for name, area in self.results.items():
            print(f"{name} -> Area = {area}")

    def find_largest(self):
        largest = max(self.results, key=self.results.get)
        return largest, self.results[largest]


shapes = {
    "Trapezoid1": Trapezoid(4, 6, 5),
    "Trapezoid2": Trapezoid(3, 7, 4),
    "Parallelogram1": Parallelogram(5, 6),
    "Parallelogram2": Parallelogram(8, 3)
}

comparison = ShapeComparison()

for name, shape in shapes.items():
    comparison.add_shape(name, shape)

comparison.display_all()

largest_shape, largest_area = comparison.find_largest()

print("\nLargest Shape:")
```

print(f"{largest_shape} -> Area = {largest_area}")

Output :