Al-Mustaqbal University
College of Science
Intelligent Medical System Department

جامعة المستقبل
AL MUSTAQBAL UNIVERSITY

كلية العلوم
قسم الانظمة الطبية الذكية

Lecture: (6)

Examples (python classes and objects)

Subject: Object oriented programming II

Class: Second

Lecturer: Dr. Maytham N. Meqdad

Examples (python classes and objects)

1. Creating a Simple Class and Object:

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

    def bark(self):
        print(f"{self.name} says Woof!")

# Create objects of the Dog class
dog1 = Dog("Buddy", 2)
dog2 = Dog("Molly", 4)

# Access object attributes and methods
print(f"{dog1.name} is {dog1.age} years old.")
dog2.bark()
```

2. Bank Account Class:

```
def deposit(self, amount):
        self.balance += amount

    def withdraw(self, amount):
        if amount <= self.balance:
            self.balance -= amount
        else:
            print("Insufficient funds.")

    def get_balance(self):
        return self.balance

# Create a bank account object and perform transactions
account = BankAccount("12345", 1000)
account.deposit(500)
account.withdraw(200)
print(f"Account balance: ${account.get_balance()}")
```

3. Car Class with Inheritance:

```
class Vehicle:
    def __init__(self, make, model):
        self.make = make
        self.model = model

    def display_info(self):
        print(f"Make: {self.make}, Model: {self.model}")

class Car(Vehicle):
    def __init__(self, make, model, year):
        super().__init__(make, model)
        self.year = year

    def display_info(self):
        print(f"Make: {self.make}, Model: {self.model}, Year: {self.year}")

# Create car objects and call methods
car1 = Car("Toyota", "Camry", 2022)
car2 = Car("Honda", "Civic", 2021)
car1.display_info()
car2.display_info()
```

4. Student Class with Multiple Objects:

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

    def display_info(self):
        print(f"Name: {self.name}, Age: {self.age}")

# Create a list of student objects and display their information
students = [Student("Alice", 20), Student("Bob", 22), Student("Charlie", 19)]

for student in students:
    student.display_info()
```

5. Rectangle Class:

```
class Rectangle:
    def __init__(self, width, height):
        self.width = width
        self.height = height

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

    def perimeter(self):
        return 2 * (self.width + self.height)

# Create rectangle objects and calculate area and perimeter
rect1 = Rectangle(5, 3)
rect2 = Rectangle(8, 4)

print("Rectangle 1 - Area:", rect1.area(), "Perimeter:", rect1.perimeter())
print("Rectangle 2 - Area:", rect2.area(), "Perimeter:", rect2.perimeter())
```

6. Bank Customer Class with Account Management:

```
python
class BankCustomer:
    def __init__(self, name):
        self.name = name
        self.accounts = {}

    def add_account(self, account_name, balance):
        self.accounts[account_name] = balance

    def display_accounts(self):
        print(f"Accounts for {self.name}:")
        for account, balance in self.accounts.items():
            print(f"{account}: ${balance:.2f}")

# Create a bank customer, add accounts, and display account information
customer = BankCustomer("Alice")
customer.add_account("Savings", 1000)
customer.add_account("Checking", 500)
customer.display_accounts()
```

7. Circle Class with Method Overriding:

```
python
import math

class Circle:
    def __init__(self, radius):
        self.radius = radius

    def area(self):
        return math.pi * self.radius ** 2

    def perimeter(self):
        return 2 * math.pi * self.radius

    def display_info(self):
        print(f"Circle - Radius: {self.radius:.2f}, Area: {self.area():.2f},
Perimeter: {self.perimeter():.2f}")

class ColoredCircle(Circle):
    def __init__(self, radius, color):
        super().__init__(radius)
        self.color = color

    def display_info(self):
        print(f"Colored Circle - Radius: {self.radius:.2f}, Area:
{self.area():.2f}, Perimeter: {self.perimeter():.2f}, Color: {self.color}")

# Create circle objects and call methods
circle1 = Circle(5)
circle2 = ColoredCircle(3, "Red")

circle1.display_info()
circle2.display_info()
```