پوهنتون کاردان
KARDAN UNIVERSITY

MODERN PROGRAMMING LANGUAGE
(Python)

Classes and Objects

Learning Outcomes

- Learn OOP concepts briefly
- Know about Classes and instances
- Can Add methods to class
- Learn Using __init__ function
- Know how about Inheritance
- Understand Polymorphism

OOP

- OOP offers a different and often more effective way of programming,
- We factor code to minimize redundancy,
- classes are created with a new statement: the class.

- Example:

```
class my_class():

    print("this is my class")
```

```
this is my class
```

Classes and instances

- The primary difference between classes and instances is that classes are a kind of *factory* for generating instances.
- With classes, we can make as many instances as we need.
- We can create instance of a class by calling class names.
- Example:

```
#creating instance of the my_class()

instance=my_class()
```

```
instance1=my_class()
instance2=my_class()
instance3=my_class()
```

Adding methods to classes

- We can add multiple functions within the class, and call them after creating object or instance of the class.
- Calling the instance name followed by "." we can call a function and pass the values to it.
- Example:

```
class Car():

    def move(self):

        print("The car is moving")
```

```
instance.move()
```

```
The car is moving
```

```
class Car():

    def setSpeed(self,speed):

        print("The car is moving at speed: ", speed)
```

```
instance.setSpeed(120)
```

```
The car is moving at speed:  120
```

Practice

- Create a class for the Bank Account named as :Account
- Add the deposit and withdraw functions into it.
- Call the deposit function and deposit the 45,000 AFN into it.
- Call the withdraw function and get the 10,000 from it.

Adding attributes using init function

- Python automatically calls a method named __init__ each time an instance is generated from a class.
- The new instance is passed in to the self argument of __init__ as usual,
- Example:

```
class Student():

    def __init__(self, name, marks):
        self.name=name
        self.marks=marks
```

```
student1=Student('Ahmad', 88)
```

```
student1.marks
```

```
88
```

Contd..

```
class Student():

    def __init__(self, name, marks):
        self.name=name
        self.marks=marks

    def printMarks(self):

        print(f"Name is: {self.name} and Marks is: {self.marks}")
```

```
student1=Student('Ahmad', 88)
```

```
student1.printMarks()
```

```
Name is: Ahmad and Marks is: 88
```

Practice

- For the previous class Account, add an __init__ function which can carry the following attributes:
  - Name, Father Name, Phone, ID card number and balance
- Create an account for the following person and print the user information using a function named, userInfo()
- Create a function called checkBalance, to print the current balance of employee

Inheritance

- We can use or get access to the classes' attributes and methods which are already made.
- This is called the reusability.
- We can inherit a class as follows:

```
class Car():

    def __init__(self):
        print("Car Created")
    def move(self):
        print("Car is moving")
```

```
class Toyota(Car):

    def __init__(self,model,color):
        Car.__init__(self)
        self.model=model
        self.color=color
```

```
LandCruiser=Toyota('2010',"Red")
```

```
Car Created
```

```
LandCruiser.move()
```

```
Car is moving
```

Polymorphism in python

- Polymorphism in python refers to a feature that to give different implementation for a superclass behavior of a function.
- For a super class employee we can have common methods used for all the employees, but the implementation details may differ for different instances.
- Example:

```
class Car():

    def __init__(self, model):
        self.model=model
        print("Car Created")
    def fill(self):
        print("Create an object of Car class")
```

```
class Corolla(Car):

    def fill(self):
        print(f"{self.model} can take Gas only")
```

```
m2010=Corolla("ED 2010 ")
```

```
Car Created
```

```
m2010.fill()
```

```
ED 2010  can take Gas only
```

Practice

Using inheritance and polymorphism, implement the following rules on different instances:

- Suppose you have a class “Bank_Account” having two methods: Deposit, Withdraw
  - There are three other kinds of Bank_Accounts
    - Saving_Account
    - Employee_Account
    - Current_Account
- There are some terms and condition with each class:
- In saving account you get 3% extra money when you are depositing
- In Employee account it is deducting the tax when the organization deposits in to your account according to the government tax policy.
  - Less than 10000 no tax
  - More than 10000, 5% tax
- For the current account you cannot withdraw more than 10000 Afn in a day.

Thank You!