CYS Programming with python

Session 4

Done By: Maryam Hajeb

01 Object-Oriented Programming (OOP)

- A programming paradigm built around objects.
- Objects combine data (attributes) and behavior (methods)
- Helps organize large programs
- Improves code reuse, readability, and maintenance.
- Python is fully object-oriented – everything is an object.

02 Classes in Python

- A class is a blueprint for objects
- Defines data (attributes) + behavior (methods)
- Created using the **class** keyword
- First string inside class = docstring
- Each class has its own namespace

03 Objects (Instances)

- An object is an instance of a class
- Created via: object_name = ClassName()
- Objects store unique instance attributes
- Objects access class functionality via methods

class.py

```
class ClassDemo:
        """This is a docstring"""
        pass

obj=ClassDemo()
```

04 Class Attributes

- Class attributes: shared among all objects
- Special attributes start with __
  - Examples: __doc__, __name__, __module__
- Class and special attributes are accessed using ClassName.attribute
- Instance Attributes: attributes defined inside __init__()
- Are unique for each object and accessed via self.attribute

05 Class Methods

- Methods are functions inside a class.
- Python automatically passes the object as first argument.

```
def methodName(self):
    #method block
```

06 Class Constructors

- Constructor (__init__) is a special method called automatically when object is created.
- Used to initialize instance attributes.
- Can accept default values

```
def __init__(self, at1=0, at2=0):
    self.attr1 = at1
    self.attr2 = at2
```

classExample.py

```
1   class Employee:
2
3       """Represents an employee in a company"""
4       def __init__(self, name, age, job_title, hourly_rate):
5
6           self.name = name
7
8           self.age = age
9
10          self.job_title = job_title
11
12          self.hourly_rate = hourly_rate
13
14          self.hours_worked = 0
      def log_hours(self, hours):
          self.hours_worked += hours
```

classExample.py

```
15
16    def calculate_salary(self):
17            return self.hours_worked * self.hourly_rate
18
19        def display_info(self):
20            print(f"Employee: {self.name}")
21            print(f"Age: {self.age}")
22            print(f"Job Title: {self.job_title}")
23            print(f"Hourly Rate: ${self.hourly_rate}")
24            print(f"Hours Worked: {self.hours_worked}")
25
26
27
28
```

classExample.py

```
29
30  emp1 = Employee("Sarah Ahmed", 28, "Software Engineer", 45)
31  emp2 = Employee("Ali Hassan", 35, "Project Manager", 60)
32
33
34  emp1.log_hours(160)
35  emp1.display_info()
36
37  print("Monthly Salary:", emp1.calculate_salary())
38
39  print()
40
41  emp2.log_hours(150)
42  emp2.display_info()
    print("Monthly Salary:", emp2.calculate_salary())
```

Lab Task

- Create a ClassCreate a class named **Product** that includes:
  - A constructor (__init__) with:

    name - price - quantity
  - Three methods:
    - add_stock(amount) : increases quantity
    - sell(amount) : decreases quantity
    - product_value() : returns total value (price × quantity)
    - show_info() : prints all product details

Lab Task

- Create two different product objects and:
  - Add stock to one product
  - Sell some quantity from another
  - Display the updated information for both products
  - Print the total value for each product

Thanks!

Do you have any questions?