PyCharm IDE

Lesson Plan

Subject/Course	Python Programming
Lesson Title	PyCharm IDE

Lesson Objectives
About PyCharm IDE
Git Integration with PyCharm
PyTest Framework and Python Database Connectivity

What is an IDE?

- An IDE (Integrated Development Environment) provides a complete environment for software development.
- It includes:
- - Code Editor
- - Debugger
- - Build Automation Tools
- - Version Control Integration

Introduction to PyCharm

- PyCharm, developed by JetBrains, is one of the most popular IDEs for Python.
- Features include:
- Smart code completion
- Real-time error detection
- Built-in terminal and debugger
- Integration with Git, virtual environments, and frameworks.

Installing PyCharm

- Steps to install PyCharm:
- 1. Visit jetbrains.com/pycharm
- 2. Download Community or Professional Edition
- 3. Install and launch
- 4. Configure interpreter and project settings

Starting a New Project

- When creating a new project in PyCharm:
- 1. Choose project name and location
- 2. Select Python interpreter
- 3. Click 'Create'
- PyCharm automatically sets up folders and virtual environments.

PyCharm User Interface

- Main sections of the PyCharm interface:
- - Project Explorer: View all files
- - Editor: Write and edit code
- - Run/Debug Panel: Execute programs
- - Terminal: Access command line inside IDE

Writing Your First Python Script

- Example code below:

```
print('Hello, PyCharm!')
```

Code Assistance

- PyCharm provides intelligent features such as:
- - Code completion
- - Syntax highlighting
- - Quick fixes

```
def greet(name):
    print(f'Hello, {name}!')
```

Running Code in PyCharm

- To run code:
- 1. Click the green Run icon
- 2. Use shortcut Shift + F10
- 3. View output in the Run window

Debugging in PyCharm

- Set breakpoints and debug step-by-step using Shift + F9.

```
def add(a, b):
    return a + b

print(add(2, 3))
```

Working with Virtual Environments

- PyCharm automatically creates virtual environments for projects.
- This ensures dependencies are isolated.

```
python -m venv venv
```

Managing Packages

- Use PyCharm’s built-in package manager:
- - Navigate to File → Settings → Project → Python Interpreter
- - Add, upgrade, or remove packages.

```
pip install requests
```

Using Git and Version Control

- PyCharm supports Git integration:
- - Clone repositories
- - Commit changes
- - Push to GitHub

Testing in PyCharm

- PyCharm supports unittest and pytest frameworks.

```
import unittest

class TestMath(unittest.TestCase):
    def test_add(self):
        self.assertEqual(2 + 3, 5)

unittest.main()
```

Advanced Python Development Tools & Integrations

Advanced Python Development Tools & Integrations

- A concise overview of modern Python development tools:
- - Git Integration with PyCharm IDE
- - PyTest Framework
- - Database Connectivity with MySQL and MongoDB

Part 1: Git Integration with PyCharm

- Git is a version control system used to track code changes and collaborate with others.

- PyCharm provides built-in Git support for easier source code management.

Setting up Git in PyCharm

1. Install Git on your system
2. Open PyCharm → Settings → Version Control → Git
3. Configure Git executable path
4. Test connection to verify setup

Basic Git Operations

- In PyCharm, you can perform all Git actions directly:
- Initialize repository
- Commit changes
- Push & Pull
- Manage branches

Common Git Commands

- Some essential Git commands are shown below:

```
git init
git add .
git commit -m 'Initial commit'
git push origin main
```

Part 2: PyTest Framework

- PyTest is a Python testing framework that makes it easy to write, run, and organize tests.

- It is simple yet powerful for both unit and functional testing.

Installing and Running PyTest

- Install PyTest using pip and run your tests easily from terminal or PyCharm.

```
pip install pytest
pytest test_sample.py
```

Writing Your First Test

- Here's a simple PyTest example:

```
def test_add():
   assert 2 + 3 == 5
```

PyTest Fixtures

- Fixtures allow you to provide setup and teardown logic for your tests.
- They help reuse code and prepare test data efficiently.

Part 3: Python Database Connectivity

- Python supports database interaction with both SQL and NoSQL systems.
- We'll explore MySQL (SQL) and MongoDB (NoSQL) with basic CRUD examples.

MySQL Connection

- Connect Python with MySQL using the mysql.connector library.
- Perform CRUD operations easily through SQL queries.

CRUD Operations in MySQL

- Example Python code for CRUD operations:

```
import mysql.connector

conn = mysql.connector.connect(
   host='localhost', user='root', password='1234', database='testdb')
cur = conn.cursor()
cur.execute('INSERT INTO students VALUES (1, "John")')
conn.commit()
```

MongoDB Connection

- MongoDB is a NoSQL database that stores data as JSON-like documents.

- Use the pymongo library for connecting and performing CRUD operations.

MongoDB CRUD Example

- Example MongoDB operations using pymongo:

```
from pymongo import MongoClient

client = MongoClient('mongodb://localhost:27017/')
db = client['school']
col = db['students']
col.insert_one({'name': 'Alice', 'age': 22})
for doc in col.find():
    print(doc)
```

Python Development Tools Summary

- 🔹 Git Integration with PyCharm IDE**
- - Version control directly inside PyCharm
- - Git operations: commit, push, pull, branch management
- - Simplifies teamwork and project tracking

- 🔹 PyTest Framework
- - Easy, powerful testing framework
- - Fixtures, assertions, parametrization for efficient test automation
- - Simple syntax and detailed reporting

- 🔹 Database Connectivity (MySQL & MongoDB)
- - MySQL for structured, relational data
- - MongoDB for flexible, document-based data
- - CRUD operations: Create, Read, Update, Delete

- 💡 Together, these tools streamline the entire Python development workflow: from coding and version control to testing and database integration.

SIX PHRASE
Democratizing Skilling
a Veranda Enterprise

Thank You