Programming in Python with Full Stack Development (303105257)

Anand Jawdekar

Assistant Professor CSE Department

Unit-3

Modules and Packages

Unit-3

Modules and Packages:

Working with modules and packages in Python

Introduction to popular Python libraries for specific tasks, such as data analysis, web development, or game development.

PyCharm IDE : GIT- Git Integration with PyCharm IDE, PyTests. Python connectivity with Databases MYSQL, MongoDB CRUD operations.

What is Modular Programming?

Modular programming is the practice of segmenting a single, complicated coding task into multiple, simpler, easier-to-manage sub-tasks. We call these subtasks modules. Therefore, we can build a bigger program by assembling different modules that act like building blocks.

Modularizing our code in a big application has a lot of benefits.

Simplification: A module often concentrates on one comparatively small area of the overall problem instead of the full task. We will have a more manageable design problem to think about if we are only concentrating on one module. Program development is now simpler and much less vulnerable to mistakes.

Flexibility: Modules are frequently used to establish conceptual separations between various problem areas. It is less likely that changes to one module would influence other portions of the program if modules are constructed in a fashion that reduces interconnectedness. (We might even be capable of editing a module despite being familiar with the program beyond it.) It increases the likelihood that a group of numerous developers will be able to collaborate on a big project.

What is Modular Programming?

Reusability: Functions created in a particular module may be readily accessed by different sections of the assignment (through a suitably established api). As a result, duplicate code is no longer necessary.

Scope: Modules often declare a distinct namespace to prevent identifier clashes in various parts of a program.

In Python, modularization of the code is encouraged through the use of functions, modules, and packages.

What are Modules in Python?

A document with definitions of functions and various statements written in Python is called a Python module.

In Python, we can define a module in one of 3 ways:

- Python itself allows for the creation of modules.
- Similar to the re (regular expression) module, a module can be primarily written in C programming language and then dynamically inserted at run-time.
- A built-in module, such as the itertools module, is inherently included in the interpreter.

A module is a file containing Python code, definitions of functions, statements, or classes. An example_module.py file is a module we will create and whose name is example_module.

We employ modules to divide complicated programs into smaller, more understandable pieces. Modules also allow for the reuse of code.

Rather than duplicating their definitions into several applications, we may define our most frequently used functions in a separate module and then import the complete module.

Example:

```
# Here, we are creating a simple Python program to show how to create a module.
# defining a function in the module to reuse it
def square( number ):
   # here, the above function will square the number passed as the input
    result = number ** 2
    return result     # here, we are returning the result of the function
```

Here, a module called example_module contains the definition of the function square(). The function returns the square of a given number.

How to Import Modules in Python?

In Python, we may import functions from one module into our program, or as we say into, another module. For this, we make use of the import Python keyword. In the Python window, we add the next to import keyword, the name of the module we need to import. We will import the module we defined earlier example_module.

Syntax:

import *example_module*

The functions that we defined in the example_module are not immediately imported into the present program. Only the name of the module, i.e., example_ module, is imported here.

We may use the dot operator to use the functions using the module name. For instance:

Example:

```
# here, we are calling the module square method and passing the value 4
result = example_module.square(  4  )
print("By using the module square of number is: ", result )
```

Output:

```
By using the module square of number is: 16
```

Python import Statement

Using the import Python keyword and the dot operator, we may import a standard module and can access the defined functions within it. Here's an illustration.

Code

```python
# Here, we are creating a simple Python program to show how to import a standard module
# Here, we are import the math module which is a standard module
import math
print( "The value of euler's number is", math.e )
# here, we are printing the euler's number from the math module
```

Output:

```
The value of euler's number is 2.718281828459045
```

Importing and also Renaming

While importing a module, we can change its name too. Here is an example to show.

Code

```
# Here, we are creating a simple Python program to show how to import a module and rename it
# Here, we are import the math module and give a different name to it
import math as mt    # here, we are importing the math module as mt
print( "The value of euler's number is", mt.e )
# here, we are printing the euler's number from the math module
```

Output:

```
The value of euler's number is 2.718281828459045
```

The math module is now named mt in this program. In some situations, it might help us type faster in case of modules having long names.

Please take note that now the scope of our program does not include the term math. Thus, mt.pi is the proper implementation of the module, whereas math.pi is invalid.

Python from...import Statement

We can import specific names from a module without importing the module as a whole. Here is an example.

Code

```
# Here, we are creating a simple Python program to show how to import specific
# objects from a module
# Here, we are import euler's number from the math module using the from keyword
from math import e
# here, the e value represents the euler's number
print( "The value of euler's number is", e )
```

Output:

```
The value of euler's number is 2.718281828459045
```

Only the e constant from the math module was imported in this case.

We avoid using the dot (.) operator in these scenarios. As follows, we may import many attributes at the same time:

Python from...import Statement

Code

```
# Here, we are creating a simple Python program to show how to import multiple
# objects from a module
from math import e, tau
print( "The value of tau constant is: ", tau )
print( "The value of the euler's number is: ", e )
```

Output:

```
The value of tau constant is: 6.283185307179586 The value of the euler's number
is: 2.718281828459045
```

Import all Names - From import * Statement

To import all the objects from a module within the present namespace, use the * symbol and the from and import keyword.

Syntax:

from name_of_module **import** *

There are benefits and drawbacks to using the symbol *. It is not advised to use * unless we are certain of our particular requirements from the module; otherwise, do so.

Here is an example of the same.

Code

```
# Here, we are importing the complete math module using *
from math import *
# Here, we are accessing functions of math module without using the dot operator
print( "Calculating square root: ", sqrt(25) )
# here, we are getting the sqrt method and finding the square root of 25
print( "Calculating tangent of an angle: ", tan(pi/6) )
# here pi is also imported from the math module
```

Output:

```
Calculating square root: 5.0 Calculating tangent of an angle: 0.5773502691896257
```

Python Packages

Python Packages are a way to organize and structure your Python code into reusable components. Think of it like a folder that contains related Python files (modules) that work together to provide certain functionality. Packages help keep your code organized, make it easier to manage and maintain, and allow you to share your code with others. They're like a toolbox where you can store and organize your tools (functions and classes) for easy access and reuse in different projects.

How to Create Package in Python?

Creating packages in Python allows you to organize your code into reusable and manageable modules. Here’s a brief overview of how to create packages:

Create a Directory: Start by creating a directory (folder) for your package. This directory will serve as the root of your package structure.

Add Modules: Within the package directory, you can add Python files (modules) containing your code. Each module should represent a distinct functionality or component of your package.

Init File: Include an __init__.py file in the package directory. This file can be empty or can contain an initialization code for your package. It signals to Python that the directory should be treated as a package.

Subpackages: You can create sub-packages within your package by adding additional directories containing modules, along with their own __init__.py files.

Importing: To use modules from your package, import them into your Python scripts using dot notation. For example, if you have a module named module1.py inside a package named mypackage, you would import its function like this: from mypackage.module1 import greet.

Distribution: If you want to distribute your package for others to use, you can create a setup.py file using Python’s setuptools library. This file defines metadata about your package and specifies how it should be installed.

Code Example

Here's a basic code sample demonstrating how to create a simple Python package:

1. Create a directory named mypackage.
2. Inside mypackage, create two Python files: module1.py and module2.py.
3. Create an __init__.py file inside mypackage (it can be empty).
4. Add some code to the modules.
5. Finally, demonstrate how to import and use the modules from the package.

```
mypackage/
|
├── __init__.py
├── module1.py
└── module2.py
```

Code Example

```
# module1.py
def greet(name):
    print(f"Hello, {name}!")
# module2.py
def add(a, b):
    return a + b
from mypackage import module1, module2

# Using functions from module1
module1.greet("Alice")
# Using functions from module2
result = module2.add(3, 5)
print("The result of addition is:", result)
```

Python Packages for Web frameworks

Flask: Flask is a lightweight and flexible web framework for Python. It's designed to make getting started with web development in Python quick and easy, with a simple and intuitive interface. Flask provides tools and libraries to help you build web applications, APIs, and other web services.
Django: Django is a Python web framework for building web applications quickly and efficiently. It follows the DRY principle and includes features like URL routing, database management, and authentication, making development easier. It's highly customizable and widely used in web development.
FastAPI: Python FastAPI is a high-performance web framework for building APIs quickly and efficiently. It's easy to use, based on standard Python-type hints, and offers automatic interactive documentation. FastAPI is designed to be fast, easy to learn, and ideal for building modern web APIs.
Pyramid: Python Pyramid is a lightweight web framework for building web applications in Python. It emphasizes flexibility, allowing developers to choose the components they need while providing powerful features for handling HTTP requests, routing, and templating.

Python Packages for Web frameworks

Falcon: Python Falcon is a lightweight web framework designed for building high-performance APIs quickly and easily. It focuses on simplicity, speed, and minimalism, making it ideal for creating RESTful APIs with minimal overhead.

CherryPy: CherryPy is a minimalist Python web framework for building web applications. It provides a simple and intuitive interface for handling HTTP requests, allowing developers to focus on their application logic without dealing with the complexities of web server management.

Bottle: Python Bottle is a lightweight web framework for building small web applications in Python with minimal effort and overhead. It's designed to be simple and easy to use, making it great for prototyping and creating simple APIs or web services.

Web2py: Web2py is a free open-source web framework for agile development of secure database-driven web applications. It's written in Python and offers features like an integrated development environment (IDE), simplified deployment, and support for multiple database backends.

Python Packages for AI & Machine Learning

Statistical Analysis

From data manipulation to machine learning and visualization, these tools offer powerful capabilities for analyzing data effectively.

NumPy
Pandas
SciPy
XGBoost
StatsModels
Yellowbrick
Arch
Dask-ML

Parul University

Python Packages for Game Development

Here, we'll explore the exciting world of game development in Python, leveraging powerful packages and libraries to bring your gaming ideas to life. Let's dive in and discover the tools that will empower you to create immersive and entertaining gaming experiences.

PyGame: PyGame is a set of libraries and tools for creating video games and multimedia applications using Python. It provides functions for handling graphics, sound, input devices, and more, making it easier to develop games with Python.

Panda3D: Python Panda3D is a game development framework that provides tools and libraries for creating 3D games and simulations using the Python programming language. It offers features for rendering graphics, handling input, and managing assets, making it suitable for both hobbyists and professional game developers.

Pyglet: Pyglet is a Python library used for creating games and multimedia applications. It provides tools for handling graphics, sound, input devices, and windowing. With Pyglet, developers can build interactive experiences efficiently in Python.

Library in Python

In recent years, Python's popularity has increased dramatically, kudos to its widespread application in, among other fields, data science, software engineering, and machine learning. The large number of libraries that Python offers is one of the reasons for its popularity.

Each of Python's open-source libraries has its own source code. A collection of code scripts that can be used iteratively to save time is known as a library. It is like a physical library in that it has resources that can be used again, as the name suggests.

A collection of modules that are linked together is also known as a Python library. It has code bundles that can be used again and again in different programs. For programmers, it makes Python programming easier and simpler. Since then, we will not need to compose the same code for various projects. Python libraries are heavily used in a variety of fields, including data visualization, machine learning, and computer science.

How Python Libraries work?

a Python library is nothing more than a collection of code scripts or modules of code that can be used in a program for specific operations. We use libraries to avoid having to rewrite existing program code. However, the process is as follows: In the MS Windows environment, the library files have a DLL (Dynamic Load Libraries) extension. The linker automatically looks for a library when we run our program and import it. It interprets the program in accordance with the functions extracted from the library.

Standard Libraries of Python

The Python Standard Library contains all of Python's syntax, semantics, and tokens. It has built-in modules that allow the user to access I/O and a few other essential modules as well as fundamental functions. The Python libraries have been written in the C language generally. The Python standard library has more than 200 core modules. Because of all of these factors, Python is a powerful programming language. The Python Standard Library plays a crucial role. Python's features can only be used by programmers who have it. In addition, Python has a number of libraries that make

Popular Python Libraries for Game Development

Python is widely used in game development for prototyping, building simple games, and developing game logic. Below are some of the most popular libraries and frameworks used for game development:

. Pygame

Description: A cross-platform library for creating video games. It includes computer graphics and sound libraries.

Key Features:

- Easy to use for beginners.
- Handles sound, graphics, and game loops.
- Ideal for 2D game development.

Example Games: Asteroids clone, Snake game.

Example

```
pip install pygame
import pygame
pygame.init()
screen = pygame.display.set_mode((400, 300))
pygame.display.set_caption('My First Game')
running = True

while running:
    for event in pygame.event.get():
        if event.type == pygame.QUIT:
            running = False

pygame.quit()
```

2. Panda3D

Description: An open-source game engine developed by Disney for 3D rendering and game development.

Key Features:

- Suitable for both 3D and 2D games.
- Built-in physics engine.
- Great for rendering complex graphics.

3. Godot (with Python scripting)

Description: Open-source game engine supporting both 2D and 3D games.

Key Features:

- Visual scripting and Python-like scripting language (GDScript).
- Lightweight and efficient.
- Active community support.

4. PyOpenGL

Description: A Python wrapper for the OpenGL API used for 3D graphics rendering.

Key Features:

- Direct access to GPU rendering.
- Supports complex 3D graphics.
- Ideal for creating 3D environments and visual effects.

5. Arcade

Description: A Python library for creating 2D games with an easy-to-learn API.

Key Features:

- Ideal for small games and educational purposes.
- Simple API for drawing shapes, sprites, and animations.
- Better performance for 2D rendering than Pygame.

Example

```
pip install arcade
import arcade

arcade.open_window(400, 300, "Simple Game")
arcade.start_render()
arcade.draw_circle_filled(200, 150, 50, arcade.color.BLUE)
arcade.finish_render()
arcade.run()
```

6. Ren'Py

Description: Specialized library for creating visual novels and storytelling games.

Key Features:

- Script-driven engine.
- Ideal for interactive stories.
- Built-in text display and image handling.

7. Pyglet

Description: A library for windowing and multimedia. Good for small games and prototypes.

Key Features:

- Handles audio, images, and video.
- Lightweight and easy to integrate.
- Supports OpenGL for rendering.

Installation:

```
pip install pyglet
```

Popular Python Libraries for Data Analysis

1. Pandas

Purpose: Data manipulation and analysis.

Key Features:

- DataFrames and Series objects for structured data.
- Handling missing data.
- Powerful group-by and aggregation functions.

Installation: pip install pandas

```
import pandas as pd
data = {'Name': ['Alice', 'Bob'], 'Age': [25, 30]}
df = pd.DataFrame(data)
print(df)
```

Popular Python Libraries for Data Analysis

. NumPy (Numerical Python)

Purpose: Numerical computations and array operations.

Key Features:

- Multi-dimensional array object (ndarray).
- Mathematical and logical operations on arrays.
- Linear algebra, Fourier analysis, and matrix computations.

Installation: pip install numpy

```
import numpy as np

arr = np.array([1, 2, 3, 4])
print(arr * 2)
```

Parul University

DIGITAL LEARNING CONTENT

Popular Python Libraries for Data Analysis

3. Matplotlib

Purpose: Data visualization through static, animated, and interactive plots.

Key Features:

- Line plots, scatter plots, and bar charts.
- Customization of plots (labels, legends, colors).
- Save plots in various formats.

Installation: pip install matplotlib

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4]
y = [10, 20, 25, 30]
plt.plot(x, y)
plt.show()
```

Popular Python Libraries for Data Analysis

4. Seaborn

Purpose: Statistical data visualization built on top of Matplotlib.

Key Features:

- Beautiful and informative statistical graphs.
- Integration with Pandas DataFrames.
- Plot distributions, regression models, and heatmaps.

Installation: pip install seaborn

```
import seaborn as sns
import pandas as pd

data = pd.DataFrame({'x': [1, 2, 3], 'y': [4, 5, 6]})
sns.barplot(x='x', y='y', data=data)
```

Parul University

DIGITAL LEARNING CONTENT

Popular Python Libraries for Data Analysis

5. SciPy (Scientific Python)

Purpose: Advanced scientific computing.

Key Features:

- Optimization, integration, and interpolation.
- Signal and image processing.
- Statistical functions and analysis.

Installation: pip install scipy

```
from scipy.stats import norm

print(norm.pdf(0))
```

Popular Python Libraries for Data Analysis

6. Plotly

Purpose: Interactive and web-based data visualization.

Key Features:

- Interactive plots and dashboards.
- Supports various chart types (scatter, bar, pie).
- Integration with Dash for web applications.

Installation: pip install plotly

```
import plotly.express as px

fig = px.bar(x=[1, 2, 3], y=[4, 5, 6])
fig.show()
```

Parul University

DIGITAL LEARNING CONTENT

Popular Python Libraries for Data Analysis

7. Statsmodels

Purpose: Statistical modeling and hypothesis testing.

Key Features:

- Regression models.
- Time series analysis.
- Statistical tests (t-tests, ANOVA).

Installation: pip install statsmodels

```
import statsmodels.api as sm

data = [1, 2, 3, 4, 5]
model = sm.OLS(data, [1, 2, 3, 4, 5]).fit()
print(model.summary())
```

Popular Python Libraries for Data Analysis

TensorFlow & PyTorch

Purpose: Machine learning and deep learning.

TensorFlow: For end-to-end machine learning pipelines.

PyTorch: For flexible deep learning model building.

Installation:

```
pip install tensorflow
pip install torch
```

Popular Python Libraries for Web Development

Python is a versatile language widely used in web development, offering powerful libraries and frameworks for building web applications, APIs, and more.

1. Django

Type: Full-Stack Web Framework

Purpose: Build secure, scalable, and robust web applications quickly.

Key Features:

- Built-in admin interface
- ORM (Object-Relational Mapping)
- Authentication and security features

Installation: pip install django

```
from django.http import HttpResponse

def home(request):
    return HttpResponse("Hello, Django!")
```

Parul University

DIGITAL LEARNING CONTENT

Popular Python Libraries for Web Development

2. Flask

Type: Micro Web Framework

Purpose: Lightweight framework for building small to medium-sized web apps and APIs.

Key Features:

- Simple and flexible architecture
- Support for Jinja2 templating
- Integrated unit testing

Installation: pip install flask

```
from flask import Flask
app = Flask(__name__)
@app.route('/')
def home():
   return "Hello, Flask!"
if __name__ == '__main__':
   app.run()
```

Popular Python Libraries for Web Development

3. FastAPI

Type: Web Framework for APIs

Purpose: Build fast and efficient APIs with automatic documentation.

Key Features:

- Automatic Swagger documentation
- Asynchronous support
- Type hints for data validation

Installation: pip install fastapi uvicorn

```
from fastapi import FastAPI

app = FastAPI()

@app.get("/")
def read_root():
    return {"message": "Hello, FastAPI!"}
```

Popular Python Libraries for Web Development

4. Pyramid

Type: Full-Stack Web Framework

Purpose: Build complex applications with flexibility and modular design.

Key Features:

- Highly customizable architecture
- Supports both small and large applications
- Built-in authentication system

Installation: pip install pyramid

Example

```
from pyramid.config import Configurator
from wsgiref.simple_server import make_server

def hello_world(request):
    return Response('Hello, Pyramid!')

with Configurator() as config:
    config.add_route('home', '/')
    config.add_view(hello_world, route_name='home')
    app = config.make_wsgi_app()

server = make_server('0.0.0.0', 6543, app)
server.serve_forever()
```

Popular Python Libraries for Web Development

5. Bottle

Type: Micro Web Framework

Purpose: Lightweight web development for smaller applications and APIs.

Key Features: Single-file deployment

Built-in template engine

Minimal dependencies

Installation: pip install bottle

```
from bottle import route, run

@route('/')
def home():
    return "Hello, Bottle!"

run(host='localhost', port=8080)
```

Popular Python Libraries for Web Development

6. Tornado

Type: Asynchronous Web Framework

Purpose: Build scalable, non-blocking web servers and applications.

Key Features:

- Handles thousands of concurrent connections
- Ideal for WebSockets
- Non-blocking I/O

Installation: pip install tornado

Example

```
import tornado.ioloop
import tornado.web

class MainHandler(tornado.web.RequestHandler):
    def get(self):
        self.write("Hello, Tornado!")

def make_app():
    return tornado.web.Application([(r"/", MainHandler)])

if __name__ == "__main__":
    app = make_app()
    app.listen(8888)
    tornado.ioloop.IOLoop.current().start()
```

Popular Python Libraries for Web Development

7. Requests
Type: HTTP Library
Purpose: Simplify making HTTP requests (e.g., REST APIs).
Key Features:

- Easy-to-use syntax for API calls
- Supports SSL, cookies, sessions

Installation: pip install requests

```
import requests

response = requests.get('https://api.github.com')
print(response.status_code)
```

Popular Python Libraries for Web Development

8. SQLAlchemy

Type: Database ORM

Purpose: Handle database operations efficiently.

Key Features:

- ORM for mapping Python objects to database tables
- Database abstraction

Installation: pip install sqlalchemy

```
from sqlalchemy import create_engine, Column, Integer, String, Base

engine = create_engine('sqlite:///test.db')
Base.metadata.create_all(engine)
```

Popular Python Libraries for Web Development

9. BeautifulSoup
Type: Web Scraping Library
Purpose: Parse HTML and XML documents.
Key Features:

- Extract data from websites
- Parse HTML and XML

Installation: pip install beautifulsoup4

```
from bs4 import BeautifulSoup

html = "<html><body><h1>Hello, World!</h1></body></html>"
soup = BeautifulSoup(html, 'html.parser')
print(soup.h1.text)
```

Popular Python Libraries for Web Development

10. Celery
Type: Task Queue
Purpose: Handle background tasks in web applications.
Key Features:

- Asynchronous task processing
- Schedule tasks and jobs

Installation: pip install celery

```
from celery import Celery

app = Celery('tasks', broker='pyamqp://guest@localhost//')

@app.task
def add(x, y):
    return x + y
```

PyCharm IDE

PyCharm makes it seamless to integrate with **Git**, enabling developers to manage their code versions directly from the IDE.

PyCharm is a cross-platform IDE that provides consistent experience on the Windows, macOS, and Linux operating systems.

PyCharm is available in two editions: Professional, and Community. The **Community** edition is an open-source project, and it's free, but it has fewer features. The **Professional** edition is commercial, and provides an outstanding set of tools and features.

PyCharm supports the following versions of Python:

Python 2: version 2.7

Python 3: from the version 3.6 up to the version 3.13

1. Git Integration with PyCharm IDE

PyCharm makes it seamless to integrate with **Git**, enabling developers to manage their code versions directly from the IDE.

1 Setting Up Git in PyCharm

Install Git

Ensure Git is installed on your system.

Verify installation using

```
git --version
```

2. Configure Git in PyCharm:

Go to **File > Settings > Version Control > Git**

Set the **Path to Git executable** (e.g., /usr/bin/git or C:\Program Files\Git\bin\git.exe)

Click **Test** to ensure everything works.

3. Enable Version Control in Project:

Go to **VCS > Enable Version Control Integration**

Select **Git** from the dropdown.

Common Git Operations in PyCharm

Clone a Repository:
Go to **File > New > Project from Version Control > Git**
Enter the repository URL and set the destination folder.
Commit Changes:
Right-click on a file > Git > Commit File
Add a commit message and click **Commit**.
Push Changes:
VCS > Git > Push
Push your changes to the remote repository.
Pull Changes:
VCS > Git > Pull
Update your local repository with changes from the remote.
View Logs:
VCS > Git > Show History
View commit history and diffs.
Resolve Merge Conflicts:
Use the built-in conflict resolution tools in PyCharm.

PyTests in PyCharm

Pytest is a testing framework for Python that makes it easy to write simple and scalable test cases. It offers features like fixtures, parameterized tests, and detailed error reporting. With Pytest, you can efficiently test your code and ensure its reliability.

Pytest Features

Some key features of Pytest is as follows −

Simple Syntax: Pytest uses a simple syntax for writing tests, making it easy to get started and understand test cases.

Fixtures: Provides a powerful fixture mechanism to set up and tear down resources needed for tests, promoting code reuse and reducing duplication.

Parameterized Tests: Supports parameterized tests, allowing you to run the same test with different inputs efficiently.

Detailed Reporting: Offers detailed and readable test reports with clear information on test failures and errors, helping you diagnose issues quickly.

Extensible: Highly extensible with plugins, enabling additional functionalities and integrations modified to your testing needs.

Pytest - Introduction

Pytest is a python based testing framework, which is used to write and execute test codes. In the present days of REST services, pytest is mainly used for API testing even though we can use pytest to write simple to complex tests, i.e., we can write codes to test API, database, UI, etc.
The advantages of Pytest are as follows −
Pytest can run multiple tests in parallel, which reduces the execution time of the test suite.
Pytest has its own way to detect the test file and test functions automatically, if not mentioned explicitly.
Pytest allows us to skip a subset of the tests during execution.
Pytest allows us to run a subset of the entire test suite.
Pytest is free and open source.
Because of its simple syntax, pytest is very easy to start with.

Example

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

def test_add():
    assert add(2, 3) == 5
    assert add(-1, 1) == 0
```

Cont..

4. Run Tests in PyCharm
Right-click on the test file.
Select **Run 'pytest in test_example.py'**
View results in the **Run Panel**.
5. Debug Tests in PyCharm
Right-click on the test and select **Debug 'pytest in test_example.py'**
Set breakpoints to inspect variables.
6. Test Coverage
Go to **Run > Run with Coverage**
Analyze the coverage report to identify untested code.

What is MongoDB?

In today's data-driven world, the ability to efficiently store and manage large amounts of data is crucial. MongoDB, a powerful **NoSQL** database, has become a go-to choice for developers looking for flexibility, scalability, and performance. MongoDB is a **document-oriented** NoSQL database system that provides high scalability, flexibility, and performance. Unlike standard relational databases, MongoDB stores data in a JSON document structure form. This makes it easy to operate with dynamic and unstructured data and MongoDB is an open-source and cross-platform database System.

It means that MongoDB isn't based on the **table-like relational database** structure but provides an altogether different mechanism for the **storage** and **retrieval** of data. This format of storage is called **BSON** (similar to **JSON** format).

Database

Database is a container for collections.
Each database gets its own set of files.
A single MongoDB server can has multiple databases.
Language Support by MongoDB
MongoDB currently provides official driver support for all popular programming languages like C, C++, Rust, C#, Java, **Node.js, Perl, PHP, Python, Ruby, Scala, Go and Erlang.**

Collection

Collection is a group of documents.
Collection is equivalent to RDBMS table.
A collection consist inside a single database.
Collections do not enforce a schema.
A Collection can have different fields within a Documents.

Why Use MongoDB?

Document Oriented Storage − Data is stored in the form of JSON documents.

Index on any attribute: Indexing in MongoDB allows for faster data retrieval by creating a searchable structure on selected attributes, optimizing query performance.

Replication and high availability: MongoDB’s replica sets ensure data redundancy by maintaining multiple copies of the data, providing fault tolerance and continuous availability even in case of server failures.

Auto-Sharding: Auto-sharding in MongoDB automatically distributes data across multiple servers, enabling horizontal scaling and efficient handling of large datasets.

Big Data and Real-time Application: When dealing with massive datasets or applications requiring real-time data updates, MongoDB’s flexibility and scalability prove advantageous.

Why Use MongoDB?

Rich queries: MongoDB supports complex queries with a variety of operators, allowing you to retrieve, filter, and manipulate data in a flexible and powerful manner.

Fast in-place updates: MongoDB efficiently updates documents directly in their place, minimizing data movement and reducing write overhead.

Professional support by MongoDB: MongoDB offers expert technical support and resources to help users with any issues or challenges they may encounter during their database operations.

Internet of Things (IoT) Applications: Storing and analyzing sensor data with its diverse formats often aligns well with MongoDB's document structure.

Where to Use MongoDB?

Mobile and Social Infrastructure
Data Hub
Previous Pag
Big Data
User Data Management
Content Management and Delivery

Prerequisites for the MongoDB

it is suitable if you have some prior knowledge of Databases, Frontend development, Text editor and execution of programs, etc. It will be beneficial if you have a basic understanding of database fundamentals because we’ll be developing high-performance databases (RDBMS).

MongoDB database features

Document Oriented: MongoDB stores the main subject in the minimal number of documents and not by breaking it up into multiple relational structures like RDBMS. For example, it stores all the information of a computer in a single document called Computer and not in distinct relational structures like **CPU, RAM, Hard disk** etc.

Indexing: Without **indexing**, a database would have to scan every document of a collection to select those that match the query which would be inefficient. So, for efficient searching Indexing is a must and MongoDB uses it to process huge volumes of data in very less time.

Scalability: MongoDB scales **horizontally** using sharding (partitioning data across various **servers**). Data is partitioned into data chunks using the **shard key** and these data chunks are evenly distributed across shards that reside across many physical servers. Also, new machines can be added to a running database.

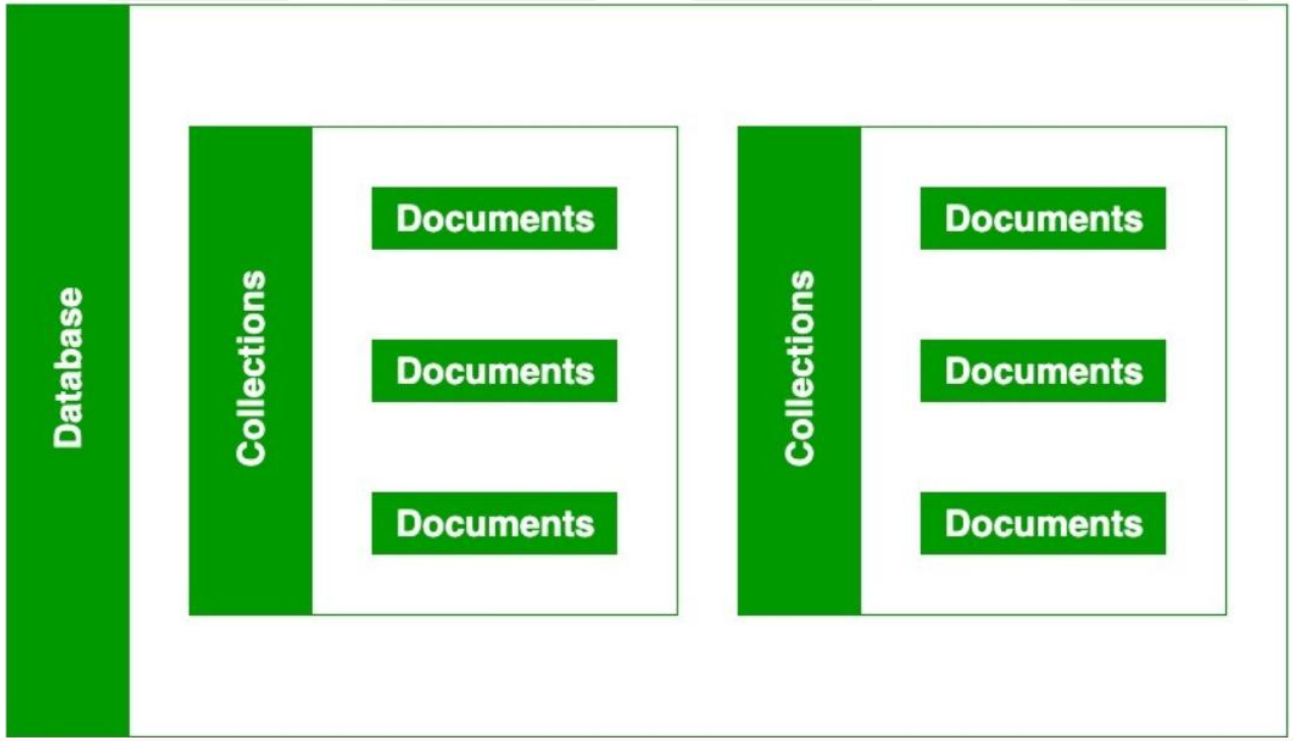
MongoDB database features

Replication and High Availability: MongoDB increases the **data availability** with multiple copies of data on different servers. By providing redundancy, it protects the database from hardware failures. If one server goes down, the data can be retrieved easily from other active servers which also had the data stored on them.

Aggregation: Aggregation operations process data records and return the computed results. It is similar to the **GROUPBY** clause in SQL. A few aggregation expressions are **sum, avg, min, max**, etc

MongoDB – Database, Collection, and Document

Databases, collections, documents are important parts of MongoDB without them you are not able to store data on the MongoDB server. A Database contains one or more collections, and a collection contains documents and the documents contain data, they are related to each other.



Cont..

Collection is a group of MongoDB documents. It is the equivalent of an RDBMS table. A collection exists within a single database. Collections do not enforce a schema. Documents within a collection can have different fields. Typically, all documents in a collection are of similar or related purpose.

A document is a set of key-value pairs. Documents have dynamic schema. Dynamic schema means that documents in the same collection do not need to have the same set of fields or structure, and common fields in a collection's documents may hold different types of data.

Sample document

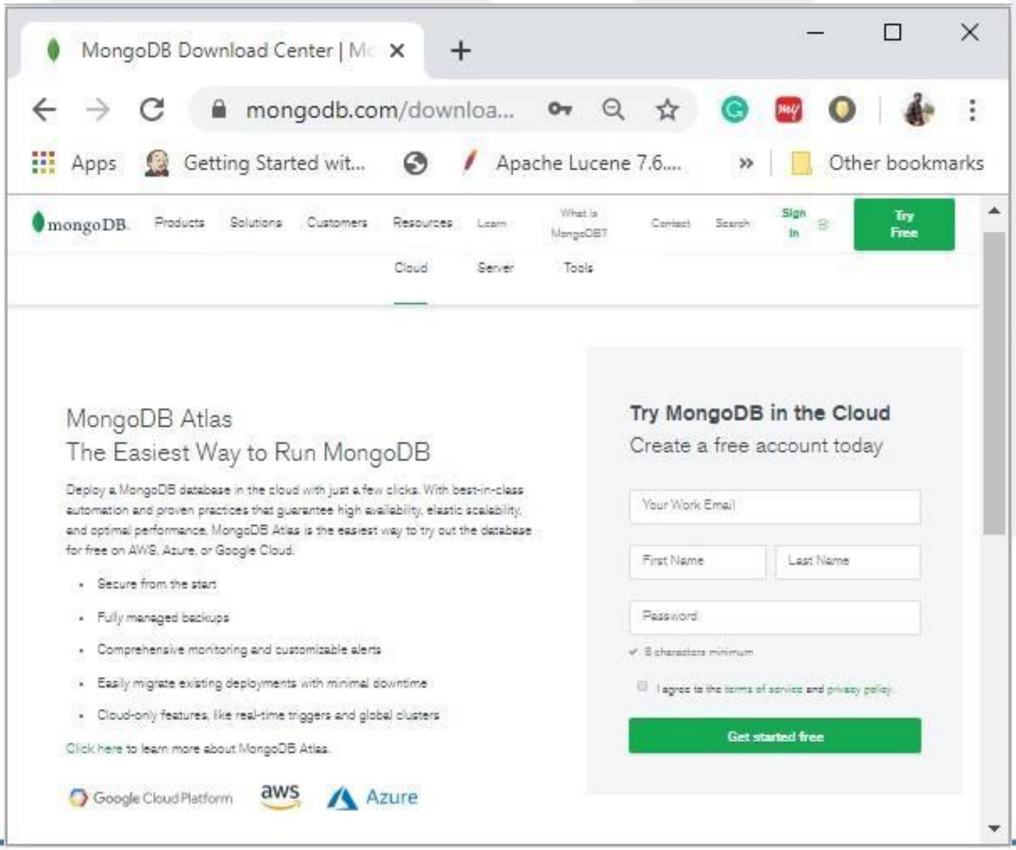
```
Employee
{
 “name”: amit,
  “age”:27,
  “city”:Noida,
  “Identity”:{
    “aadhar”:222211113333,
     “pan”: TUFEU5905D”
    }
}
```

Relationship of RDBMS terminology with MongoDB

RDBMS	MongoDB
Database	Database
Table	Collection
Tuple/Row	Document
column	Field
Table Join	Embedded Documents
Primary Key	Primary Key (Default key _id provided by MongoDB itself)
Database Server and Client	
mysqld/Oracle	mongod
mysql/sqlplus	mongo

Install MongoDB On Windows

To install MongoDB on Windows, first download the latest release of MongoDB from https://www.mongodb.com/download-center.



MongoDB Atlas
The Easiest Way to Run MongoDB

Deploy a MongoDB database in the cloud with just a few clicks. With best-in-class automation and proven practices that guarantee high availability, elastic scalability, and optimal performance, MongoDB Atlas is the easiest way to try out the database for free on AWS, Azure, or Google Cloud.

- Secure from the start
- Fully managed backups
- Comprehensive monitoring and customizable alerts
- Easily migrate existing deployments with minimal downtime
- Cloud-only features, like real-time triggers and global clusters

Click here to learn more about MongoDB Atlas.

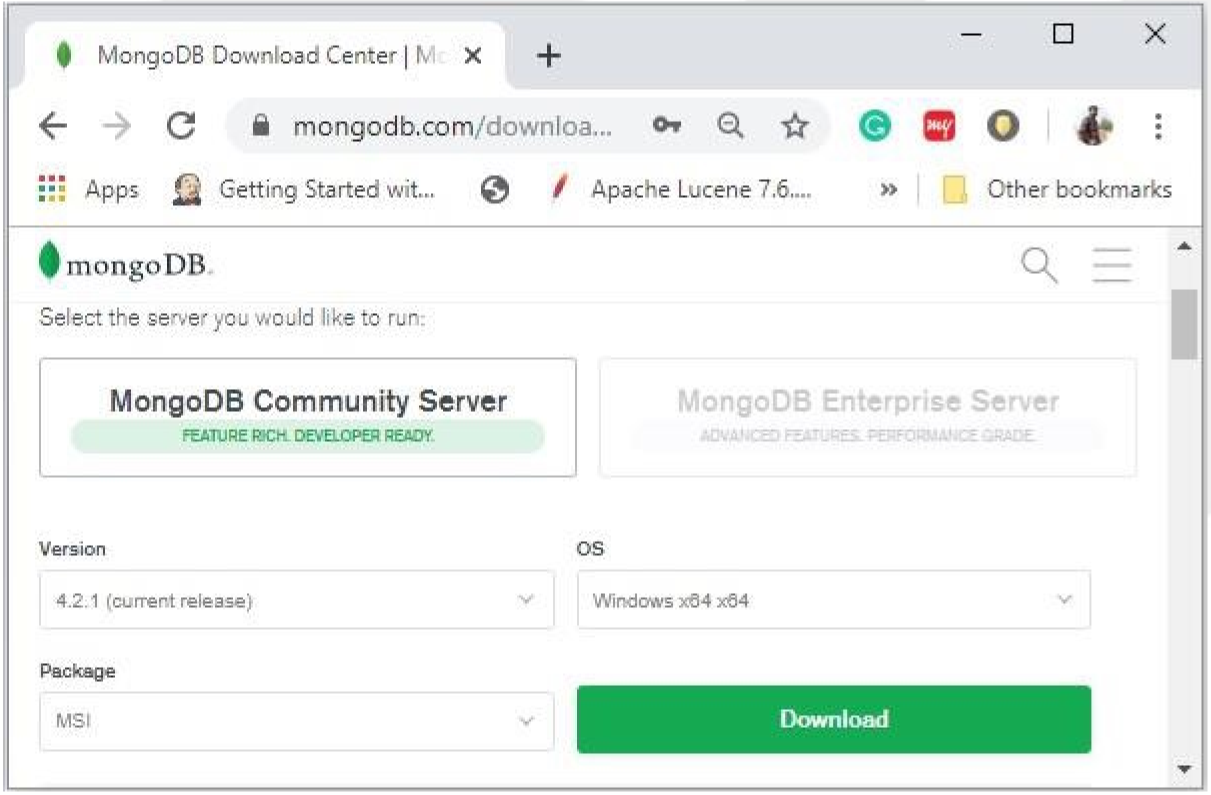
Google Cloud Platform aws Azure

Try MongoDB in the Cloud
Create a free account today

Your Work Email
First Name
Last Name
Password
✓ 8 characters minimum
☐ I agree to the terms of service and privacy policy
Get started free

Install MongoDB On Windows

Enter the required details, select the ***Server*** tab, in it you can choose the version of MongoDB, operating system and, packaging as:



Select the server you would like to run:

MongoDB Community Server
FEATURE RICH. DEVELOPER READY.

MongoDB Enterprise Server
ADVANCED FEATURES. PERFORMANCE GRADE.

Version: 4.2.1 (current release)

OS: Windows x64 x64

Package: MSI

Download

Install MongoDB On Windows

Now install the downloaded file, by default, it will be installed in the folder **C:\Program Files**.
MongoDB requires a data folder to store its files. The default location for the MongoDB data directory is c:\data\db. So you need to create this folder using the Command Prompt. Execute the following command sequence.
C:\>md data C:\md data\db Then you need to specify set the **dbpath** to the created directory in **mongod.exe**. For the same, issue the following commands.
In the command prompt, navigate to the bin directory current in the MongoDB installation folder. Suppose my installation folder is **C:\Program Files\MongoDB**
C:\Users\XYZ>d:cd C:\Program Files\MongoDB\Server\4.2\bin C:\Program Files\MongoDB\Server\4.2\bin>mongod.exe --dbpath "C:\data"

Install MongoDB On Windows

This will show **waiting for connections** message on the console output, which indicates that the mongod.exe process is running successfully.

Now to run the MongoDB, you need to open another command prompt and issue the following command.

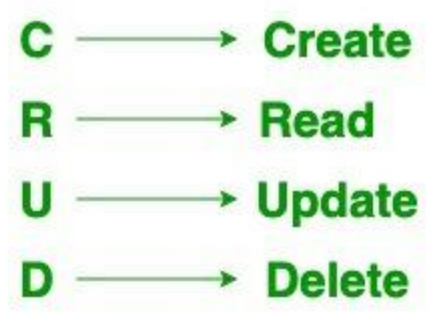
C:\Program Files\MongoDB\Server\4.2\bin>mongo.exe MongoDB shell version v4.2.1 connecting to: mongodb://127.0.0.1:27017/?compressors=disabled&gssapiServiceName=mongodb Implicit session: session { "id" : UUID("4260beda-f662-4cbe-9bc7-5c1f2242663c") } MongoDB server version: 4.2.1 >

This will show that MongoDB is installed and run successfully. Next time when you run MongoDB, you need to issue only commands.

C:\Program Files\MongoDB\Server\4.2\bin>mongod.exe --dbpath "C:\data" C:\Program Files\MongoDB\Server\4.2\bin>mongo.exe

MongoDB CRUD Operations

CRUD Operations (Create, Read, Update, and Delete) are the basic set of operations that allow users to interact with the MongoDB server.
As we know, to use MongoDB we need to interact with the MongoDB server to perform certain operations like entering new data into the application, updating data into the application, deleting data from the application, and reading the application data.



1. Create Operations

The **create or insert operations** are used to insert or add new documents in the collection. If a collection does not exist, then it will create a new collection in the database.
You can perform, create operations using the following methods provided by the MongoDB:

Method	Description
db.collection.insertOne()	It is used to insert a single document in the collection.
db.collection.insertMany()	It is used to insert multiple documents in the collection.
db.createCollection()	It is used to create an empty collection.

2. Read Operations

The Read operations are used to retrieve documents from the collection, or in other words, read operations are used to query a collection for a document.
You can perform read operation using the following method provided by the MongoDB:

Method	Description
db.collection.find()	It is used to retrieve documents from the collection.
db.collection.findone()	

3. Update Operations

The update operations are used to update or modify the existing document in the collection. You can perform update operations using the following methods provided by the MongoDB:

Method	Description
db.collection.updateOne()	It is used to update a single document in the collection that satisfy the given criteria.
db.collection.updateMany()	It is used to update multiple documents in the collection that satisfy the given criteria.
db.collection.replaceOne()	It is used to replace single document in the collection that satisfy the given criteria.

4. Delete Operations

The delete operation are used to delete or remove the documents from a collection. You can perform delete operations using the following methods provided by the MongoDB:

Method	Description
db.collection.deleteOne()	It is used to delete a single document from the collection that satisfy the given criteria.
db.collection.deleteMany()	It is used to delete multiple documents from the collection that satisfy the given criteria.