Got it 👍 You want me to give you step-by-step answers, code, sample output, and Jenkins setup instructions for all the projects listed in your table (Java + Python, Free Style + Pipeline).

Since there are 20+ projects, answering all at once would make this reply way too long to handle. The best way is to go one by one (or a few grouped together) so that each answer is complete with:

✅ Steps to do it (project setup + Jenkins setup)

✅ Code implementation (Java/Python program)

✅ Sample output (console/app output screenshots in text)

✅ Jenkins Free Style OR Pipeline configuration (with Jenkinsfile if Pipeline)

---

Example Walkthrough (for your first project: Java To-Do List)

1. System Name: Java To-Do List

Description: Manage tasks with CRUD operations

Jenkins Type: Free Style

Expected Outcome: Updated to-do app after commit

---

🔹 Steps to Do It

1. Install Java & Git on your system.

2. Create a folder JavaToDoList and add ToDoList.java.

3. Initialize a Git repo (git init) and push to GitHub.

4. In Jenkins:

Create a Free Style Project → Name: JavaToDoListJob.

Source Code Management → Git → Add your repo URL.

Build → Add build step → “Execute shell” → javac ToDoList.java && java ToDoList.

Post-build → Archive artifacts (\*.class).

Save & Build.

---

🔹 Java Code (ToDoList.java)

import java.util.ArrayList;

import java.util.Scanner;

public class ToDoList {

private static ArrayList<String> tasks = new ArrayList<>();

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

while (true) {

System.out.println("\n--- To-Do List ---");

System.out.println("1. Add Task");

System.out.println("2. View Tasks");

System.out.println("3. Delete Task");

System.out.println("4. Exit");

System.out.print("Enter choice: ");

int choice = sc.nextInt();

sc.nextLine();

switch (choice) {

case 1:

System.out.print("Enter task: ");

String task = sc.nextLine();

tasks.add(task);

System.out.println("Task added!");

break;

case 2:

System.out.println("Tasks:");

for (int i = 0; i < tasks.size(); i++) {

System.out.println((i + 1) + ". " + tasks.get(i));

}

break;

case 3:

System.out.print("Enter task number to delete: ");

int index = sc.nextInt();

if (index > 0 && index <= tasks.size()) {

tasks.remove(index - 1);

System.out.println("Task deleted!");

} else {

System.out.println("Invalid task number!");

}

break;

case 4:

System.out.println("Exiting...");

return;

default:

System.out.println("Invalid choice!");

}

}

}

}

---

🔹 Sample Output

--- To-Do List ---

1. Add Task

2. View Tasks

3. Delete Task

4. Exit

Enter choice: 1

Enter task: Complete assignment

Task added!

--- To-Do List ---

1. Add Task

2. View Tasks

3. Delete Task

4. Exit

Enter choice: 2

Tasks:

1. Complete assignment

---

🔹 Jenkins Free Style Setup Output

When you push code to GitHub, Jenkins will:

1. Pull latest code

2. Compile & run program

3. Show To-Do List output in Jenkins console

---

⚡ This is the full template I can follow for each project.

👉 Do you want me to continue with all projects in this format one by one (Java + Python, 20+ answers),

or should I bundle them in groups (like Java apps first, then Python apps)?

Perfect 👍 bundling makes it more organized and readable.

We’ll do:

1. Java Projects (Free Style Jobs) – To-Do List, Quiz App, Temperature Conversion/Timer, Contact Book, News App, Currency Converter, Book Finder, Dark/Light Theme, Age Calculator, Discount Calculator.

2. Python Projects (Pipeline Jobs) – Calculator, Notes App, Expense Tracker, BMI Calculator, Dictionary App, Recipe Finder, Form Validation, Counter App, Calendar Picker, Online Shopping (Notify), Bill Calculator.

For each group:

I’ll give steps (setup + Jenkins job)

code (Java/Python)

sample output

Jenkins Free Style or Pipeline config (Pipeline with Jenkinsfile)

---

✅ Let’s start with Group 1: Java Projects (Free Style Jobs)

---

1. Java To-Do List (CRUD)

(Already given above – manage tasks with add, view, delete. Jenkins Free Style compiles & runs.)

---

2. Java Quiz App (MCQ Quiz with Score)

Steps:

1. Create QuizApp.java.

2. Push to GitHub.

3. Jenkins Free Style Job → Git SCM → javac QuizApp.java && java QuizApp.

Code (QuizApp.java)

import java.util.Scanner;

public class QuizApp {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String[] questions = {

"What is the capital of France?",

"2 + 2 = ?",

"Java is \_\_\_?"

};

String[][] options = {

{"1. Paris", "2. London", "3. Berlin"},

{"1. 3", "2. 4", "3. 5"},

{"1. A drink", "2. A language", "3. A car"}

};

int[] answers = {1, 2, 2};

int score = 0;

for (int i = 0; i < questions.length; i++) {

System.out.println(questions[i]);

for (String opt : options[i]) System.out.println(opt);

System.out.print("Enter answer: ");

int ans = sc.nextInt();

if (ans == answers[i]) score++;

}

System.out.println("Your Score: " + score + "/" + questions.length);

}

}

Sample Output

What is the capital of France?

1. Paris

2. London

3. Berlin

Enter answer: 1

2 + 2 = ?

1. 3

2. 4

3. 5

Enter answer: 2

Java is \_\_\_?

1. A drink

2. A language

3. A car

Enter answer: 2

Your Score: 3/3

---

3. Java Temperature Conversion (Timer Simulation)

Steps:

Create TimerApp.java.

Free Style job → compile & run.

Code (TimerApp.java)

public class TimerApp {

public static void main(String[] args) throws InterruptedException {

System.out.println("Starting Timer...");

for (int i = 1; i <= 5; i++) {

System.out.println("Seconds: " + i);

Thread.sleep(1000); // 1 second delay

}

System.out.println("Timer Stopped!");

}

}

Sample Output

Starting Timer...

Seconds: 1

Seconds: 2

Seconds: 3

Seconds: 4

Seconds: 5

Timer Stopped!

---

4. Java Contact Book (Add/Search Contacts)

Code (ContactBook.java)

import java.util.\*;

public class ContactBook {

private static HashMap<String, String> contacts = new HashMap<>();

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

while (true) {

System.out.println("\n--- Contact Book ---");

System.out.println("1. Add Contact");

System.out.println("2. Search Contact");

System.out.println("3. Exit");

System.out.print("Enter choice: ");

int choice = sc.nextInt();

sc.nextLine();

switch (choice) {

case 1:

System.out.print("Enter Name: ");

String name = sc.nextLine();

System.out.print("Enter Phone: ");

String phone = sc.nextLine();

contacts.put(name, phone);

System.out.println("Contact added!");

break;

case 2:

System.out.print("Enter Name to search: ");

String search = sc.nextLine();

if (contacts.containsKey(search))

System.out.println("Phone: " + contacts.get(search));

else

System.out.println("Not found!");

break;

case 3:

return;

}

}

}

}

Sample Output

--- Contact Book ---

1. Add Contact

2. Search Contact

3. Exit

Enter choice: 1

Enter Name: Alice

Enter Phone: 9876543210

Contact added!

Enter choice: 2

Enter Name to search: Alice

Phone: 9876543210

---

5. Java News App (Display Dummy News)

public class NewsApp {

public static void main(String[] args) {

String[] news = {

"Breaking: AI transforming industries!",

"Sports: Team A wins championship!",

"Weather: Sunny tomorrow."

};

System.out.println("Today's News:");

for (String n : news) {

System.out.println("- " + n);

}

}

}

Sample Output:

Today's News:

- Breaking: AI transforming industries!

- Sports: Team A wins championship!

- Weather: Sunny tomorrow.

---

6. Java Currency Converter

import java.util.Scanner;

public class CurrencyConverter {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter amount in USD: ");

double usd = sc.nextDouble();

double inr = usd \* 83;

System.out.println("Amount in INR: " + inr);

}

}

Sample Output:

Enter amount in USD: 10

Amount in INR: 830.0

---

7. Java Book Finder (Google Books Mock)

public class BookFinder {

public static void main(String[] args) {

String[] books = {"Java Basics", "Python for Beginners", "Data Structures"};

System.out.println("Available Books:");

for (String b : books) System.out.println("- " + b);

}

}

---

8. Java Dark/Light Theme Toggle

import java.util.Scanner;

public class ThemeSwitcher {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

String theme = "Light";

while (true) {

System.out.println("Current Theme: " + theme);

System.out.print("Toggle? (y/n): ");

String ans = sc.nextLine();

if (ans.equalsIgnoreCase("y")) {

theme = theme.equals("Light") ? "Dark" : "Light";

} else break;

}

}

}

---

9. Java Age Calculator

import java.util.Scanner;

import java.time.Year;

public class AgeCalculator {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

int currentYear = Year.now().getValue();

System.out.print("Enter birth year: ");

int birth = sc.nextInt();

System.out.println("Your age: " + (currentYear - birth));

}

}

---

10. Java Discount Calculator

import java.util.Scanner;

public class DiscountCalculator {

public static void main(String[] args) {

Scanner sc = new Scanner(System.in);

System.out.print("Enter price: ");

double price = sc.nextDouble();

System.out.print("Enter discount %: ");

double disc = sc.nextDouble();

double finalPrice = price - (price \* disc / 100);

System.out.println("Final Price: " + finalPrice);

}

}

---

✅ That’s all 10 Java projects (Free Style Jenkins jobs) done.

Each job in Jenkins:

Select Free Style Project

SCM → Git → Repo URL

Build → Execute Shell → javac FileName.java && java ClassName

---

👉 Next, do you want me to move to Group 2: Python Projects (Pipeline Jobs with Jenkinsfile) with code + Jenkinsfile + sample output?

Great 🚀 Let’s move on to Group 2: Python Projects (Pipeline Jobs with Jenkinsfile).

For each project:

Steps (setup + Jenkins pipeline job)

Python code

Sample output

Jenkinsfile (Pipeline script to automate build & run)

---

Group 2: Python Projects (Pipeline)

---

1. Python Calculator (Basic Calculator)

Steps

1. Create calculator.py.

2. Push to GitHub.

3. In Jenkins → Create Pipeline Job.

4. Pipeline script → Use Jenkinsfile below.

Code (calculator.py)

def calculator():

print("Simple Calculator")

a = int(input("Enter first number: "))

b = int(input("Enter second number: "))

print("Choose operation: + - \* /")

op = input("Enter operation: ")

if op == '+':

print("Result:", a + b)

elif op == '-':

print("Result:", a - b)

elif op == '\*':

print("Result:", a \* b)

elif op == '/':

print("Result:", a / b if b != 0 else "Error! Division by zero")

else:

print("Invalid operation!")

if \_\_name\_\_ == "\_\_main\_\_":

calculator()

Sample Output

Simple Calculator

Enter first number: 10

Enter second number: 5

Choose operation: + - \* /

Enter operation: \*

Result: 50

Jenkinsfile

pipeline {

agent any

stages {

stage('Checkout') {

steps {

git 'https://github.com/your-repo/python-calculator.git'

}

}

stage('Run Calculator') {

steps {

sh 'python3 calculator.py'

}

}

}

}

---

2. Python Notes App (Sticky Notes in Browser)

Code (notes\_app.py)

from flask import Flask, request, render\_template\_string

app = Flask(\_\_name\_\_)

notes = []

TEMPLATE = """

<h2>Sticky Notes</h2>

<form method="POST">

<input type="text" name="note" placeholder="Enter note" required>

<button type="submit">Add</button>

</form>

<ul>

{% for n in notes %}

<li>{{n}}</li>

{% endfor %}

</ul>

"""

@app.route("/", methods=["GET", "POST"])

def home():

if request.method == "POST":

notes.append(request.form["note"])

return render\_template\_string(TEMPLATE, notes=notes)

if \_\_name\_\_ == "\_\_main\_\_":

app.run(debug=True)

Sample Output (Browser)

Sticky Notes

[Input box + Add button]

Notes displayed below

Jenkinsfile

pipeline {

agent any

stages {

stage('Checkout') {

steps {

git 'https://github.com/your-repo/python-notes-app.git'

}

}

stage('Install Dependencies') {

steps {

sh 'pip3 install flask'

}

}

stage('Run Notes App') {

steps {

sh 'python3 notes\_app.py &'

}

}

}

}

---

3. Python Expense Tracker

Code (expense\_tracker.py)

expenses = []

def add\_expense(item, amount):

expenses.append({"item": item, "amount": amount})

def show\_expenses():

print("\nExpenses:")

for e in expenses:

print(f"{e['item']} - ${e['amount']}")

print("Total:", sum(e['amount'] for e in expenses))

if \_\_name\_\_ == "\_\_main\_\_":

add\_expense("Food", 20)

add\_expense("Transport", 15)

show\_expenses()

Sample Output

Expenses:

Food - $20

Transport - $15

Total: 35

Jenkinsfile

pipeline {

agent any

stages {

stage('Checkout') {

steps {

git 'https://github.com/your-repo/python-expense-tracker.git'

}

}

stage('Run Tracker') {

steps {

sh 'python3 expense\_tracker.py'

}

}

}

}

---

4. Python BMI Calculator

Code (bmi\_calculator.py)

def bmi\_calc():

weight = float(input("Enter weight (kg): "))

height = float(input("Enter height (m): "))

bmi = weight / (height \*\* 2)

print("Your BMI:", round(bmi, 2))

if \_\_name\_\_ == "\_\_main\_\_":

bmi\_calc()

Sample Output

Enter weight (kg): 60

Enter height (m): 1.7

Your BMI: 20.76

Jenkinsfile

pipeline {

agent any

stages {

stage('Checkout') {

steps {

git 'https://github.com/your-repo/python-bmi-calculator.git'

}

}

stage('Run BMI Calculator') {

steps {

sh 'python3 bmi\_calculator.py'

}

}

}

}

---

5. Python Dictionary App (Search Word Meanings)

Code (dictionary\_app.py)

dictionary = {

"python": "A programming language",

"java": "Another programming language",

"jenkins": "Automation server"

}

word = input("Enter a word: ").lower()

print("Meaning:", dictionary.get(word, "Not found"))

Sample Output

Enter a word: python

Meaning: A programming language

Jenkinsfile

pipeline {

agent any

stages {

stage('Checkout') {

steps {

git 'https://github.com/your-repo/python-dictionary-app.git'

}

}

stage('Run Dictionary App') {

steps {

sh 'python3 dictionary\_app.py'

}

}

}

}

---

⚡ To keep this clean, I’ve given you 5 Python pipeline apps.

Next batch will cover:

Recipe Finder

Form Validation

Counter App

Calendar Picker

Online Shopping (with Jenkins notify)

Bill Calculator

---

👉 Do you want me to continue with the next Python batch now or should I prepare a combined PDF/Doc with all projects (Java + Python) together for easier reference?

Alright 👍 Let’s continue with the remaining Python Pipeline projects.

This is Batch 2 (Recipe Finder → Bill Calculator).

---

Group 2: Python Projects (Pipeline, Part 2)

---

6. Python Recipe Finder

Code (recipe\_finder.py)

recipes = {

"pasta": "Boil pasta, add sauce, serve hot.",

"tea": "Boil water, add tea leaves, strain and serve.",

"cake": "Mix flour, sugar, eggs, bake at 180°C for 30 min."

}

search = input("Enter recipe name: ").lower()

print("Recipe:", recipes.get(search, "Not found"))

Sample Output

Enter recipe name: tea

Recipe: Boil water, add tea leaves, strain and serve.

Jenkinsfile

pipeline {

agent any

stages {

stage('Checkout') {

steps {

git 'https://github.com/your-repo/python-recipe-finder.git'

}

}

stage('Run Recipe Finder') {

steps {

sh 'python3 recipe\_finder.py'

}

}

}

}

---

7. Python Form Validation (Login/Signup)

Code (form\_validation.py)

import re

def validate\_email(email):

return re.match(r"[^@]+@[^@]+\.[^@]+", email)

def validate\_password(password):

return len(password) >= 6

email = input("Enter email: ")

password = input("Enter password: ")

if validate\_email(email) and validate\_password(password):

print("Form submitted successfully!")

else:

print("Invalid email or password (must be 6+ chars).")

Sample Output

Enter email: user@test.com

Enter password: secret123

Form submitted successfully!

Jenkinsfile

pipeline {

agent any

stages {

stage('Checkout') {

steps {

git 'https://github.com/your-repo/python-form-validation.git'

}

}

stage('Run Validation') {

steps {

sh 'python3 form\_validation.py'

}

}

}

}

---

8. Python Counter App (Increment/Decrement)

Code (counter\_app.py)

count = 0

while True:

print(f"Current Count: {count}")

action = input("Enter (+ / - / exit): ")

if action == '+':

count += 1

elif action == '-':

count -= 1

elif action.lower() == "exit":

break

else:

print("Invalid input!")

Sample Output

Current Count: 0

Enter (+ / - / exit): +

Current Count: 1

Enter (+ / - / exit): -

Current Count: 0

Jenkinsfile

pipeline {

agent any

stages {

stage('Checkout') {

steps {

git 'https://github.com/your-repo/python-counter-app.git'

}

}

stage('Run Counter') {

steps {

sh 'python3 counter\_app.py'

}

}

}

}

---

9. Python Calendar Picker

Code (calendar\_picker.py)

import calendar

year = int(input("Enter year: "))

month = int(input("Enter month: "))

print(calendar.month(year, month))

Sample Output

Enter year: 2025

Enter month: 9

September 2025

Mo Tu We Th Fr Sa Su

1 2 3 4 5 6 7

8 9 10 11 12 13 14

15 16 17 18 19 20 21

22 23 24 25 26 27 28

29 30

Jenkinsfile

pipeline {

agent any

stages {

stage('Checkout') {

steps {

git 'https://github.com/your-repo/python-calendar-picker.git'

}

}

stage('Run Calendar') {

steps {

sh 'python3 calendar\_picker.py'

}

}

}

}

---

10. Python Online Shopping (Jenkins Notify)

Code (shopping\_app.py)

cart = []

def add\_item(item):

cart.append(item)

def checkout():

print("\nItems in cart:")

for i in cart:

print("-", i)

print("Checkout successful!")

if \_\_name\_\_ == "\_\_main\_\_":

add\_item("Shoes")

add\_item("Shirt")

checkout()

Sample Output

Items in cart:

- Shoes

- Shirt

Checkout successful!

Jenkinsfile (with email notification)

pipeline {

agent any

stages {

stage('Checkout') {

steps {

git 'https://github.com/your-repo/python-shopping-app.git'

}

}

stage('Run Shopping App') {

steps {

sh 'python3 shopping\_app.py'

}

}

}

post {

success {

mail to: 'team@example.com',

subject: 'Shopping App Deployment Success',

body: 'The shopping app has been deployed successfully.'

}

}

}

---

11. Python Bill Calculator (Final Project)

Code (bill\_calculator.py)

def calculate\_bill(items):

total = sum(price for \_, price in items)

tax = total \* 0.05 # 5% tax

final\_amount = total + tax

print("Items:")

for name, price in items:

print(f"{name}: ${price}")

print(f"Total: ${total}")

print(f"Tax (5%): ${tax:.2f}")

print(f"Final Bill: ${final\_amount:.2f}")

if \_\_name\_\_ == "\_\_main\_\_":

items = [("Milk", 50), ("Bread", 30), ("Eggs", 70)]

calculate\_bill(items)

Sample Output

Items:

Milk: $50

Bread: $30

Eggs: $70

Total: $150

Tax (5%): $7.50

Final Bill: $157.50

Jenkinsfile

pipeline {

agent any

stages {

stage('Checkout') {

steps {

git 'https://github.com/your-repo/python-bill-calculator.git'

}

}

stage('Run Bill Calculator') {

steps {

sh 'python3 bill\_calculator.py'

}

}

}

}

---

✅ Now you have all 20 projects (Java + Python) with:

Code

Sample output

Jenkins Free Style (Java) & Pipeline (Python) setup

---

👉 Do you want me to now create a compiled single document (Word/PDF) with all these answers structured neatly, so you can use it directly for your seminar/project report?

Good question 👍

In Java (Free Style) we explicitly write:

javac FileName.java && java ClassName

because Java needs compilation before running.

But in Python (Pipeline jobs) we don’t use javac, since Python is interpreted, not compiled. So:

For Python files (.py), you just run them directly:

python3 filename.py

---

🔹 Detailed Comparison

Language File Extension Build Command in Jenkins Why?

Java .java javac FileName.java && java ClassName Java is compiled into bytecode first, then executed by JVM

Python .py python3 filename.py Python is interpreted directly by the Python runtime

---

🔹 Example

Java Quiz App (Free Style)

javac QuizApp.java && java QuizApp

Python Calculator (Pipeline)

python3 calculator.py

---

🔹 What to include in Jenkins for Python

If it’s Free Style →

1. Add build step → Execute shell.

2. Write:

python3 yourfile.py

If it’s Pipeline with Jenkinsfile → inside steps {} block, write:

sh 'python3 yourfile.py'

---

👉 So, in your documentation/report, for Java files always show both compile + run.

For Python files, just show run command (python3 filename.py).

---

Do you want me to add a short "Build Commands Table" for all your 20 projects (Java + Python) so you can keep it handy in your seminar?