Question - Can we build a bdd framework with a combination of selenium, testng using python??

Answer - In Python, **Selenium** is commonly used for browser automation, and **BDD (Behavior Driven Development)** is typically implemented using tools like **Behave** or **pytest-bdd**. However, **TestNG** is a **Java-based testing framework** and is **not used with Python**.

So, to answer your question directly:

❌ You **cannot** build a BDD framework in Python using **TestNG**, because TestNG is not compatible with Python.

**✅ What you can do instead in Python:**

To build a **BDD framework with Selenium in Python**, consider this combination:

**✅ Tools to Use:**

* **Selenium** – for browser automation
* **Behave** or **pytest-bdd** – for BDD
* **Allure or HTMLTestRunner** – for reporting (optional)
* **Python’s unittest or pytest** – for structuring tests (if not using pure BDD)
* **Page Object Model (POM)** – for maintainability
  1. **Example: Behave + Selenium**

**Install dependencies:**

bash

pip install behave selenium

* 1. **Project Structure:**

pgsql

CopyEdit

features/

│

├── login.feature

├── steps/

│ └── login\_steps.py

└── environment.py

pages/

└── login\_page.py

* 1. **Sample Feature File (login.feature):**

gherkin

Feature: Login Feature

Scenario: Successful login

Given I open the login page

When I enter valid credentials

Then I should be redirected to the dashboard

* 1. **Page Object (pages/login\_page.py):**

python

CopyEdit

class LoginPage:

def \_\_init\_\_(self, driver):

self.driver = driver

def open(self):

self.driver.get("https://example.com/login")

def login(self, username, password):

self.driver.find\_element("id", "username").send\_keys(username)

self.driver.find\_element("id", "password").send\_keys(password)

self.driver.find\_element("id", "submit").click()

**Summary:**

You **can build a BDD framework in Python using Selenium**, but you **cannot** use TestNG, as it is Java-specific. Use **Behave** or **pytest-bdd** instead.

Question - Can maven be used with python?

Answer - **No**, **Maven is not used with Python**. **Maven** is a **Java build and dependency management tool**, designed specifically for Java-based projects.

**Python** has its own ecosystem and tools for dependency management and project setup, such as:

* **pip** or **pipenv** – for dependency management
* **setuptools**, **poetry**, or **flit** – for building and packaging
* **virtualenv** – for environment isolation

Java Vs Python

|  |  |  |
| --- | --- | --- |
| **Task** | **Java Tool (Maven)** | **Python Equivalent** |
| Dependency Management | pom.xml (Maven) | requirements.txt, Pipfile, or pyproject.toml |
| Build Tool | Maven | setuptools, poetry |
| Test Runner | TestNG / JUnit | pytest, unittest |
| BDD Support | Cucumber | Behave, pytest-bdd |

Question - Can we use classes in python?

Answer - Yes, absolutely! ✅ You **can and should use classes in Python**, especially when building **structured**, **reusable**, and **scalable** code — like in test automation frameworks (e.g., with **Selenium**, **BDD**, or **POM** patterns).

**Python Class Basics**

Here’s a simple example of a Python class:

class Calculator:

def \_\_init\_\_(self, a, b):

self.a = a

self.b = b

def add(self):

return self.a + self.b

python

CopyEdit

calc = Calculator(5, 3)

print(calc.add()) # Output: 8

Question - **Why Use Classes in Test Automation?**

In BDD or Selenium frameworks, classes help with:

* **Page Object Model (POM)**: Encapsulate locators and actions
* **Test modularization**: Group related functionality
* **Code reusability**: Use common setups, data handlers, or utilities
* Example: Selenium with Page Object Class

class LoginPage:

def \_\_init\_\_(self, driver):

self.driver = driver

self.username\_input = "id=username"

self.password\_input = "id=password"

self.login\_button = "id=submit"

def load(self):

self.driver.get("https://example.com/login")

def login(self, username, password):

self.driver.find\_element("id", "username").send\_keys(username)

self.driver.find\_element("id", "password").send\_keys(password)

self.driver.find\_element("id", "submit").click()

* Use it in your step definitions (for Behave or pytest-bdd):

page = LoginPage(driver)

page.load()

page.login("admin", "admin123")

Question - Can we push payloads to IBM MQ using python automation

Yes, you can push payloads to IBM MQ using Python automation.

**How It Works:**

IBM MQ is a message-oriented middleware that supports messaging between applications via queues. Python can interact with IBM MQ using IBM's **official Python client library**, or via **MQ REST APIs**.

**Option 1: Use IBM MQ’s Python Library (pymqi)**

**Install the library:**

pip install pymqi

Example Code: Send a message (payload) to a queue

import pymqi

# Connection details

queue\_manager = 'QM1'

channel = 'DEV.APP.SVRCONN'

host = 'localhost'

port = '1414'

queue\_name = 'TEST.QUEUE'

conn\_info = f'{host}({port})'

# Credentials if needed

user = 'app'

password = 'passw0rd'

# Message (payload)

payload = 'This is a test message'

# Connect and put message

cd = pymqi.CD()

cd.ChannelName = channel.encode()

cd.ConnectionName = conn\_info.encode()

cd.ChannelType = pymqi.CMQC.MQCHT\_CLNTCONN

# Authentication setup

sco = pymqi.SCO()

qmgr = pymqi.QueueManager(None)

qmgr.connect\_with\_options(queue\_manager, user, password, cd, sco)

queue = pymqi.Queue(qmgr, queue\_name)

queue.put(payload)

print("Message sent successfully!")

queue.close()

qmgr.disconnect()

**Option 2: Use IBM MQ REST API (if enabled on your MQ server)**

**Send via requests (HTTP/HTTPS):**

import requests

import json

url = 'https://your-mq-host:9443/ibmmq/rest/v1/messaging/qmgr/QM1/queue/TEST.QUEUE/message'

headers = {

'ibm-mq-rest-csrf-token': 'dummy',

'Content-Type': 'text/plain'

}

auth = ('app', 'passw0rd')

data = 'This is a test message from REST API'

response = requests.post(url, data=data, headers=headers, auth=auth, verify=False)

print(f"Status: {response.status\_code}, Response: {response.text}")