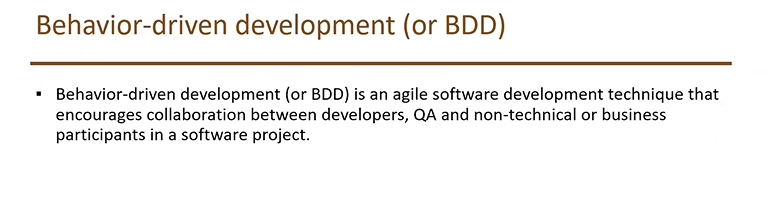
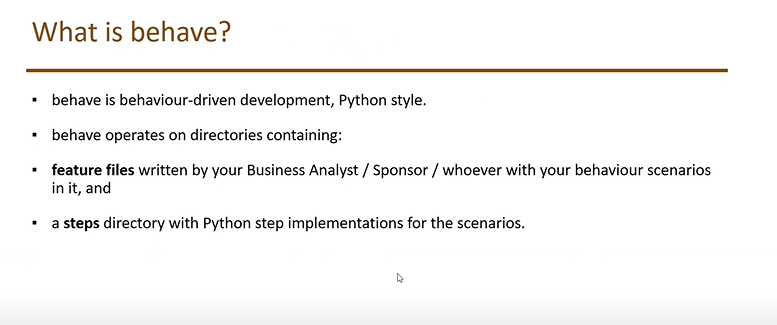
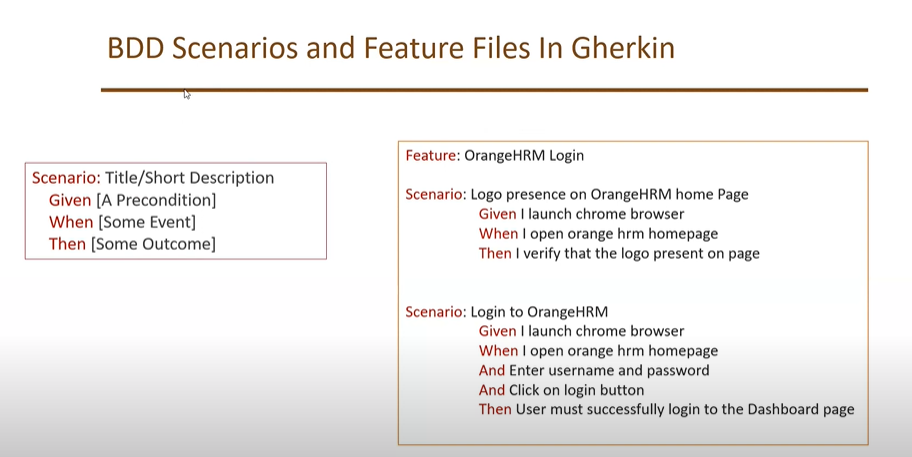
Python With BDD:







List of Softwares for Python with BDD:

1. Python
2. Pycharm IDE
3. Selenium
4. Behave

We need to install specific packages related to behave:

We need to go to the path of feature file and then enter behave <feature\_file\_name> to generate unimplemented feature

If we don’t want to give driver executable path we can place it in inside script folder and add the path to environment variables

To run any scenario go to the parent project path of terminal and enter input as behave, it will run all features

Set for Rules while creating Folder:

**All feature files should come under features**

**All steps should come under steps**

Paramertes in BDD Behave

Step:

When I enter credentails username as "atulb" and password as "ab00338092"

Stepdef:

@when(u'I enter credentails username as "{username}" and password as "{password}"')  
def enter\_user\_name\_and\_password(context,username,password):

Variables names mentioned in parameter and step should match

Scenario Outline Examples:

Feature: Check logout functionality  
 Scenario Outline: Check user is able to logout after logging in  
 Given I launch the webiste  
 When I enter credentails username as "<username>" and password as "<password>"  
 When I click on logout link  
 Then I should be redirected back to login page  
 Examples:  
 |username|password|  
 |atulb |ab00338092|  
 |atulb |ab00338092|

Cntrl+shift+F 🡪 Find word places

Hooks mechanism can be achived using environment in behave, we need to create environment.py under features folder and add below methods based on requirements

**before\_step(context, step), after\_step(context, step)**

These run before and after every step.

**before\_scenario(context, scenario), after\_scenario(context, scenario)**

These run before and after each scenario is run.

**before\_feature(context, feature), after\_feature(context, feature)**

These run before and after each feature file is exercised.

**before\_tag(context, tag), after\_tag(context, tag)**

These run before and after a section tagged with the given name. They are invoked for each tag encountered in the order they’re found in the feature file. See [controlling things with tags](https://behave.readthedocs.io/en/latest/tutorial.html#controlling-things-with-tags).

**before\_all(context), after\_all(context)**

Background is nothing but executing number of steps before each scenario

Feature: Check logout functionality  
  
 Background:  
 Given I launch the webiste  
  
 Scenario: Check user is able to logout after logging in  
 When I enter credentails username as "atulb" and password as "ab00338092"  
 When I click on logout link  
 Then I should be redirected back to login page

**Controlling Things With Tags**

You may also “tag” parts of your feature file. At the simplest level this allows behave to selectively check parts of your feature set.

@slow

**Scenario:** Weaker opponent

**Given** the ninja has a third level black-belt

**When** attacked by a samurai

**Then** the ninja should engage the opponent

then running behave --tags=slow will run just the scenarios tagged @slow. If you wish to check everything *except* the slow ones then you may run behave --tags="not @slow".

Another common use-case is to tag a scenario you’re working on with @wip and then behave --tags=wip to just test that one case.

Tag selection on the command-line may be combined:

* **--tags="@wip or @slow"**

This will select all the cases tagged *either* “wip” or “slow”.

* **--tags="@wip and @slow"**

This will select all the cases tagged *both* “wip” and “slow”.

I figured it out after spending more time reading the documentation. It is actually quite simple. By default, behave does **not** display any output (i.e. by using print()) unless there is a failure in the test. To force displaying all output regardless of the outcome of the test (pass/fail), all you need is to change some of the default settings. The easiest way to achieve that is to create a file named behave.ini in the root of your project's directory and put the following:

Filename: **behave.ini**

**Behave.ini should not be inside feature it should be inside project**

**To read command line values**

The userdata settings can be accessed as dictionary in hooks and steps by using the context.config.userdata dictionary.

behave -D server=obelix features/

behave --define server=obelix features/

**behave --define browser="chrome" --define env="stage" --tags=regression**

**Behave Udemy Tutorials:**

BDD f/w for Python:

Behave is BDD cucumber, not full owned by cucumber

Pip install behave

All feature files should be kept in a folder called features not any other folder

All steps should be kept in package called steps

To run all feature files we should use command called behave

To run with logs print behave --no-capture

Steps should be kept inside feature folder

For hooks we need to create python file name **environment.py**