Understanding TDD & BDD -BDD(Behave) & Gherkin

**Intro-What is Test driven development?**

* **TDD is a method/approach to develop a program/software where we write code using a Red-Green-Refactor cycle …**

1. **Start with a problem that needs a solution**/User story/high level software development/etc…
2. First you create a test based off a functionality you would want for your program.**(This is considered the red zone/phase as we have not yet introduced a function/unit yet and the test will always fail)**
3. Create a function that you previously wanted in step 1 & that will make the test pass**(This is known as the green zone/phase as we can properly test the function )**
4. Now we can optionally fix up all the code we just wrote if it’s needed**(This is known as the refractor zone/phase and this is where we improve all the code that has currently been made and is an optional cycle)**
5. **Restart the cycle**(step1- step3) **to create a new code/ a new cycle**

**Why is this beneficial?**

1. TDD will make sure that **every code you write is tested**.
2. This method **works well** with faster coding practices such as **agile**
3. **Prevents** possible **bug** issues quickly **and early** in the development cycle.

**simple example of TDD:**

* Problem: I want a user to enter a number and I will show them 1 - that user number
* **(Red cycle) create a test**:

#Import testing suite like unittest

Import unittest

Def is\_A\_Number(userNumber):

try:

If isinstance(userNumber,int) # if an integer

Return True

Else: # if not an integer raise an error

raise Exception(‘User restart and please enter a number, “)

Except Exception as err:

print(f “because, {err}”)

Class TestExample(unittest.TestCase)

Def TestNum(self); # test that userNumber is an int

# Get user input

userNumber = input(“Enter a number”)

# test user input

self.assertTrue(is\_A\_Number(userNumber))

# call green function to output correct code:

self.GreenFunction(userNumber)

* **(Green cycle) Create function code**

Def GreenFunction(self,userNumber): # outputs number 1-userNumber

For elem in range(1,userNumber + 1):

print(elem)

**Intro-What is Behavior driven development, BEHAVE, and Gherkin?**

* BDD is a method(a way to do something)by being a liaison between the developer and non-developers. Think of it as a means to communicate the program’s purpose in a more human readable way to someone with little to no knowledge of code.

BDD uses a GIVEN WHEN THEN phase where:

1. **GIVEN** = Initial action performed to create the scenario/code/ **the starting point**
2. **WHEN** = Secondary action that is important to the code’s functionality/ the **primary action you want to happen**
3. **THEN** = **Result** of WHEN
4. (OPTIONAL) **AND** = **Additional step** & can be used inside any of the phases.

* Gherkin is a language used to enact BDD.
* BDD looks very similar to pseudocode,but with the use of a BDD framework(such as BEHAVE(BDD framework used for Python)) we can actually execute the BDD’s writing.

**So in all this is how we can use BDD BEHAVE, and Gherkin Together:**

1. **BDD Is the idea of explaining the behavior/purpose of code in a human readable fashion/way**
2. **Gherkin is one way (there are other options as well) we can create a human readable explanation of the code(that can be executable).**
3. **Behave is the way we would execute this BDD from Gherkin (in Python specifically)**

**How to execute Gherkin/Example:**

1. **Choose a BDD framework (in this example we will use BEHAVE)**
2. **Create a Gherkin file ex:**

Feature: Addition Calculation

I want to add two numbers together that I personally pick

So that I can quickly calculate the results

Scenario: Add two numbers using GUI

Given the calculator is open

When I enter 10 into the calculator

And I select the "Addition" button

And I enter 3 into the calculator

And I press the "Equal" button

Then the calculator displays the result as 13

Scenario: Add a negative number using GUI

Given the calculator is open

When I enter 7 into the calculator

And I select the "Addition" button

And I enter -2 into the calculator

And I press the "Equal" button

Then the calculator displays the result as 5

1. **First pip install behave**
2. **Use BEHAVE to execute(GHERKIN) in Python ex:**

from behave import given, when, then #get the ability to use behave

@given('the calculator is open')

current\_calc = Calculator() # initializes calc object

@when('I enter a number into the calculator as the first number')

current\_calc.enter\_number() #input number

@when(‘I press enter for the first time’)

current\_calc.store\_number() # stores user input number

@when('I enter a number into the calculator as the second number')

current\_calc.enter\_number() #input number

@when(‘I press enter for the second time’)

current\_calc.store\_number() # stores user input number

@when('I press the "Equal" button')

current\_calc.press\_equal\_button() #processes the calculation of all numbers that were #entered by user

@then('the calculator displays the result')

current\_calc.get\_displayed\_result() # shows the result

# class creation

class Calculator:

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

self.result = 0

def enter\_number(self):

# enter the number

Num =int( input(“Enter a number”))

def store\_number(self,Num):

self.Numbers = []

if Num != 0:

self.Numbers.append(Num)

def press\_equal\_button(self):

# calculate the result

result = 0

For elem in self.Numbers:

self.result += elem

def get\_displayed\_result(self):

# displayed result

print(self.result)

**Works Cited**

**Gogna, Vandan. “A Practical Example Using Test Driven Development.” *Medium*, 14 July 2021,** [**https://vandangogna.medium.com/a-practical-example-using-test-driven-development-88b4536ac574**](https://vandangogna.medium.com/a-practical-example-using-test-driven-development-88b4536ac574)**.**

**Accessed 30 Oct. 2023.**

**Automation Panda, editor. “PYTHON TESTING 101: BEHAVE.” *Automation Panda*, 14 July 2021, https://automationpanda.com/2018/05/11/python-testing-101-behave/. Accessed 31 Oct. 2023.**

**Automation Panda, editor. “BDD 101: INTRODUCING BDD.” *Automation Panda*, 14 July 2021, https://automationpanda.com/2017/01/25/bdd-101-introducing-bdd/. Accessed 31 Oct. 2023.**

**CSER, TAMAS. “Behavior-Driven Development.” *Functionize*, 15 Aug. 2023, https://www.functionize.com/automated-testing/behavior-driven-development. Accessed 31 Oct. 2023.**

**behave.readthedocs, editor. “Tutorial.” *Behave.readthedocs*, https://behave.readthedocs.io/en/stable/tutorial.html. Accessed 31 Oct. 2023.**

**behat, editor. “Writing Features - Gherkin Language.” *Behat*, https://docs.behat.org/en/v2.5/guides/1.gherkin.html. Accessed 31 Oct. 2023.**