Go Track

Week 9 Practical

NGEE ANN POLYTECHNIC

Git

Basic Git

Description	*In this lesson, you will learn about basic Git.*
Learning Objectives	Understand the basic of using Git
Duration	30 ~ 45 minutes

Creation of GitHub account.

GitHub link: https://github.com/
Select “Sign Up” and follow the instructions given.

Activity #1: Simple Go Hello World Git Console

1. Create a new public repository in GitHub and name it “HelloWorldTest”.
2. Clone the repository into your local workspace using the command prompt.
3. Create a simple HelloWorld.go file that prints “Hello world” on run using Visual Studio Code.
4. Commit the initial copy of the file into the local repository using the command prompt.
5. Push the file to the remote repository using command prompt.
6. Create a branch called “HelloNewWorld”
7. Switch to the new branch.
8. Edit the simple HelloWorld.go file to print “Hello to the new world from <Your name>” instead.
9. Commit the copy of the file into the local repository using the command prompt.
10. Switch back to the main branch.
11. Do a merge to update the HelloWorld.go file.
12. Push the file to the remote repository using command prompt.

Activity #2: Go Hello World Again? Let's use GitHub instead

This activity requires the use of your team registered.
Register your Git Username and the Simple Go Hello World Git Repository link.

Assign each pair in a round-robin sequence. I.e. Member 1 will modify Member 2, Member 2 will modify Member 3 and Member 3 will modify Member 1 etc.

Member 2 will add Member 1 as the collaborator under Settings → Manage access and invite Member 1.

Member 1 shall be the receiving developer
Member 2 shall be the changing developer

By Console:

Each changing developer will on their own local cloned repository

1. Create a new branch
2. Switch to the new branch
3. Add in an additional line of code of their choice (Please be nice!)
4. Commit the new line of code
5. Push the branch to the main using

```
git push –u origin <branchName>
```

Receiving Developers shall receive a simple compare and pull request popup in Github and do the following

1. Start the pull request
2. Accept and merge the pull request.
3. Receiving Developers can choose to delete the branch if they wish.

Activity #3: Go Hello World GitHub Pull Requests

This activity will be similar to activity #2 except all code manipulations will be done in GitHub.

Changing Developer shall navigate to Receiving Developer repo and create a branch. Again, new lines of your choice can be added and propose the changes.

Receiving Developers shall then perform the pull request as per normal.

Activity #4: Simple Calculator with GitHub

Create a new public repository in GitHub and name it "MyGoCalculator".

Add the user name of the developer B as your collaborator under Settings → Manage access and invite the developer B using his or her username and vice versa.

Making use of Git Issues, begin using TDD methodology to assign to one another the role of developer and QA tester and vice versa.

Ensure TDD steps are followed through the communications of Git Issues.

NGEE ANN POLYTECHNIC

Testing and Monitoring

Testing Basics

Description	In this lesson, you will learn about Testing and Monitoring in Go.
Learning Objectives	Understand testing basics
Duration	30 ~ 45 minutes

Activity #1: Boundary and Equivalence Testing

Consider a simple setting for an air conditioning unit.

1. The setting of an air conditioning unit shall
    a. Have 4 different modes
        i. Auto mode
        ii. Cool mode
        iii. Dry mode
        iv. Fan mode
    b. Have a range of 17 Degree Celsius to 38 Degree Celsius
    c. Have a on and off switch
    d. Have a fan speed of low, medium, high and auto.
2. A beep would be sounded upon valid selection from the user.
3. Construct simple test cases using boundary and equivalence testing.

Activity #2: Simple Calculator Test Cases

Consider a simple calculator.

1. The simple calculator shall consist of the following simple functions
    a. Add
    b. Subtract
    c. Multiply
    d. Divide
2. All functions are only allowed to take in two integers.
3. All functions are to return with integers except divide which will return a float64 value.
4. Construct simple test cases using boundary and equivalence testing. This will be in preparation for Test Driven Development.

Activity #3: Simple Calculator Test Cases

Consider a simple calculator test cases and create the calculator in go.

NGEE ANN POLYTECHNIC

Development in TDD

Use of TDD methods in development

Description	***In this lesson, you will learn about the TDD in Go.***
Learning Objectives	**Understand the application of TDD in software development.**
Duration	**30 ~ 45 minutes**

The following activities shall make use of the following set of data.

Category	Value of first integer	Value of second integer	Want
Add	1	2	3
Add	1	0	1
Add	2	-2	0
Subtract	1	1	0
Subtract	10	5	5
Subtract	-5	-5	0
Multiply	1	5	5
Multiply	5	5	25
Multiply	100	0	0
Divide	100	5	20
Divide	100	1000	0.1
Divide	100	0	0 (Error)

Activity #1: Simple Calculator using TDD with Standard Test

1. Refer to the simple calculator test cases developed. Apply TDD and create a simple go calculator and its relevant test cases.

2. Create a test suite using Standard Test with the values given.

Activity #2: Simple Calculator using TDD with Testify

1. Refer to the simple calculator test cases developed. Apply TDD and create a simple go calculator and its relevant test cases.

2. Create a test suite using Testify with the values given.

3. Make slight changes to the code to skip the one or two category.

Activity #3: Simple Calculator using TDD with GoCheck

1. Refer to the simple calculator test cases developed. Apply TDD and create a simple go calculator and its relevant test cases.

2. Create a test suite using GoCheck with the values given.

3. Using the command prompt, control the test scenarios using flag -*check.f*

Activity #4: Simple Calculator using TDD with Goblin

1. Refer to the simple calculator test cases developed. Apply TDD and create a simple go calculator and its relevant test cases.

2. Create a test suite using Goblin with the values given.

Activity #5: Simple Web Calculator Testing

Consider a simple web calculator server.

1. The simple calculator shall consist of the following simple functions
   a. Add
   b. Subtract
   c. Multiply
   d. Divide
2. All functions are only allowed to take in two integers.
3. All functions are to return with integers except divide which will return a float64 value.
4. Activate the calculator by using URL POST "localhost:8080/calc?action=add&value1=value&value2=value"
5. Return a simple printout on the client browser.
6. Develop the codes for a Simple Web Calculator.
7. Modify the Web Calculator code to include test for Handlers and Routing.
8. Create a test suite for the Web Calculator.

Monitoring in Go

Coverage

Description	*In this lesson, you will learn about monitoring in Go.*
Learning Objectives	**Understand coverage to software development**
Duration	**30 ~ 45 minutes**

Activity #1: Monitoring the Go Tests

1. Refer to the previous development of the simple calculator.
2. Run coverage tests on each kind of test and observe the different response.
3. Output the coverage into HTML and observe the different coverage statements.

NGEE ANN POLYTECHNIC

YAML in CICD

Basic YML

Description	***In this lesson, you will learn about the CICD in Go.***
Learning Objectives	**Understand the basic of using YAML files in CICD**
Duration	**30 ~ 45 minutes**

Activity #1: Simple Actions Setup

Construct the directory for the YAML file to reside in the Github repository.

Create a .yml file and attempt to build an OS of choice.

Experiment with the basic settings and print an output.

A Sample is provided below:

```
jobs:
  my-job:
    name: My Job
    runs-on: ubuntu-latest
    steps:
    - name: Print a greeting
      run: |
        echo Hello there!
```

Activity #2: Virtual Machine using YML and GitHub Actions

Create a new YAML file and create a virtual machine that runs on Ubuntu.

Setup the virtual machine to have Go version 1.13.7 installed.

Publish the version of Go to verify the installation

Activity #3: GitHub Actions Remote Checkouts

Use the following token:
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

Add a token to the repository under settings > secrets

Modify the YAML file to remotely download a text file

The remote example repo is under
lowkh2/GoTrackRemoteExample

More info on checkout actions:
https://github.com/actions/checkout

Activity #4: Build and Test using YML and GitHub Actions

Add onto the YAML file created in activity #1 and include the relevant tools to create the following capabilities to the automation.

Addon	Capability
JasonEtco/create-an-issue@v2	To raise notifications in Issues
github.com/franela/goblin	To enable test using Goblin
github.com/tebeka/go2xunit	To convert report from test to XML
EnricoMi/publish-unit-test-result-action@v1.5	To read XML and display test results

Activity #5: Release/ Deployment using YML and GitHub Actions

Add onto the YAML file created in activity #2 and include the relevant tools to create the release/ deployment capabilities of your choice to the automation.

A simple tag triggered method will be shown.