**Practical Exercise 30 - Working with Outputs in Custom Actions**

**Exercise Description**

**In this practical exercise, our goal is to understand how to set outputs from all three different types of custom actions available in GitHub Actions.**

Here are the instructions for the exercise:

1. Let's start with the composite custom action. Extend the composite custom action action.yaml file under .github/actions/composite-cache-deps by:
   1. Adding an outputs top-level key, and under it a single output named installed-deps. It should have the description of Whether dependencies were installed or not, and a value of ${{ steps.cache.outputs.cache-hit != 'true' }}. Composite actions allow us to directly reference outputs from individual steps, so setting up an output within the custom action is very straightforward.
2. Extend the workflow 17-1-custom-actions-composite.yaml by:
   1. Adding an id of setup-deps to the step named Setup Node and NPM Dependencies.
   2. Adding an additional step after the setup-deps step:
      1. The new step should be named Print setup deps output.
      2. It should print "Installed dependencies: <value of the installed-deps output from the setup-deps step>".
3. Commit the changes and push the code. Trigger the 17-1-custom-actions-composite.yaml workflow from the UI and take a few moments to inspect the output of the workflow run.
4. Let's now move on to the JavaScript custom action. Extend the JavaScript custom action action.yaml file under .github/actions/js-dependency-update by:
   1. Adding an outputs top-level key, and under it a single output named updates-availabe. It should have the description of Whether there are updates available. JavaScript custom actions require us to set the values of the outputs from within the JavaScript code, so let's do that next.
5. From within the index.js file, use the setOutput method from the @actions/core package to set the updates-available output to either true or false depending on whether there were updates available.
6. Extend the workflow 17-2-custom-actions-js.yaml by:
   1. Adding an id of update-deps to the step named Check for dependency updates.
   2. Adding an additional step after the update-deps step:
      1. The new step should be named Print custom action output.
      2. It should print "Updates available: <value of the updates-available output from the setup-deps step>".
7. Commit the changes and push the code. Trigger the 17-2-custom-actions-js.yaml workflow from the UI and take a few moments to inspect the output of the workflow run.
8. Last but not least, let's move on to the Docker custom action. Extend the Docker custom action action.yaml file under .github/actions/docker-ping-url by:
   1. Adding an outputs top-level key, and under it a single output named url-reachable. It should have the description of Whether the URL is reachable. Docker custom actions require us to set the values of the outputs from within the code, so let's do that next.
9. For Docker custom actions which are not written in JavaScript or TypeScript, we cannot leverage the @actions/core package to set outputs. Therefore, we must append the values to the output file, as we have done in **Section 12: Inputs & Outputs**:
   1. From within the main.py file, open the file which path is stored under the GITHUB\_OUTPUT environment variable. Use the append mode by providing 'a' as the second argument to the open() function.
   2. Use the print statement to write a line in the format <key>=<value> to the opened file (this can be done like so: print(f'url-reachable=<interpolate return value of ping\_url here>', file=file))
10. Extend the workflow 17-3-custom-actions-docker.yaml by:
    1. Adding an id of ping-url to the step named Ping URL.
    2. Adding an additional step after the ping-url step:
       1. The new step should be named Print output from ping url.
       2. It should print "URL reachable: <value of the url-reachable output from the ping-url step>".
11. Commit the changes and push the code. Trigger the 17-3-custom-actions-docker.yaml workflow from the UI and take a few moments to inspect the output of the workflow run.