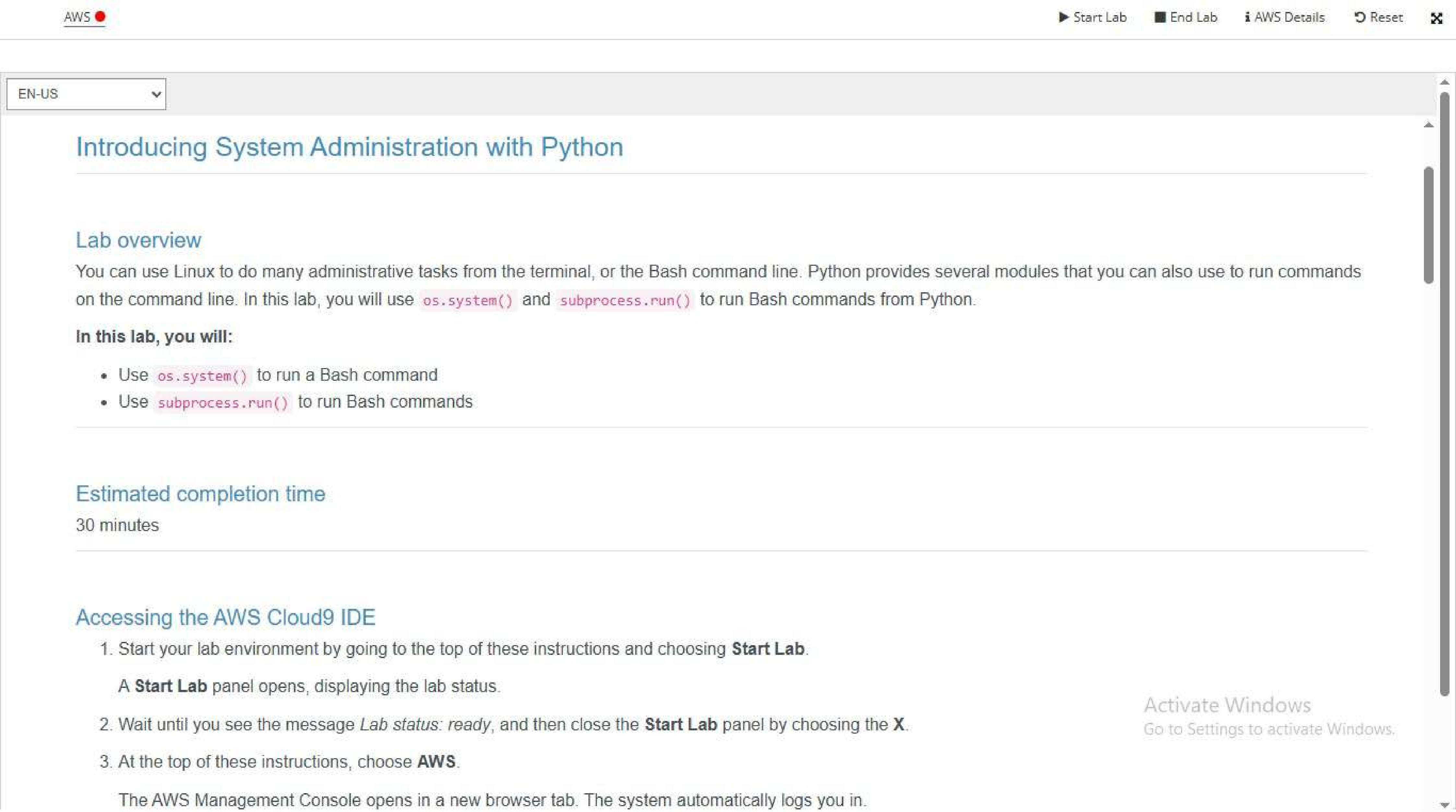


Introducing System Administration with Python

Lab overview

You can use Linux to do many administrative tasks from the terminal, or the Bash command line. Python provides several modules that you can also use to run commands on the command line. In this lab, you will use `os.system()` and `subprocess.run()` to run Bash commands from Python.

In this lab, you will:

- Use `os.system()` to run a Bash command
- Use `subprocess.run()` to run Bash commands

Estimated completion time

30 minutes

Accessing the AWS Cloud9 IDE

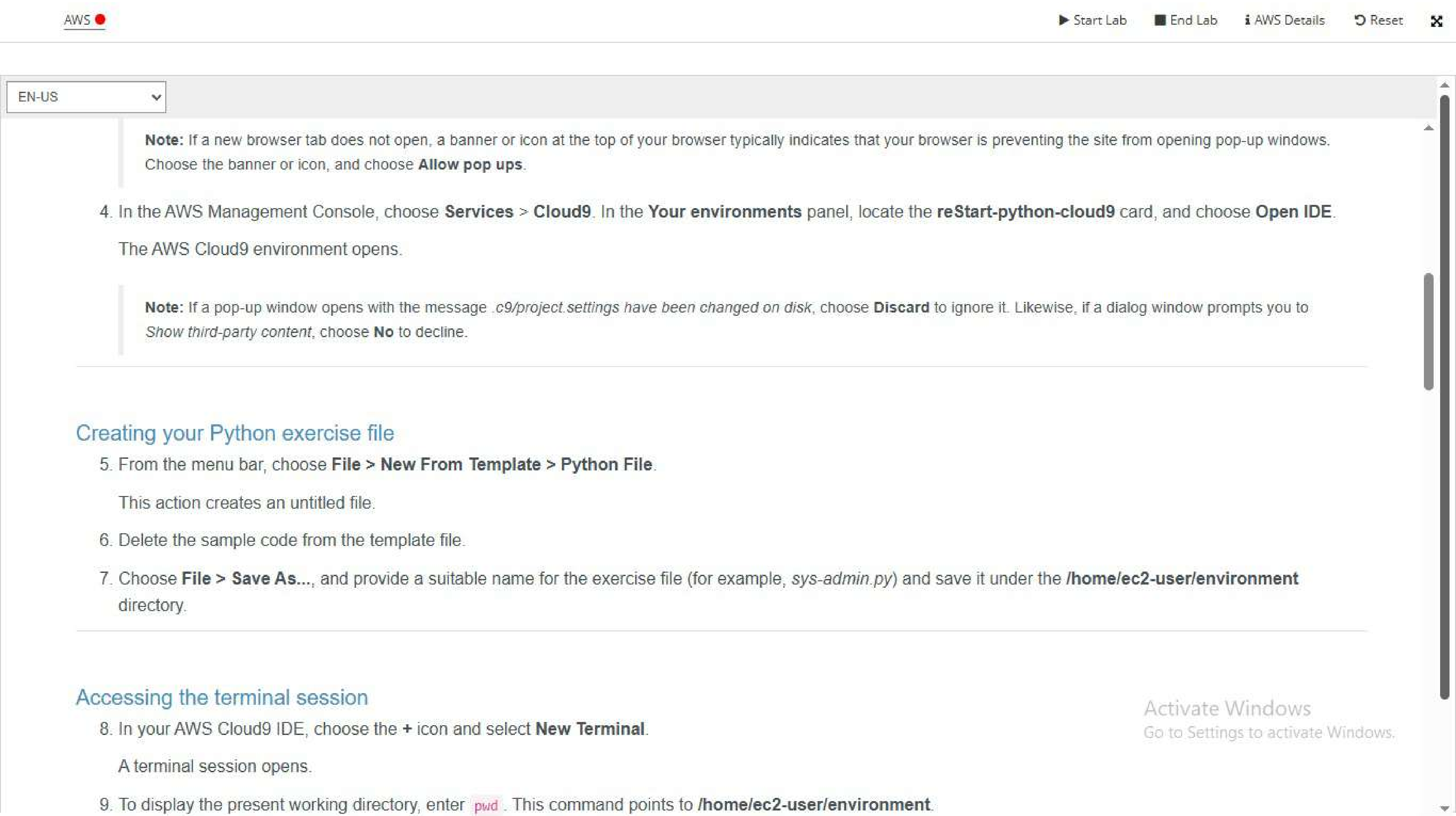
1. Start your lab environment by going to the top of these instructions and choosing **Start Lab**.

   A **Start Lab** panel opens, displaying the lab status.

2. Wait until you see the message *Lab status: ready*, and then close the **Start Lab** panel by choosing the **X**.

3. At the top of these instructions, choose **AWS**.

   The AWS Management Console opens in a new browser tab. The system automatically logs you in.



> **Note:** If a new browser tab does not open, a banner or icon at the top of your browser typically indicates that your browser is preventing the site from opening pop-up windows. Choose the banner or icon, and choose **Allow pop ups**.

4. In the AWS Management Console, choose **Services** > **Cloud9**. In the **Your environments** panel, locate the **reStart-python-cloud9** card, and choose **Open IDE**.

   The AWS Cloud9 environment opens.

   > **Note:** If a pop-up window opens with the message *.c9/project.settings have been changed on disk*, choose **Discard** to ignore it. Likewise, if a dialog window prompts you to *Show third-party content*, choose **No** to decline.

Creating your Python exercise file

5. From the menu bar, choose **File > New From Template > Python File**.

   This action creates an untitled file.

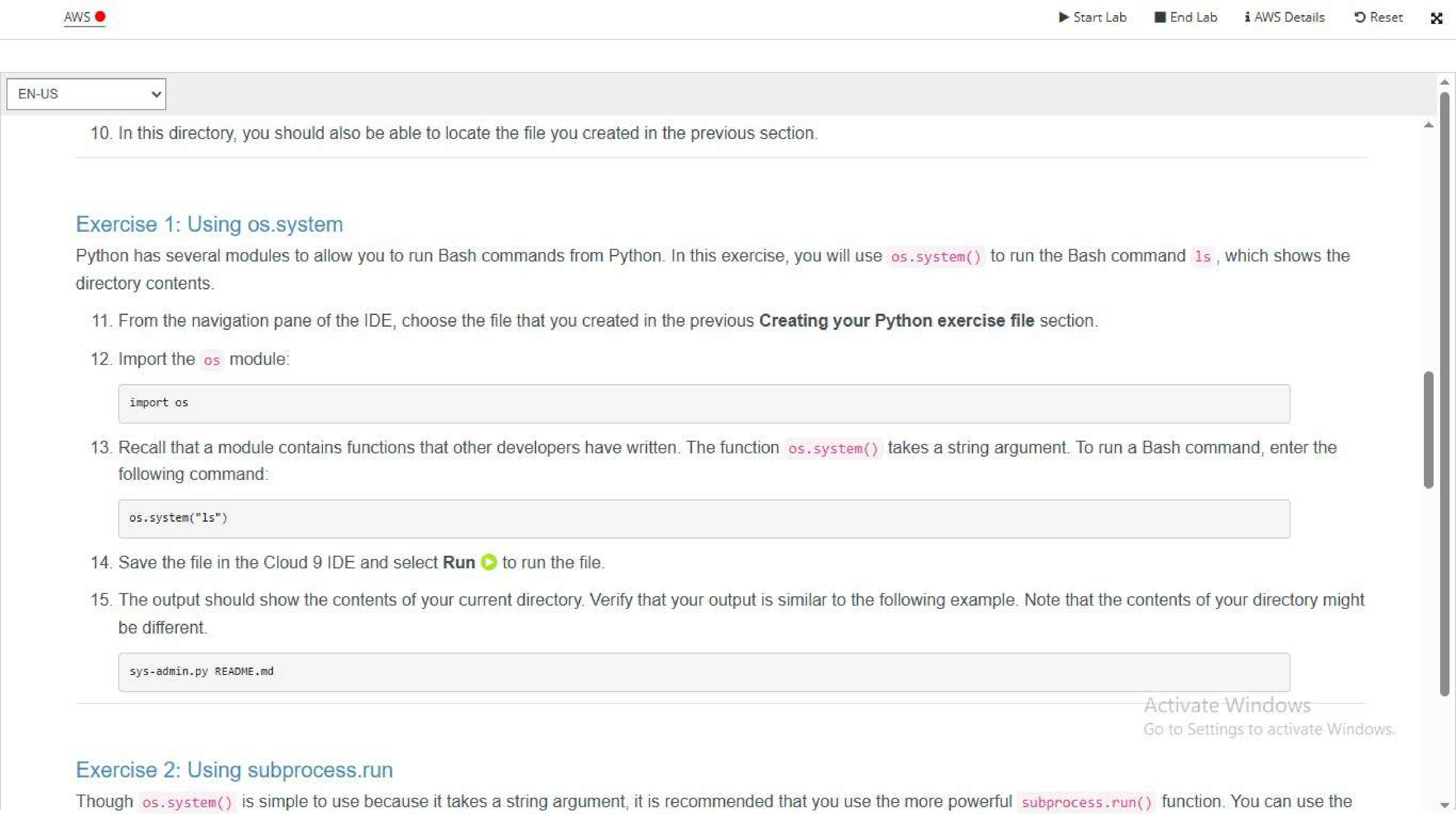
6. Delete the sample code from the template file.
7. Choose **File > Save As...**, and provide a suitable name for the exercise file (for example, *sys-admin.py*) and save it under the **/home/ec2-user/environment** directory.

Accessing the terminal session

8. In your AWS Cloud9 IDE, choose the **+** icon and select **New Terminal**.

   A terminal session opens.

9. To display the present working directory, enter `pwd`. This command points to **/home/ec2-user/environment**.



10. In this directory, you should also be able to locate the file you created in the previous section.

Exercise 1: Using os.system

Python has several modules to allow you to run Bash commands from Python. In this exercise, you will use `os.system()` to run the Bash command `ls`, which shows the directory contents.

11. From the navigation pane of the IDE, choose the file that you created in the previous **Creating your Python exercise file** section.
12. Import the `os` module:

```
import os
```

13. Recall that a module contains functions that other developers have written. The function `os.system()` takes a string argument. To run a Bash command, enter the following command:

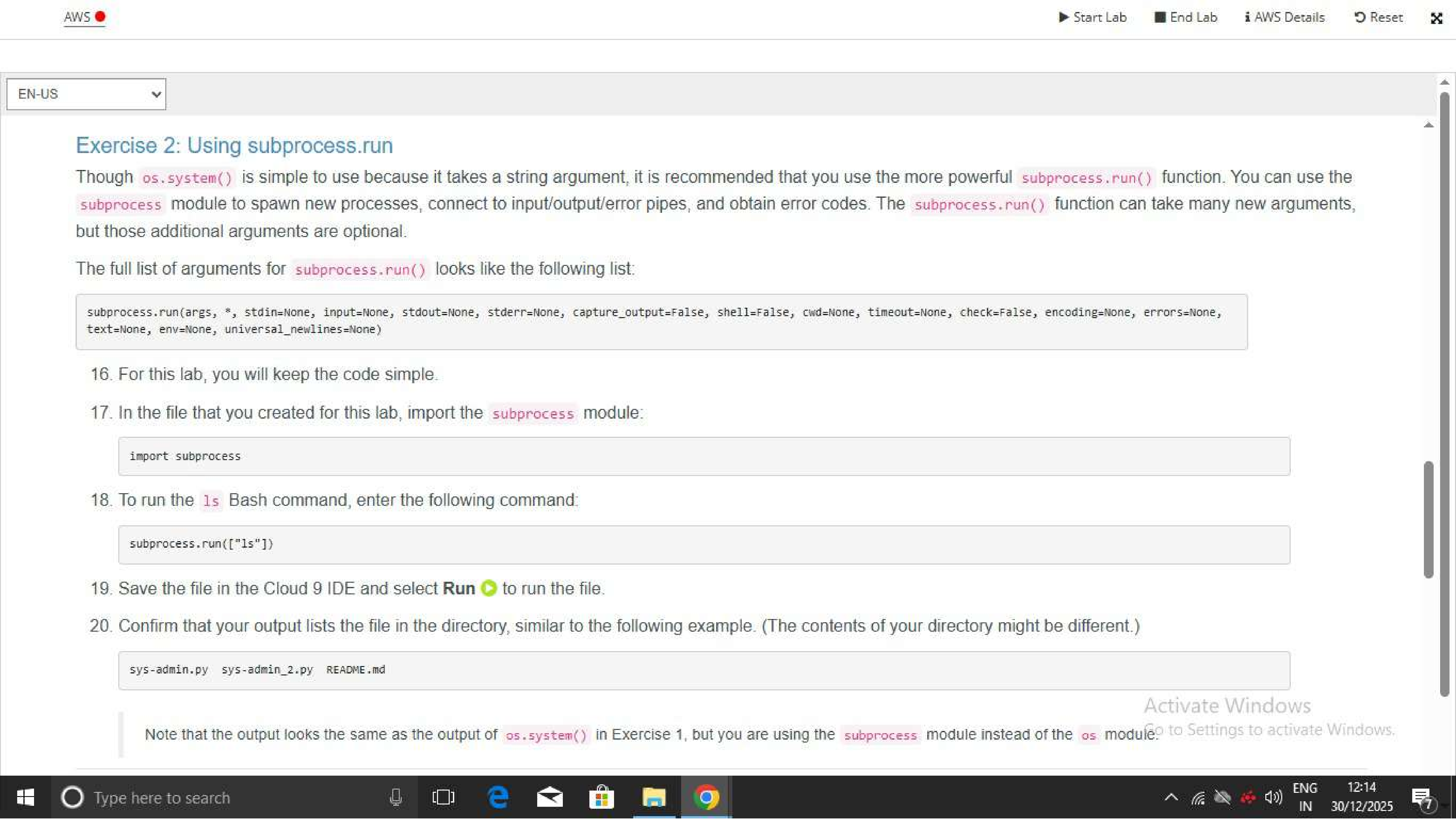
```
os.system("ls")
```

14. Save the file in the Cloud 9 IDE and select **Run** to run the file.
15. The output should show the contents of your current directory. Verify that your output is similar to the following example. Note that the contents of your directory might be different.

```
sys-admin.py README.md
```

Exercise 2: Using subprocess.run

Though `os.system()` is simple to use because it takes a string argument, it is recommended that you use the more powerful `subprocess.run()` function. You can use the



Exercise 2: Using subprocess.run

Though `os.system()` is simple to use because it takes a string argument, it is recommended that you use the more powerful `subprocess.run()` function. You can use the `subprocess` module to spawn new processes, connect to input/output/error pipes, and obtain error codes. The `subprocess.run()` function can take many new arguments, but those additional arguments are optional.

The full list of arguments for `subprocess.run()` looks like the following list:

```
subprocess.run(args, *, stdin=None, input=None, stdout=None, stderr=None, capture_output=False, shell=False, cwd=None, timeout=None, check=False, encoding=None, errors=None,
text=None, env=None, universal_newlines=None)
```

16. For this lab, you will keep the code simple.
17. In the file that you created for this lab, import the `subprocess` module:

    ```
    import subprocess
    ```

18. To run the `ls` Bash command, enter the following command:

    ```
    subprocess.run(["ls"])
    ```

19. Save the file in the Cloud 9 IDE and select **Run** to run the file.
20. Confirm that your output lists the file in the directory, similar to the following example. (The contents of your directory might be different.)

    ```
    sys-admin.py  sys-admin_2.py  README.md
    ```

    > Note that the output looks the same as the output of `os.system()` in Exercise 1, but you are using the `subprocess` module instead of the `os` module.

EN-US

Exercise 3: Using subprocess.run with two arguments

In Python, the square brackets are list data types, which means that `run()` can take a list of arguments. Continue to add to the Python script.

21. In the lab file for this exercise, modify the final line of the script to include an additional argument:

```
subprocess.run(["ls","-l"])
```

22. The `"-l"` is an argument that tells the `ls` command to use a long-listing format.
23. Save the file in the Cloud 9 IDE and select **Run** to run the file again.
24. Confirm that your output is similar to the following example.

```
total 12
-rw-r--r-- 1 ec2-user ec2-user  55 Apr 16 20:20 sys-admin.py
-rw-r--r-- 1 ec2-user ec2-user 343 Apr 16 19:07 sys-admin_2.py
-rw-r--r-- 1 ec2-user ec2-user 569 Apr  6 02:17 README.md
```

Exercise 4: Using subprocess.run with three arguments

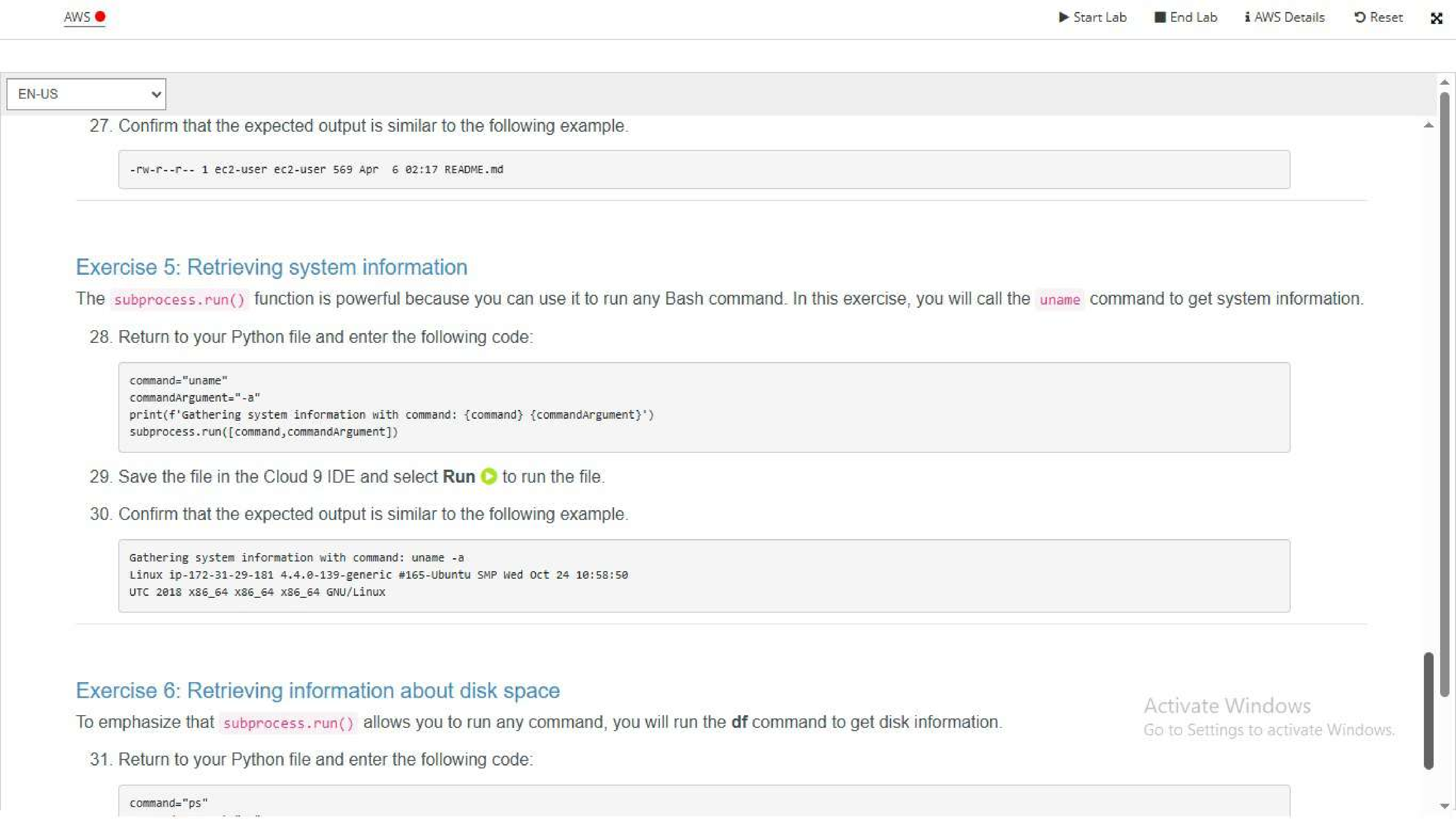
You will now call `subprocess.run()` with three arguments. The third argument will be a directory name.

25. Return to your Python file and modify the final line of the script:

```
subprocess.run(["ls","-l","README.md"])
```

26. Save the file in the Cloud 9 IDE and select **Run** the file.
27. Confirm that the expected output is similar to the following example.

```
-rw-r--r-- 1 ec2-user ec2-user 569 Apr  6 02:17 README.md
```



27. Confirm that the expected output is similar to the following example.

```
-rw-r--r-- 1 ec2-user ec2-user 569 Apr  6 02:17 README.md
```

Exercise 5: Retrieving system information

The `subprocess.run()` function is powerful because you can use it to run any Bash command. In this exercise, you will call the `uname` command to get system information.

28. Return to your Python file and enter the following code:

```
command="uname"
commandArgument="-a"
print(f'Gathering system information with command: {command} {commandArgument}')
subprocess.run([command,commandArgument])
```

29. Save the file in the Cloud 9 IDE and select **Run** to run the file.

30. Confirm that the expected output is similar to the following example.

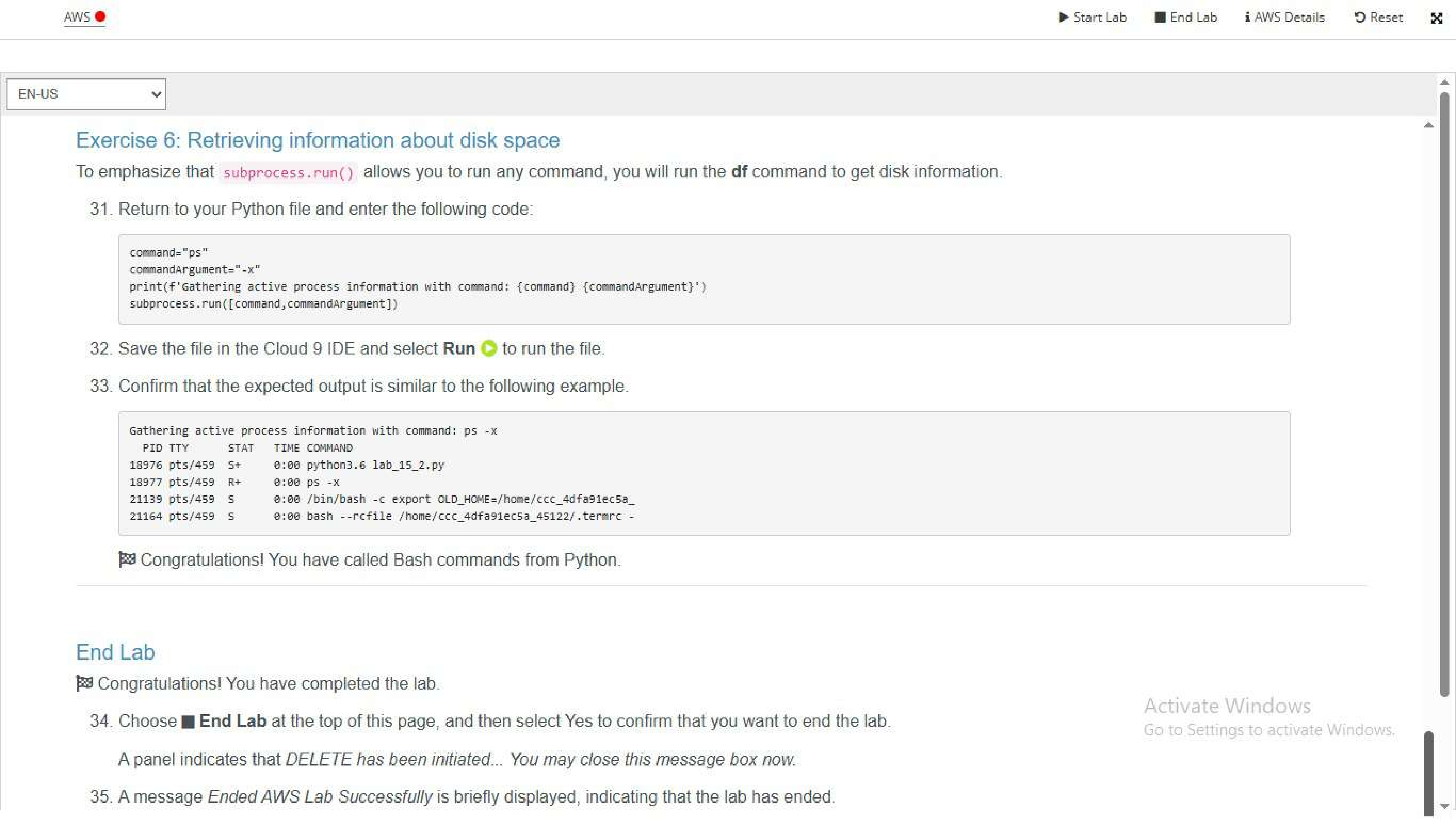
```
Gathering system information with command: uname -a
Linux ip-172-31-29-181 4.4.0-139-generic #165-Ubuntu SMP Wed Oct 24 10:58:50
UTC 2018 x86_64 x86_64 x86_64 GNU/Linux
```

Exercise 6: Retrieving information about disk space

To emphasize that `subprocess.run()` allows you to run any command, you will run the **df** command to get disk information.

31. Return to your Python file and enter the following code:

```
command="ps"
```



Exercise 6: Retrieving information about disk space

To emphasize that `subprocess.run()` allows you to run any command, you will run the **df** command to get disk information.

31. Return to your Python file and enter the following code:

```
command="ps"
commandArgument="-x"
print(f'Gathering active process information with command: {command} {commandArgument}')
subprocess.run([command,commandArgument])
```

32. Save the file in the Cloud 9 IDE and select **Run** to run the file.

33. Confirm that the expected output is similar to the following example.

```
Gathering active process information with command: ps -x
  PID TTY      STAT   TIME COMMAND
18976 pts/459  S+     0:00 python3.6 lab_15_2.py
18977 pts/459  R+     0:00 ps -x
21139 pts/459  S      0:00 /bin/bash -c export OLD_HOME=/home/ccc_4dfa91ec5a_
21164 pts/459  S      0:00 bash --rcfile /home/ccc_4dfa91ec5a_45122/.termrc -
```

🏁 Congratulations! You have called Bash commands from Python.

End Lab

🏁 Congratulations! You have completed the lab.

34. Choose ■ **End Lab** at the top of this page, and then select Yes to confirm that you want to end the lab.

    A panel indicates that *DELETE has been initiated... You may close this message box now.*

35. A message *Ended AWS Lab Successfully* is briefly displayed, indicating that the lab has ended.