Source: <https://learn.umgc.edu/d2l/le/news/766913/3013162/view?ou=766913>

Homework 1 - Guidance - Requirement Clarification

Posted May 18, 2023 12:01 AM

Before we get into the details - [Homework 1 - Using S3](https://learn.umgc.edu/d2l/common/dialogs/quickLink/quickLink.d2l?ou=766913&type=dropbox&rcode=UMUC-1706565)**is due on May 30, 2023 @ 11:59PM ET.** I am posting this announcement very EARLY to encourage students to start EARLY.

The University produced Lab PDFs are sometimes a bit short on details. This can lead to confusion and students needing to ask a number of questions. Questions are always welcome - but I am going to provide some clarity on the assignment and the deliverable requirements. **I will post a solution to Homework 1 on June 10, 2023 @ 12:00AM ET for students to review. After this answer is posted - I cannot accept anymore Homework 1 submissions.** Make sure you get your **Homework 1** in **before** the solution is posted **June 10, 2023 @ 12:00AM ET**.

The attached video will go over much of this post - please take some time to watch the attached video - **NOTE: It is from a previous semester so the dates are off - but the content is accurate ;-) . AWS Educate has changed to AWS Academy - but the other content is accurate, minus the dates.**

**The Homework 1 is worth 17.5% of your overall grade.**

It is broken into 4 parts. The largest and most complicated part is Part 3 - start early... start early... start early.

If you get stuck - you can take advantage of my "virtual office hours" and ask me questions over email or post questions to the [Ask the Professor](https://learn.umgc.edu/d2l/common/dialogs/quickLink/quickLink.d2l?ou=766913&type=discuss&rcode=UMUC-6966279) section.  **If you are confused by any of what I am going to detail below - Please post a question to the**[**Ask the Professor**](https://learn.umgc.edu/d2l/common/dialogs/quickLink/quickLink.d2l?ou=766913&type=discuss&rcode=UMUC-6966279)**section, email me directly, or dial into the optional Weekly Monday ZOOM (**[**ZOOM Meeting Information**](https://learn.umgc.edu/d2l/le/766913/discussions/threads/28221502/View)**).**

**Part 4 - Overall Report (20% of assignment overall)**

You are going to be providing a "formal" report. Much of the below guidance is standard APA formatting you should be used to for University assignments. Here is a link to the [University APA Guidance](https://libguides.umgc.edu/apa-examples)and a link with [video guidance on paper formatting](https://libguides.umgc.edu/apa-document-formatting).

The assignment specifically requires the following submission details:

1. Page numbers should be included for all pages (except the title page) and be at the top right of the page.

2. Paragraphs should be double-spaced with 1" margins on all sides.

3. 12 pt. Times New Roman font or similar should be used.

4. Figures should have titles and numbers.

5. The document should contain minimal spelling and grammar errors.

6. References are included (you should reference the code used) and provided in APA format.

**Part 3 - Python Portion (50% of assignment overall)**

You are asked to create a Python 3 application that will provide some functionality in interfacing with S3 via the command-line/console of your Cloud9IDE. The Week 2 content provides a number of code snippets ([Learning Resources](https://learn.umgc.edu/d2l/common/dialogs/quickLink/quickLink.d2l?ou=728214&type=content&rcode=UMUC-6966673)  [ZIP - Week2-S3-PythonCode](https://leocontent.umgc.edu/content/dam/permalink/f7e83659-0138-4ab8-9fa9-fbf91e829d87.html))  that can be used to construct many of the features requested. These are AWS examples and are NOT Pylint'ed nor do they provide all of the error checking you will have to implement in your application. They do provide a quick starting point when looking for an AWS boto3 code example to perform a requested action. You must submit your actual code of this portion of the assignment.

In the "hints" section is says:

- **The menu should be launched from the AWS Cloud9 command line interface and contain options to perform each of the functionality listed above. (Review the Python textbook for basic functionality as needed)**

- **Use the code examples provided in the content area to cobble together the application(**[**Learning Resources**](https://learn.umgc.edu/d2l/common/dialogs/quickLink/quickLink.d2l?ou=728214&type=content&rcode=UMUC-6966673)[**ZIP - Week2-S3-PythonCode**](https://leocontent.umgc.edu/content/dam/permalink/f7e83659-0138-4ab8-9fa9-fbf91e829d87.html)**)**

- **Be sure to thoroughly test each function and document the results of your tests that include the input, expected output, actual output and if the test passed. Screen captures should be included with each test case.**

- **Document your code with comments and use small functions as opposed to larger main functions**.

We are in a 400-level class so there are some expectations about your deliverables. This class required SDEV300 as a prerequisite. In SDEV300 - Building Secure Python Applications you learned Python, how to properly comment/document within your applications, how to build functions/classes/modules, and how to Pylint your code to PEP8 compliance. All assignments in SDEV300 had a test table (like bullet 3 references). All assignments should be Pylint'ed - if you don't have proper docstrings your Pylint score will be terrible and you will lose 10 points.

**You must run Pylint or some other PEP8 code checking application on your submitted work.** Assignments require that you use the PEP Python Style guide found here:

<https://www.python.org/dev/peps/pep-0008/>

Some examples of Python Coding Style best practices include:

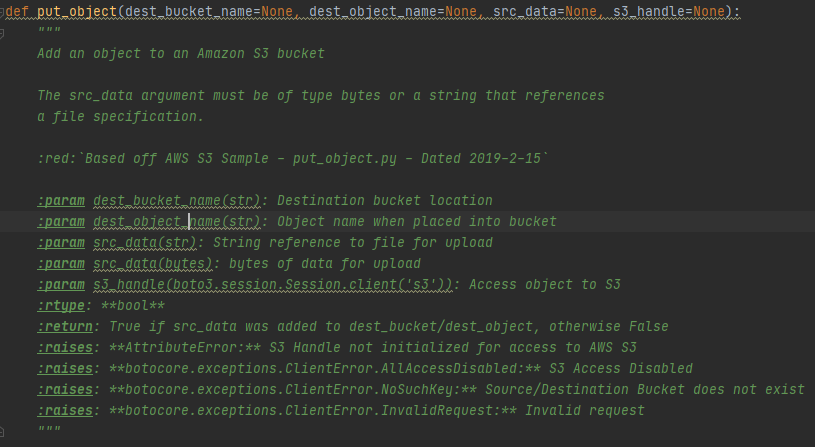
* Limit all lines to a maximum of 79 characters (**for this class we will deviate and use 100 character width - this is technically the pylint default**). I have attached a .pylintrc file that will enforce this.
  + You can do this on the command line (assuming we were scanning hello\_world.py, in the current directory):
    - pylint --max-line-length=100  hello\_world.py
    - Technically, pylint already passes the 100 in for you - as that is the default for pylint.
* Imports are always put at the top of the file, just after any module comments and before module globals and constants.
* Use 4 spaces for indentation.

Using a lint application like Pylint will automate these types of checks for you before you submit your code for evaluation. I will be grading code using Pylint. **You must receive an 80% or greater in Pylint on your code. You will lose -10pts for poorly coded Pylint scores.**

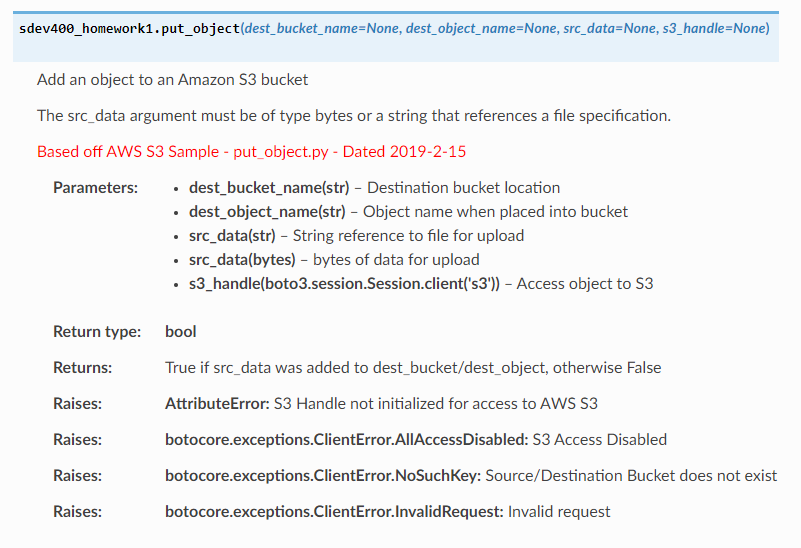
You can find some Pylint guidance in this announcement [Python Application Template - PyLint Scoring](https://learn.umgc.edu/d2l/le/news/766913/3013145/view).

**You must document your code.** The Official Python documentation standard is [reStructured text](https://en.wikipedia.org/wiki/ReStructuredText" \t "_blank) (reST)  . The examples provided by AWS for Week 2 **(**[**Learning Resources**](https://learn.umgc.edu/d2l/common/dialogs/quickLink/quickLink.d2l?ou=728214&type=content&rcode=UMUC-6966673)[**ZIP - Week2-S3-PythonCode**](https://leocontent.umgc.edu/content/dam/permalink/f7e83659-0138-4ab8-9fa9-fbf91e829d87.html)**)**use this syntax at the top of each function in the docstring. Using this syntax allows you to automatically generate PDF / HTML / and other formats of documentation using the [Sphinx](https://www.sphinx-doc.org/en/master/) tool.

Here is a code example from my actual Homework 1 Solution:



This results in HTML documentation that looks like the below when using a [theme](https://github.com/readthedocs/sphinx_rtd_theme) provided by [Read the Docs](https://readthedocs.org/):

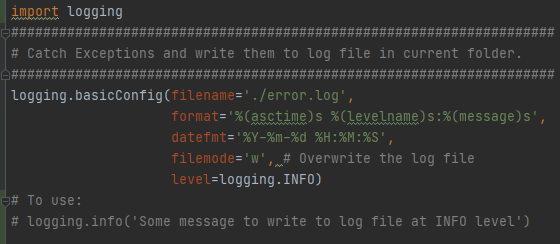


You are not required to use Sphinx to generate marked up code documentation. You are required, however, to build an appropriate docstring for your functions - I recommend using [reStructured text](https://en.wikipedia.org/wiki/ReStructuredText" \t "_blank) (reST) docstrings for your code. One line docstrings to prevent Pylint from flagging it are not acceptable as "documentation". You should also check out my example Python file template to help with the Python header requirements. The template is attached as the hello\_world.py sample ([Python Application Template - PyLint Scoring](https://learn.umgc.edu/d2l/le/news/766913/3013145/view)).

The assignment lists these as some assignment general requirements for options / menu items on your command-line driven Python Application. I have tried to call out various error checking that should be done when using these features. You would demonstrate these types of things in your test table that will be included with your formal report.

* Creates a S3 bucket with the name consisting of your first name, last name and a random 6-digit suffix.
  + Ask for first name (validates DNS safe), asks for last name (validates DNS safe), generates a random 6 digit number, append all 3 items together (<firstname><lastname>-<sixdigitnumber>) and creates new bucket with the appended name.
  + ERROR CHECKING - First and Last name can only contain valid **name** characters (i.e. A-Z, a-z, and -). The application does not allow a name to have an apostrophe (**‘**) as it is not DNS safe. A name should not contain numbers except for the randomly generated 6-digit suffix.
  + ERROR CHECKING - Should not create a bucket with the same name that already exists. Bucket names must be unique.

Puts objects in a previously created bucket.

* Lists available buckets, user selects a bucket, known file uploaded to selected bucket.
* You can implemented to just upload a known file - like the error.log that application generates while running. This could be enhanced to ask for a specific file to be uploaded - asking the end user for a local file path - then uploading that file.
  + Here is a some sample code to output to a error.log file using the standard Python logging library:
  + 
  + ERROR CHECKING - If you don't upload a known file, you will need to error check access to the file and make sure the user isn't uploading "system" files - like /etc/passwd....
* Deletes an object in a bucket.
  + List available buckets, user selects a bucket, list objects in bucket, the user selects an object to delete, object deleted.
  + ERROR CHECKING - You cannot select a bucket outside of the range of listed buckets for the object source.
  + ERROR CHECKING - You cannot select an object outside of the range of listed objects to delete.
* Deletes a bucket.
  + List available buckets, user selects a bucket for deletion, bucket is deleted.
  + Bucket must be empty to be deleted.
  + ERROR CHECKING - The application will print an error if the selected bucket is not empty.
  + ERROR CHECKING - You cannot select a bucket outside of the range of listed buckets for target bucket.
* Copies and object from one bucket to another.
  + List available buckets, user selects a Source bucket, list objects in bucket, the user selects an object to copy, list available buckets, user selects a Destination bucket, object is copied to bucket.
  + Source and Destination bucket selected must be different.
  + ERROR CHECKING - If the user attempts to select the same Destination as the Source an error message will be printed.
  + ERROR CHECKING - You cannot select a bucket outside of the range of listed buckets for Source or Destination.
  + ERROR CHECKING - You cannot select an object outside of the range of listed objects to copy.
* Downloads an existing object from a bucket.
  + List available buckets, user selects a Source bucket, list objects in bucket, the user selects an object to download, object is downloaded to local machine.
  + You can download the selected object and place it in the current directory. The file downloaded can be named **temp\_file**or whatever its name was in AWS.
* Exit the program. Upon exit, the application should list the current date and time at exit.

I have attached a sample run (sampleRun.txt) of the application to show you how the application might behave. I have also attached the error.log file just so you could see some errors/status it wrote out. You are NOT required to make an error.log file - but I have given you the code and it will make your life easier.

**Part 2 - Demonstration of copying files around AWS Console/Web GUI (10% of assignment overall)**

This is pretty clearly described in the PDF.

Just make sure once you have completed this you provide screen captures in your overall report that clearly demonstrate your successful completion. Be sure to provide a figure number and title as well as a description of the screen captures. Your screen captures should clearly show the bucket names created, the folders residing in each bucket and the files sent to each folder.

I do not need screen captures of the intermediate steps. I need screen captures of the final buckets/folders/files created. I need to see the files in the respective folders/buckets.

**Part 1 - Demonstration of bucket and "folder" creation using the AWS Console/Web GUI (20% of assignment overall)**

This is pretty clearly described in the PDF.

Just make sure once you have completed this you provide screen captures in your overall report that clearly demonstrate your successful completion. Be sure to provide a figure number and title as well as a description of the screen captures. Your screen captures should clearly show the bucket names created, the folders residing in each bucket and the files sent to each folder.

I do not need screen captures of the intermediate steps. I need screen captures of the final buckets/folders created.

Respectfully,