Excellent resource: <https://hands-on.cloud/boto3-s3-tutorial/#Listing-S3-Buckets-using-Boto3-resource>

DO NOT UPDATE PIP

sudo python3 -m pip install boto3

sudo python3 -m pip install numpy

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

79 character limits above

the syntax and structure for the response variable returned in this case are:

Response Syntax

{

'Buckets': [

{

'Name': 'string',

'CreationDate': datetime(2015, 1, 1)

},

],

'Owner': {

'DisplayName': 'string',

'ID': 'string'

}

}

Response Structure

(dict) --

Buckets (list) --

(dict) --

Name (string) --

The name of the bucket.

CreationDate (datetime) --

Date the bucket was created.

Owner (dict) --

DisplayName (string) --

ID (string) –

buckets = [bucket['Name'] for bucket in response['Buckets']] – This statement retrieves a list of buckets based on the ‘Name’ attribute. There is actually quite a bit happening in this one statement. The response[‘Buckets’] code is referring to the list of Buckets in the response. You could have also used response[‘Owner’]. The for statement just cycles through each of the Buckets with bucket[‘Name’] looks specifically for the Name of the bucket. If you wanted to retrieve the CreationDate, use bucket[‘CreationDate’]. Note the [ ] enclosing the syntax. This is required as

the results will be a list of bucket names. **Experimentation and a detailed review of Python data structure use is the best approach for truly understanding how this powerful syntax works**.

e. print("Bucket List: %s" % buckets) – Finally, this statement takes the list of buckets generate in the above statement and prints them to the screen.

Boto3

[Boto3 1.26.140 documentation (amazonaws.com)](https://boto3.amazonaws.com/v1/documentation/api/latest/index.html)

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

jimroberts-132882

David,

1) With regard to Pylint'ing the code - the only assignments that I impose the Pylint requirement on are the Lab 1, 2, and 4. Lab 3 is is a LAMBDA assignment and the code snippets you built are pretty short for it - so don't worry about linting it. Yes - 10 points off for a score lower than 8/10.

2) For your doc-string question - I don't take points off for your doc strings. I strongly encourage students to build and write code that follows a standard one would be expected to comply with in the real world. In my example announcement, you can see what that might look like:

However - no - I don't grade against those. If you are missing a doc-string you would take a hit on the Pylint scan.

When you actually start working in the real world - developer standards have things like "pre-commit" scanning. It will lint and do a number of things to your code before you are even allowed to commit to the repo.

Here is a "quick example"

|  |
| --- |
| --- |
|  | repos: |
|  | - repo: <https://github.com/PyCQA/isort> |
|  | rev: 5.10.1 |
|  | hooks: |
|  | - id: isort |
|  | args: |
|  | - "--profile" |
|  | - "black" |
|  | - repo: <https://github.com/psf/black> |
|  | rev: 22.1.0 |
|  | hooks: |
|  | - id: black |
|  | - repo: <https://github.com/adrienverge/yamllint> |
|  | rev: v1.26.3 |
|  | hooks: |
|  | - id: yamllint |
|  | - repo: <https://github.com/pre-commit/pre-commit-hooks> |
|  | rev: v4.1.0 |
|  | hooks: |
|  | - id: check-ast |
|  | - id: check-json |
|  | - id: check-toml |
|  | - id: check-xml |
|  | - id: check-yaml |
|  | - id: end-of-file-fixer |
|  | - id: requirements-txt-fixer |
|  | - id: trailing-whitespace |

Failing any of those "step" would result in you not being able to commit the code. So - I encourage students to develop knowing when they are writing code professionally - they will have to comply.

3) To answer some comments from below - I do not build the course documentation. The University standardizes and provides this across all classes/sections. At times there are out of date items - like the screen grabs for Lab 0. This is why I provided an example in the Ask the Professor section. You will find as the class moves along - I post very detailed guidance for the various assignments and try my best to "clear up" any data points that might be at issue.

4) I would encourage students to dial into the Weekly Monday ZOOM 8-9pm ET. I would encourage students to work on the assignments "early" and email them to me ahead of submission for "office hours" review. I will tell you "where" you would loose points or how to fix something if you are stuck. If you work "proactively" in this class and take advantage of my "virtual office hours" reviews and the Monday ZOOMs - you should do pretty well in the class.

Respectfully,

Craig Poma  
Adjunct Associate Professor