Introduction to boto3

- Boto3 is the name of the Python SDK for AWS.
- Boto3 allows us to directly create, update, and delete AWS services from our Python scripts.
- Boto3 is built on the top of botocore module.
- We have to Install boto3 to work with AWS Services using Python Scripts.
- How to install boto3 ?
    - Python-2.x:
        - pip install boto3
    - Python-3.x
        - pip3 install boto3

Learn how to automate AWS common tasks using boto3 and Lambda

Installing Python-3.x and boto3 on Windows Server

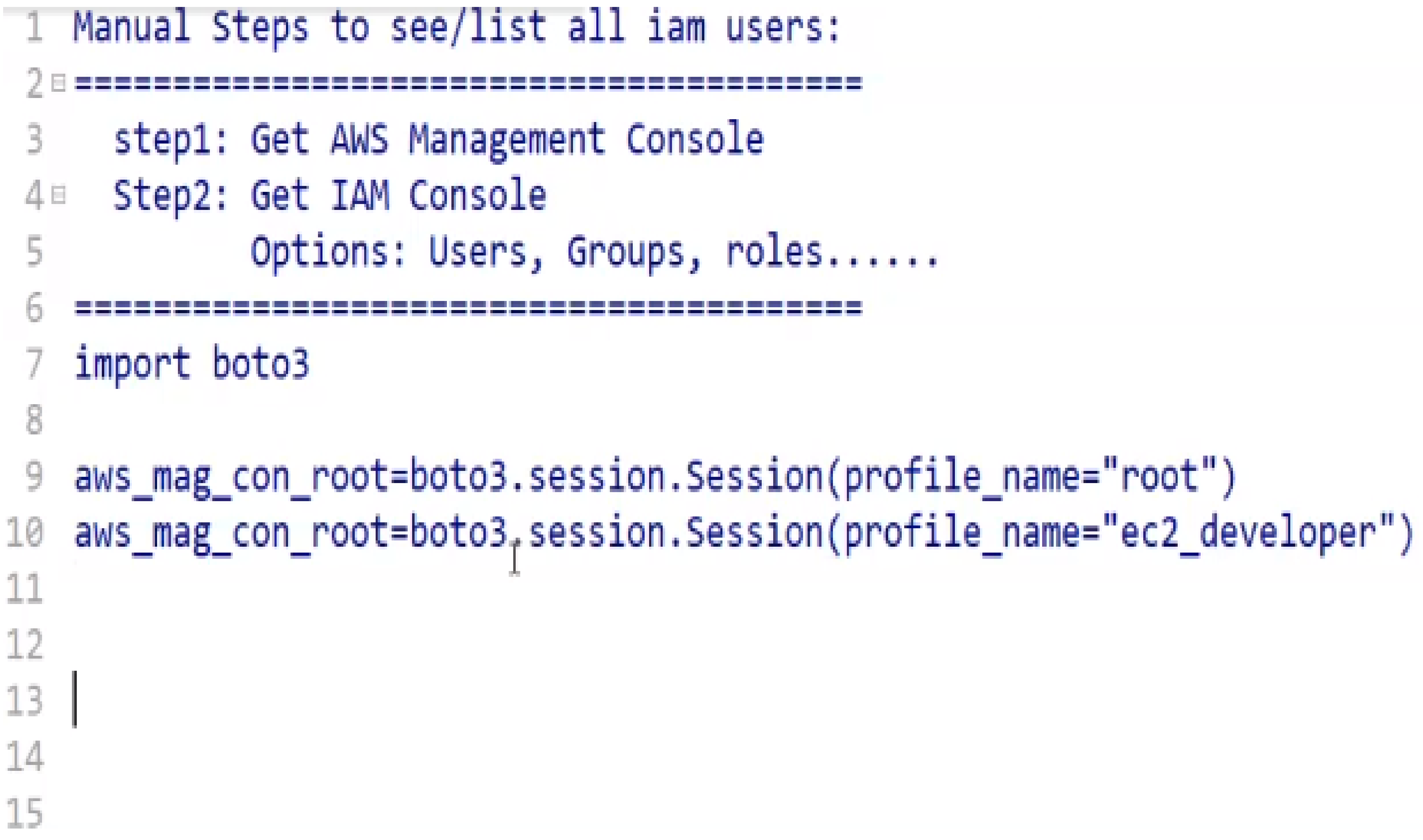
- Python-3.7.4
- Go to www.python.org
- Set Paths for python and pip3
- Install boto3
    - pip3 install boto3

Boto3 Environment setup on Windows Server…

- Configure credentials of your AWS account on windows server using awscli commands.
  - Install awscli
    - pip3 install awscli
  - Configure root/IAM user access-keys/credentials using:
    - aws configure --profile root
    - aws configure --profile non_prod

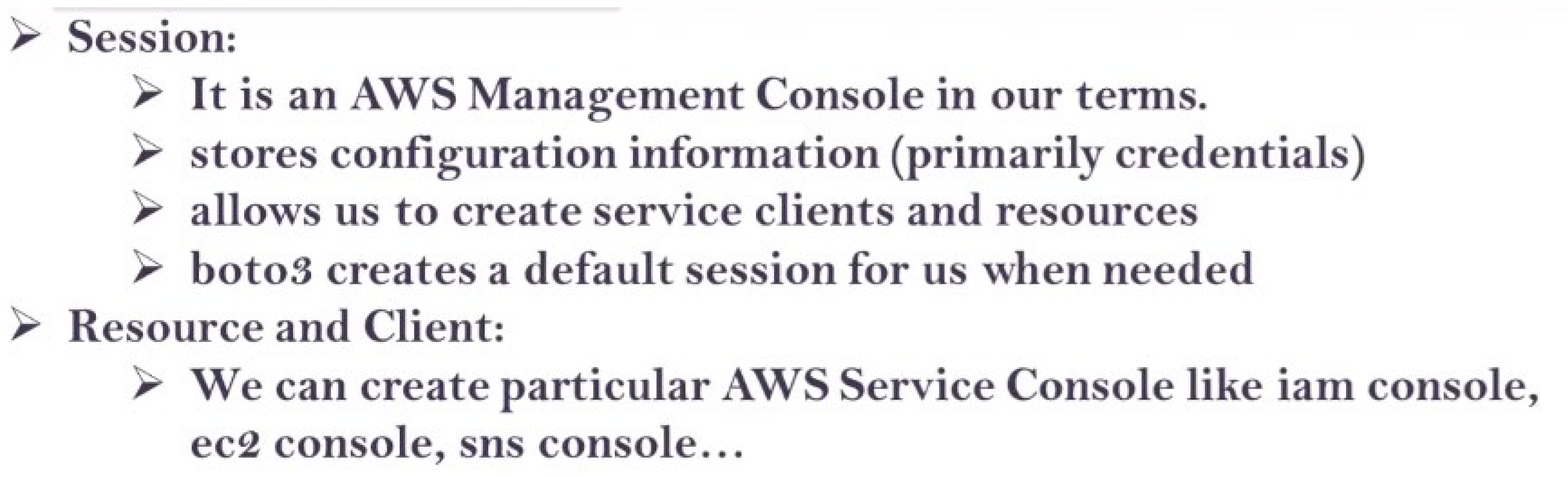
Boto3 Concepts:

- The core concepts of boto3 are:
  - Session
  - Resource
  - Client
  - Meta
  - Collections
  - Waiters
  - Paginators

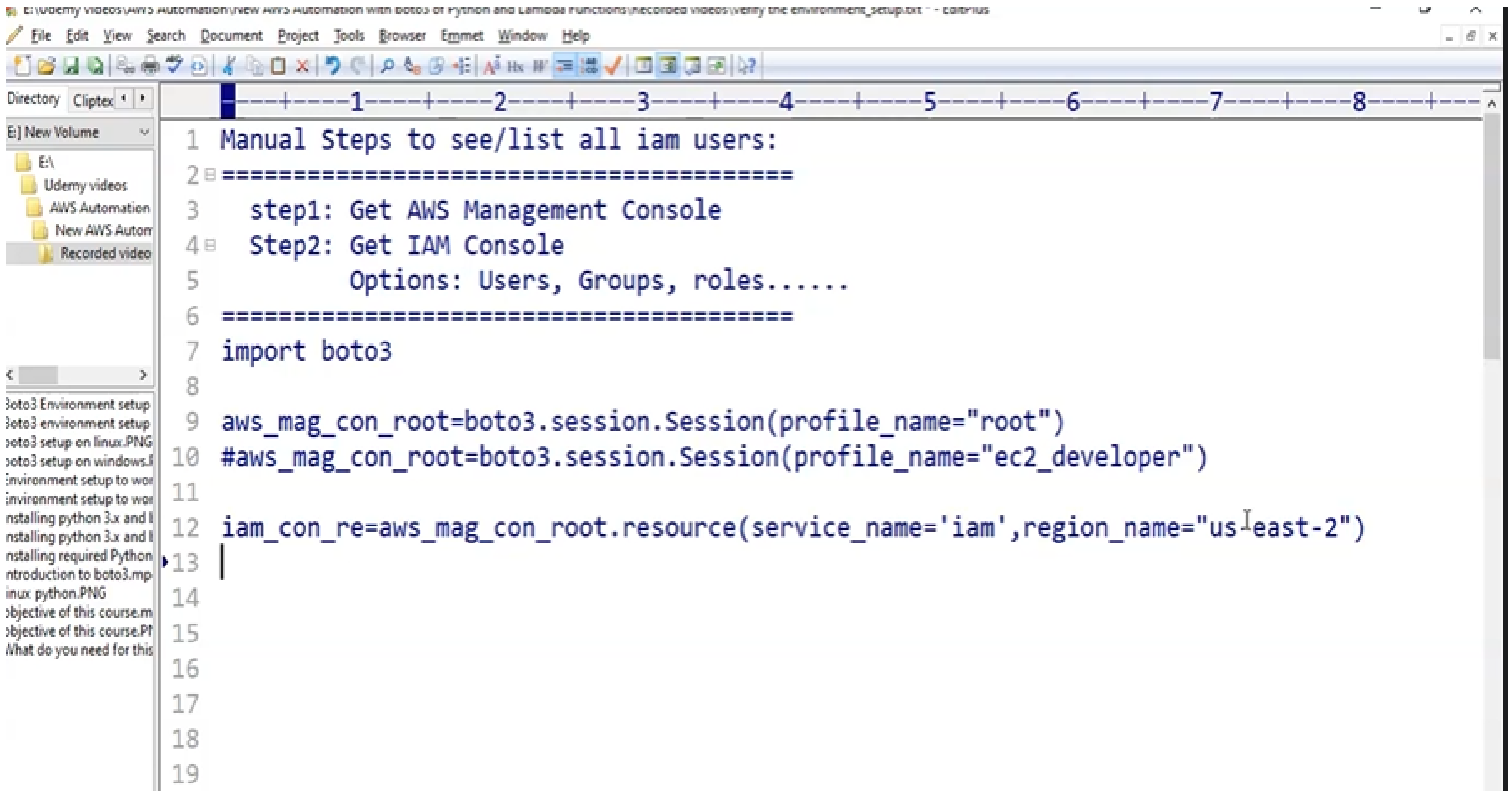


```
Manual Steps to see/list all iam users:
========================================
  step1: Get AWS Management Console
  Step2: Get IAM Console
         Options: Users, Groups, roles......
========================================
import boto3

aws_mag_con_root=boto3.session.Session(profile_name="root")
aws_mag_con_root=boto3.session.Session(profile_name="ec2_developer")
```

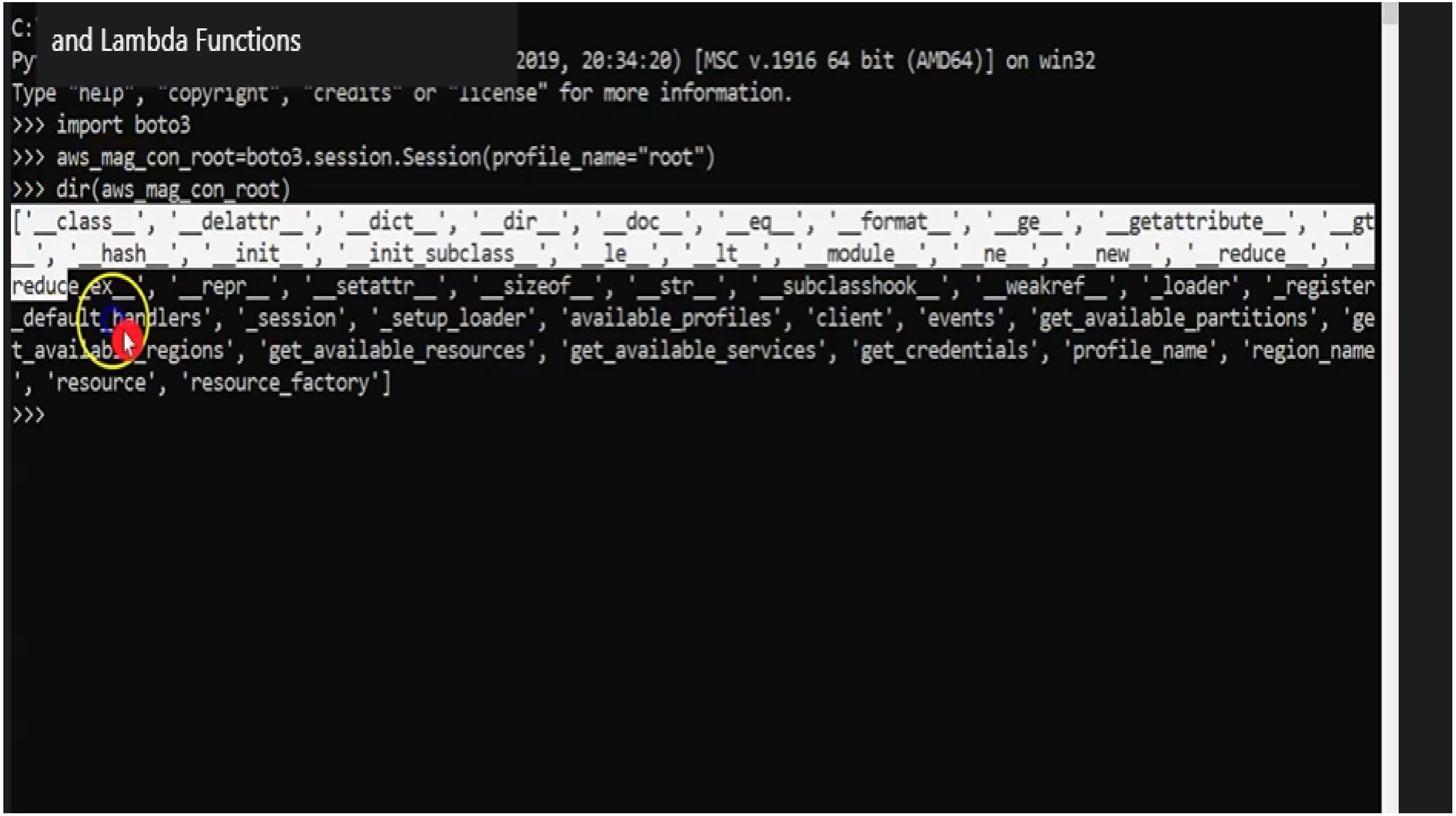
Boto3 : session,resource and client



- **Session:**
  - **It is an AWS Management Console in our terms.**
  - **stores configuration information (primarily credentials)**
  - **allows us to create service clients and resources**
  - **boto3 creates a default session for us when needed**
- **Resource and Client:**
  - **We can create particular AWS Service Console like iam console, ec2 console, sns console…**

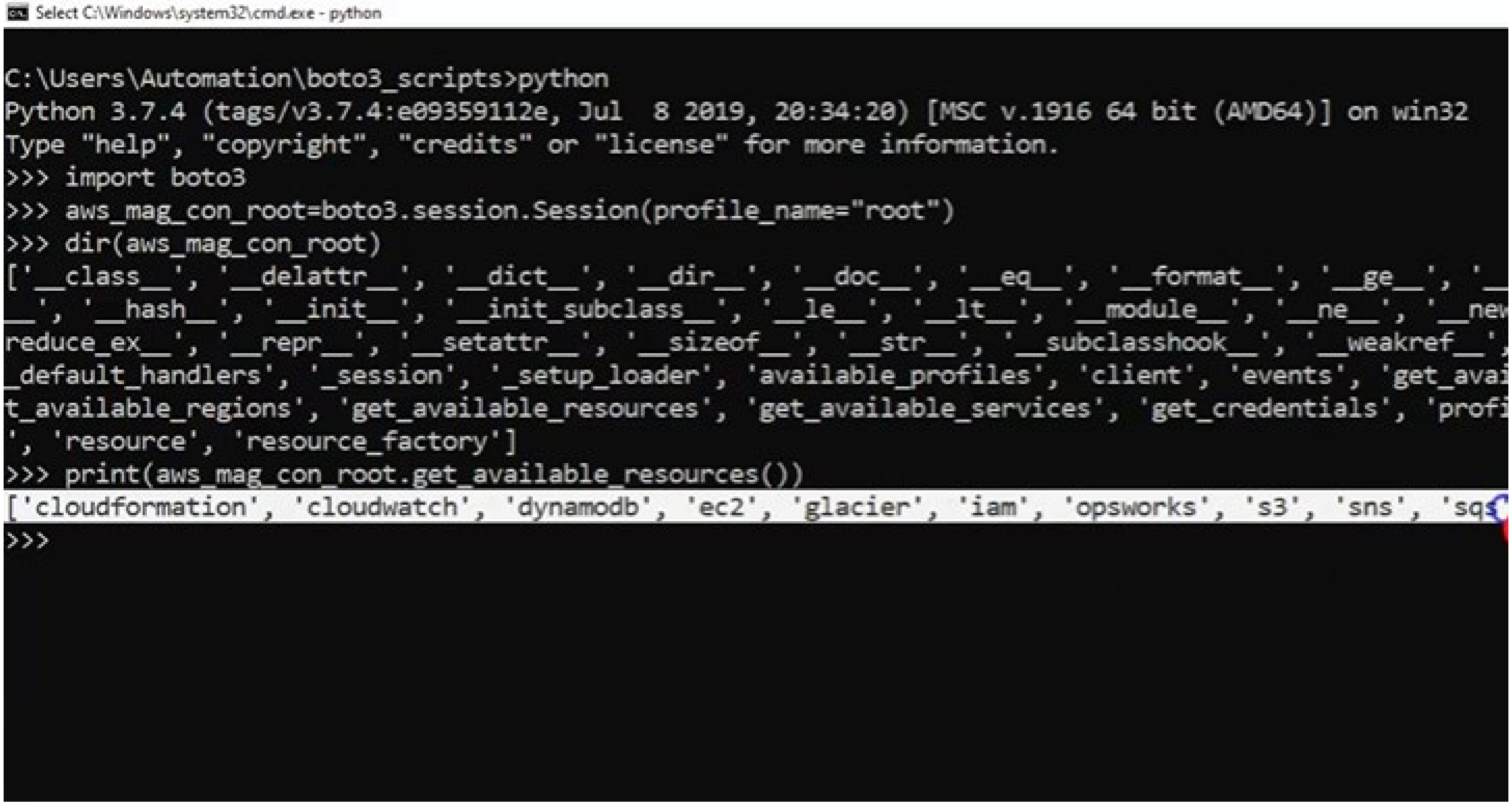


```
Manual Steps to see/list all iam users:
=======================================
  step1: Get AWS Management Console
  Step2: Get IAM Console
         Options: Users, Groups, roles......
=======================================
import boto3

aws_mag_con_root=boto3.session.Session(profile_name="root")
#aws_mag_con_root=boto3.session.Session(profile_name="ec2_developer")

iam_con_re=aws_mag_con_root.resource(service_name='iam',region_name="us-east-2")
```



and Lambda Functions

```
C:
Py                                    2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import boto3
>>> aws_mag_con_root=boto3.session.Session(profile_name="root")
>>> dir(aws_mag_con_root)
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__gt
__', '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__', '__module__', '__ne__', '__new__', '__reduce__', '__
reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__', '_loader', '_register
_default_handlers', '_session', '_setup_loader', 'available_profiles', 'client', 'events', 'get_available_partitions', 'ge
t_available_regions', 'get_available_resources', 'get_available_services', 'get_credentials', 'profile_name', 'region_name
', 'resource', 'resource_factory']
>>>
```



Select C:\Windows\system32\cmd.exe - python

```
C:\Users\Automation\boto3_scripts>python
Python 3.7.4 (tags/v3.7.4:e09359112e, Jul  8 2019, 20:34:20) [MSC v.1916 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> import boto3
>>> aws_mag_con_root=boto3.session.Session(profile_name="root")
>>> dir(aws_mag_con_root)
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '_
_', '__hash__', '__init__', '__init_subclass__', '__le__', '__lt__', '__module__', '__ne__', '__new
reduce_ex__', '__repr__', '__setattr__', '__sizeof__', '__str__', '__subclasshook__', '__weakref__',
_default_handlers', '_session', '_setup_loader', 'available_profiles', 'client', 'events', 'get_avai
t_available_regions', 'get_available_resources', 'get_available_services', 'get_credentials', 'profi
', 'resource', 'resource_factory']
>>> print(aws_mag_con_root.get_available_resources())
['cloudformation', 'cloudwatch', 'dynamodb', 'ec2', 'glacier', 'iam', 'opsworks', 's3', 'sns', 'sqs
>>>
```

- Session:
    - It is an AWS Management Console in our terms.
    - stores configuration information (primarily credentials)
    - allows us to create service clients and resources
    - boto3 creates a default session for us when needed
- Resource and Client:
    - We can create particular AWS Service Console like iam console, ec2 console, sns console…
    - Resource is *higher-level object-oriented service access and it is available for some of the aws services.*
    - Client is *low-level service access*

Boto3 Session Concept:

- The are two types of Sessions
  - They are:
    - Custom Session
    - Default Session

Introduction to AWS Lambda or AWS Lambda Function:

- *AWS Lambda is a server-less computing platform that allows engineers to create a small function, configure the function in the AWS console, and have the code executed without the need to provision servers—paying only for the resources used during the execution.*
- *Simply it is like an editor(vim, Pycharm, sublime text, atom) with some extra features.*
- *It supports to run different languages like python, Go, java, Node.js etc…*
- *It is installed or running on Amazon Linux Server and we can access /tmp using Lambda Function.*

Requirements for AWS Lambda Function:

- *A Lambda function has a few requirements.*
- *The first requirement you need to satisfy is to provide a handler.*
    - *The handler is the entry point for the Lambda.*
    - *A Lambda function accepts JSON-formatted input and will usually return the same.*
- *The second requirement is that you'll need to specify the runtime environment for the Lambda. The runtime will usually correlate directly with the language you selected to write your function.*
- *The final requirement is a trigger.*
    - *Manual trigger or Run by us.*
    - *You can configure a Lambda invocation in response to an event, such as a new file uploaded to S3, a change in a DynamoDB table, or a similar AWS event. You can also configure the Lambda to respond to requests to AWS API Gateway, or based on a timer triggered by AWS Cloudwatch.*

How AWS Lambda Function executes the code for AWS services:

Two ways:

Use programmatic access keys
Create a AWS IAM Role and attach the role to AWS Lambda.

Every Day

Start EC2 Instances at 8 am Mon-Fri

Stop EC2 Instances at 5pm Mon-Fri

===

Step1: Create a Role for Lambda Function

Every Day

Start EC2 Instances at 8 am Mon-Fri

Stop EC2 Instances at 5pm Mon-Fri

===============================

Step1: Create a Role for Lambda Function

Step2: Write a Lambda Function using boto3 of python

Step3: Schedule the job

Paginators

- Paginators plays a role when we use boto3 to query AWS resource.
- Like get all ec2 instances , iam users, buckets, objects etc.
- For query,API calls are made to AWS through boto3
- Generally each API call will return 50 or 100 results.
- Note: s3 will return up to 1000 results

paginators

- Boto3 provides Paginators to automatically issue multiple API requests to retrieve all the pages

- Paginators are straightforward to use

- But not all boto3 services provide paginator support. For those services you will need to write your own paginator in python

How to use paginators ?

- Step1 : Create a paginator
- Step2 : Paginate through created paginator go get pages one by one

EBS Volumes

- An Amazon EBS volume is **a durable, block-level storage device that you can attach to your instances**. After you attach a volume to an instance, you can use it as you would use a physical hard drive. EBS volumes are flexible. ... EBS volumes persist independently from the running life of an EC2 instance.

- AWS Elastic Block Store (EBS) is Amazon's block-**level storage solution used with the EC2 cloud service to store persistent data**. This means that the data is kept on the AWS EBS servers even when the EC2 instances are shut down

EBS Volumes using lambda

- Write a code to list all EBS Volumes based on requirement