🚀 AWS Automation with Python Boto3 and Lambda Functions

[PART - 1] Introduction

🎯 **Aim:** Learn how to automate AWS common tasks using Boto3 and Lambda Functions.

🎯 Objective of this course:

- Cover the core concepts of Boto3 and Lambda.
- Understand Boto3 and Lambda concepts with real-time scenarios.
- Running Boto3 scripts on your local machine and triggering Lambda functions.
- By the end of this course, you will gain the knowledge to apply different concepts of Boto3 and Lambda for different AWS Services.

Pre-requisites: What do you need for this course?

- AWS Account: It is great if you have a free tier account.
- Good if you have basic Knowledge on AWS Services and Python (Not mandatory).
- Knowledge on Any Python IDE (Not mandatory).

💡 Introduction to Boto3

- Boto3 is the name of the Python SDK/Library/Module/API for AWS.
  🔧 Boto3 allows us to directly create, update, and delete AWS services from our Python scripts.
  🔝 Boto3 is built on the top of the boto core module

🔧 Installation:

- **Python-2.x**: pip install boto3
- **Python-3.x**: pip3 install boto3

💻 Install Python and Boto3 on Windows Machine:

1. **Python-3.7.4**: Visit www.python.org.
2. Set Paths for Python and pip3.
3. Install boto3 using pip3 install boto3.

🐧 Install Python and Boto3 on Linux Machine:

1. **Dependencies**
   - yum install gcc openssl-devel bzip2-devel libffi-devel
2. **Download Python**
   - cd /usr/src
   - wget https://www.python.org/ftp/python/3.7.4/Python-3.7.4.tgz
   - tar xzf Python-3.7.4.tgz
   - cd Python-3.7.4

3. **Configure and Install Python**
    - ./configure --enable-optimizations
    - make altinstall
4. **Set Up Python Binaries**
    - cd /usr/local/bin/
    - ./python3.7 --version
    - ./pip3.7 --version
    - ln -s /usr/local/bin/python3.7 /bin/python3
    - python3 --version
    - ln -s /usr/local/bin/pip3.7 /bin/pip3
    - pip3 --version
5. **Install Boto3**
    - pip3 install boto3

🔧 Boto3 Environment Setup

Setting up your environment to use Boto3 for AWS automation is a crucial first step. Here's a detailed guide to get you started:

1. Configure AWS Credentials:

- 🛠 **AWS CLI**: The AWS Command Line Interface (AWS CLI) is your go-to tool for managing AWS services from the command line.
- 📥 **Downloading AWS CLI**: Guide
- 💻 **Configuration**:
    - Login to AWS Management Console and create a new user with programmatic access, granting AdministratorAccess.
    - Configure access keys/credentials:
        - aws configure (Creates DEFAULT profile)

2. First Automation Script: List IAM Users

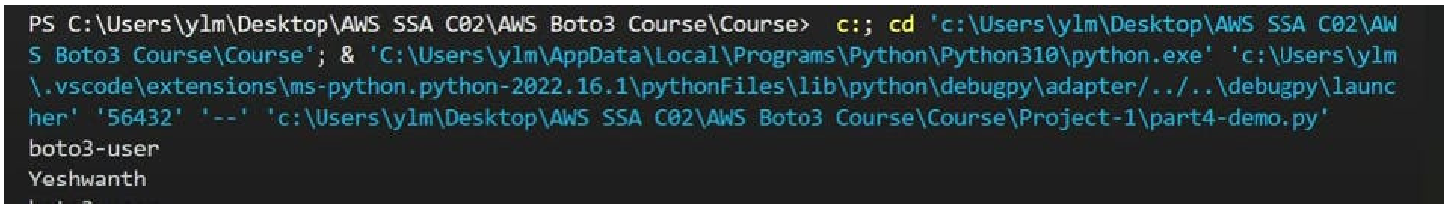
- 📝 **Manual Steps**:
    - **Step 1**: Access AWS Management Console AWS Management Console
    - **Step 2**: Navigate to IAM Console
        - In IAM Console, explore options like:
            - Users
            - Groups
            - Roles
            - Policies, etc.

```
import boto3
```

```
# Create a session object named 'aws_management_console' using the default profile

aws_management_console =boto3.session.Session(profile_name="default")



# Create an IAM resource object named 'iam_console_resource' using the session

iam_console_resource = aws_management_console.resource('iam')



# Iterate through all IAM users and print their names

for each_user in iam_console_resource.users.all():

    print(each_user.name)
```

🔍 **Explanation**:

- boto3.session.Session(profile_name="default"): Creates a session object named aws_management_console using the default AWS profile. This session object will store configuration information like credentials.
- aws_management_console.resource('iam'): Creates an IAM resource object named iam_console_resource using the session. This resource object allows you to interact with IAM resources.
- iam_console_resource.users.all(): Fetches all IAM users using the all() method provided by the resource object.
- for each_user in iam_console_resource.users.all():: Iterates through each IAM user fetched.
- print(each_user.name): Prints the name of each IAM user.

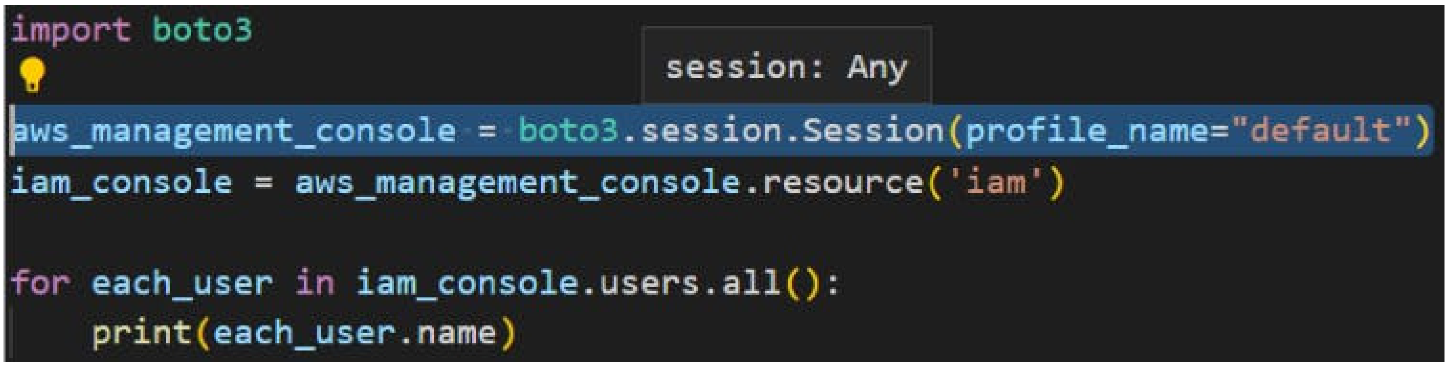


```
PS C:\Users\ylm\Desktop\AWS SSA C02\AWS Boto3 Course\Course>  c:; cd 'c:\Users\ylm\Desktop\AWS SSA C02\AW
S Boto3 Course\Course'; & 'C:\Users\ylm\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\ylm
\.vscode\extensions\ms-python.python-2022.16.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launc
her' '56432' '--' 'c:\Users\ylm\Desktop\AWS SSA C02\AWS Boto3 Course\Course\Project-1\part4-demo.py'
boto3-user
Yeshwanth
```

💡 **Concepts of Boto3**

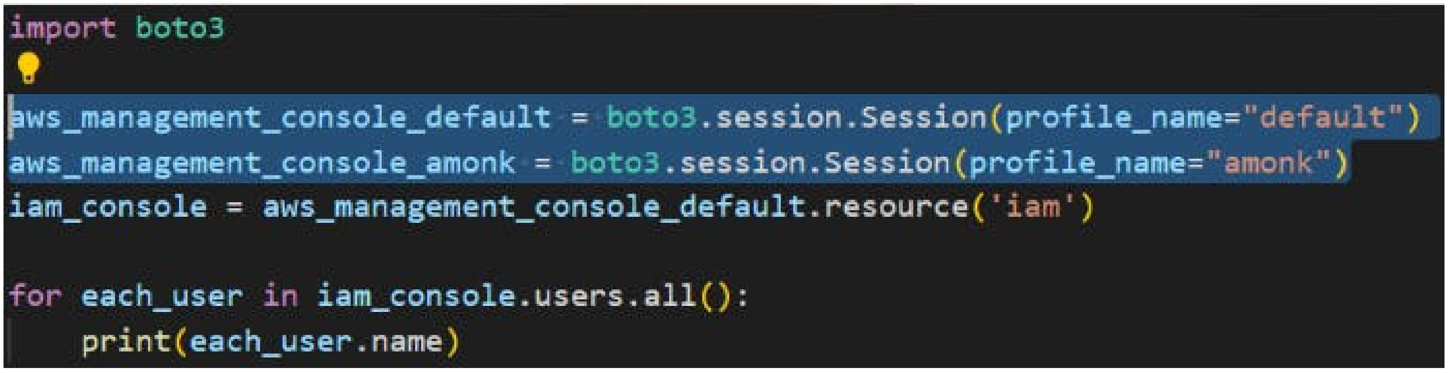
- Session
- Resource
- Client
- Meta
- Collections
- Waiters
- Paginators

Session



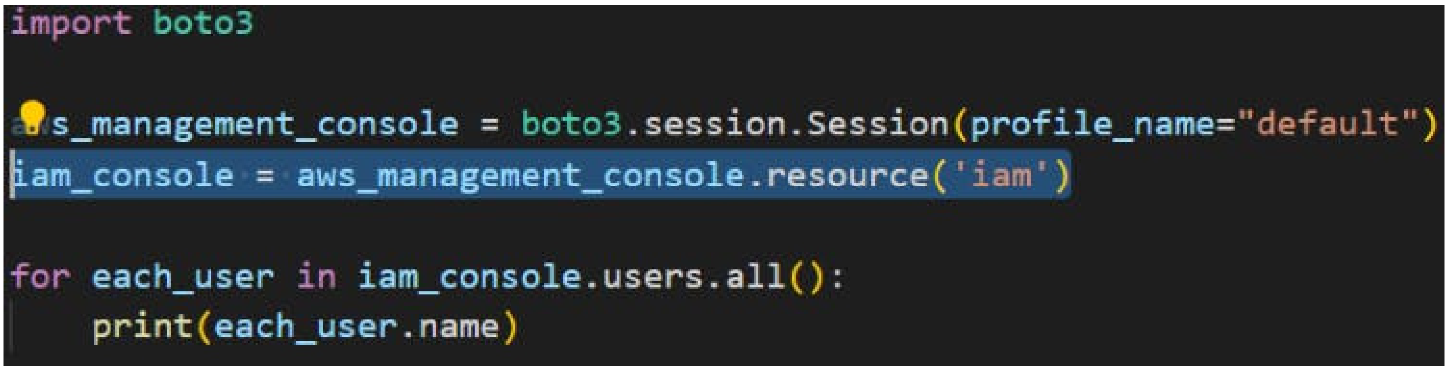
```
import boto3

aws_management_console = boto3.session.Session(profile_name="default")
iam_console = aws_management_console.resource('iam')

for each_user in iam_console.users.all():
    print(each_user.name)
```

- In simple words, it's like the AWS Management Console.
- Stores configuration information (Credentials of Default user etc...).
- Allows us to create Service, Clients, and Resources.
- It creates a default session for us when we need it.
- We can create multiple sessions in the same script!



```
import boto3

aws_management_console_default = boto3.session.Session(profile_name="default")
aws_management_console_amonk = boto3.session.Session(profile_name="amonk")
iam_console = aws_management_console_default.resource('iam')

for each_user in iam_console.users.all():
    print(each_user.name)
```

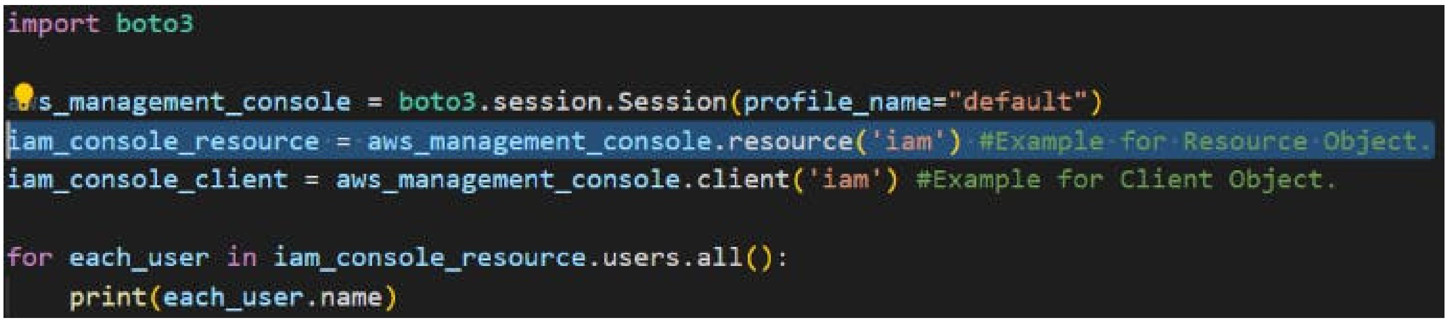
Resource and Client



```python
import boto3

aws_management_console = boto3.session.Session(profile_name="default")
iam_console = aws_management_console.resource('iam')

for each_user in iam_console.users.all():
    print(each_user.name)
```

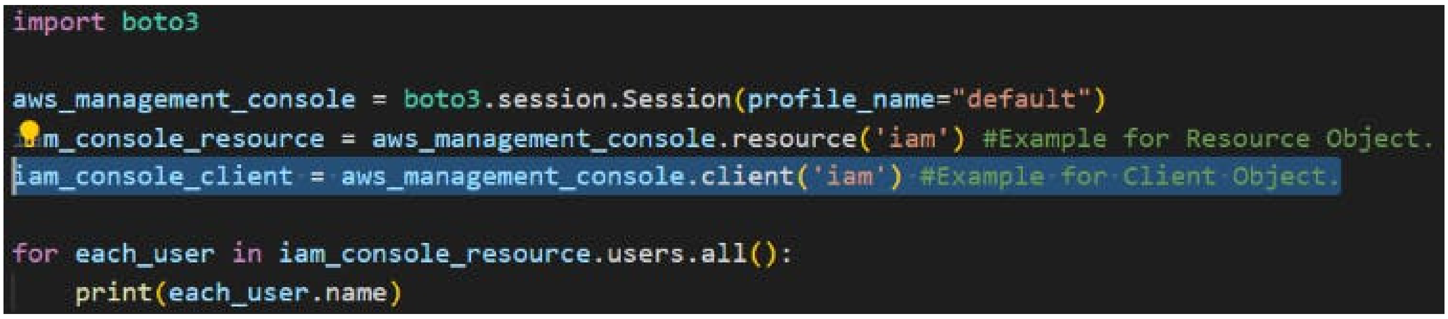
- We can create particular AWS Service consoles examples: IAM Console, EC2 Console, etc...
- You can create an AWS Service console from your Session object.
- Region name can be specified after the Profile name.

Example for Resource Object:



```python
import boto3

aws_management_console = boto3.session.Session(profile_name="default")
iam_console_resource = aws_management_console.resource('iam') #Example for Resource Object.
iam_console_client = aws_management_console.client('iam') #Example for Client Object.

for each_user in iam_console_resource.users.all():
    print(each_user.name)
```

Example for Client Object:



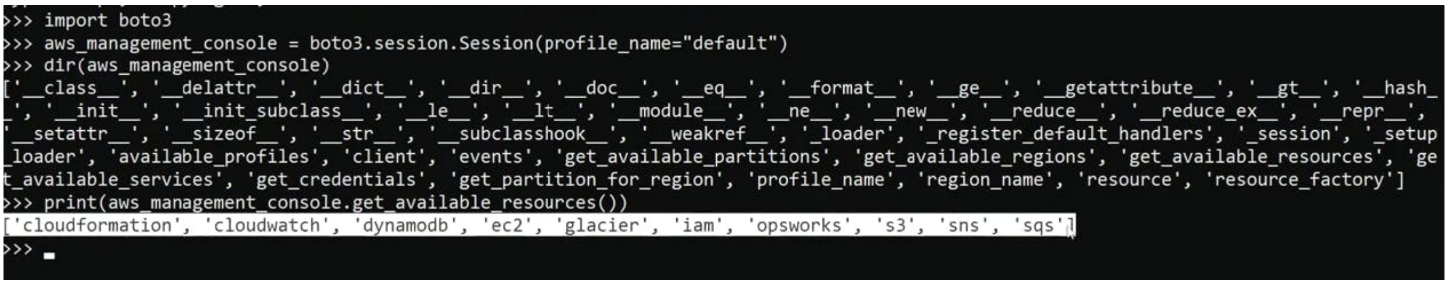
```python
import boto3

aws_management_console = boto3.session.Session(profile_name="default")
iam_console_resource = aws_management_console.resource('iam') #Example for Resource Object.
iam_console_client = aws_management_console.client('iam') #Example for Client Object.

for each_user in iam_console_resource.users.all():
    print(each_user.name)
```

🤔 Should I choose Resource or Client?

You can choose anyone depending on your use case.

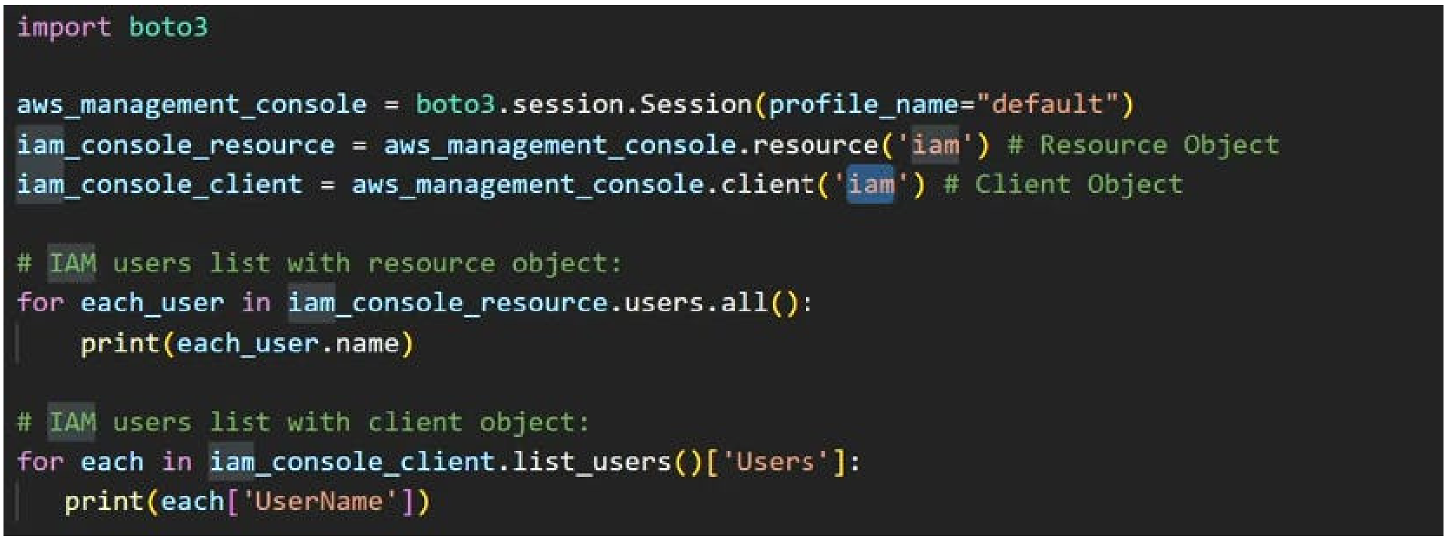
- Resource is **Higher Level** Object oriented service access.
- Resource objects are only available for a few AWS Services.
- Let us check which AWS Service has a Resource Object!!! - DEM😉

- ['cloudformation', 'cloudwatch', 'dynamodb', 'ec2', 'glacier', 'iam', 'opsworks', 's3', 'sns', 'sqs'] - Resource Object Available.



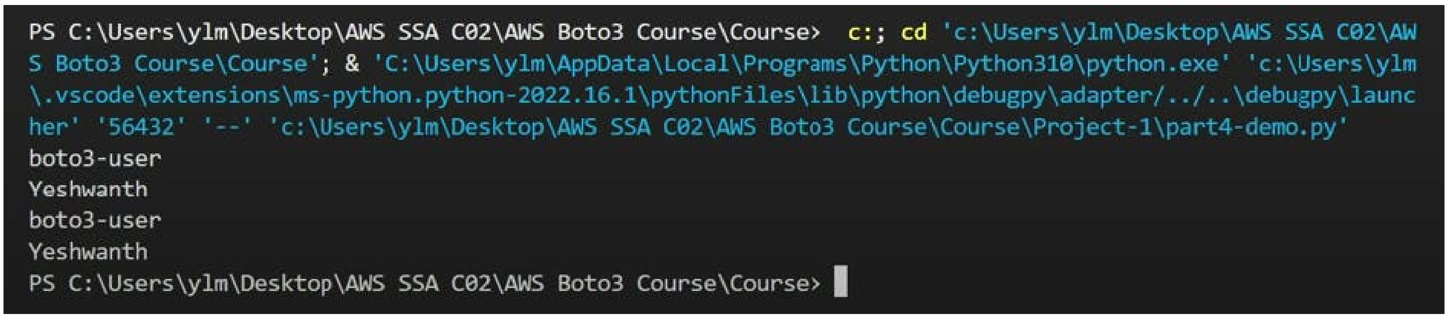
```
>>> import boto3
>>> aws_management_console = boto3.session.Session(profile_name="default")
>>> dir(aws_management_console)
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__gt__', '__hash_
_', '__init__', '__init_subclass__', '__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', 'get_available_regions', 'get_available_resources', 'ge
t_available_services', 'get_credentials', 'get_partition_for_region', 'profile_name', 'region_name', 'resource', 'resource_factory']
>>> print(aws_management_console.get_available_resources())
['cloudformation', 'cloudwatch', 'dynamodb', 'ec2', 'glacier', 'iam', 'opsworks', 's3', 'sns', 'sqs']
>>> 
```

- Client is **Low-Level** Service Access.
- In simple terms, the output/response in case of Client will be in Dictionary, which needs more effort in implementing boto3 scripts.
- Whereas Resource is an object, **we can use simple (.) operation.**
- All operations that we see in AWS Management Console can be done in Client whereas Resource does not guarantee you that. Some operations may not be supported.
- If we do not have some operations in Resource we can enter into Client by using the **"Meta"** concept. Let us talk about this later! 😉
- Let us see how much effort is needed for both Resource and Client. - DEM😀



```python
import boto3

aws_management_console = boto3.session.Session(profile_name="default")
iam_console_resource = aws_management_console.resource('iam') # Resource Object
iam_console_client = aws_management_console.client('iam') # Client Object

# IAM users list with resource object:
for each_user in iam_console_resource.users.all():
    print(each_user.name)

# IAM users list with client object:
for each in iam_console_client.list_users()['Users']:
   print(each['UserName'])
```

output:



```
PS C:\Users\ylm\Desktop\AWS SSA C02\AWS Boto3 Course\Course>  c:; cd 'c:\Users\ylm\Desktop\AWS SSA C02\AW
S Boto3 Course\Course'; & 'C:\Users\ylm\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\ylm
\.vscode\extensions\ms-python.python-2022.16.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launc
her' '56432' '--' 'c:\Users\ylm\Desktop\AWS SSA C02\AWS Boto3 Course\Course\Project-1\part4-demo.py'
boto3-user
Yeshwanth
boto3-user
Yeshwanth
PS C:\Users\ylm\Desktop\AWS SSA C02\AWS Boto3 Course\Course> 
```

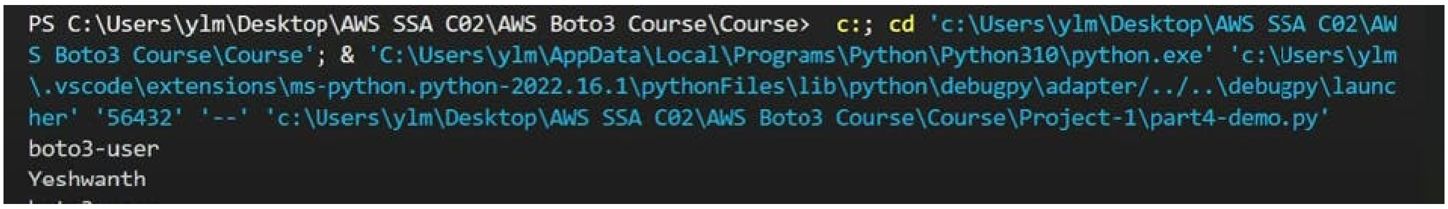
Example 1: List all the IAM users in AWS Account using client objects.

```
import boto3


# Create a session object named 'aws_management_console' using the default profile

aws_management_console =boto3.session.Session(profile_name="default")


# Create an IAM client object named 'iam_console_client' using the session

iam_console_client = aws_management_console.client('iam')


# Retrieve a list of all IAM users

response = iam_console_client.list_users()


# Iterate through all IAM users and print their names

for user in response['Users']:

    print(user['UserName'])
```

🔍 **Explanation**:

1. **Session Creation**:
   - boto3.session.Session(profile_name="default"): Creates a session object named aws_management_console using the default AWS profile. This session object will store configuration information like credentials.
2. **IAM Client Creation**:
   - aws_management_console.client('iam'): Creates an IAM client object named iam_console_client using the session. The client object allows you to interact with the IAM service.

3. **Listing IAM Users**:
   - iam_console_client.list_users(): Calls the list_users() method of the IAM client object to retrieve a list of all IAM users in the AWS account.
   - The response from list_users() is stored in the variable response.
4. **Iterating and Printing**:
   - for user in response['Users']:: Iterates through each IAM user in the list of users returned in the response.
   - print(user['UserName']): Prints the name of each IAM user. The username is accessed using the key 'UserName' in the user dictionary.



```
PS C:\Users\ylm\Desktop\AWS SSA C02\AWS Boto3 Course\Course>  c:; cd 'c:\Users\ylm\Desktop\AWS SSA C02\AW
S Boto3 Course\Course'; & 'C:\Users\ylm\AppData\Local\Programs\Python\Python310\python.exe' 'c:\Users\ylm
\.vscode\extensions\ms-python.python-2022.16.1\pythonFiles\lib\python\debugpy\adapter/../..\debugpy\launc
her' '56432' '--' 'c:\Users\ylm\Desktop\AWS SSA C02\AWS Boto3 Course\Course\Project-1\part4-demo.py'
boto3-user
Yeshwanth
```

Example 2: List all the running EC2 Instances in your AWS Account using client objects.

Example 3: List all the IAM users in AWS Account using resource objects.

['cloudformation', 'cloudwatch', 'dynamodb', 'ec2', 'glacier', 'iam', 'opsworks', 's3', 'sns', 'sqs'] - Resource Object Available.

Example 4: List all the running EC2 Instances in your AWS Account using resource objects.