spot by NetApp

Introduction to Python

October 2022

Ido Haskel & Maor Elfassy

python™

Agenda

- Lesson 1 - Recap
- Python - Files
- Cloud & AWS Intro
- Python interface with AWS (Boto3)
- Exercise

spot by NetApp

Recap - Question 1

What’s the difference between price and score variables?

```
score = 10
price = 9.9
```

Variables

We use variables to temporarily store data in the computer's memory

```
score = 10

price = 9.9
```

- `score` is an integer (a whole number without a decimal point)
- `price` is a float (a number with a decimal point)

Recap - Question 2

What does the following method do?

```
name = input()
```

Receiving Input

You can receive an input from the user using the input() function:

```
name = input('what is your name?')
print ('Hello ' + name);
```

Recap - Question 3

What does the x variable hold after executing each line?

```
name = 'Maor Elfassy'
x = name[1]
x = name[-1]
x = name[:-1]
x = name[:1]
x = name[3:7]
x = name[:]
```

spot by NetApp

Strings

We can get individual characters in a string using square brackets []

- If we leave out the start index, 0 will be assumed
- If we leave out the end index, the length of the string will be assumed

```
name = 'Maor Elfassy'

x = name[1] # returns the second character: a

x = name[-1]# returns the first character from the end: y

x = name[:-1] # returns the characters at position 0 to latest (last
character excluded): Maor Elfass

x = name[:1] # returns the characters at position 0 to 0 (1 excluded): M

x = name[3:7] # returns the characters at position 3 to 6 (7 excluded): r El

x = name[:] # returns the whole string: Maor Elfassy
```

Recap - Question 4

What does the x variable hold after executing each line?

```
name = 'Maor Elfassy'
x = name.lower()
x = 'Maor' in x
x = name.title()
x = x == name
```

Strings

What does the x variable hold after executing each line?

```
name = 'Maor Elfassy'

x = name.lower() # to convert to lowercase - maor elfassy

x = 'Maor' in x # False (To check if a string contains another string (or character), use the in operator)

x = name.title() # to capitalize the first letter of every word - Maor Elfassy

x = x == name # True
```

Recap - Question 5

What does the *age* variable hold after executing those lines?

```
birth_year = “1985”
age = 2022 - birth_year
```

Strings - Type Conversion

In order to convert a String representation of a number to the actual number, use **int()** or **float()**

```
birth_year = "1985"

age = 2022 - birth_year
```

```
Traceback (most recent call last):
  File "/Users/maore/PycharmProjects/python-course/recap.py", line 10, in <module>
    main()
  File "/Users/maore/PycharmProjects/python-course/recap.py", line 7, in main
    age = 2022 - birth_year
TypeError: unsupported operand type(s) for -: 'int' and 'str'
```

Instead:

```
birth_year = "1985"

age = 2022 - int(birth_year)
```

Recap - Question 6

What does each operator mean?

Operator	Description	Example
and		x < 5 and y < 10
or		x < 5 or y < 4
not		not (x < 5 and y < 10)

Recap - Logical Operators

Logical operators are used to combine conditional statements:

Operator	Description	Example
and	Returns True if both statements are true	x < 5 and y < 10
or	Returns True if one of the statements is true	x < 5 or y < 4
not	Reverse the result, returns False if the result is true	not (x < 5 and y < 10)

Recap - Question 7

Comparison operators are used to compare two values. What does each operator mean?

Operator	Name	Example
==		x == y
!=		x != y
>		x > y
<		x < y
>=		x >= y
<=		x <= y

Comparison Operators

Comparison operators are used to compare two values:

Operator	Name	Example
==	Equal	x == y
!=	Not equal	x != y
>	Greater than	x > y
<	Less than	x < y
>=	Greater than or equal to	x >= y
<=	Less than or equal to	x <= y

Recap - Question 8

What will be printed?

```
temperature = 23

if temperature > 30:
    print("It is a hot day")
elif temperature < 20:
    print("It is a cold day")
else:
    print("It is a beautiful day")
```

Recap - Question 9

Imagine a shopping cart with several items, each with its own price.

What would be printed?

```
prices = [20, 37, 120, 52, 13, 240]
total = 0
for price in prices:
    total+= price
print(total)
```

Recap - Question 10

What is the main difference between a tuple and a list?

```
list_values = [1, 2, 3]
tuple_values = (1, 2, 3)
```

Tuple

Besides the different kinds of brackets used to delimit them, the main difference between a tuple and a list is that the tuple object is **immutable**.

Tuple is a **read-only** list. Once created, its content **cannot be changed.**

Functions

A function is a block of code which only runs when it is called

```
def say_hello():
  print("Hello world!")


say_hello() # call the function
```

Recap - Question 11

What will happen?

```
def say_hello(name):
  print(f'Hello {name}!')


say_hello() # call the function
```

Function with Arguments

```
TypeError: say_hello() missing 1 required positional argument:
'name'
```

Possible solution: Set the default value for the argument

```
def say_hello(name=None):

  print(f'Hello {name}!')



say_hello() # call the function - Hello None!
```

Recap - Question 12

A function can also return **multiple values:**

```
def square_point(x, y, z):

  x_squared = x * x

  y_squared = y * y

  z_squared = z * z

  return x_squared, y_squared, z_squared      # return all three values:


result = square_point(1, 2, 3)                # returns (1, 4, 9)
print(type(result))                           # What would be printed?
```

Function Return Value

A function can also return **multiple values:**

```
def square_point(x, y, z):

  x_squared = x * x

  y_squared = y * y

  z_squared = z * z

  return x_squared, y_squared, z_squared      # return all three values:


result = square_point(1, 2, 3)                # returns (1, 4, 9)
print(type(result))                           # Tuple
```

Classes

We use classes to define new types. It is like a **template**, or **blueprint**

```
class MyClass:

  x = 5
```

Now we can use the class named MyClass to create objects, or instances:

```
p1 = MyClass()

print(p1.x)
```

Class Methods

Class can also contain **methods**

Methods in classes are functions that belong to the class/object

```
class Dog:
  def bark(self):
    print("woof!")

charlie = Dog()     # create a new instance of the class
charlie.bark()          # call the class method
```

The __init__() method

All classes have a method called **__init__()**, which is executed when the class is being initiated

Use the __init__() function to assign values to object properties, or other operations that are necessary to do when the object is being created:

```
class Person:
  def __init__(self, name, age):
    self.name = name
    self.age = age

p1 = Person("John", 36)

print(p1.name)
print(p1.age)
```

Note: The **__init__()** function is called **automatically** every time an object of the class is created

Module

A module is a code library

A file containing a set of functions you want to include in your application

To **create** a module - just save the code in a file with **.py** extension

Example: Save this code in a file named mymodule.py

```
def greeting(name):
  print("Hello, " + name)
```

Module

To **use** a module, we use the **import** statement

```
# import the entire module
import mymodule
mymodule.greeting("Jonathan")

# import only specific function(s) within the module
from mymodule import greeting
greeting("Jonathan")
```

Modules can contain functions and variables of all types (arrays, dictionaries, objects etc)

Module

There is a huge library of built-in modules in Python which you can import

Examples:

```
import platform
x = platform.system()      # Returns the system/OS name, e.g. 'Linux'
print(x)

import random
x = random.randint(1,10)  # Returns a random number in the range 1-10
print(x)
```

Files

File handling is an important part of any application.

Python has several functions for creating, reading, updating, and deleting files.

To open a file for reading, use the built-in open() function:

```
f = open("demofile.txt")
```

The open() function returns a file object, which has a read() method for reading the content of the file:

```
print(f.read())
```

Files

Use the readline() function to read a single line from the file:

```
f = open("demofile.txt")
print(f.readline())
f.close()
```

Call readline() multiple times to read more lines

To return all lines in the file (as a list), use readlines():

```
lines = f.readlines()
for line in lines:
    print(line)
f.close()
```

Files

To **write** to an existing file, you must add a parameter to the open() function:

"a" - Append - will append to the end of the file

"w" - Write - will overwrite any existing content

```
f = open("demofile2.txt", "a")

f.write("Now the file has more content!")
```

```
f = open("demofile3.txt", "w")

f.write("Woops! I have deleted the content!")
```

Exercise

- Receive an input file (*input.txt*)
- Count the number of **occurrences** (frequency) of each word in the file
- Print the result (as dictionary) to a new file (*result.txt*)

Example (input.txt):

```
'this is the text file, and it is used to take words and count'
```

Expected output (result.txt):

```
'this': 1
'is': 2
'the': 3
```

...

Cloud Intro

Service providers (Amazon, Google, Microsoft) enables public use of resources:

- Virtual machines (EC2)
- Applications (Gmail)
- Storage (S3)

Public cloud services may be free or offered on a pay-per-usage model.

Before the cloud - a touch of history

- Each company had its own data center and managed its own infrastructure
- Physical machines were needed to be maintained
- Problems:
  - Starting a new company was very expensive
  - Large IT departments with experienced people

Can you think about the public cloud advantages?

Public cloud advantages

- Scale easy
- More Flexibility
- Pay only for company usage
- Manage only the application level, but not the infrastructure or the hardware

AWS - Amazon Web Service

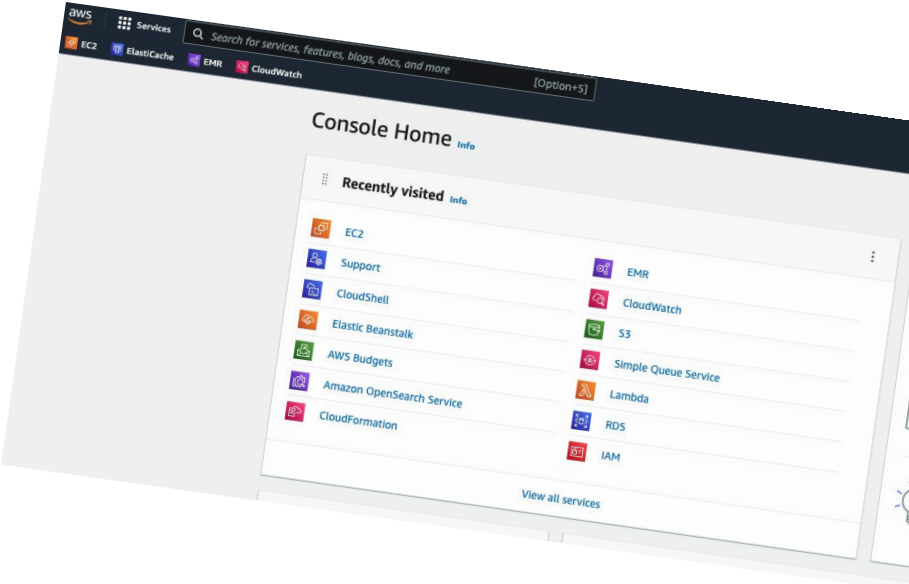
Amazon.com was founded as an online bookstore by Jeff Bezos in 1994 in his garage, in Bellevue, Washington. Initially an online marketplace for books.

Today, it is the world's largest online retailer and marketplace, smart speaker provider, cloud computing service through AWS, live-streaming service through Twitch, and Internet company as measured by revenue and market share.

AWS - Amazon Web Service

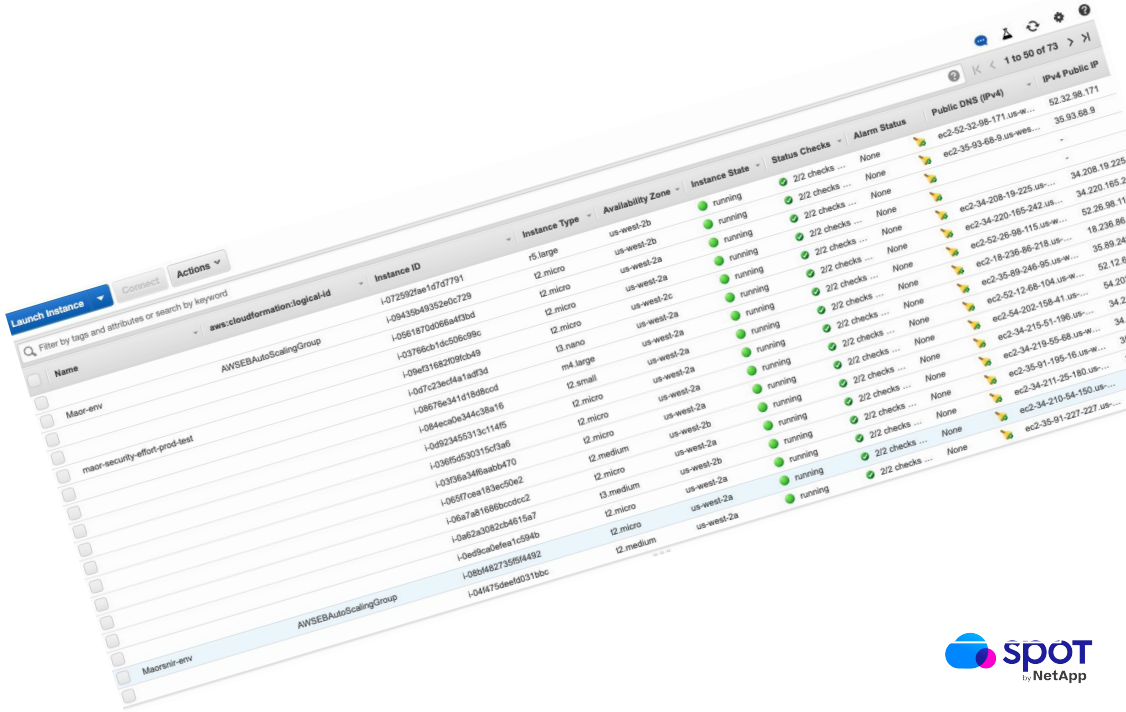
- Subsidiary of Amazon that provides on-demand cloud computing platforms to individuals, companies, and governments, on a metered pay-as-you-go basis.

- Re-launched in 2006 with 3 main services:
  - S3, EC2, SQS



AWS Main services

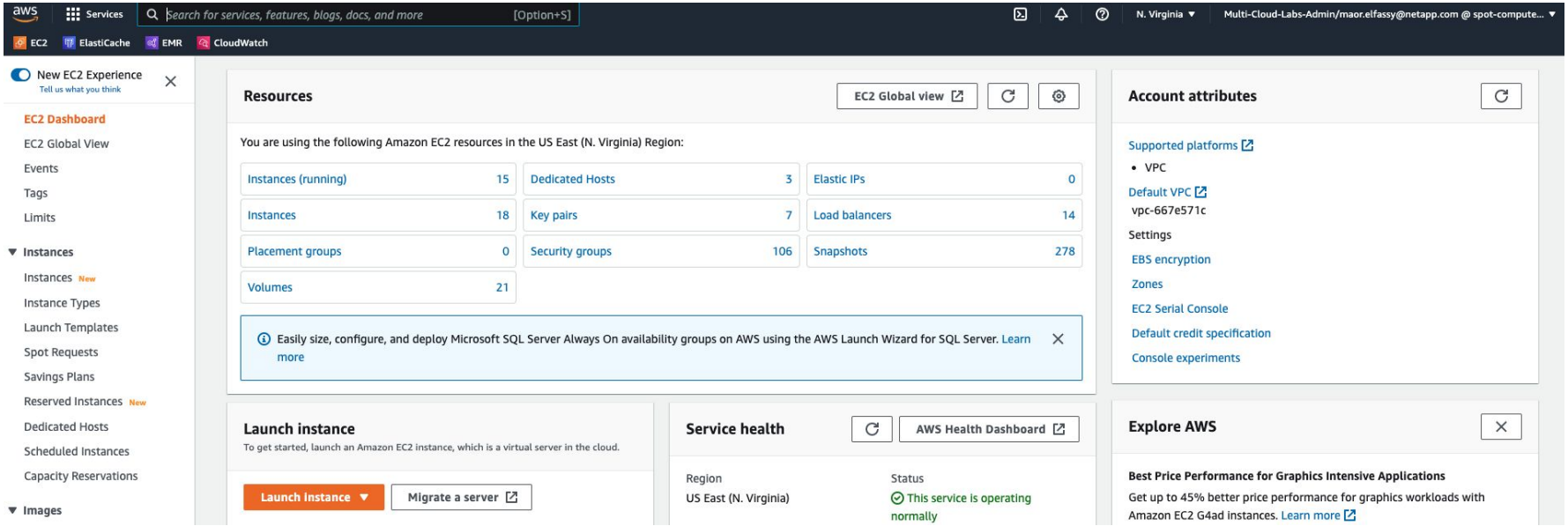
- **EC2** - Elastic Compute Cloud Service
- **SQS** - Simple Queue Service
- **S3** - Simple Storage Service
- **ELB** - Elastic Load Balancer Service
- **RDS** - Relational Database Service



spot by NetApp

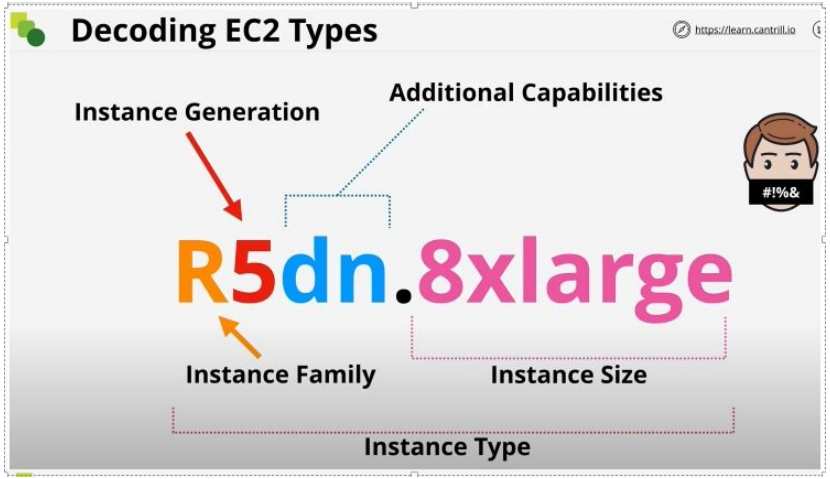
AWS Console

Demo



spot by NetApp

Instance types



Instance Type

- Additional Capabilities:
  - a - AMD CPU
  - d - NVMe Storage
  - n - Network optimized
  - e - Extra capacity, RAM or storage
- https://aws.amazon.com/ec2/instance-types
- Instance family - each of these are design for a specific of computing
  - T - General Purpose
  - M - General Purpose
  - I - Storage Optimized
  - R - Memory Optimized
  - X - Memory Optimized
  - C - Compute Optimized

EC2 Instance Types

https://learn.cantrill.io adriancan

Categories	Type	Details / Notes
General Purpose	A1, M6g	Graviton (A1) Graviton 2 (M6g) ARM based processors. Efficient.
	T3, T3a	Burst Pool - Cheaper assuming nominal low levels of usage, with occasional Peaks.
	M5, M5a, M5n	Steady state workload alternative to T3/3a - Intel / AMD Architecture
Compute Optimized	C5, C5n	Media encoding, Scientific Modelling, Gaming Servers, General Machine learning
Memory Optimized	R5, R5a	Real time analytics, in-memory caches, certain DB applications (in-memory operations)
	X1, X1e	Large scale in-memory applications .. lowest $ per GiB memory in AWS
	High Memory (u-Xtbl)	Highest memory of all AWS instances
	z1d	Large memory and CPU - with directly attached NVMe
	P3	GPU instances (Tesla v100 GPUs) - parallel processing & machine learning

What is Boto3?

Boto3 is the **AWS SDK for Python** provides **a Python API for AWS infrastructure services**. Using the SDK for Python, you can build applications on top of Amazon S3, Amazon EC2, Amazon DynamoDB, and more.

Documentation: https://docs.aws.amazon.com/pythonsdk/?id=docs_gateway

AWS Boto3 - Create your AWS keys

aws | Services | Search for services, features, blogs, docs, and more [Option+S] | Global | Multi-Cloud-Labs-Admin/maor.elfassy@netapp.com @ spot-compute...

EC2 | ElastiCache | EMR | CloudWatch

Identity and Access Management (IAM)

Search IAM

- Dashboard
- Access management
  - User groups
  - Users
  - Roles
  - Policies
  - Identity providers
  - Account settings
- Access reports
  - Access analyzer

Introducing the new Users list experience
We've redesigned the Users list experience to make it easier to use. Let us know what you think.

IAM > Users

Users (31) Info
An IAM user is an identity with long-term credentials that is used to interact with AWS in an account.

Delete | Add users

Find users by username or access key

1 2

User name	Groups	Last activity	MFA	Password age	Active key age
a-planzone	None	627 days ago	None	811 days ago	811 days ago
alex_v_user_ami	Administrator	453 days ago	None	None	714 days ago
AmirIMac	sqs_admin	558 days ago	None	684 days ago	684 days ago
Azure-market-predictions-admin	admin	14 days ago	None	None	161 days ago
cloud-analyzer	None	924 days ago	None	None	933 days ago

spot by NetApp

AWS Boto3 - Create your AWS keys

Add user

1 2 3 4 5

Set user details

You can add multiple users at once with the same access type and permissions. Learn more

User name* maor-python-course-1

Add another user

Select AWS access type

Select how these users will primarily access AWS. If you choose only programmatic access, it does NOT prevent users from accessing the console using an assumed role. Access keys and autogenerated passwords are provided in the last step. Learn more

Select AWS credential type*

- [x] **Access key - Programmatic access**
  Enables an **access key ID** and **secret access key** for the AWS API, CLI, SDK, and other development tools.
- [ ] **Password - AWS Management Console access**
  Enables a **password** that allows users to sign-in to the AWS Management Console.

* Required

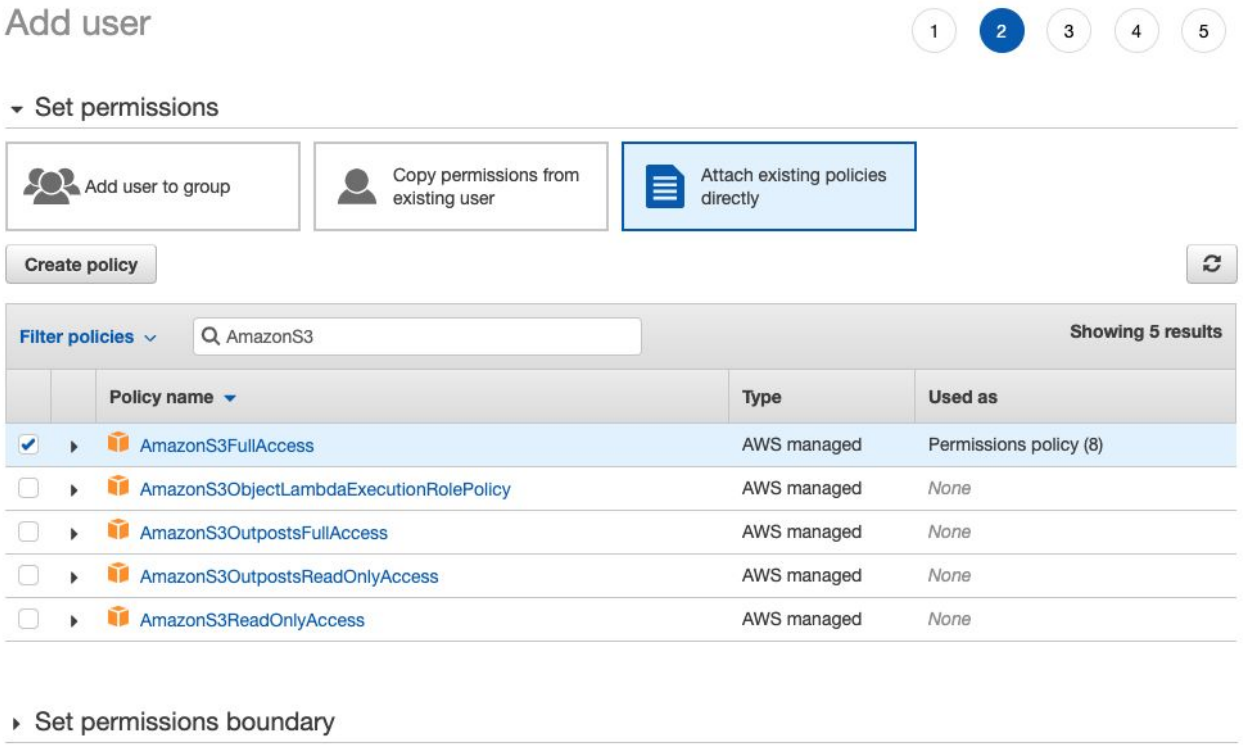
Cancel | Next: Permissions

spot by NetApp

AWS Boto3 - Create your AWS keys

Search and mark the following Policies:

- AmazonEC2FullAccess
- AmazonS3FullAccess



AWS Boto3 - Create your AWS keys

-

Add user

1 2 3 4 5

Review

Review your choices. After you create the user, you can view and download the autogenerated password and access key.

User details

User name	maor-python-course-1
AWS access type	Programmatic access - with an access key
Permissions boundary	Permissions boundary is not set

Permissions summary

The following policies will be attached to the user shown above.

Type	Name
Managed policy	AmazonEC2FullAccess
Managed policy	AmazonS3FullAccess
Managed policy	AmazonRDSFullAccess

Tags

The new user will receive the following tag

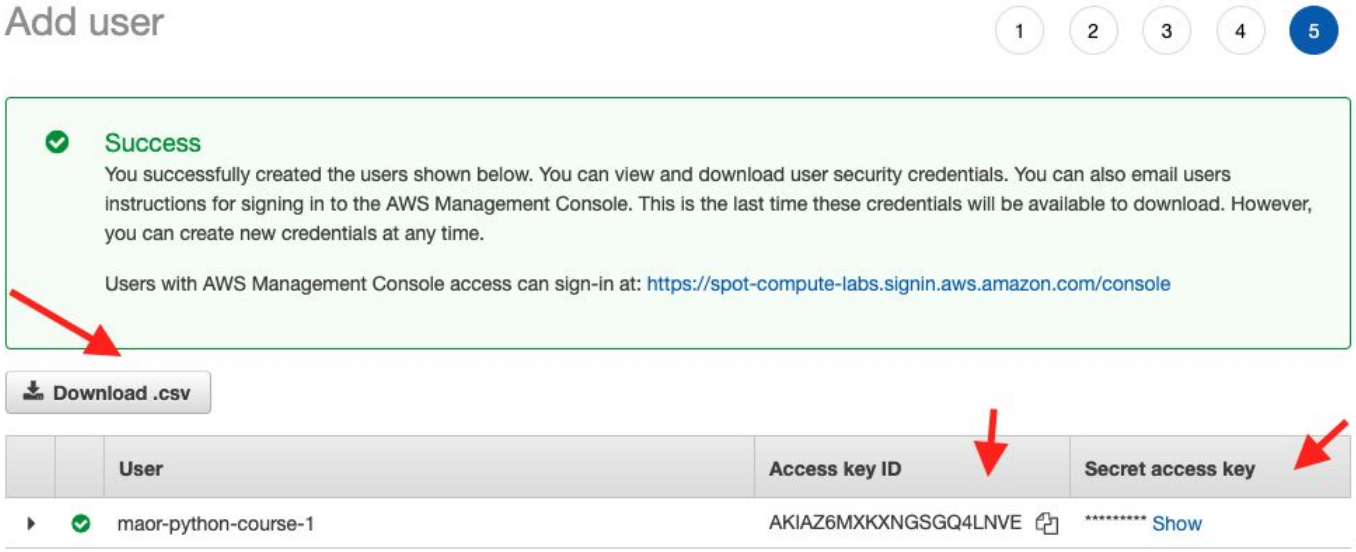
Key	Value
Creator	Maor Elfassy

Cancel Previous Create user

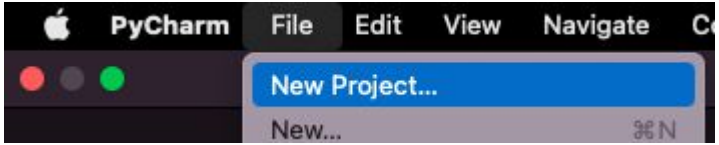
spot by NetApp

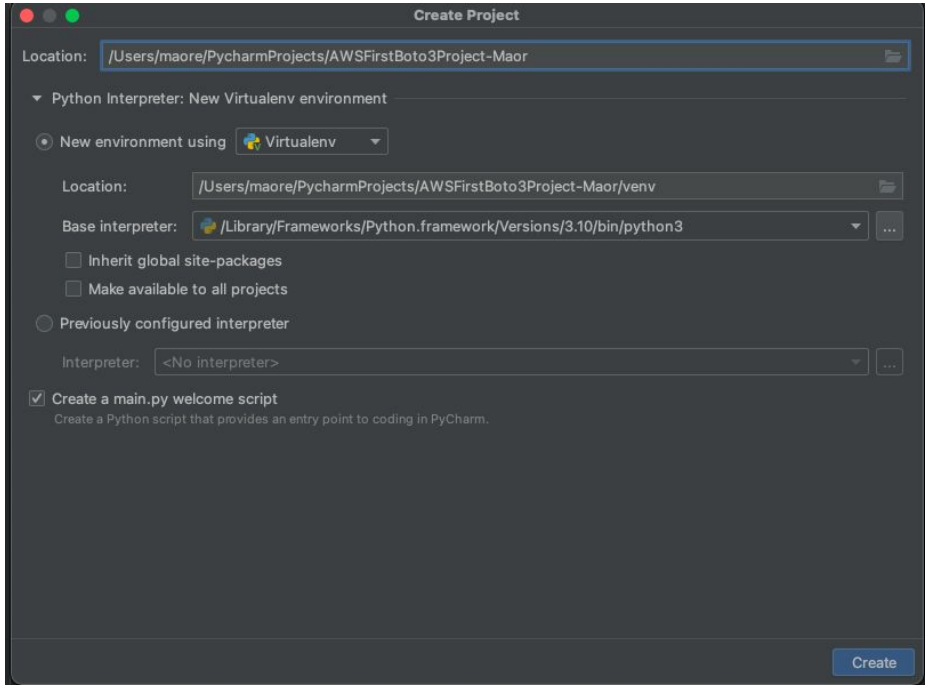
AWS Boto3 - Create your AWS keys

Save your Access key ID & Secret access key!

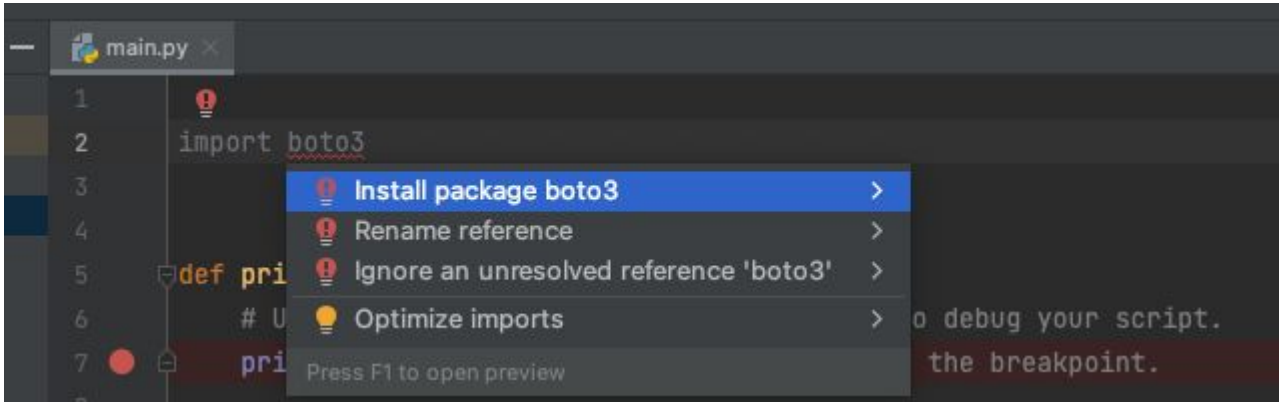


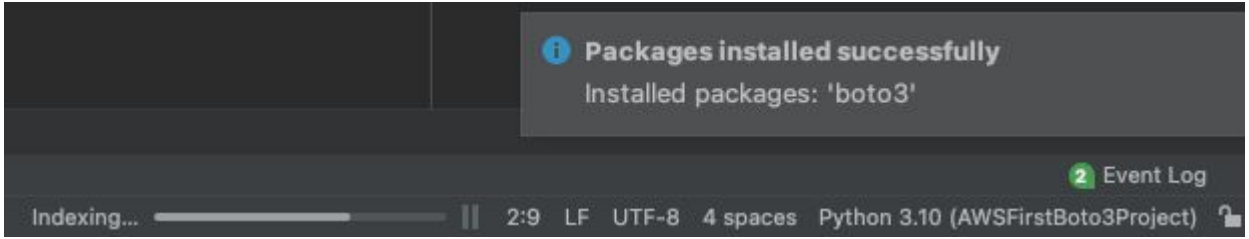
AWS Boto3 - Create your first Boto3 project





AWS Boto3 - Import & Install Boto3





AWS Boto3 - Init client and run your first Boto3 App

main.py

```python
import boto3


def main():
    print('Initializing Boto3 EC2 client')
    ec2_client = boto3.client('ec2', aws_access_key_id='A[redacted]VE',
                              aws_secret_access_key='d[redacted]R5a')
    print("Start getting EC2 instances")
    ec2_instances = ec2_client.describe_instances()
    print(ec2_instances)


if __name__ == '__main__':
    main()
```

Exercise - 1

- Launch 1 EC2 instance via the AWS Console on a specific region. (us-west-2)
- Create your own AWS credentials (AWS Access key id + AWS Secret access key)
- Create your first Boto3 project:
  - Create an EC2 client
  - Print all running EC2 instances
- Use the documentation:
  - Python SDK - https://boto3.amazonaws.com/v1/documentation/api/latest/reference/services/ec2.html#EC2.Client.describe_instances
  - API Reference - https://docs.aws.amazon.com/goto/WebAPI/ec2-2016-11-15/DescribeInstances

Terminate all EC2 instances - now!

spot by NetApp

More Questions?
Thank You!

spot by NetApp