**PYTHON FOR DEVOPS**

**Python VS Shell Scripting:**

Mostly Devops engineers deploy their applications in the Linux machines and they are more secured than windows.

To get the details of file or to create a file, CPU utilization, free space and memory usage we use the Shell commands.

Instead of using the commands one by one to get the details. We can simply create a file with extension as **Test.sh** called Shell scripting file we can get all these data at one place.

**Two reasons to use Python instead of Shell scripting:**

1. Not all the deployments are done in Linux, some can deploy in windows also. Although we have ansible for that where we can simply write a playbook but the Ansible is also written in Python.
2. When we are working with complex tasks, API’s and Data manipulation. Python makes it easy for us to achieve all these tasks than Shell scripting.

**Data Types:**

1. Numeric Data Types:
   * int: Represents integers (whole numbers). Example: x = 5
   * float: Represents floating-point numbers (numbers with decimal points). Example: y = 3.14
   * complex: Represents complex numbers. Example: z = 2 + 3j
2. Sequence Types:
   * str: Represents strings (sequences of characters). Example: text = "Hello, World"
   * list: Represents lists (ordered, mutable sequences). Example: my\_list = [1, 2, 3]
   * tuple: Represents tuples (ordered, immutable sequences). Example: my\_tuple = (1, 2, 3)
3. Mapping Type:
   * dict: Represents dictionaries (key-value pairs). Example: my\_dict = {'name': 'John', 'age': 30}
4. Set Types:
   * set: Represents sets (unordered collections of unique elements). Example: my\_set = {1, 2, 3}
   * frozenset: Represents immutable sets. Example: my\_frozenset = frozenset([1, 2, 3])
5. Boolean Type:
   * bool: Represents Boolean values (True or False). Example: is\_valid = True
6. Binary Types:
   * bytes: Represents immutable sequences of bytes. Example: data = b'Hello'
   * bytearray: Represents mutable sequences of bytes. Example: data = bytearray(b'Hello')
7. None Type:
   * NoneType: Represents the None object, which is used to indicate the absence of a value or a null value.
8. Custom Data Types:
   * You can also define your custom data types using classes and objects.

**String Data Type in Python:**

* In Python, a string is a sequence of characters, enclosed within single (' '), double (" ")
* You can access individual characters in a string using indexing, e.g., my\_string[0] will give you the first character.
* Python provides many built-in methods for string manipulation, such as split(), join(), and startswith()

**Split():**

arn = "arn:aws:iam::123456789012:user/johndoe"

print (arn.split("/")[1)

Output:

Johndoe

--------------------------------------------------------------------------------------------------------------------------------------

**Upper()/Lower():**

str1="Hello"  
str2="WORLD"

print(str1.upper()+"! "+ str2.lower())

Output:

HELLO! world

--------------------------------------------------------------------------------------------------------------------------------------

**Concatination:**

str1="Hello"  
str2="World"  
result=str1+"! "+str2  
print(result)

Output:

Hello! World

--------------------------------------------------------------------------------------------------------------------------------------

**Len():**

arn = "arn:aws:iam::123456789012:user/johndoe"

length=len(arn)  
print("length of the string is :",length)

Output:

length of the string is : 38

--------------------------------------------------------------------------------------------------------------------------------------

**Replace():**

text = "Python is awesome"  
new\_text = text.replace("awesome", "great")  
print("Modified text:", new\_text)

Output:

Modified text: Python is great

--------------------------------------------------------------------------------------------------------------------------------------

**REGEX:**

**Regular Expressions for Text Processing:**

* Regular expressions (regex or regexp) are a powerful tool for pattern matching and text processing.
* The re module in Python is used for working with regular expressions.
* Common metacharacters: . (any character), \* (zero or more), + (one or more), ? (zero or one), [] (character class), | (OR), ^ (start of a line), $ (end of a line), etc.
* Examples of regex usage: matching emails, phone numbers, or extracting data from text.
* re module functions include re.match(), re.search(), re.findall(), and re.sub() for pattern matching and replacement.

import re

text = "The quick brown fox"

pattern = r"brown"

search = re.search(pattern, text)

if search:

print("Pattern found:", search.group())

else:

print("Pattern not found")

--------------------------------------------------------------------------------------------------------------------------------------