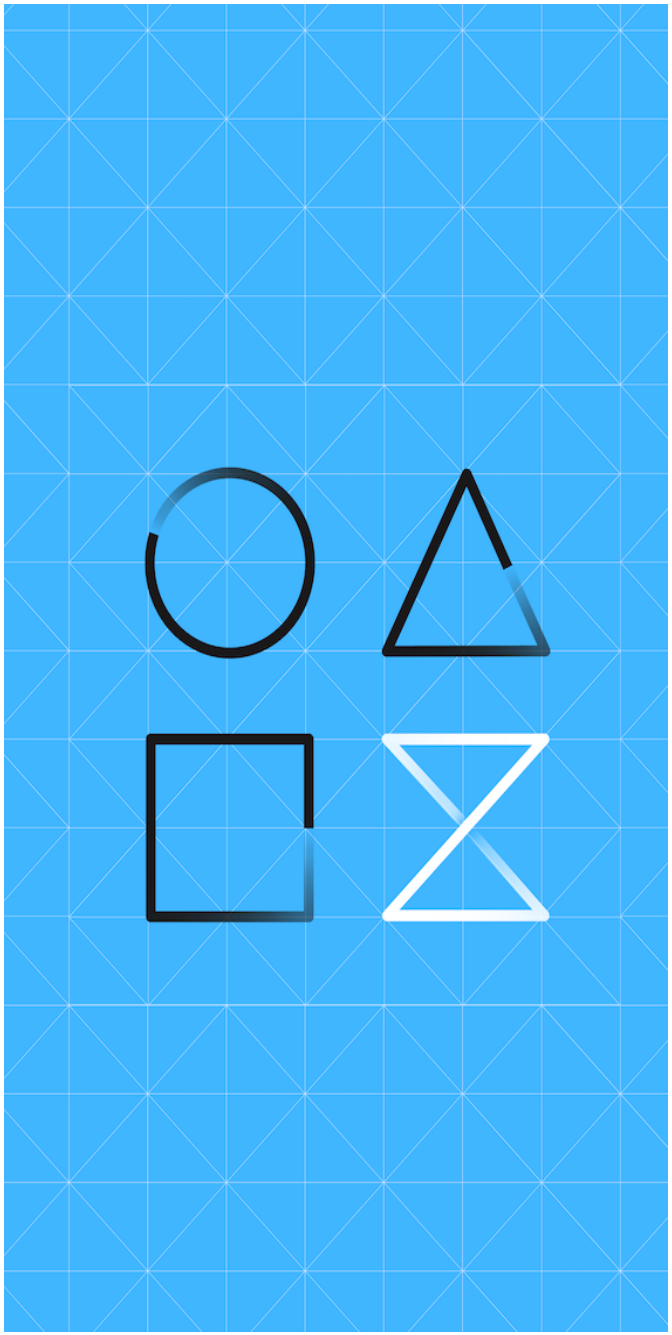


Fundamentos

250121-1118

WRAPUP

Work Smarter

Tradução automatizada usando IBM Globalization Service

250121-1118

TIME TO BRING IN Z OPEN AUTOMATION UTILITIES

O Desafio

Agora que você conhece alguns dos principais fundamentos do z/OS e também explorou um pouco os scripts e o Python, vamos reunir tudo isso com algo chamado ' Z Open Automation Utilities ' ou, para economizar espaço daqui para frente, *ZOAU*.

Combine as principais habilidades do IBM Z que você adquiriu, aprendendo sobre conjuntos de dados e JCL, com as habilidades de script encontradas no Python, para automatizar algumas tarefas que um programador de sistemas pode realizar no dia a dia.

Antes De Começar

A esta altura, você já sabe como trabalhar com conjuntos de dados, enviar trabalhos e navegar pelo site USS.

Você colocará tudo isso em prática aqui e concluirá sua compreensão fundamental do z/OS.

Investimento

Etapas	Duração
12	90 minutos

WRAPUP250121-1118

1 DECHO ... DECHO ... DECHO

Você usará um conjunto de dados sequenciais para este desafio.

Se você ainda tiver o conjunto de dados sequenciais JCL3OUT, use-o. Caso contrário, crie um novo conjunto de dados clicando com o botão direito do mouse no perfil de conexão VSCode em ' Data Sets ', selecionando "Create New Data Set" e seguindo as instruções.

Tudo pronto?

Em seguida, acesse **novamente o sistema do mainframe em SSH para obter um shell USS** e digite a versão correta do comando a seguir, substituindo seu próprio ID de usuário e o conjunto de dados que deseja usar.

```
decho -a "This line goes at the bottom" 'Zxxxxx.JCL3OUT'
```

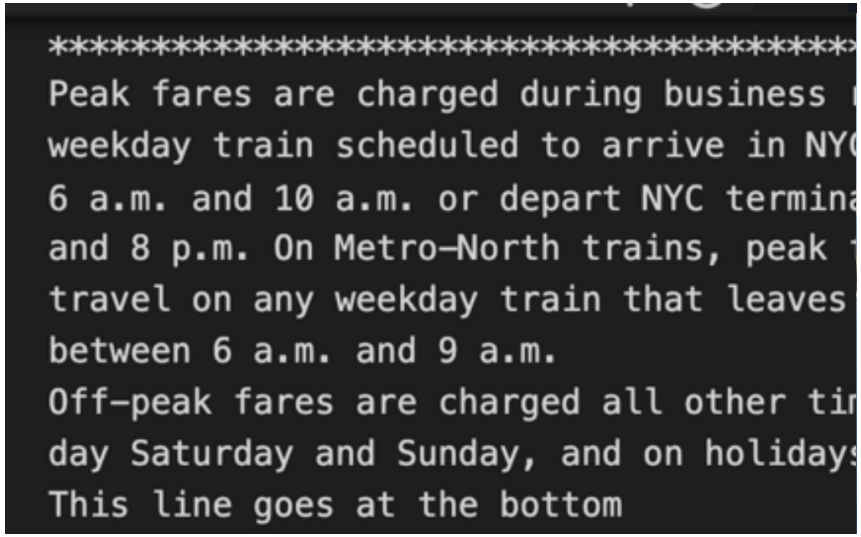
Isso pode levar alguns segundos para ser totalmente executado, portanto, seja paciente.

O comando decho grava texto em um conjunto de dados. O sinalizador -a direciona o texto a ser anexado à parte inferior do conjunto de dados. Se o sinalizador -a não estiver definido, o texto substituirá o conteúdo existente no conjunto de dados.

2 HELLO DOWN THERE

Edite o conjunto de dados JCL3OUT em VSCode.

Se você o criou novamente, talvez seja necessário clicar com o botão direito do mouse nele e selecionar "Pull From Mainframe"; isso garante que você tenha a versão mais recente absoluta em sua sessão de edição.



Você deverá ver a linha que acabou de decho 'd anexada à parte inferior do conjunto de dados.

Um truque interessante, embora provavelmente não seja mais fácil do que simplesmente abri-lo e digitá-lo manualmente.

O verdadeiro poder dessas ferramentas vem da capacidade de integrar ações do z/OS a programas e scripts de shell novos e existentes do Python.

3 ALL SET WITH DATASETS

Certifique-se de que você tenha uma cópia do **dslist.py** em seu diretório pessoal em USS.

Você deve dar uma olhada rápida no código para descobrir o que ele estará fazendo.

```
#!/usr/bin/python3
# Let's just import the datasets module from zoautils
from zoautil_py import datasets

# This line creates a *list* of the data set members
# inside the data set specified in the argument.
# A list, in Python, is a type of data set object that
# can easily be sorted, appended, counted, reversed, and more
members_list = datasets.list_members("ZXXXXX.JCL")

# Here we meet our old friend the *for* loop.
# The loop is saying create a new variable called "member"
# Then, from that number to however long members_list is,
# print out that number in the list.
```


É um script Python (se você não adivinhou pelo sufixo .py) e você pode ver que ele obtém uma listagem de todos os membros em um determinado conjunto de dados e, em seguida, usa um "loop for " para imprimir cada nome de membro nele.

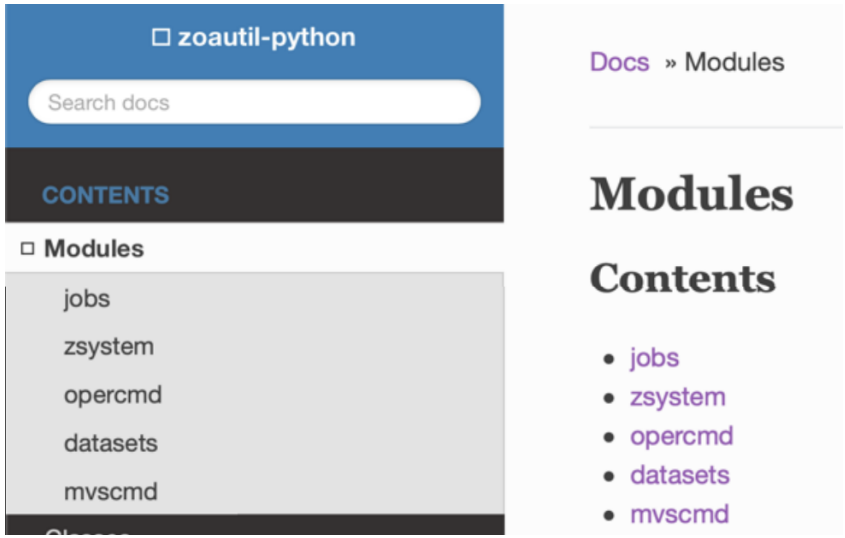
Pelo menos, isso acontecerá quando você tiver editado o script para apontar para um dos conjuntos de dados particionados na linha número 9 (a linha que começa em members_list =). Experimente agora, especificando seu conjunto de dados JCL ou OUTPUT.

**Salve o arquivo e execute-o em um login do shell USS.

OBSERVAÇÃO: lembre-se de que no desafio **CODE** que, toda vez que você executou um script python, usou o comando python3, e *não* o velho e simples python.

Simples e agradável, certo?

4 WAITER - THERE'S A PYTHON IN MY Z



Escrever um script Python apenas para listar o que há em um conjunto de dados pode parecer um exagero e, nesse caso... sim, é.

Entretanto, com um programa muito simples, você pode ver como isso pode ser feito.

E depois de ver e entender, talvez isso lhe dê algumas ideias.

Agora que você já sabe como executar as principais tarefas do z/OS no código Python. Este exemplo envolveu apenas conjuntos de dados. Você também pode trabalhar com trabalhos, com o próprio sistema e com dois tipos de comandos.

Dê uma olhada na página ZOAU Python APIs e leia mais sobre esses módulos em https://www.ibm.com/docs/en/zoau/1.2.0?topic=reference-classes

(esse local muda com as novas versões - se você precisar encontrar uma versão diferente, faça uma pesquisa na Internet por " *zoau python reference classes* ")

5 INSERT SOME EXTRA LOGIC

Agora você pode trabalhar com um script que faz um pouco mais, incluindo a criação de um novo conjunto de dados sequenciais, a coleta de alguns dados e a gravação desses dados no conjunto de dados sequenciais recém-criado.

```
#!/usr/bin/python3
# Let's just import the datasets module from zoautils
from zoautil_py import datasets, jobs, zsystem
import sys

#Prompt for data set name:
dsname = input("Enter the Sequential Data Set name:")

#if it exists, say we foudn it, and we'll use it. Otherw
if (datasets.exists(dsname) == True):
    print("Data set found! We will use it")
else:
    create_new = input("Data set not found. Should we cr
    if (create_new.upper() == "Y"):
```

Você pode começar com algumas dessas funções que já funcionam, e seu trabalho será fazer com que o restante delas funcione sem problemas.

Abra o arquivo ' members.py ' no seu diretório pessoal e dê uma olhada no código-fonte.

6 GET YOUR HACK ON!

A essa altura, você pode estar vendo muito código e ficando nervoso.

O negócio é o seguinte... mesmo que você não queira ser um programador, se trabalhar com sistemas, terá que passar algum tempo analisando códigos e fazendo pequenos ajustes nos programas de outras pessoas.

Esse é o próprio espírito do hacking.

Grande parte da codificação envolve descobrir por onde começar, reunir todos os recursos de que você precisa e criar algo do zero.

Hackear é pegar algo que já faz algo útil e fazer pequenas alterações para adicionar um recurso, corrigir problemas ou simplesmente deixar sua própria marca nele.

Sempre que há código envolvido, há MUITOS comentários para ajudá-lo a entender o que está acontecendo, e saiba que (pelo menos para esse desafio) você nunca precisará escrever mais do que algumas linhas de código novo.

Trabalhar o fluxo e a estrutura do que já existe provavelmente será a coisa mais difícil.

Mas não se preocupe - você tem o que é preciso!

POR QUE APRENDER DE OUTRA FORMA? PORQUE É BOM TER OPÇÕES.

Se você começou a trabalhar no z/OS há alguns anos, é provável que esteja muito confortável com o JCL e com os vários métodos já incorporados ao sistema operacional para fazer as coisas.

No entanto, se você estiver acostumado a trabalhar com Linux e Python, talvez queira continuar escrevendo código da mesma forma, sem deixar de ter acesso à funcionalidade principal de z/OS. É nesse ponto que os módulos da biblioteca ZOAU são úteis.

Aprender novos métodos além do que já sabemos permite mais opções e mais oportunidades de fazer escolhas eficientes.

Esses exercícios foram feitos para lhe dar algumas ideias do que é possível fazer quando você combina a capacidade de script e automação do Python com os recursos de back-end do z/OS.

WRAPUP250121-1118

7 LOADING MODULES

```
#!/usr/bin/python3
# Let's just import the datasets module from zoaut
from zoautil_py import datasets, jobs, zsystem
import sys

#Prompt for data set name:
dsname = input("Enter the Sequential Data Set name

#if it exists, say we foudn it, and we'll use it.
if (datasets.exists(dsname) == True):
    print("Data set found! We will use it")
else:
```

Veja a linha nº 3 de **members.py**.

Três módulos estão sendo importados da biblioteca ' zoautil.py ': datasets, jobs, e zsystem.

Cada um desses módulos fornece um conjunto de funções que você pode usar em seu código.

Você pode ler tudo sobre os módulos e o que eles fazem aqui:

https://www.ibm.com/docs/en/zoau/1.2.0?topic=python-api-reference

8 GET THAT DATASET

```
#Prompt for data set name:
dsname = input("Enter the Sequential Data Set name:")

#if it exists, say we foudn it, and we'll use it. Otherw
if (datasets.exists(dsname) == True):
    print("Data set found! We will use it")
else:
    create_new = input("Data set not found. Should we cr
    if (create_new.upper() == "Y"):
        # User wants to create a file
        # This is the part where we create a new data se
        datasets.create(dsname,type="???",primary_space=
    else: sys.exit("Without a data set name, we cannot c
```

As linhas 6 a 18 de **members.py** solicita ao usuário um nome de conjunto de dados sequencial e atribui esse valor à variável ' dsname '.

Se o conjunto de dados já existir, o programa o usará e continuará.

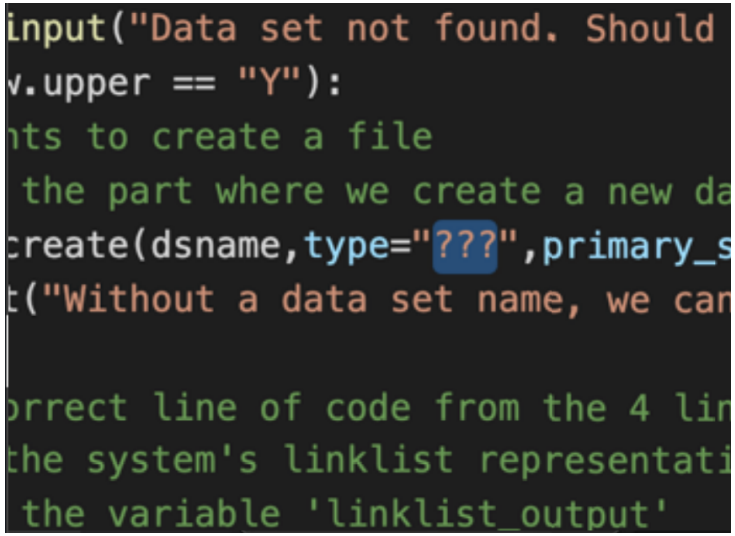
Se o conjunto de dados *não* existir, você poderá criá-lo para o usuário (o que será feito na próxima etapa).

Se o conjunto de dados NÃO existir e o usuário disser que não deseja criá-lo, não faz muito sentido continuar o restante do código, portanto, ' sys.exit( ) ' é usado para encerrar o programa.

Até agora, tudo bem? Muito bem, vamos começar a criar alguns códigos.

9 IN THE RIGHT SEQUENCE

Para este desafio, você usará o script (**members.py**) para coletar dados e gravá-los em um conjunto de dados sequencial.



A linha 17 está usando a função create do módulo zoau ' datasets ' para tentar criar esse conjunto de dados sequenciais. Mas, como está agora, não funcionará até que você faça um pequeno ajuste. Dê uma olhada na função *datasets.create( )* e descubra o que precisa substituir a função **???** para criar um conjunto de dados sequencial.

Use **Zxxxxx.COMPLETE** como o nome do conjunto de dados sequencial quando solicitado pelo script. É aí que o trabalho de validação procurará validar seu trabalho, bem como executar seu ' members.py '.

Dica: um conjunto de dados sequencial é um tipo específico de conjunto de dados. Outros tipos de conjuntos de dados são KSDS , PDS , ESDS , mas, nesse caso, o tipo que você precisa criar é um conjunto de dados sequencial.

Consulte a documentação e os comentários para obter ajuda.

KSDS? ESDS? PENSEI QUE FOSSE APENAS UM PARTICIONADO E SEQUENCIAL?

Às vezes, um conjunto de dados é pouco mais do que um simples arquivo; um local para armazenar alguns dados que você deseja ler ou processar de cima para baixo. Os membros do Dataset particionado e os conjuntos de dados sequenciais funcionam bem para esses casos.

No entanto, às vezes os aplicativos precisam de acesso rápido a um registro específico e precisam de uma maneira melhor de chegar a esse registro do que ler todo o conjunto de dados de cima para baixo. Nessas situações, é possível usar um Conjunto de *Dados Sequenciados por Chave* (KSDS), *Conjunto de Dados Sequenciados por Entrada* (ESDS) ou Conjunto de Dados de Registro Relativo (RRDS). Todos esses são exemplos de conjuntos de dados do Virtual Storage Access Method (VSAM), que serão abordados em desafios posteriores.

Ter opções quando se trata de acesso a dados significa que o desenvolvedor ou programador de sistemas pode escolher a melhor, mais rápida e mais eficiente maneira de trabalhar com todos esses bits e bytes.

z/OS Compreender as opções e como fazer tudo funcionar faz de você um profissional valioso, e os empregadores definitivamente gostam desse tipo de coisa.

WRAPUP250121-1118

10 GRAB THE LINKLIST

Lembra-se de como você importou alguns módulos no início deste script? Quando o sistema operacional z/OS é iniciado, ele faz mais ou menos a mesma coisa, carregando módulos e bibliotecas cheios de tipos de recursos que o usuário pode precisar usar.

A lista completa de bibliotecas carregadas é conhecida como linklist, e é o que permite que o usuário digite um único comando em vez de ter que referenciá-lo pelo caminho completo todas as vezes.

Muito útil!

De qualquer forma, é importante saber como é o site linklist completo, que está disponível no módulo zsystem de zoautil.py.

```
# Uncomment the correct line of code from the 4 lines
# which will get the system's linklist representation
# its contents to the variable 'linklist_output'
# https://www.ibm.com/docs/en/zoau/1.1.1?topic=SSKFYE_
#linklist_output = zsystem.get_linklist()
#linklist_output = zsystem.list_linklist()
#linklist_output = zsystem.link_linklist()
#linklist_data = zsystem.list_linklist()

#Format the output so each member is on its own line
linklist_output = str(linklist_output).replace(',','\n
print(linklist_output)
```

Observe as linhas 20-27. Como você pode ver, há quatro linhas de código ali (e muitos comentários que você provavelmente deveria ler).

Sua tarefa é identificar e descomentar a única linha de código (de 24 a 27) que é a linha correta que capturará a representação linklist e a atribuirá à variável ' linklist_output '.

Observação : o link para a documentação on-line no código pode ficar desatualizado; você encontrará isso com frequência - os comentários ficam desatualizados, mas nem sempre são atualizados com as alterações no código.

11 LAST STOP: WRITING OUT

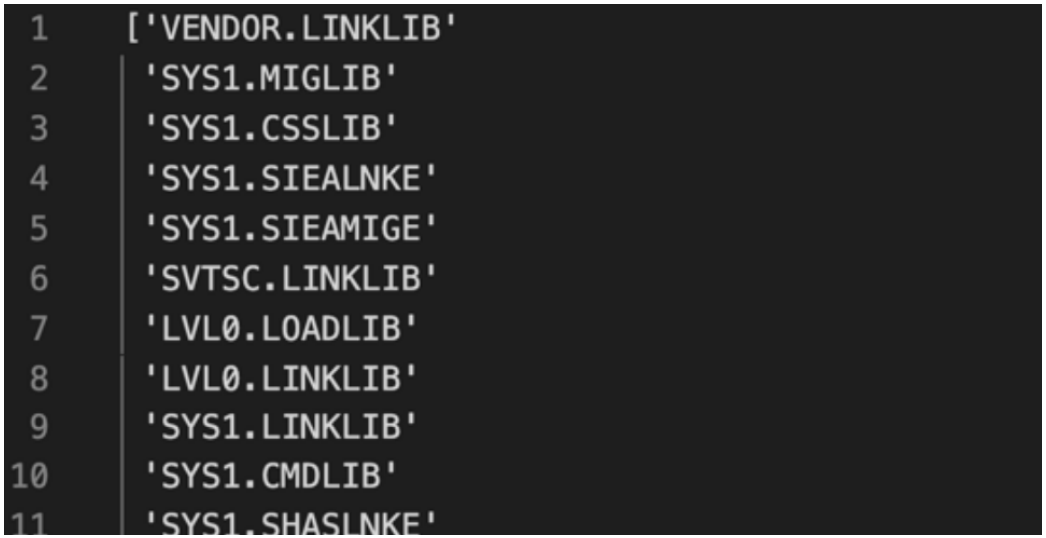
Até agora, você criou (ou pelo menos reutilizou) um conjunto de dados sequenciais e coletou alguns dados. Agora você precisa gravar esses dados no conjunto de dados no final do script.

```
#linklist_data = zsystem.list_linklist()

#This is just here to show the value of linklist_output
print(linklist_output)

# Write the value of linklist_info into our sequential
# This is the data set we created back on line xx
datasets.write(linklist_output,dsname,append=False)

# If everything looks good, run the JCL for this challe
# For bonus points (bonus points may not actually exist
# see if you can submit the JCL through this script.
```

Todas as informações estão lá, mas algo está errado e você precisará dar uma olhada no método ' datasets.write( ) ' e em sua documentação para descobrir o que precisa ser corrigido.

Depois de ter tudo esclarecido e em ordem, execute o programa e verifique se o script funciona corretamente para você.

12 CHECK IT; FINISH IT



Depois de concluir todas as etapas necessárias, você deverá ter um conjunto de dados sequenciais COMPLETE que contém uma listagem do sistema linklist. Consulte a captura de tela acima se precisar de esclarecimentos.

Encontre e envie o **CHKAUTO** para marcar este desafio como concluído.

Você pode fazer isso da mesma forma que tem feito até agora por meio de VSCode, mas pode apostar que há uma função Python para enviar JCL que também pode funcionar muito bem.

De qualquer forma, parabéns por ter concluído todos os desafios do Fundamentals!

Você está pronto para ir mais longe.

Bom trabalho - vamos recapitular	A seguir .
Você elaborou uma série de ações fundamentais e profundas do z/OS diretamente de um script do Python ! Mantivemos a simplicidade neste exemplo, mas agora que você sabe como conectar os pontos, deve ser capaz de criar qualquer número de novos utilitários que possam ser úteis no processamento de texto, na verificação da saída do trabalho, na verificação de alterações no sistema e no trabalho com conjuntos de dados. Você já sabia como fazer a maioria das coisas antes, mas agora conhece outra maneira. Como dissemos, é bom ter mais opções.	Você completou todos os desafios do site Fundamental! Você tem o que é preciso para passar para os desafios do Advanced. A esta altura, você já deve ter alguma ideia do que lhe interessa e provavelmente encontrará isso na próxima série de desafios disponíveis.