CONCLUSÃO: TRABALHE COM MAIS INTELIGÊNCIA

Hora de trazer o *Z Open Automation Utilities*

12 steps 90 minutes

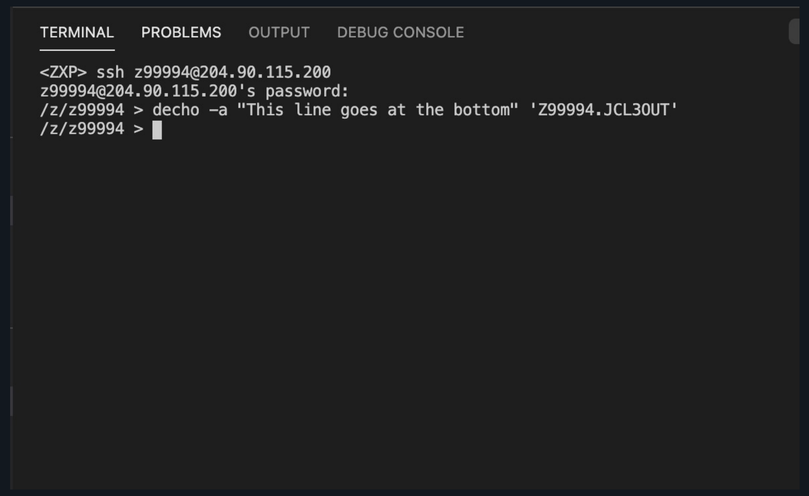
O DESAFIO

Agora que você conhece alguns dos principais fundamentos do z/OS e também explorou um pouco sobre scripts e Python, vamos reunir tudo isso com algo chamado Z Open Automation Utilities ou, para economizar espaço a partir de agora, ZOAU.

Combinaremos as principais habilidades Z, que conquistamos ao aprender sobre conjuntos de dados e JCL, com as habilidades de script encontradas em Python para automatizar algumas tarefas que um programador de sistema pode executar no dia-a-dia.

ANTES DE VOCÊ COMEÇAR

Neste ponto, você sabe como trabalhar com conjuntos de dados, enviar trabalhos e navegar pelo USS. Você usará tudo isso aqui e encerrará sua compreensão fundamental do z/OS.

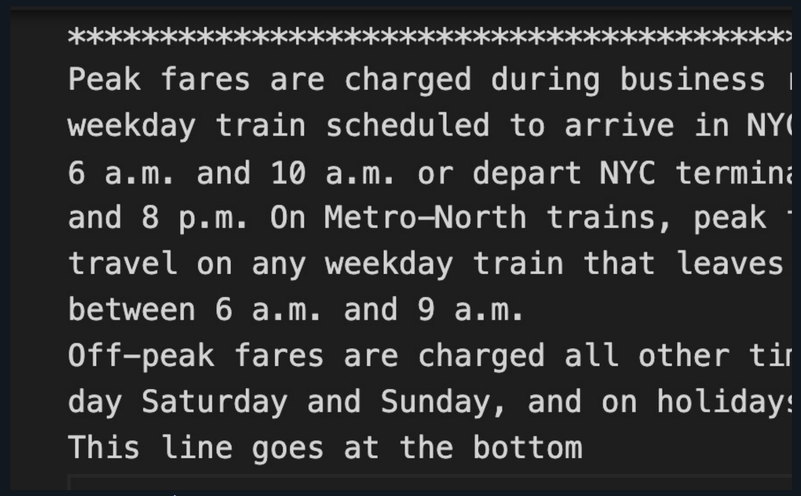


1. DECHO... DECHO... DECHO

Você precisará de um data set sequencial para esse desafio. Se você ainda não tiver seu data set sequencial JCL3OUT, faça um clicando com o botão direito do mouse em seu perfil de conexão em Data Sets e selecionando *"Create New Data Set"* e seguindo as instruções.
Tudo pronto? Tudo bem, SSH no sistema novamente para que você obtenha um shell e insira a versão correta deste comando, substituindo seu próprio ID de usuário e o data set que deseja usar.

decho -a "This line goes at the bottom" 'Z99994.JCL3OUT'

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

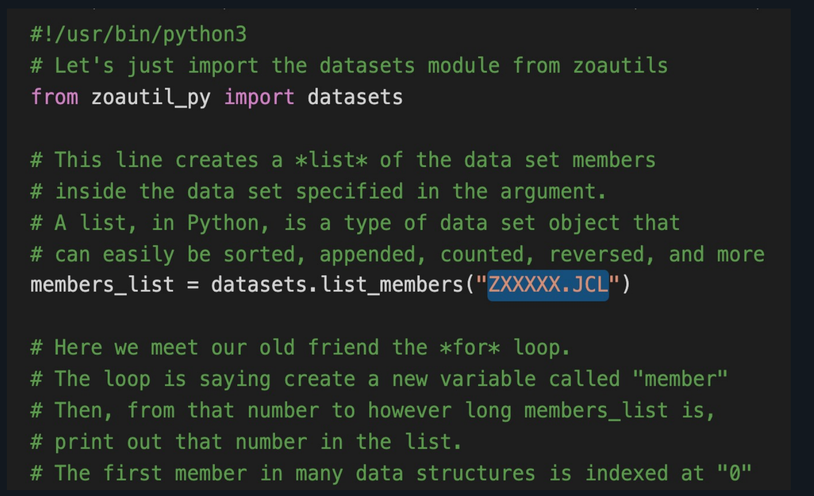


2. OI AÍ EMBAIXO

Abra o data set que você usou no comando no VS Code. Você pode precisar clicar com o botão direito do mouse e selecionar *"Pull From Mainframe"* (Puxar Do Mainframe) se já estiver aberto para obter a versão mais recente absoluta.

Você deve ver a linha que acabou de decodificar anexada à parte inferior do data set. Truque legal, embora provavelmente não seja mais fácil do que apenas abri-lo e digitá-lo manualmente.

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



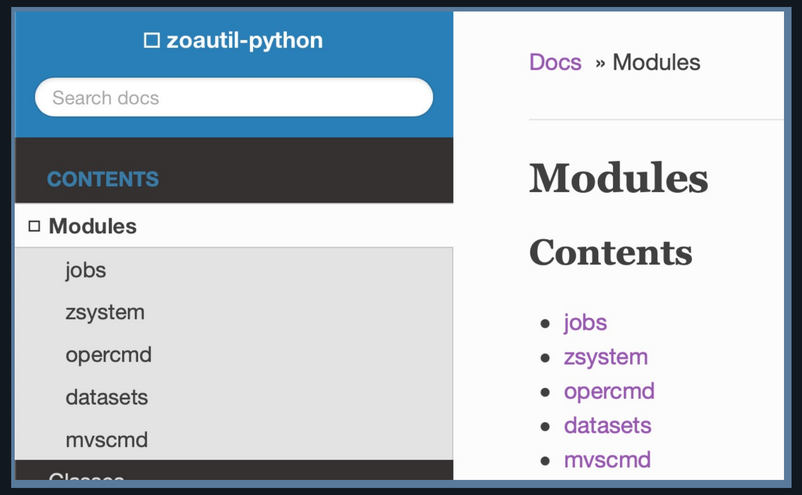
```
#!/usr/bin/python3
# Let's just import the datasets module from zoautils
from zoautil_py import datasets

# 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.
# The first member in many data structures is indexed at "0"
```

3. TUDO CERTO COM OS DATA SETS

Procure por **dslist.py** em /z/public/ e copie-o para seu próprio diretório pessoal. Você pode dar uma olhadinha no código para descobrir o que ele está fazendo.

É um script Python (se você não adivinhou pelo sufixo .py) e você pode ver que ele obtém uma lista de tudo em um data set específico e usa um loop for para imprimir cada nome de membro nele (pelo menos, ele irá, assim que você entrar e editar para apontar para um de seus data sets, na linha número 9). Faça essa mudança agora. Tente com seu data set JCL ou OUTPUT.

Simples e bacana, não é mesmo?



4. TEM UM PYTHON NO MEU Z

Escrever um script Python apenas para listar o que está em um data set pode parecer um exagero e, neste caso... sim, é. No entanto, queremos mostrar a você, em um programa muito simples, como isso pode ser feito. E agora que você vê, talvez você tenha algumas idéias.

Agora sabemos que podemos executar as principais tarefas do z/OS no código Python. Este exemplo envolveu apenas data sets. Você também pode trabalhar com jobs, o próprio sistema e dois tipos de comandos. Dê uma olhada na página de APIs do Python e leia mais sobre esses módulos AQUI.

```python
#!/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"):
```

5. INSERINDO ALGUMA LÓGICA

Em seguida, trabalharemos com um script que faz um pouco mais, incluindo a criação de um novo data set sequencial, a coleta de alguns dados e a gravação desses dados em nosso data set sequencial recém-criado.

Vamos começar com algumas dessas funções, e seu trabalho será fazer o resto delas funcionar sem dificulades.

Caso ainda não esteja em seu diretório pessoal, copie **members.py** de /z/public e dê uma olhada no código.

```
# These are comments
# They are here to help
```

6. FAÇA SEUS HACKS

Neste ponto, você pode estar vendo muito código e ficando nervoso(a). Aqui está o negócio... mesmo que você não queira ser um programador, você terá que gastar algum tempo olhando o código e fazendo pequenos ajustes no código de outras pessoas. Esse é o próprio espírito do hacking.

Muita codificação envolve descobrir por onde começar, reunir todos os recursos de que você precisa e criar algo do zero. Hackear é pegar algo que jã faz alguma coisa e fazer pequenas alterações para adicionar um recurso, corrigir problemas ou apenas deixar sua própria marca nele.

Sempre que há código envolvido, deixamos MUITOS comentários para ajudá-lo a descobrir o que está acontecendo e saber que, pelo menos para este desafio, você nunca precisará escrever mais do que algumas linhas de código novo. Descobrir o fluxo e a estrutura do que já existe provavelmente será a coisa mais difícil.

Você pegou essa!

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

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

No entanto, se você está acostumado a trabalhar com Linux e Python, talvez queira continuar escrevendo código da mesma maneira, enquanto ainda tem acesso à funcionalidade principal do z/OS. É aí que os módulos da biblioteca ZOAU são úteis.

Aprendemos novos métodos, além do que já conhecemos, para que tenhamos mais opções e mais oportunidades de fazer escolhas eficientes. Esses exercícios estão aqui para fornecer algumas ideias do que é possível quando você combina a capacidade de script e automação do Python com os recursos de backend do z/OS.

```
#!/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:
```

7. CARREGANDO MÓDULOS

Veja a linha #3 do arquivo **members.py**. Estamos escolhendo importar três módulos da biblioteca **zoautil.py**: datasets, jobs, e zsystem. Cada um desses módulos fornece algum conjunto de funções que agora podemos usar em nosso código.

Você pode ler tudo sobre os módulos e o que eles fazem aqui: https://www.ibm.com/docs/api/v1/content/SSKFYE_1.1.1/python_doc_zoautil/index.html?view=embed

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

8. PEGUE AQUELE DATA SET

As linhas 6-18 pedem ao usuário um nome de data set sequencial e atribui esse valor à variável dsname.
Se o data set já existir, usaremos ele e continuaremos. Se o data set não existir, podemos criá-lo para o usuário (o que faremos no próximo passo).

Se o data set NÃO existir e o usuário não quiser criá-lo, não há muito sentido em continuar o resto do código, então usamos *sys.exit* para sair do programa.

Até agora tudo bem? Certo, vamos começar a hackear algum código.

```
input("Data set not found. Should we crea
v.upper() == "Y"):
nts to create a file
 the part where we create a new data set
create(dsname,type="???",primary_space="1
t("Without a data set name, we cannot con

orrect line of code from the 4 lines belo
the system's linklist representation and
 the variable 'linklist_output'
```

9. NA SEQUÊNCIA CERTA

Para este desafio, usaremos este script (**members.py**) para coletar dados e gravá-los em um data set sequencial.

A linha #17 está usando a função create do módulo ***datasets*** para tentar criar este data set sequencial. 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 o ??? para criar um ***data set sequencial***.

Use ZXXXXXXX.COMPLETE como o data set sequencial quando solicitado pelo script. É aí que procuraremos validar seu trabalho, além da cópia de members.py.

(Lembre-se de usar seu ID de usuário em vez de ZXXXXX!).
Dica: Um data set sequencial é um tipo de data set. Outros tipos de data sets são KSDS, PDS, ESDS, mas neste caso, o tipo de data set que queremos criar é um data set sequencial.

KSDS? ESDS? PENSEI QUE SÓ TINHA PARTICIONADO E SEQUENCIAL?

Às vezes, um data set é um pouco mais do que um simples arquivo; um local para armazenar alguns dados que você deseja ler ou processar de cima para baixo. Membros do data set e data sets sequenciais funcionam muito 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 acessar esse registro do que ler todo o data set de cima para baixo. Nessas situações, podemos usar um *Key Sequenced Data Set* (KSDS). *Entry Sequenced Data Set* (ESDS) ou *Relative Record Data Set* (RRDS). Esses são todos exemplos de data sets *Virtual Storage Access Method* (VSAM) e os revisaremos mais tarde.
Ter opções quando se trata de acesso a dados significa que podemos escolher a melhor, mais rápida e mais eficiente maneira de trabalhar com todos esses bits e bytes. Entender as opções e como fazer tudo funcionar faz de você um profissional valioso do z/OS, e os empregadores definitivamente gostam desse tipo de coisa.

```
# 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)

# Write the value of linklist_info into our sequential
```

10. PEGUE A LISTA DE LINKS

Lembra como importamos alguns módulos no início deste script? Quando o sistema operacional z/OS é inicializado, ele faz a mesma coisa, carregando módulos e bibliotecas cheios de recursos que o usuário pode precisar usar.

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

De qualquer forma, saber como é a lista de links completa é um tanto quanto importante, bem no módulo *zsystem* do zoautil.py

Veja as linhas 20-27. Como você pode ver, existem quatro linhas de código lá (e muitos comentários caso você queira lê-los... o que você provavelmente deveria fazer).

Sua tarefa é identificar e descomentar a linha de código (de 24 a 27) essa é a linha correta que pegará a representação da linklist e a atribuirá à variável *linklist_output*.

```
#linklist_output = zsystem.link_linklist()
#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.
```

11. ÚLTIMA PARADA: ESCREVER

Até agora, criamos (ou reutilizamos) um data set sequencial e coletamos alguns dados. Vamos escrever esses dados no data set no final do nosso script.

Todas as informações estão lá, mas **algo** está errado e você precisará dar uma olhada no método datasets.write para descobrir o que é.

Depois de ter tudo arrumado e em ordem, rode o script e certifique-se de que ele funciona para você.

```
1  ['VENDOR.LINKLIB'
2   'SYS1.MIGLIB'
3   'SYS1.CSSLIB'
4   'SYS1.SIEALNKE'
5   'SYS1.SIEAMIGE'
6   'SVTSC.LINKLIB'
7   'LVL0.LOADLIB'
8   'LVL0.LINKLIB'
9   'SYS1.LINKLIB'
10  'SYS1.CMDLIB'
11  'SYS1.SHASLNKE'
12  'SYS1.SHASMIG'
13  'ISF.SISFLOAD'
14  'ISF.SISFLINK'
```

12. CONFIRA. FINALIZE.

Se você completou todas as etapas, você deve ter um data set sequencial ZXXXXX.COMPLETE que contém uma leitura da lista de links do sistema. Consulte a captura de tela acima se precisar de algum esclarecimento.

Encontre e envie CHKAUTO para marcar este como concluído. Você pode fazer isso do jeito que tem feito até agora, ou... aposto que há uma função para enviar JCL que pode funcionar bem.

De qualquer forma, parabéns por completar os desafios fundamentais! Você agora é capaz de ir cada vez mais longe.

BOM TRABALHO! VAMOS RECAPITULAR

Você criou um script de uma série de ações profundamente fundamentais do z/OS diretamente de um script Python! Mantivemos isso bastante simples neste exemplo, mas agora que você sabe como conectar os pontos, você deve ser capaz de criar qualquer número de novos utilitários que possam ser úteis no processamento de texto, verificação de saída de trabalho, verificação de alterações do sistema e trabalho com data sets.

Você sabia como fazer a maior parte disso antes, e agora você conhece outra maneira. Como dissemos, é bom ter mais opções.

A SEGUIR...

Você completou os desafios fundamentais!
Você tem o que é preciso para passar para os desafios avançados. Até agora, você provavelmente tem alguma ideia do que é de seu interesse e provavelmente encontrará isso na próxima série de desafios disponíveis.