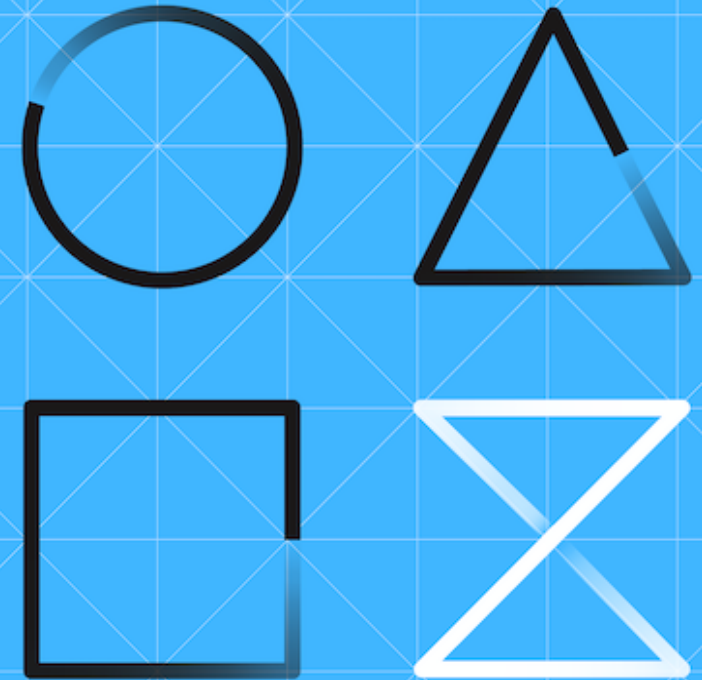


WRAPUP

Work Smarter

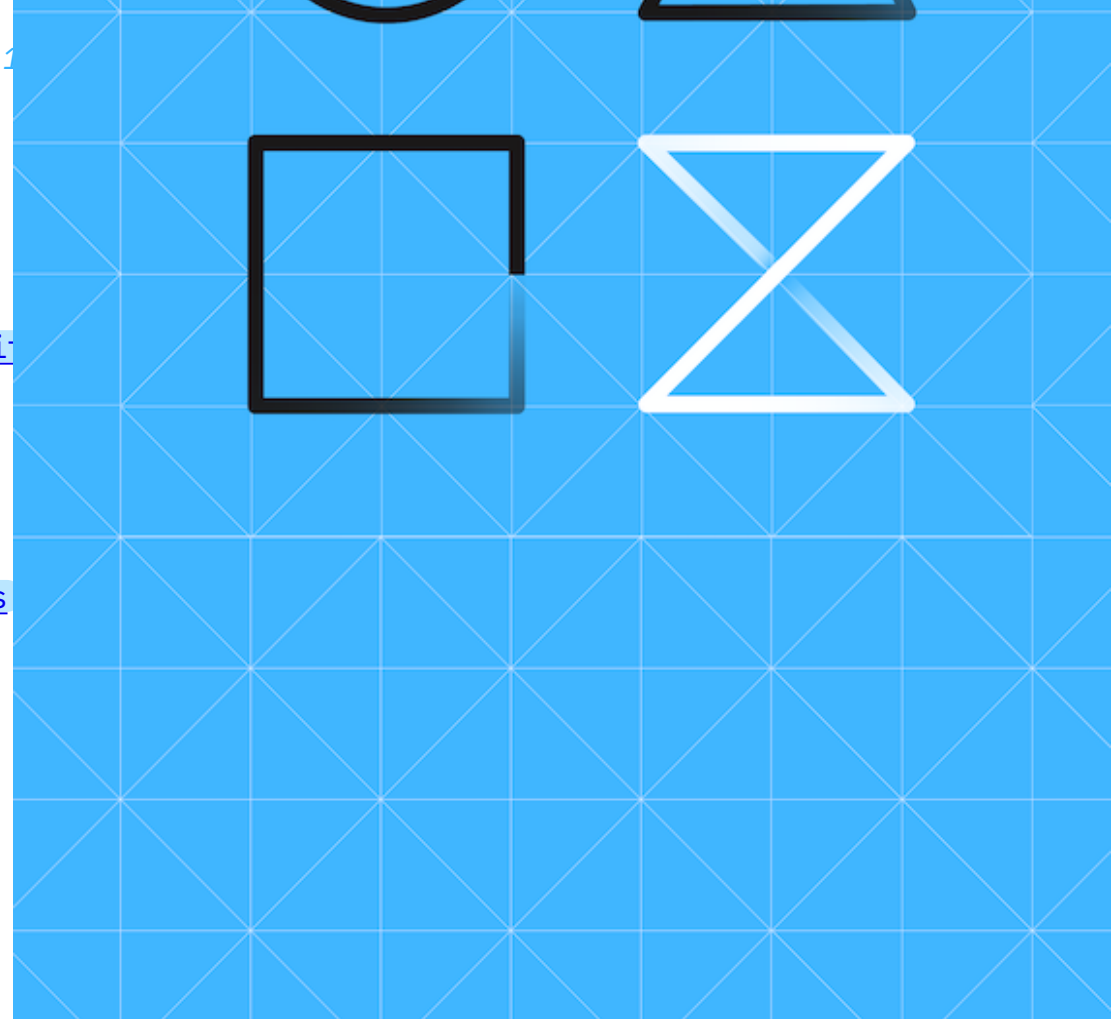
Automated Translation by Watson Language Translator



WRAPUP

Trabalhar Mais Inteligente

- Tempo para Trazer o Z Open Automation Utili
 - 0 desafio
 - Antes de começar
 - Investimento
- 1 decho ... decho ... decho
- 2 olá lá em baixo
- 3 todos configurados com conjuntos de dados
- 4 Garçom-há um python no meu z
- 5 inserir alguma lógica extra
- 6 Pegue o seu hack!
- 7 carregando módulos
- 8 obtenha esse conjunto de dados
- 9 Na sequência certa
- 10 pegue a lista de links
- 11 última parada: escrevendo para fora
- 12 Verifique; Finalize-o



TEMPO PARA TRAZER O Z OPEN AUTOMATION UTILITIES

O Desafio

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

Combine as principais qualificações do IBM Z que você adquiriu, aprendendo sobre conjuntos de dados e JCL, com as habilidades de script localizadas em Python, para automatizar algumas tarefas que um Programador de Sistema pode executar diariamente.

Antes De Começar

Nesse ponto, você sabe como trabalhar com conjuntos de dados, enviar tarefas e navegar pelo USS..

Você colocará tudo isso em uso aqui e encerrará seu entendimento fundamental do z/OS.

Investimento

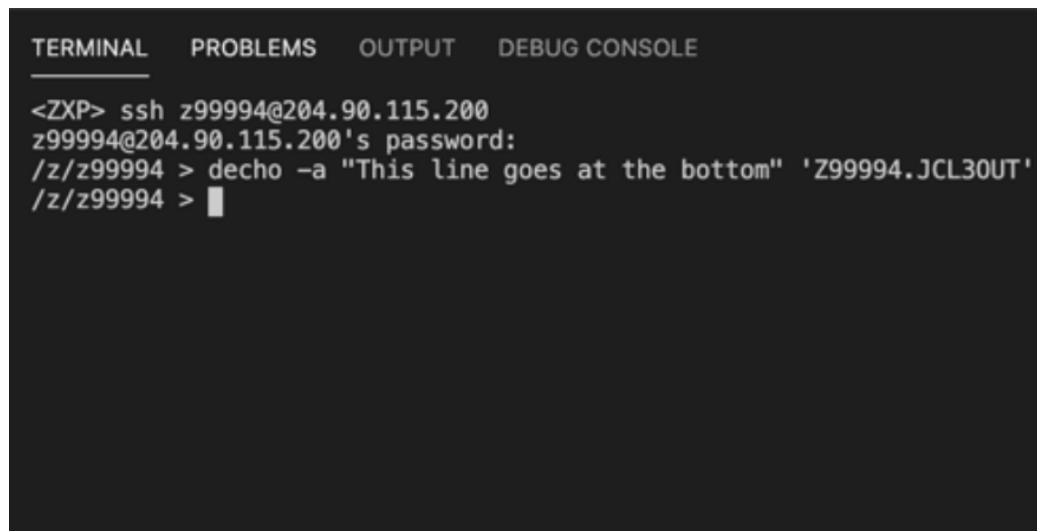
Etapas	Duração
12	90 minutos

1 DECHO ... DECHO ... DECHO

Você estará usando um conjunto de dados sequenciais para este desafio

Se você ainda tiver seu conjunto de dados sequenciais JCL30UT, use isso; caso contrário, faça um novo clicando com o botão direito do mouse no perfil de conexão do VSCode em 'Conjuntos de Dados ', selecionando "Criar Novo Conjunto de Dados" e seguindo os prompts.

Tudo pronto?

A screenshot of a terminal window with a dark background. At the top, there are tabs labeled 'TERMINAL', 'PROBLEMS', 'OUTPUT', and 'DEBUG CONSOLE'. The terminal shows the following text: '<ZXP> ssh z99994@204.90.115.200', 'z99994@204.90.115.200's password:', '/z/z99994 > decho -a "This line goes at the bottom" 'Z99994.JCL30UT'', and '/z/z99994 > ' followed by a cursor. On the right side of the image, there is vertical text: 'WPA/PUP [230301-1739]'.

```
TERMINAL  PROBLEMS  OUTPUT  DEBUG CONSOLE

<ZXP> ssh z99994@204.90.115.200
z99994@204.90.115.200's password:
/z/z99994 > decho -a "This line goes at the bottom" 'Z99994.JCL30UT'
/z/z99994 > █
```

Em seguida, execute SSH no sistema mainframe novamente para que você obtenha um shell USS e, em seguida, insira a versão correta do comando a seguir, substituindo seu próprio ID do usuário e o conjunto de dados que você deseja usar.

```
decho -a "This line goes at the bottom" 'Zxxxxx.JCL30UT'
```

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

2 OLÁ LÁ EM BAIXO

Editar o conjunto de dados JCL30UT no VSCode. Se você o criou novamente, pode ser necessário clicar com o botão direito do mouse nele e selecionar "Puxar do Mainframe"; isso garante que você tenha a versão mais recente absoluta em sua sessão de edição.

```
*****  
Peak fares are charged during business  
weekday train scheduled to arrive in NYC  
6 a.m. and 10 a.m. or depart NYC termin  
and 8 p.m. On Metro-North trains, peak  
travel on any weekday train that leaves  
between 6 a.m. and 9 a.m.  
Off-peak fares are charged all other tim  
day Saturday and Sunday, and on holidays  
This line goes at the bottom
```

Você deveria ver a linha que você apenas `decho` d anexado à parte inferior do conjunto de dados..

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

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

WPA/PUP [230301-1739]

3 TODOS CONFIGURADOS COM CONJUNTOS DE DADOS

Certifique-se de ter uma cópia de **dslist.py** em sua casa, no 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.
```

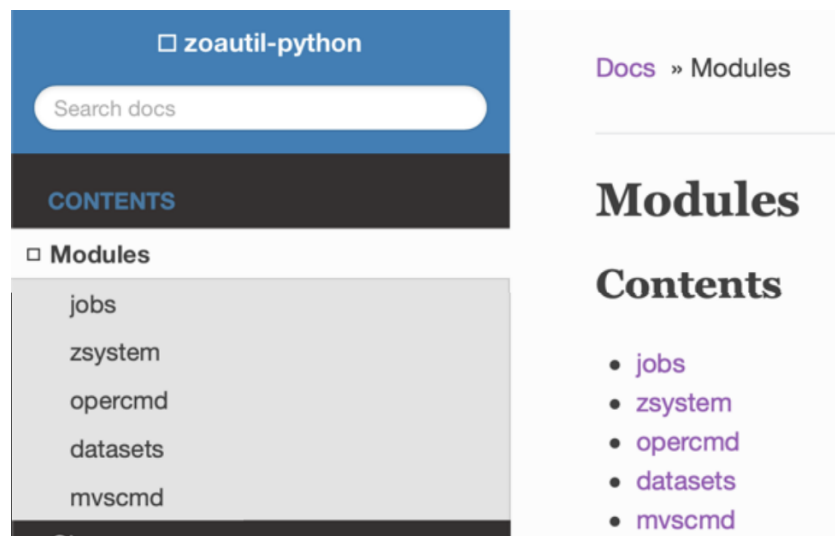
É um script Python (se você não adivinhou pelo sufixo .py) e você pode ver que ele obtém uma listagem de todos os membros em um determinado conjunto de dados, em seguida, usa um 'loop for' para imprimir cada nome de membro nele.

Pelo menos, ele irá, uma vez que você tenha editado o script para apontar para um de seus conjuntos de dados particionados na linha número 9 (a linha que começa `members_list =`). Tente agora, especificando o conjunto de dados JCL ou OUTPUT.

Salve o arquivo e execute-o a partir de um login de shell do USS

Agradável e simples, certo?

4 GARÇOM-HÁ UM PYTHON NO MEU Z



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

No entanto, com um programa muito simples, você pode ver como isso pode ser feito.

E uma vez que você vê e entende, talvez isso lhe dê algumas ideias.

Agora que você sabe como é possível executar tarefas principais do z/OS no código Python. Este exemplo envolvia apenas conjuntos de dados Também é possível trabalhar com tarefas, o próprio sistema e dois tipos de comandos.

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

(este local muda-se-se você precisar encontrar uma versão diferente, use uma pesquisa de internet para "zoau python classes de referência")

5 INSERIR ALGUMA LÓGICA EXTRA

Agora você pode trabalhar com um script que faz um pouco mais, incluindo a criação de um novo conjunto de dados sequenciais, reunindo alguns dados e, em seguida, gravando esses dados em seu conjunto de dados sequenciais recém-criado.

```
#!/usr/bin/python3
# Let's just import the datasets module from zoautils
from zoutil_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 found it, and we'll use it. Otherwise
if (datasets.exists(dsname) == True):
    print("Data set found! We will use it")
else:
    create_new = input("Data set not found. Should we create it? (Y/N): ")
    if (create_new.upper() == "Y"):
```

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

Abra seu arquivo 'members.py' em seu diretório inicial e dê uma olhada na origem.

6 PEGUE O SEU HACK!

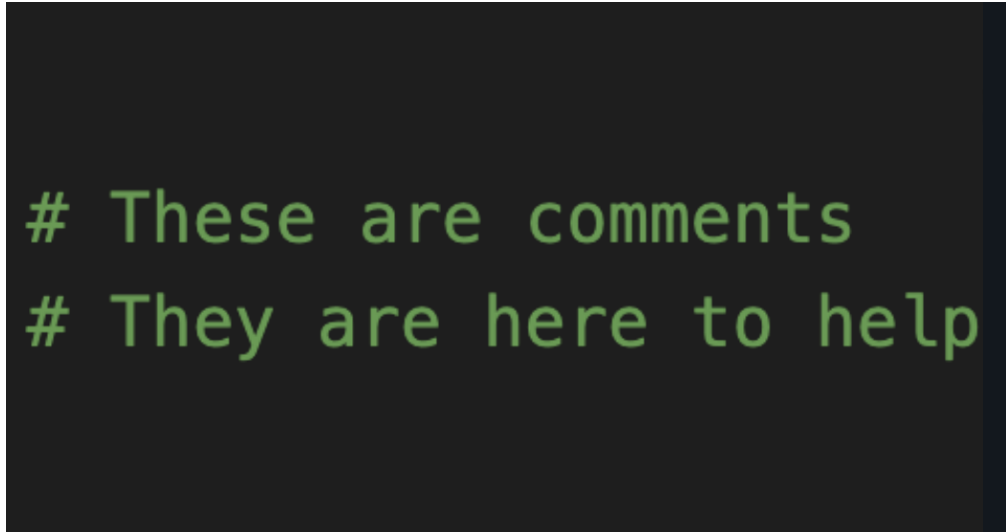
Neste ponto, você pode estar vendo muito código e ficando nervoso.

Aqui está o negócio ... mesmo que você não queira ser um programador, se você trabalha com sistemas, você terá que passar algum tempo olhando o código e fazendo pequenos ajustes nos programas de outras pessoas.

Esse é o próprio espírito de hackear.

Um monte de codificação envolve descobrir por onde começar, juntar todos os recursos que você precisa, e criar algo do zero.

Hacking é sobre tomar algo que já faz algo útil e fazer pequenas mudanças para adicionar um recurso, corrigir problemas, ou apenas deixar sua própria pequena marca nele.



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

Sempre que há código envolvido, há LOTS de 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 novo código.

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

Mas não se preocupe-você tem isso!

POR QUE APRENDER OUTRA MANEIRA? PORQUE É BOM TER OPÇÕES.

Se você começou a trabalhar no z/OS anos atrás, as chances são que você está muito confortável com a JCL e os muitos métodos já construídos no 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 maneira, enquanto ainda tem acesso à funcionalidade principal do z/OS. É aí que os módulos da biblioteca ZOAU são úteis.

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

Esses exercícios estão aqui para fornecer algumas ideias do que é possível ao combinar a capacidade de script e automação do Python com os recursos de backend do z/OS.

7 CARREGANDO MÓDULOS

```
#!/usr/bin/python3
# Let's just import the datasets module from zoaut
from zoutil_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:
```

Consulte a linha #3 de **members.py** .

Três módulos estão sendo importados da biblioteca 'zoutil.py': conjuntos de dados, tarefas e zsystem.

Cada um desses módulos fornece um conjunto de funções que agora 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 OBTENHA ESSE CONJUNTO DE DADOS

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

#if it exists, say we found it, and we'll use it. Otherwise
if (datasets.exists(dsname) == True):
    print("Data set found! We will use it")
else:
    create_new = input("Data set not found. Should we create it?")
    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=1)
    else: sys.exit("Without a data set name, we cannot continue")
```

Linhas 6-18 de **members.py** solicitar ao usuário um nome de conjunto de dados sequencial e designar esse valor à variável 'dsname'.

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

Se o conjunto de dados não *não* existe, você pode criar isso para o usuário (que você fará na próxima etapa).

Se o conjunto de dados NÃO existe, e o usuário diz que não quer criá-lo, então não há muito sentido em continuar o resto do código, então 'sys.exit ()' é usado para sair do programa.

Até agora, tão bom? Certo, vamos começar a hackear algum código ...

9 NA SEQUÊNCIA CERTA

Para este desafio, você estará usando o script (**members.py**) para reunir dados e gravá-los em um conjunto de dados sequenciais

```
input("Data set not found. Should  
v.upper == "Y"):  
nts to create a file  
the part where we create a new da  
create(dsname,type="???",primary_s  
t("Without a data set name, we can  
  
orrect line of code from the 4 lin  
the system's linklist representati  
the variable 'linklist_output'
```

A linha #17 está usando a função de criação do módulo 'datasets' do zoau para tentar criar esse conjunto de dados sequencial. Mas como é agora, não vai funcionar até que você faça um pequeno ajuste. Dê uma olhada no *datasets.create()* e trabalhar para fora o que precisa substituir o **? ??** para criar um conjunto de dados sequencial.

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

Sugestão: Um conjunto de dados sequencial é um tipo particular de conjunto de dados... Outros tipos de conjuntos de dados são KSDS, PDS, ESDS, mas, neste caso, o tipo que você precisa criar é um conjunto de dados sequencial. Verifique a documentação e os comentários para obter ajuda

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

Às vezes, um conjunto de dados é pouco mais do que um arquivo simples; um lugar para conter alguns dados que você deseja ler ou processar de cima para baixo. Membros do conjunto de dados particionados e 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 de Chave Sequenciados* (KSDS), *Conjunto de Dados Sequenciados de Entrada* (ESDS) ou Relative Record Data Set (RRDS). Esses são todos os exemplos de conjuntos de dados do Virtual Storage Access Method (VSAM) e serão cobertos em desafios posteriores.

Ter opções quando se trata de acesso a dados significa que o desenvolvedor ou programador de sistema pode 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 torna você um profissional valioso do z/OS, e os empregadores definitivamente gostam desse tipo de coisa.

10 PEGUE A LISTA DE LINKS

Lembra como você importou alguns módulos no início deste script? Quando o sistema operacional z/OS é inicializado, ele faz a mesma coisa, carregando módulos e bibliotecas cheias dos 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 um usuário apenas digite um único comando em vez de ter que referenciá-lo por seu caminho completo a cada vez.

Muito útil!

De qualquer forma, saber como é a lista de links completa é meio importante, e está disponível diretamente no módulo zsystem do zoutil.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)
```

Veja as linhas 20-27. Como você pode ver, existem quatro linhas de código lá (e um monte de comentários que você provavelmente deveria ler).

Sua tarefa é identificar e remover o comentário de uma linha de código (de 24-27) que é a linha correta que capturará a representação linklist e a designará à variável 'linklist_output'.

Nota : o link para a documentação on-line no código pode estar desatualizado; você vai encontrar isso muito-comentários ficam desatualizados, mas nem sempre são atualizados com as mudanças de código.

11 ÚLTIMA PARADA: ESCRREVENDO PARA FORA

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

```
#linklist_output = zsystem.linklist_info()
#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 challenge
# 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 sua documentação, para descobrir o que precisa ser corrigido.

Depois de ter tudo resolvido e em ordem, execute o programa e certifique-se de que o script funciona corretamente para você.

12 VERIFIQUE; FINALIZE-0

```
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'
```

Ao concluir todas as etapas necessárias, você deve ter um conjunto de dados sequencial Zxxxxx.COMPLETE que contenha uma listagem da linklist do sistema. Consulte a captura de tela acima se precisar de esclarecimento.

Localize e envie o **CHKAUTO** tarefa para marcar este desafio como concluído

Você pode fazer isso da maneira que você tem feito até agora através do VSCode, mas eu aposto que há uma função Python para enviar JCL que pode funcionar bem, também.

De qualquer forma, parabéns por completar todos os desafios Fundamentals!

Você está pronto para ir maior e mais longe.

Bom trabalho-vamos recapitular	Em seguida ...
<p>Você fez o script de uma série de ações profundas fundamentais do z/OS diretamente de um script Python!</p> <p>Nós 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 podem ser úteis no processamento de texto, verificação de saída de tarefa, verificação de mudanças do sistema e trabalho com dados. conjuntos.</p> <p>Você sabia como fazer a maior parte disso antes, e agora você sabe de outra maneira. Como dissemos, mais opções são boas para se ter.</p>	<p>Você completou todos os desafios fundamentais! Você tem o que é preciso para avançar para os desafios avançados. Até agora, você provavelmente tem alguma ideia do que você está interessado, e você provavelmente vai encontrá-lo na próxima série de desafios disponíveis.</p>