MAINFRAME SOCIETY

UNLEASH YOUR POTENTIAL

When they open Powerpoint and you see slide “1 out of 243”

Snakes & SysProgs

Henri Kuiper / the Wizard of z/OS

GSE-UK Security Workinggroup Meetup
February 27th, Winnersh, UK

Notices and disclaimers

This document is distributed “as is,” just like an untested JCL job—proceed at your own risk. We provide zero warranties, implied or express.

If your LPAR crashes, your batch job goes rogue, or your coffee machine stops working, we are not responsible. This includes, but is not limited to, loss of datasets, CICS meltdowns, REXX scripts spiraling out of control, or any profit that evaporates into thin air faster than your 4HRA on a misconfigured z/OS.

Customer examples are like carefully optimized assembler code: they show how some sysadmins have conquered the mainframe, but your mileage may vary.

Your results might differ based on your operating system, skill level, or whether you still run on COBOL from 1974.

Workshops and sessions may be crafted by independent mainframe gurus who dwell in the deepest corners of their data centers. Their ideas may not represent the views of Mainframe Society, especially if they suggest using punch cards (please, no).

Oh, and not all offerings are available in every galaxy. So if you're running on a legacy server in a parallel universe—sorry, no Mainframe Society support for you!

Any statements about future directions or plans for world domination with Mainframe Society are purely speculative, subject to spontaneous quantum shifts, and could be withdrawn at warp speed with no prior notice.

Mainframe Society, the Mainframe Society logo, and MainframeSociety.com are proudly trademarked in multiple realities. Other product names might belong to us or some other tech overlords from the 21st century.

Live long and compile!

My (big) promise for today…

At the end of this session, you'll have another (better?, faster?, cooler?) way to analyze and work with your RACF unloads…

For those of you that don’t know me…

This is me

IBM Champion 2025

Awarded through the program

IBM Champion

GSE GUIDE SHARE EUROPE

MFPANDAS

Belastingdienst

Email: henri@mainframesociety.com | wizard@zdevops.com

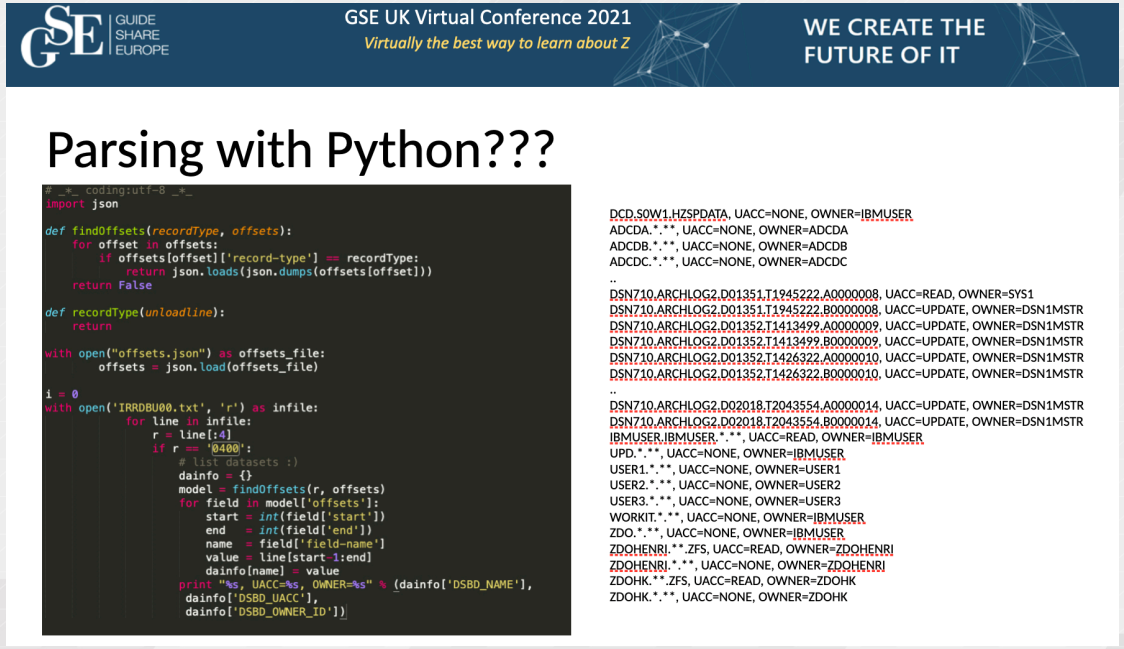
LinkedIn: https://www.linkedin.com/in/wizardofzos/

GitHub: https://github.com/wizardofzos

MS-Profile: https://mainframesociety.com/users/7138614

Credly: https://www.credly.com/users/wizardofzos

This story started a long time ago…



Yes, typo :)

I’ve got my RACF setup in python dictionaries!!

- Not all the record types available yet
- WorksOnMyMachine™…
  - Parsing my ZPDT unload took about 5mins
  - Guess how long it took for a ‘real’ RACF unload?
  - You’re wrong :)
- So things needed done to make it work (better)
- But why did I need this to work?

Frustration

Question(s)…

How long does it take you to extract *meaningful* reports from your RACF environment?

And share them (with auditors?) in a format they can work with?

Or generate corrective actions based on your findings?


The Roku Channel
It takes time.


WOW. THAT IS A HECK
OF A LONG TIME.


I'LL BE THERE
SOON
MOMMY


it's been 84 years...


IT FEELS LIKE FOREVER!


©bombaysoftwares
STILL WAITING...


STILL WAITING


The struggle is Real !
MakeAGIF.com


AS LONG AS IT TAKES

There’s nothing wrong with CARLa, but…


```
newlist type=racf title="Profiles where IBMUSER is on access list"
 define acl subselect acl(id=ibmuser)
 s acl(id=ibmuser)
 sortlist class profile acl(aclaccess,7,"Access")

P R O F I L E   L I S T I N G   12 Aug 2008 00:20
Profiles where IBMUSER is on access list

Class     Profile key                                    Access
ACCTNUM   **                                             ALTER
DATASET   CBC.**                                         NONE
DATASET   CRMQARUN.ACCESS.B.**                           READ
DATASET   CRMQARUN.NOACCESS.B.**                         NONE
FACILITY  $CNF.RACF                                      ALTER
FACILITY  CKF.RACF                                       ALTER
STARTED   BLSJPRMI.*                                     ALTER
STARTED   CATALOG.*                                      ALTER
STARTED   CIC410A.*                                      ALTER
```

…it doesn't really work for me

- *Usually I take too long to 'craft the perfect CARLa'*
- *Once the results are there, I'm spending too much time creating an XLS or other 'sharable' display of my findings.*
- *When I needed an update to the report, that pain starts all over again*
- *Provided you've zSecure/Consul available on the system*

First Version (pyRACF)

The first release of PyRACF

pyracf 0.0.1

Newer version available (0.8.8)

```
pip install pyracf==0.0.1
```

Released: Apr 9, 2022

Parsing IRRDBU00 unloads in panda dataframes.

Navigation

- Project description
- Release history
- Download files

Verified details

These details have been verified by PyPI

Project description

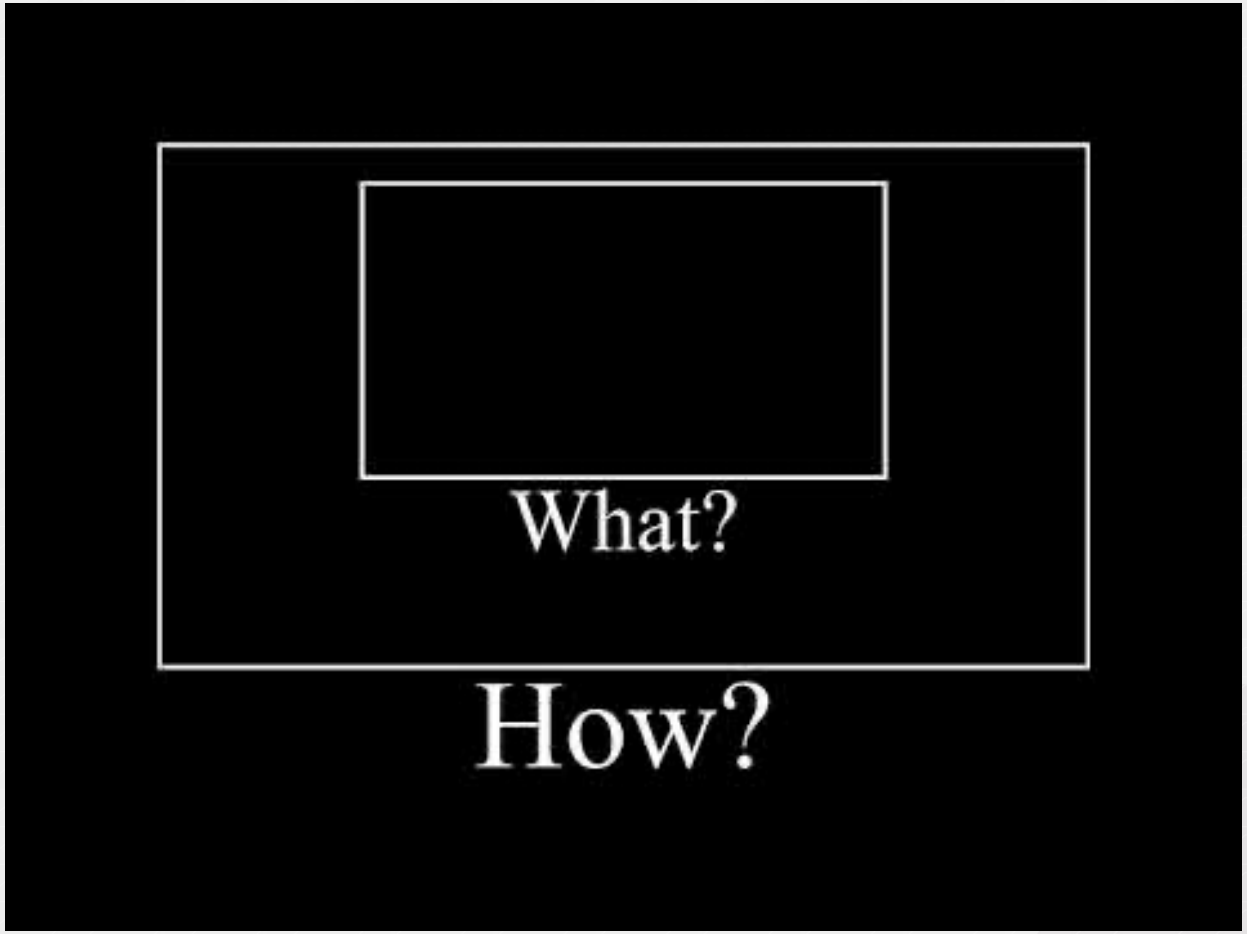
pyracf

parsing IRRDBU00 unloads like a boss

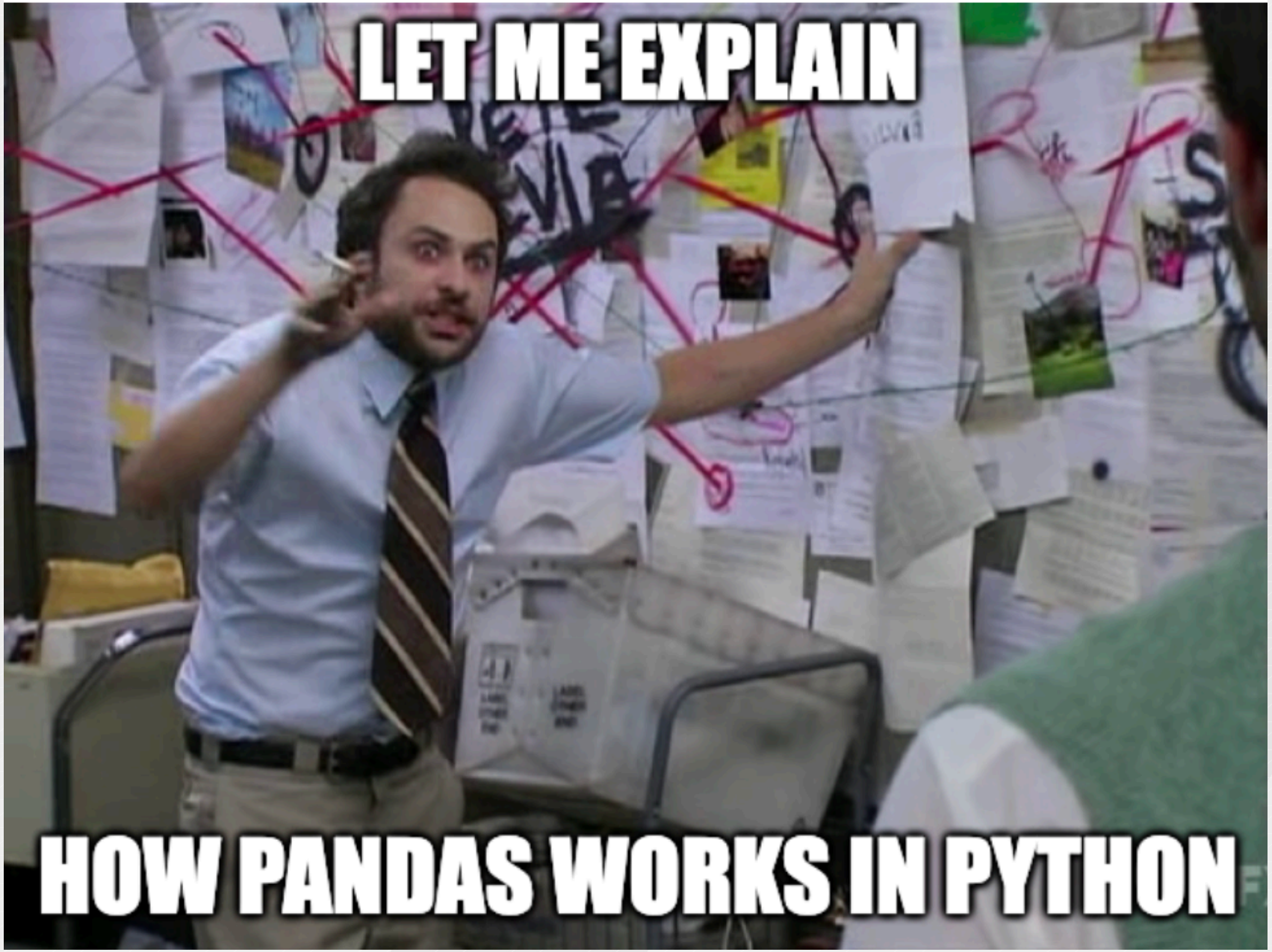
```
from pyracf import RACF
RACF.parse('path/to/irrdbu00')
RACF.users.info()
```

- ☑ *Easy selection of various RACF ‘facts’ using python and Pandas DataFrames*
- ☑ *Easily create XLS-reports*
- ☑ *Make it repeatable*
- ☑ *Get all the possibilities of python power*

Panda DataFrames you say?



- *A* **Python library** *for* **data analysis** – *Think Excel but with superpowers.*
- *Works with* **structured data** – *Rows, columns, and tables (like a spreadsheet).*
- **Fast and flexible** – *Filter, sort, and analyze large datasets easily.*
- *Supports* **multiple formats** – *Works with CSV, Excel, JSON, and more.*
- *Used in* **data science & finance** – *Trusted by industries handling big data.*
- **Lightweight & open-source** – *Free, widely used, and community-driven.*


LET ME EXPLAIN
HOW PANDAS WORKS IN PYTHON

So how does it parse the IRRDBU00 unload?

- *Automatically generated ‘offsets.json’ based on the online IBM documentation.*
- *Describing every record type and every field in it*
- *Parser is fully dynamic, changes in json, result in changes in the ‘DataFrames’.*

So how does it parse the IRRDBU00 unload?

```
urls = [
"https://www.ibm.                                                    00/format.htm",
"https://www.ibm.                                                    00/usr.htm",
"https://www.ibm.                                                    00/dsr.htm",
"https://www.ibm.                                                    00/grr.htm"
]
```

```
for wtype in w.find_all(["h2","h3"]):
  try:
    [rdesc,rtype,*_] = re.split("[\(\)]",wtype.string)
  except:
    print("Funny header:",wtype.string)
  else:
    rdesc = re.sub("\s"," ",rdesc)  # newlines in description...
    print(rtype,":",rdesc)
    rdesc = rdesc.strip().lower().replace(" ","-")
    wtable = wtype.find_next_sibling().find("tbody")
    rfields = []
    for wrow in wtable.find_all("tr"):
      wfields = wrow.find_all("td")
      # some <td>s contain <svg> tags for changes, so the string is not the only descendant of <td> a
      wf = [re.sub("\W","",str(list(wfields[i].strings)[0])) for i in range(4)]  # remove strash that
      wf.append(re.sub("[\s\u00ae]"," ",str(list(wfields[4].strings)[0])))  # remove newlines and (R)
      rfields.append({
          "field-name": wf[0] if wf[0]!="" else "RESERVED",
          "type": wf[1],
          "start": wf[2],
          "end": wf[3],
          "field-desc": wf[4]
      })
    model.update({
      rdesc: {
        "record-type": rtype.upper(),  # 05k0 is in fact 05K0
        "ref-url": url,
        "offsets": rfields
```

https://github.com/wizardofzos/mfpandas/blob/main/src/mfpandas/irrdbu00-offsets.json

```
{
    "group-basic-data-record": {
        "record-type": "0100",
        "ref-url": "https://www.ibm.com/docs/en/SSLTBW_3.1.0/com.ibm.zos.v3r1.icha300/format.htm",
        "offsets": [
            {
                "field-name": "GPBD_RECORD_TYPE",
                "type": "Int",
                "start": "1",
                "end": "4",
                "field-desc": "Record type of the Group Basic Data record (0100)."
            },
            {
                "field-name": "GPBD_NAME",
                "type": "Char",
                "start": "6",
```

```python
# load irrdbu00 field definition
# strictly speaking only needed
with importlib.resources.open_te
    _offsets = json.load(file)
for offset in _offsets:
    rtype = _offsets[offset]['re
    if rtype in _recordtype_info
      _recordtype_info[rtype].up
try:
    del file, rtype, rinfo, offs
except NameError:
    pass
```

```python
with open(self._irrdbu00, 'r', encoding="utf-8", errors="replace") as infile:
    for line in infile:
        lineno += 1
        r = line[:4]
        # check if we can support this recordtype
        if r not in self._recordtype_info.keys():
            self.errors.append(f"Unsupported recordtype '{r}' on line {lineno} ignored.")
            continue
        if r in self._records:
            self._records[r]['seen'] += 1
        else:
            self._records[r] = {'seen': 1, 'parsed': 0}
        try:
            offsets = IRRDBU00._recordtype_info[r]["offsets"]
        except:
            offsets = False
        if offsets:
            irrmodel = {}
            for model in offsets:
                start = int(model['start'])
                end   = int(model['end'])
                name  = model['field-name']
                value = line[start-1:end].strip()
                irrmodel[name] = str(value)
            self._parsed[r].append(irrmodel)
            self._records[r]['parsed'] += 1
```

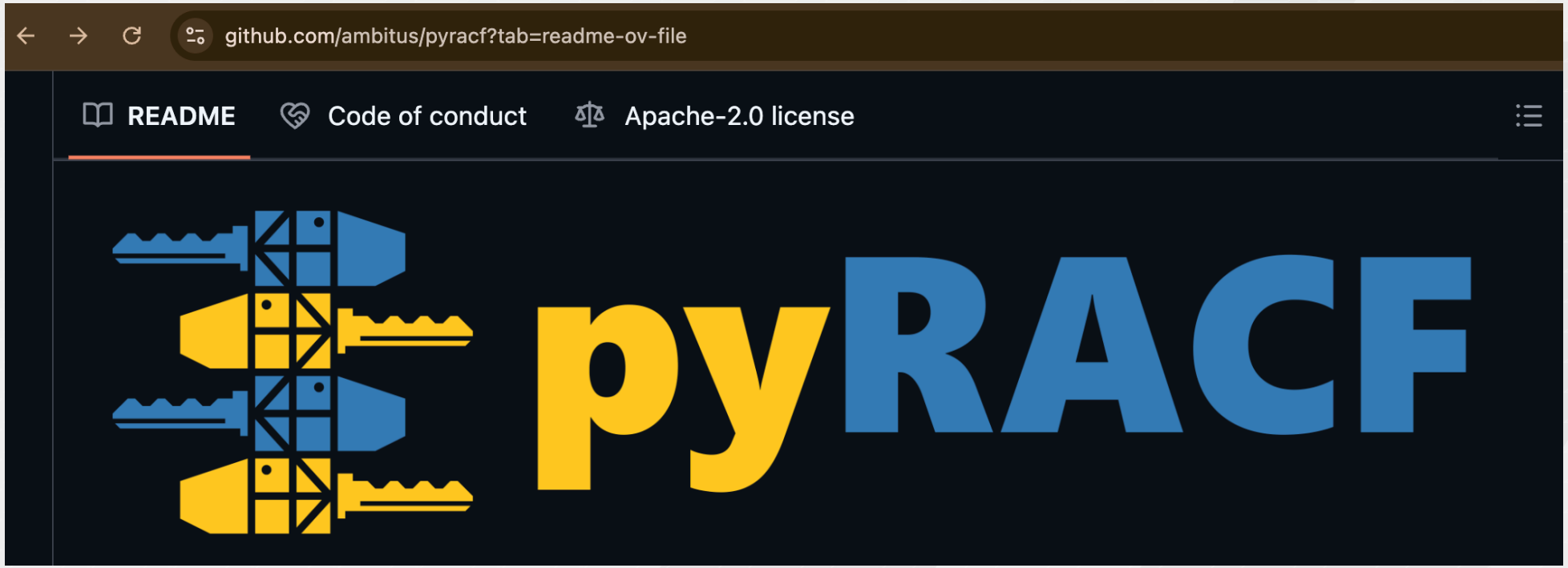
Feature Creep?

But I wanted more features…

- *Don't want to code the same queries over and over*
  - *Who are my 'special' users? Do I have any orphan definitions? Groups without members?*
- *It needs some documentation too*
- *So this is the access list for SOME.ARCANE.PROFILE.**, but what actual datasets is this protecting?*
- *What about my SETROPTS settings?*
- *I needed a 'broader' framework than 'just' RACF*

Plus another pyracf was also being developed…

SHARE
August 2023


github.com/ambitus/pyracf?tab=readme-ov-file
README
Code of conduct
Apache-2.0 license
pyRACF

Futureproof it: MFPandas

What is MFPandas?

For decades, z/OS has generated vast amounts of critical system data, storing it in structured datasets. Security logs, access controls, and storage records are all captured in formats that are essential for managing and auditing mainframe environments. Traditionally, if you wanted to extract insights from this data, you'd rely on tools like DFSort or ICETOOL—robust, but rigid solutions that require extensive JCL scripting and careful formatting to generate meaningful reports. While effective, these approaches are slow, inflexible, and not easily adaptable to modern analytics needs.

But what if you could query and analyze this data dynamically, without writing JCL, and export it effortlessly into formats like Excel or CSV? Enter MFPandas—a modern, Python-powered solution that brings flexibility and efficiency to mainframe data analysis. Instead of predefined report structures and cumbersome batch jobs, MFPandas allows you to interactively explore RACF and DCOLLECT data, filter results in real time, and apply advanced data transformations with just a few lines of Python code.

By leveraging pandas, a powerful data manipulation library widely used in finance, machine learning, and data science, MFPandas turns static mainframe reports into live, interactive datasets that can be sorted, filtered, and exported in seconds. No more waiting for batch jobs to run. No more struggling with rigid report formats. Just fast, flexible, and insightful security analysis at your fingertips.

MFPandas isn't just about modernization—it's about empowering security teams and mainframe professionals with tools that keep up with today's data-driven world. Whether you're identifying orphaned datasets, auditing user access permissions, or detecting unusual security patterns, MFPandas makes it easier, faster, and far more efficient than traditional methods.

So why stick to legacy tools that slow you down? With MFPandas, you can unlock the full potential of z/OS security data—all with the power and simplicity of Python.


I'M GONNA SHOW YOU SOME STUFF

Future Features being thought of…

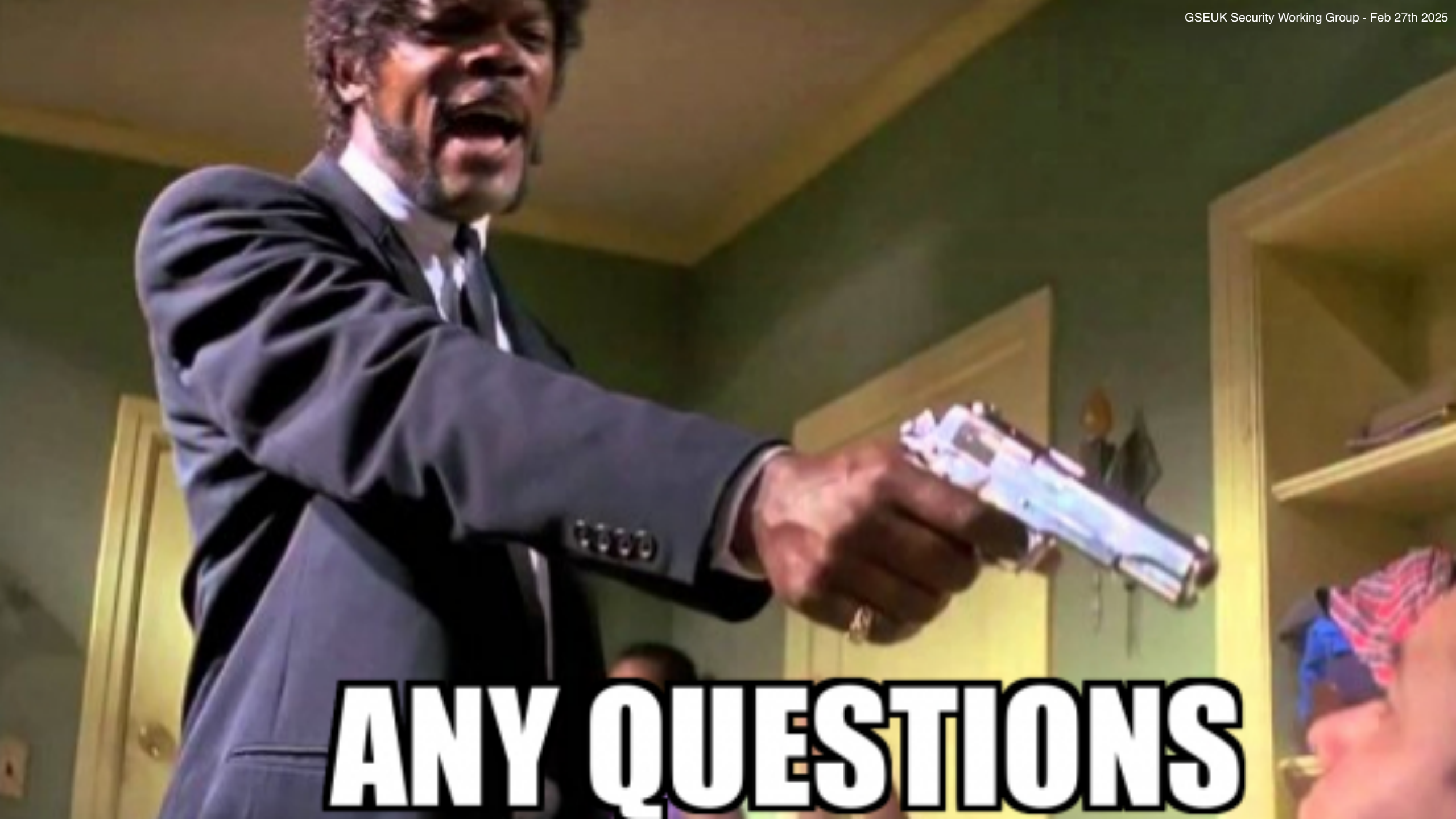
- *RMM Extract data?*
- *Catalog data?*
- *IODF?*
- *Top Secret / ACF2 unloads?*
- *SMF80? 30? 120?*

You might have ideas?
Found bugs?
Why not contribute?

Links

Documentation: *https://mfpandas.readthedocs.io*
GitHub: *https://github.com/wizardofzos/mfpandas*
PyPi: *https://pypi.org/project/mfpandas/*
Demo Repo: *https://github.com/wizardofzos/gseuk-winnersh-25*

Pandas Documentation: *https://pandas.pydata.org/docs/index.html*


ANY QUESTIONS


Thank You For Your Attention!
Presentation is over!