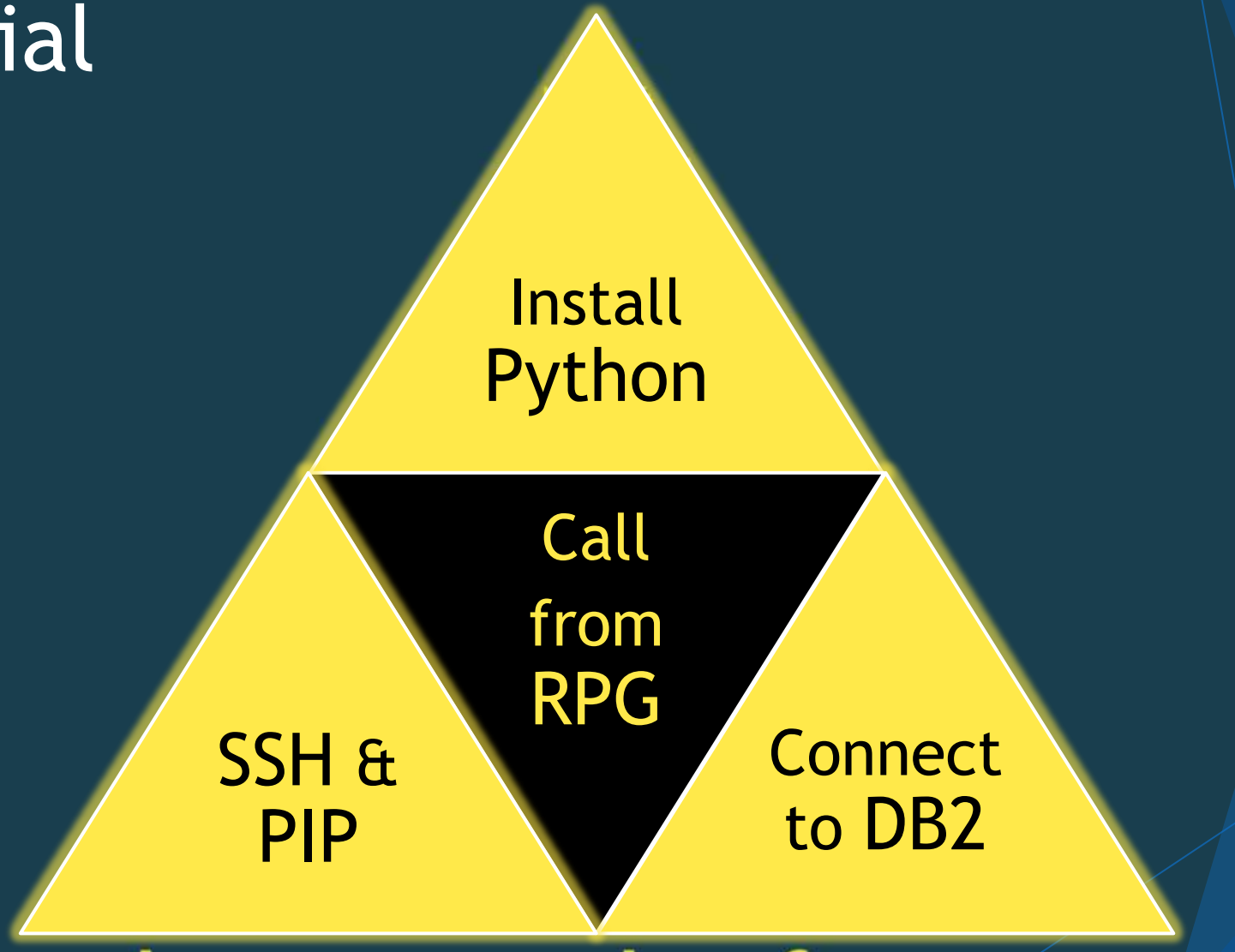
# Level Up Your RPG with Python

Joseph Wright
@Wright4i

# Why *SHOULD* you care?

- It's EASY
  - Stop reinventing the wheel (or saying I can't) when you should start saying I can… in Python!
- It's FREE
  - The opensource community behind Python is incredibly strong and very open. With hundreds of thousands of free packages available you can usually find a quick solution to your problem in a package.
- It's COOL
  - And you will be too.

# The Tutorial

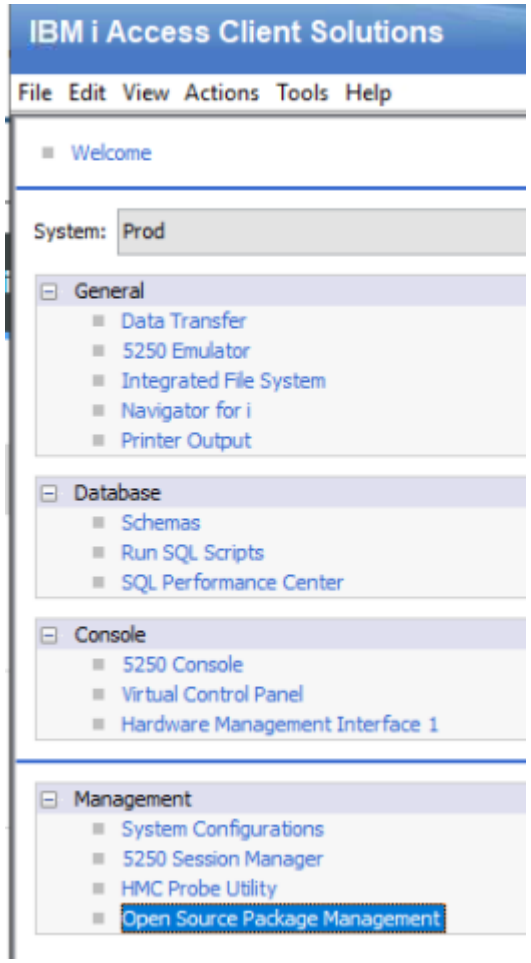


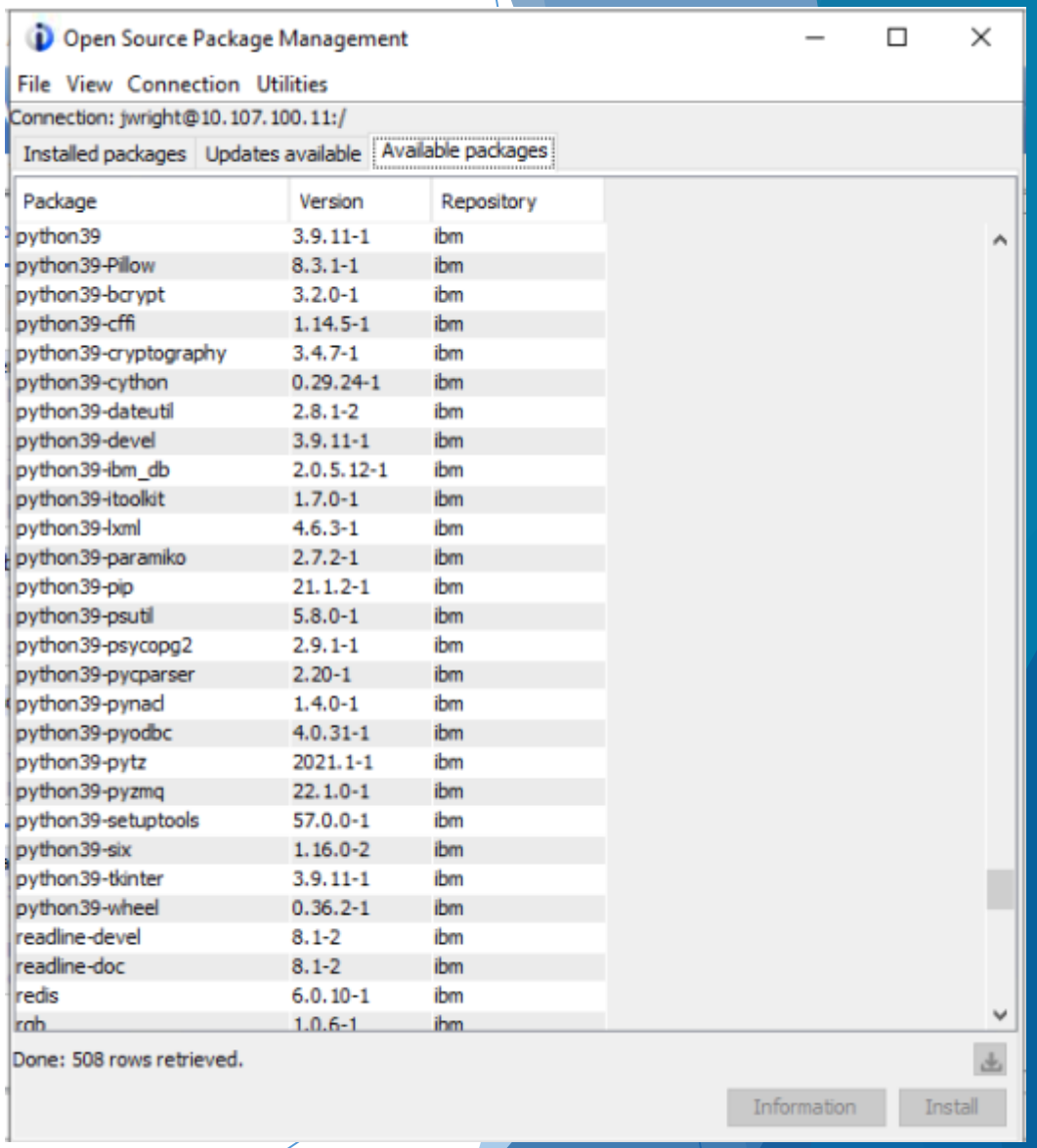

The Legend Of Python

# Installing Python

## Use the Open Source Package Manager in ACS

- Update your ACS to the latest version
- Click Open Source Package Management
- Sign in using SSH. (STRTCPSVR *SSHD if necessary).
- Go to Available Packages
- Select python39 and click install.



# Using SSH and PIP

## Git SCM (Git for Windows)

- Bourne Again SHell (aka BASH)
- ssh **your_server**
- Or ssh **ibmiuser@your_server**

## Using PIP

- PIP is the package manager for python packages.
- The command will be pip3 for python3
- pip3 install **package_name**

MINGW64:/c/Users/wrightj07

```
$ pwd
/home/JWRIGHT
$ python3 --version
Python 3.6.10
$ pip3 install python-fire
Collecting python-fire
  Downloading python_fire-0.1.0-py2.py3-none-any.whl (14 kB)
Collecting pygments
  Downloading Pygments-2.14.0-py3-none-any.whl (1.1 MB)
     |###                                | 122 kB 402 kB/s eta 0:00:03
```

# Connecting to DB2 from Python

```python
import ibm_db_dbi as dbi

conn = dbi.connect()
cur = conn.cursor()
cur.execute('SELECT * FROM SYSIBM.SYSDUMMY1')

for i, row in enumerate(cur, start=1):
    print(row)

# Expected Output ('Y',)
```

# Running Python from CL/RPG (Quick and Dirty)

```
CHGVAR      VAR(&CMD) VALUE('/QOpenSys/pkgs/bin/python3 +
              /Path_To_Your_Source/myprogram.py +
              > /dev/null')

QSH         CMD(&CMD)
```

```
sCmd = 'QSH CMD(''/QOpenSys/pkgs/bin/python3 ' +
        '/Path_To_Your_Source/myprogram.py > /dev/null'')';
system(sCmd);
```

# Running Python from RPG

# (Unix CMD, Chroot, Proc in SRVPGM)

# Pt 1

```
// ---------------------------------------------------------------------------
// Call_Python
// - Takes a script, up to 20 args (at 128 char each), and a chroot environment
// ---------------------------------------------------------------------------
 dcl-proc Call_Python export;
     dcl-pi Call_Python ind;
         script char(256)  const;
         args   char(2560) const options(*nopass:*omit);
         env    char(256)  const options(*nopass:*omit);
     end-pi;

     dcl-f unix disk(1000) handler('UNIXCMDOA': unixCmd) usropn;

     dcl-ds record len(1000) end-ds;

     dcl-ds args;
         arg char(128) dim(20);
     end-ds

     dcl-s unixCmd       char(3560);
     dcl-s chroot        char(256)     inz;
     dcl-s idx           packed(2:0);
     dcl-s success       ind           inz(*on);
     dcl-s logArgs       like(args)    inz;
     dcl-s logEnv        like(env)     inz;
     dcl-s errorMessage  varchar(1000) inz;

     if %parms >= %parmnum(env) and %addr(env) <> *null;
         logEnv = env;
     endif;

     if %parms >= %parmnum(args) and %addr(args) <> *null;
         args = args;
         logArgs = args;
     endif;
```

# Running Python from RPG

# (Unix CMD, Chroot, Proc in SRVPGM)

# Pt 2

```
39          // Build command string
40          unixCmd = 'PATH=/QOpenSys/pkgs/bin:$PATH && ';
41
42          // Chroot
43          chroot = 'chroot /QOpenSys/envs/' + %trim(env);
44          unixCmd = %trim(unixCmd) + ' ' + chroot;
45
46          // Script
47          unixCmd = %trim(unixCmd) +
48                    ' python3 "' + %trim(script) + '"';
49
50          for idx = 1 to %elem(arg);
51              if arg(idx) <> '';
52                  unixCmd = %trim(unixCmd) + ' "' + %trim(arg(idx)) + '"';
53              endif;
54          endfor;
55
56          // Executes the unixCmd string via the RPG OA Handler()
57          open unix;
58
59          // Read anything sent to STDOUT and log it
60          read unix record;
61
62          dow not %eof(unix);
63              LogUnixCmd(record);
64
65              read unix record;
66          enddo;
67
68          // Close unix will fail when there were script errors
69          monitor;
70              close unix;
71          on-error;
72              success = *off;
73          endmon;
74
75          return success;
76      end-proc;
```

# LEVEL UP!

- Date and Time functions with Arrow
- Extract Text with PDF Miner
- Convert Spreadsheets with Pandas
- Currency Exchange Rates with Forex

# Date and Time functions with Arrow

```python
import arrow

arrow.get('2013-05-11T21:23:58.970460+07:00')
# <Arrow [2013-05-11T21:23:58.970460+07:00]>

utc = arrow.utcnow()
# <Arrow [2013-05-11T21:23:58.970460+00:00]>

utc = utc.shift(hours=-1)
# <Arrow [2013-05-11T20:23:58.970460+00:00]>

local = utc.to('US/Pacific')
# <Arrow [2013-05-11T13:23:58.970460-07:00]>

local.timestamp()
# 1368303838.970460

local.format()
# '2013-05-11 13:23:58 -07:00'

local.format('YYYY-MM-DD HH:mm:ss ZZ')
# '2013-05-11 13:23:58 -07:00'

local.humanize()
# 'an hour ago'

local.humanize(locale='ko-kr')
# '한시간 전'
```

# Convert Spreadsheets with pandas

```
from sys import argv
import pandas as pd

# Read all sheets
data_xls = pd.ExcelFile(argv[1].strip(), index_col=None, sheet_name=None)
sheets = data_xls.book.sheets()

# Loop through the sheets
for sheet in sheets:

    #Find the first visible sheet
    if sheet.visibility==0:

        # Read in the first sheet
        first_sheet = pd.read_excel(argv[1].strip(), index_col=None, sheet_name=sheet.name)

        #Export to CSV
        first_sheet.to_csv(argv[2].strip(), encoding='utf-8', index=False, float_format='%.5f')
        break
```

# Extract Text with pdf-miner

```
from pdfminer.high_level import extract_text
text = extract_text("example.pdf")
print(text)
```

Yeah, that's it.

# Currency Exchange Rates with forex

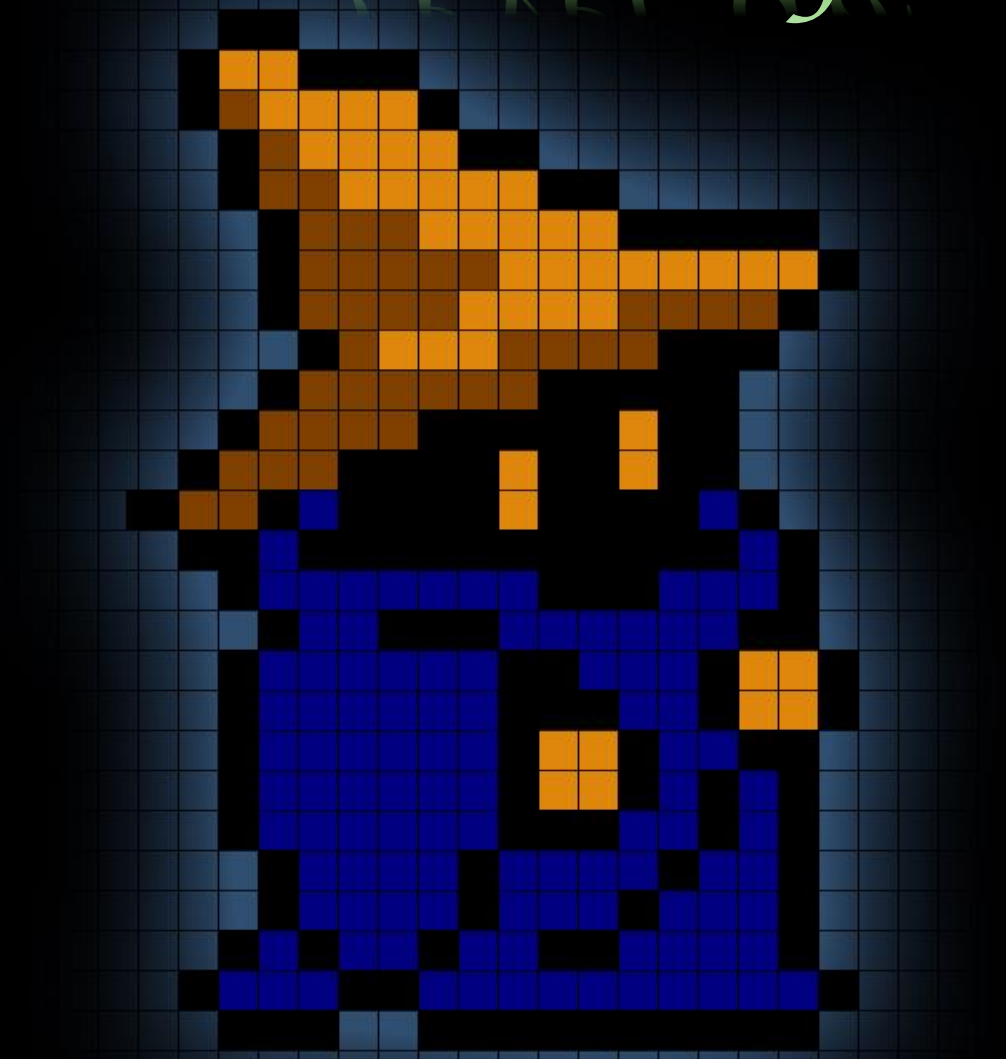
```python
from forex_python.converter import CurrencyRates
c = CurrencyRates()
c.get_rates('USD')
# {u'IDR': 13625.0, u'BGN': 1.7433, u'ILS': 3.8794, u'GBP': 0.68641, ....

c.get_rate('USD', 'JPY')
# 110.36

from forex_python.bitcoin import BtcConverter
b = BtcConverter()
b.get_latest_price('USD')
# 29185.40  # return type float

from forex_python.converter import CurrencyCodes
c = CurrencyCodes()
c.get_symbol('GBP')
# £
c.get_symbol('EUR')
# €
```

# So You Want to Be a Wizard?
# Level Up!



- Arguments
  - ArgParse
  - Python Fire
- Validation
  - JSON
  - Phone #s
  - Emails
  - URLs
  - Coordinates

# Argument Parsing with argparse

```python
import argparse

parser = argparse.ArgumentParser(description='Display netstat information.')
parser.add_argument('--limit', type=int,
    help='Only show X rows')
parser.add_argument('--offset', type=int,
    help='Skip first X rows')
parser.add_argument('--port', type=int,
    help='Look for only local port')
args = parser.parse_args()
```

Thanks Club-Seiden & ThePrez!

# Argument Magic with python-fire

```python
import fire

def hello(name="World"):
    return "Hello %s!" % name

if __name__ == '__main__':
    fire.Fire(hello)
```

```
python3 hello.py
>>> Hello World!

python3 hello.py --name=OCEAN
>>> Hello OCEAN!

python3 hello.py --help
>>> Shows usage information.
```

# Validate your JSON with jsonschema

```python
from jsonschema import validate

# A sample schema, like what we'd get from json.load()
schema = {
        "type" : "object",
        "properties" : {
            "price" : {"type" : "number"},
            "name" : {"type" : "string"},
        },
}

# If no exception is raised by validate(), the instance is valid.
validate(instance={"name" : "Eggs", "price" : 34.99}, schema=schema)
```

# Validate Phone #s with python-phonenumbers

```python
import phonenumbers

z = phonenumbers.parse("+120012301", None)
# Country Code: 1 National Number: 20012301 Leading Zero: False

phonenumbers.is_possible_number(z)  # too few digits for USA
# False

phonenumbers.is_valid_number(z)
# False

z = phonenumbers.parse("+12001230101", None)
# Country Code: 1 National Number: 2001230101 Leading Zero: False

phonenumbers.is_possible_number(z)
# True

phonenumbers.is_valid_number(z)  # NPA 200 not used
# False
```

# Validate Emails with flanker

```python
from flanker.addresslib import address

address.parse('Foo foo@example.com')
# Foo <foo@example.com>

address.parse('@example.com')
# None

address.parse_list(['foo@example.com, bar@example.com, @example.com'], as_tuple=True)
# [foo@example.com, bar@example.com], ['@example.com']
```

# Validate URLs with furl

```
from furl import furl
f = furl('http://user:pass@www.google.com:99/')
f.scheme, f.username, f.password, f.host, f.port
# ('http', 'user', 'pass', 'www.google.com', 99)
```

# Validate Coordinates with geopy

```python
from geopy.geocoders import Nominatim
geolocator = Nominatim(user_agent="specify_your_app_name_here")
location = geolocator.geocode("175 5th Avenue NYC")
print(location.address)
# Flatiron Building, 175, 5th Avenue, Flatiron, New York, NYC, New York, ...

print((location.latitude, location.longitude))
# (40.7410861, -73.9896297241625)

print(location.raw)
#{'place_id': '9167009604', 'type': 'attraction', ...}
```

It is pitch black.
You are likely to be
eaten by a Grue.

Zork

# Pt 1

## More Pandas
## PF to XLSX

## Arg1: Library
## Arg2: Table
## Arg3: Filename

```python
from sys import argv
import numpy as np
import pandas as pd
import ibm_db_dbi as dbi

# Set Save Path
savePath= '/tmp/xlsx/' + argv[3].strip() + '.xlsx'

# Get data from query and read into dataframe
sql = ('SELECT * FROM ' + argv[1] + '.' + argv[2])
conn = dbi.connect()
df1 = pd.read_sql(sql, conn)

# Start building output file for dataframe 1
writer = pd.ExcelWriter(savePath, engine='xlsxwriter')
df1.to_excel(writer, sheet_name='Sheet 1', index=None)
worksheet1 = writer.sheets['Sheet 1']
```

# Pt 2

```python

# Grab header infromation from syscolumns
sql = ("""
    SELECT
        REGEXP_REPLACE(
            REGEXP_REPLACE(
            VALUE(NULLIF(column_text,''), NULLIF(column_heading,''), column_name)
            '[\''\"]',
            ''
            ),
            '\s+',
            ' '
        ) as column_text
    FROM
        qsys2.syscolumns
    WHERE
        system_table_schema = upper('""" + argv[1] + """')
        AND system_table_name = upper('""" + argv[2] + """')
    ORDER BY ordinal_position
""")

# Read into another dataframe
df2 = pd.read_sql(sql, conn)

```

# Pt 3

```python
# Set column headers
columns = []
for col_num, value in enumerate(df2.values):
    header = value[0]
    i = 0
    while next((item for item in columns if item["header"] == header), None):
        i += 1
        header = value[0] + '_' + str(i)
    columns.append({'header':header})

# Determine last row/column
lastRow = len(df1.values)
if lastRow == 0:
    lastRow = 1

lastCol = len(df1.columns) - 1
if lastCol == 0:
    lastCol = 1

```

# Pt 4

```python
# Format As Table
worksheet1.add_table(
    0,
    0,
    lastRow,
    lastCol,
    {
        'autofilter': True,
        'style': 'Table Style Medium 16',
        'columns': columns
    }
)
```

# Pt 5

```python
# Auto width
for col_num, value in enumerate(df1.columns):
    try:
        dataLen = len(str(df1.values[0][col_num]).strip())
    except:
        dataLen = 0

    try:
        headerLen = len(str(df2.values[col_num][0].strip()
    except:
        headerLen = 0
    newLen = 0

    if dataLen > headerLen:
        newLen = dataLen
    else:
        newLen = headerLen

    newLen += 4
    worksheet1.set_column(col_num, col_num, newLen)

# Save
writer.save()
```

# Pt 6

# The Results!

| | A | B | C | D | E | F | G | H | I | J | K | L | M | N |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 1 | EMPNO | FIRSTNME | MIDINIT | LASTNAME | WORKDEPT | PHONENO | HIREDATE | JOB | EDLEVEL | SEX | BIRTHDATE | SALARY | BONUS | COMM |
| 2 | 000010 | CHRISTINE | I | HAAS | A00 | 3978 | 1965-01-01 | PRES | 18 | F | 1933-08-24 | 52750 | 1000 | 4220 |
| 3 | 000020 | MICHAEL | L | THOMPSON | B01 | 3476 | 1973-10-10 | MANAGEI | 18 | M | 1948-02-02 | 41250 | 800 | 3300 |
| 4 | 000030 | SALLY | A | KWAN | C01 | 4738 | 1975-04-05 | MANAGEI | 20 | F | 1941-05-11 | 38250 | 800 | 3060 |
| 5 | 000050 | JOHN | B | GEYER | E01 | 6789 | 1949-08-17 | MANAGEI | 16 | M | 1925-09-15 | 40175 | 800 | 3214 |
| 6 | 000060 | IRVING | F | STERN | D11 | 6423 | 1973-09-14 | MANAGEI | 16 | M | 1945-07-07 | 32250 | 500 | 2580 |
| 7 | 000070 | EVA | D | PULASKI | D21 | 7831 | 1980-09-30 | MANAGEI | 16 | F | 1953-05-26 | 36170 | 700 | 2893 |
| 8 | 000090 | EILEEN | W | HENDERSON | E11 | 5498 | 1970-08-15 | MANAGEI | 16 | F | 1941-05-15 | 29750 | 600 | 2380 |
| 9 | 000100 | THEODORE | Q | SPENSER | E21 | 0972 | 1980-06-19 | MANAGEI | 14 | M | 1956-12-18 | 26150 | 500 | 2092 |
| 10 | 000110 | VINCENZO | G | LUCCHESSI | A00 | 3490 | 1958-05-16 | SALESREP | 19 | M | 1929-11-05 | 46500 | 900 | 3720 |
| 11 | 000120 | SEAN | | O'CONNELL | A00 | 2167 | 1963-12-05 | CLERK | 14 | M | 1942-10-18 | 29250 | 600 | 2340 |
| 12 | 000130 | DELORES | M | QUINTANA | C01 | 4578 | 1971-07-28 | ANALYST | 16 | F | 1925-09-15 | 23800 | 500 | 1904 |
| 13 | 000140 | HEATHER | A | NICHOLLS | C01 | 1793 | 1976-12-15 | ANALYST | 18 | F | 1946-01-19 | 28420 | 600 | 2274 |
| 14 | 000150 | BRUCE | | ADAMSON | D11 | 4510 | 1972-02-12 | DESIGNEF | 16 | M | 1947-05-17 | 25280 | 500 | 2022 |
| 15 | 000160 | ELIZABETH | R | PIANKA | D11 | 3782 | 1977-10-11 | DESIGNEF | 17 | F | 1955-04-12 | 22250 | 400 | 1780 |
| 16 | 000170 | MASATOSHI | J | YOSHIMURA | D11 | 2890 | 1978-09-15 | DESIGNEF | 16 | M | 1951-01-05 | 24680 | 500 | 1974 |
| 17 | 000180 | MARILYN | S | SCOUTTEN | D11 | 1682 | 1973-07-07 | DESIGNEF | 17 | F | 1949-02-21 | 21340 | 500 | 1707 |
| 18 | 000190 | JAMES | H | WALKER | D11 | 2986 | 1974-07-26 | DESIGNEF | 16 | M | 1952-06-25 | 20450 | 400 | 1636 |
| 19 | 000200 | DAVID | | BROWN | D11 | 4501 | 1966-03-03 | DESIGNEF | 16 | M | 1941-05-29 | 27740 | 600 | 2217 |
| 20 | 000210 | WILLIAM | T | JONES | D11 | 0942 | 1979-04-11 | DESIGNEF | 17 | M | 1953-02-23 | 18270 | 400 | 1462 |

# Expansion Packs

## Authentication

- oauthlib
- pyjwt

## APIs

- fastapi
- boto3
- google-api-client

## Web Scraping

- scrapy
- lassie

## OCR

- opencv
- pytesseract

# Continuing your Quest

## Links

- https://www.python.org/doc/
- https://github.com/gto76/python-cheatsheet
- https://code.visualstudio.com/
- https://ibm.github.io/ibmi-oss-resources/
- https://github.com/vinta/awesome-python

# References

- https://git-scm.com/downloads
- https://www.scottklement.com/unixcmd/
- https://github.com/IBM/ibmichroot/
- https://github.com/arrow-py/arrow
- https://github.com/pdfminer/pdfminer.six
- https://github.com/pandas-dev/pandas
- https://github.com/MicroPyramid/forex-python
- https://github.com/google/python-fire
- https://docs.python.org/3/library/argparse.html
- https://github.com/python-jsonschema/jsonschema
- https://github.com/daviddrysdale/python-phonenumbers
- https://github.com/mailgun/flanker
- https://github.com/gruns/furl
- https://github.com/geopy/geopy

# Thank you

Joseph Wright

@Wright4i

https://wright4i.com