# Creating PDF's with Python

# Creating PDF's with Python

Mike Larsen

Project Manager & Senior Developer

Central Park Data Systems

mlarsen@centralparkdata.com

http://linkedin.com/in/mike-larsen-a4414a20

# What we'll cover

- Creating a sample database
- Installing packages
- Show me the code
- Creating a PDF
  - Adding a Header & Footer
  - Populating the PDF from a Db2 table
  - Saving the PDF to the IFS
- Executing the Python script
  - From an SSH terminal
  - From a CLP
- Bonus! (time permitting)

# Creating a sample database

- Run a stored procedure to create the sample database

```
CALL QSYS.CREATE_SQL_SAMPLE ('SAMPLE1');
```

```
[ 08/21/2020, 01:42:47 PM ]  Run All...
  CALL QSYS.CREATE_SQL_SAMPLE ('SAMPLE1')
✓ Return Code = 0
✓ Statement ran successfully    (32,168 ms  =  32.168 sec)
```
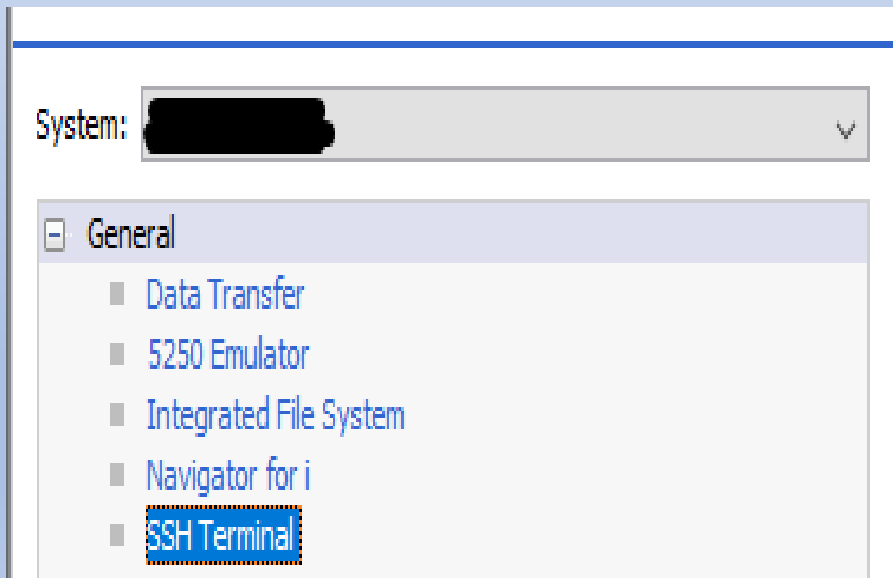
# Creating a sample database

- The stored procedure creates & populates around 20 tables
- I'm using the 'Employee' table in my example

File  Edit  View

| EMPNO | FIRSTNME | MIDINIT | LASTNAME | WORKDEPT | PHONENO | HIREDATE | JOB | EDLEVEL | SEX | BIRTHDATE | SALARY | BONUS | COMM |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 000010 | CHRISTINE | I | HAAS | A00 | 3978 | 1965-01-01 | PRES | 18 | F | 1933-08-24 | 52750.00 | 1000.00 | 4220.00 |
| 000020 | MICHAEL | L | THOMPSON | B01 | 3476 | 1973-10-10 | MANAGER | 18 | M | 1948-02-02 | 41250.00 | 800.00 | 3300.00 |
| 000030 | SALLY | A | KWAN | C01 | 4738 | 1975-04-05 | MANAGER | 20 | F | 1941-05-11 | 38250.00 | 800.00 | 3060.00 |
| 000050 | JOHN | B | GEYER | E01 | 6789 | 1949-08-17 | MANAGER | 16 | M | 1925-09-15 | 40175.00 | 800.00 | 3214.00 |
| 000060 | IRVING | F | STERN | D11 | 6423 | 1973-09-14 | MANAGER | 16 | M | 1945-07-07 | 32250.00 | 500.00 | 2580.00 |
| 000070 | EVA | D | PULASKI | D21 | 7831 | 1980-09-30 | MANAGER | 16 | F | 1953-05-26 | 36170.00 | 700.00 | 2893.00 |
| 000090 | EILEEN | W | HENDERSON | E11 | 5498 | 1970-08-15 | MANAGER | 16 | F | 1941-05-15 | 29750.00 | 600.00 | 2380.00 |
| 000100 | THEODORE | Q | SPENSER | E21 | 0972 | 1980-06-19 | MANAGER | 14 | M | 1956-12-18 | 26150.00 | 500.00 | 2092.00 |
| 000110 | VINCENZO | G | LUCCHESSI | A00 | 3490 | 1958-05-16 | SALESREP | 19 | M | 1929-11-05 | 46500.00 | 900.00 | 3720.00 |
| 000120 | SEAN | | O'CONNELL | A00 | 2167 | 1963-12-05 | CLERK | 14 | M | 1942-10-18 | 29250.00 | 600.00 | 2340.00 |
| 000130 | DELORES | M | QUINTANA | C01 | 4578 | 1971-07-28 | ANALYST | 16 | F | 1925-09-15 | 23800.00 | 500.00 | 1904.00 |
| 000140 | HEATHER | A | NICHOLLS | C01 | 1793 | 1976-12-15 | ANALYST | 18 | F | 1946-01-19 | 28420.00 | 600.00 | 2274.00 |
| 000150 | BRUCE | | ADAMSON | D11 | 4510 | 1972-02-12 | DESIGNER | 16 | M | 1947-05-17 | 25280.00 | 500.00 | 2022.00 |

# Installing packages

- Install packages using an SSH terminal.  I use PuTTY or MobaXterm
- In ACS, click on 'SSH Terminal'.
    - This will show up after you install PuTTY on your PC

# Installing packages

- If you get an error launching SSH, you may need to start SSH on IBM i
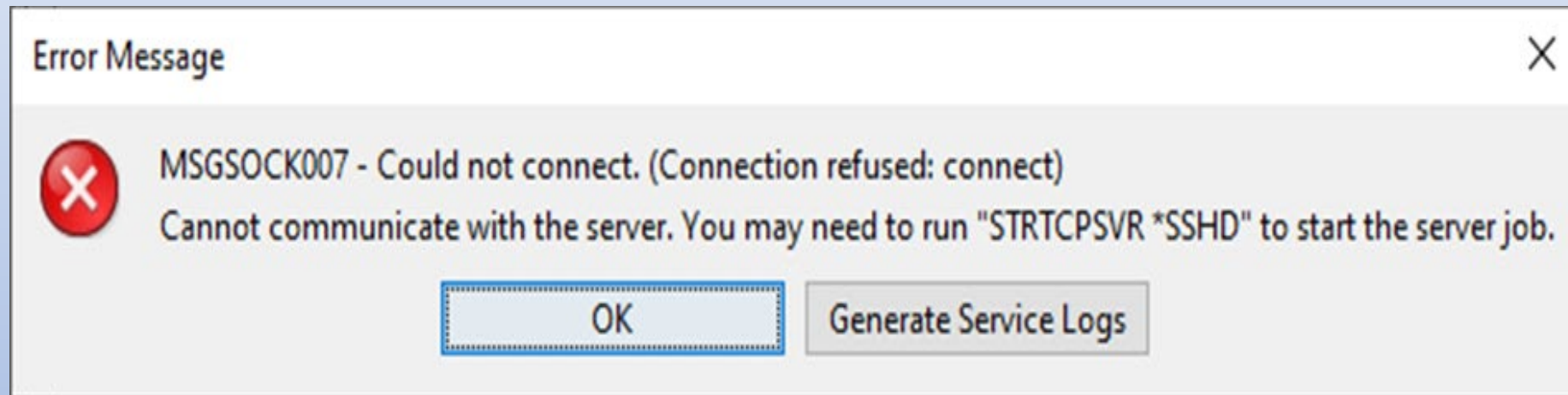


- Just run the command on IBM i:

# Installing packages

- In PuTTY, use 'pip' to install 'ibm_db' and 'fpdf'
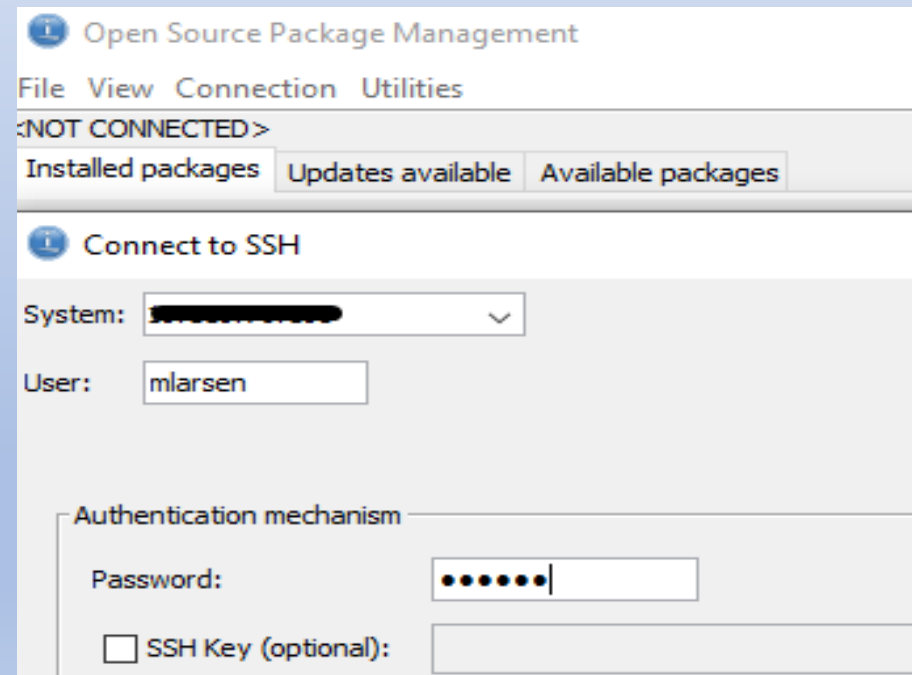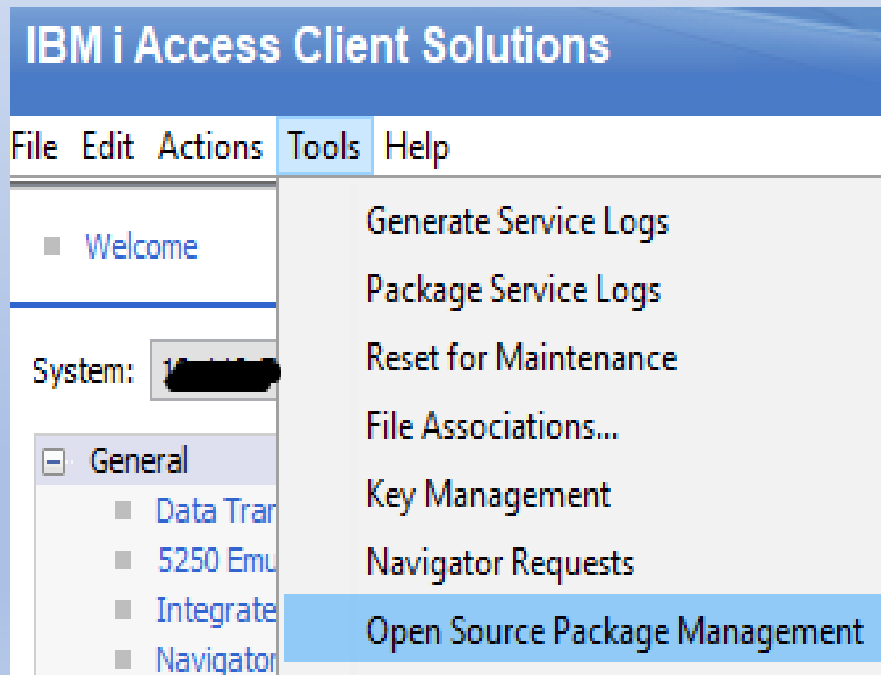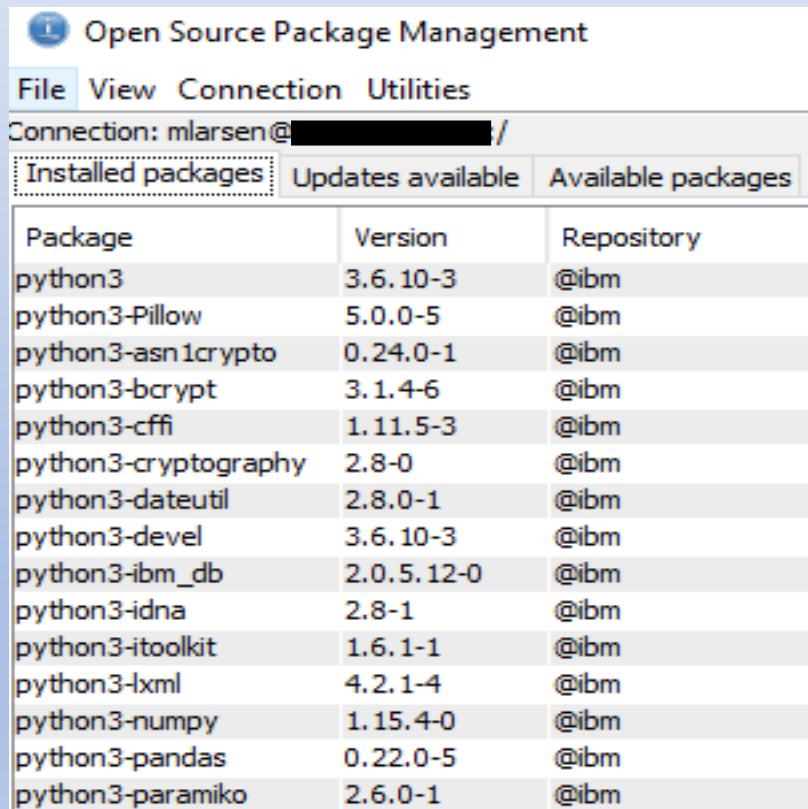- 'pip' is a package-management system used to install and manage software packages written in Python.

# Installing packages

- Update Python if necessary (optional)
- Use ACS to see if any updates exist.

# Installing packages

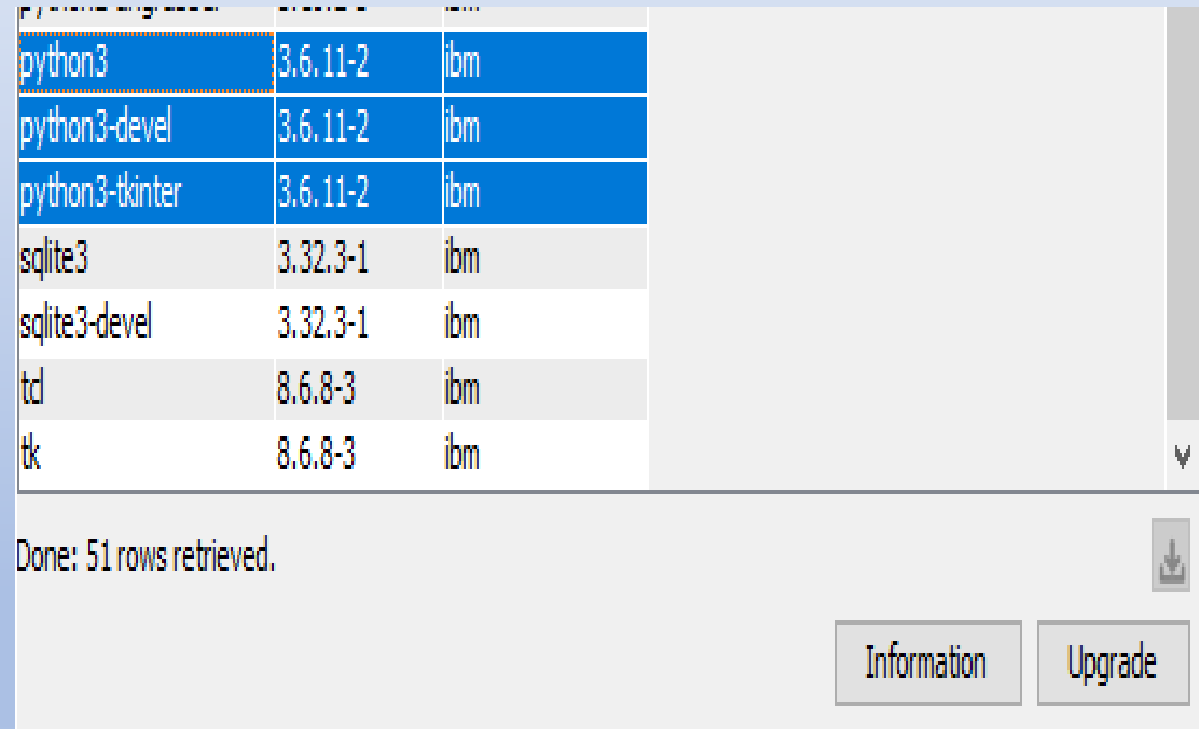- Update Python if necessary (optional)

# Installing packages

# Installing packages

```
Transaction Summary
================================================================
Install      1 Package
Upgrade      3 Packages

Total download size: 38 M
Is this ok [y/N]: y
Downloading Packages:
(1/4): libreadline8-8.0-1.ibmi7.2.ppc64.rpm                    | 517 kB  00:00
(2/4): python3-3.6.11-2.ibmi7.2.ppc64.rpm                     |  36 MB  00:31
(3/4): python3-devel-3.6.11-2.ibmi7.2.ppc64.rpm              | 165 kB  00:00
(4/4): python3-tkinter-3.6.11-2.ibmi7.2.ppc64.rpm            | 787 kB  00:00
----------------------------------------------------------------
Total                                         1.1 MB/s |  38 MB     00:34
Running Transaction Check
Running Transaction Test
Transaction Test Succeeded
Running Transaction
  Installing : libreadline8-8.0-1.ppc64                                      1/7
  Updating   : python3-3.6.11-2.ppc64 [#####                              ] 2/7
```

# Installing packages

```
Dependency Installed:
  libreadline8.ppc64 0:8.0-1

Updated:
  python3.ppc64 0:3.6.11-2                    python3-devel.ppc64 0:3.6.11-2
  python3-tkinter.ppc64 0:3.6.11-2

Complete!
```

# Creating a PDF

- A quick view of the PDF we're creating

# Creating a PDF

- The first page of the PDF created



**Employee listing**

| First name | MI | Last name | Department | Phone no. | Job title |
|---|---|---|---|---|---|
| CHRISTINE | I | HAAS | A00 | 3978 | PRES |
| MICHAEL | L | THOMPSON | B01 | 3476 | MANAGER |
| SALLY | A | KWAN | C01 | 4738 | MANAGER |
| JOHN | B | GEYER | E01 | 6789 | MANAGER |
| IRVING | F | STERN | D11 | 6423 | MANAGER |
| EVA | D | PULASKI | D21 | 7831 | MANAGER |

# Creating a PDF

- The last page of the PDF created

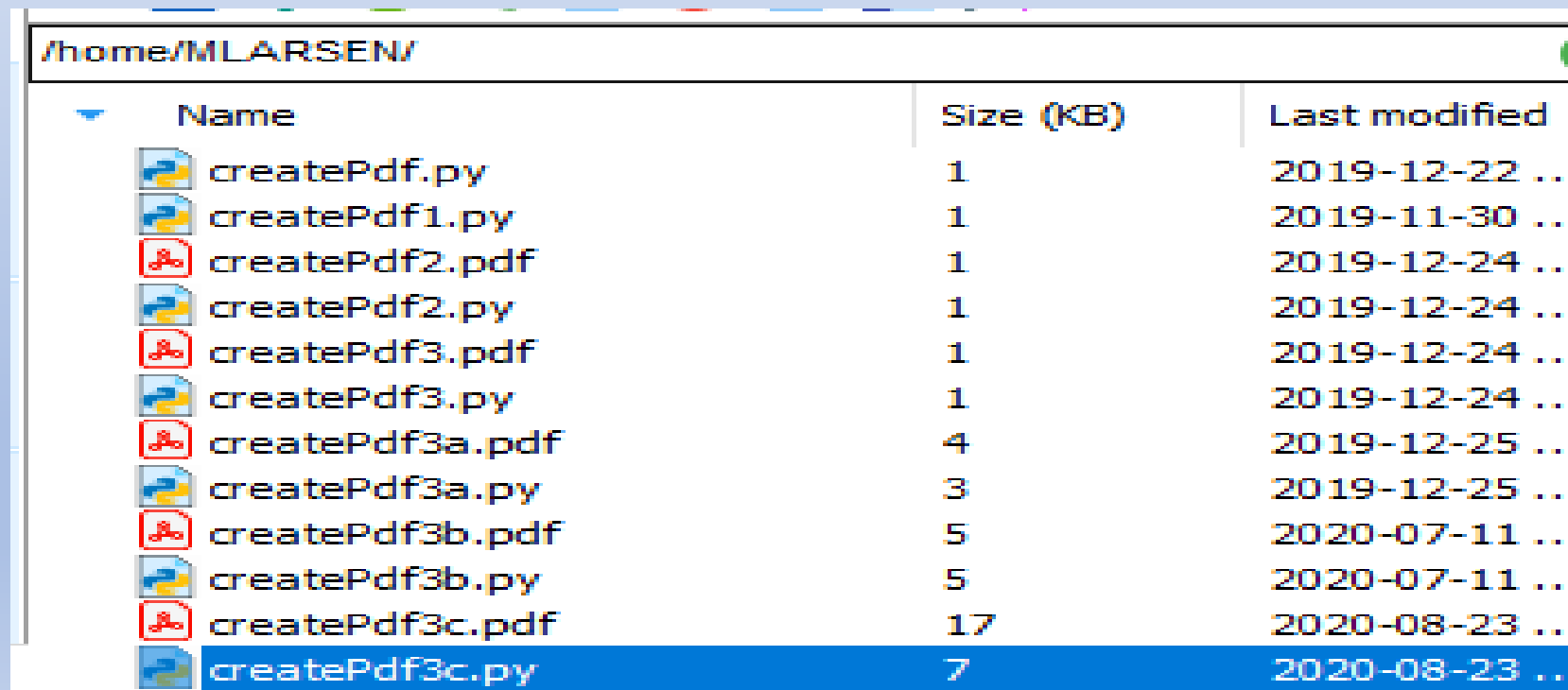| MICHELLE | F | SPRINGER | E11 | 3332 | OPERATOR |
|----------|---|----------|-----|------|----------|
| HELENA | | WONG | E21 | 2103 | FIELDREP |
| ROY | R | ALONZO | E21 | 5698 | FIELDREP |
| Joe | E | cool | A00 | 8550 | MANAGER |
| Joe | D | cool | A00 | 8551 | MANAGER |

Number of Employees     44

# Show me the code

- Let's step through the code
- I put detailed comments in the code

# Show me the code

- Python classes
  - User-defined blueprint or prototype from which objects are created
  - Provide a means of bundling data and functionality together
  - Creating a new class creates a new type of object, allowing new instances of that type to be made
  - Class instances can also have methods
  - We can override a class and add our own functionality
  - We can loosely compare a class to a program and a method to a subprocedure or subroutine

# Executing the Python script

- I use either PuTTY of MobaXterm

- I used MobaXterm in this example

# Executing the Python script

- I use either PuTTY of MobaXterm

- I used MobaXterm in this example

# Executing the Python script

- I use either PuTTY of MobaXterm

- I used MobaXterm in this example

| /home/MLARSEN/ | | |
|---|---|---|
| Name | Size (KB) | Last modified |
| createPdf.py | 1 | 2019-12-22 .. |
| createPdf1.py | 1 | 2019-11-30 .. |
| createPdf2.pdf | 1 | 2019-12-24 .. |
| createPdf2.py | 1 | 2019-12-24 .. |
| createPdf3.pdf | 1 | 2019-12-24 .. |
| createPdf3.py | 1 | 2019-12-24 .. |
| createPdf3a.pdf | 4 | 2019-12-25 .. |
| createPdf3a.py | 3 | 2019-12-25 .. |
| createPdf3b.pdf | 5 | 2020-07-11 .. |
| createPdf3b.py | 5 | 2020-07-11 .. |
| createPdf3c.pdf | 17 | 2020-08-23 .. |
| createPdf3c.py | 7 | 2020-08-23 .. |

# Executing the Python script

- It's great that I can execute the script from a terminal, but in a production process, you'd likely want to run it from a program

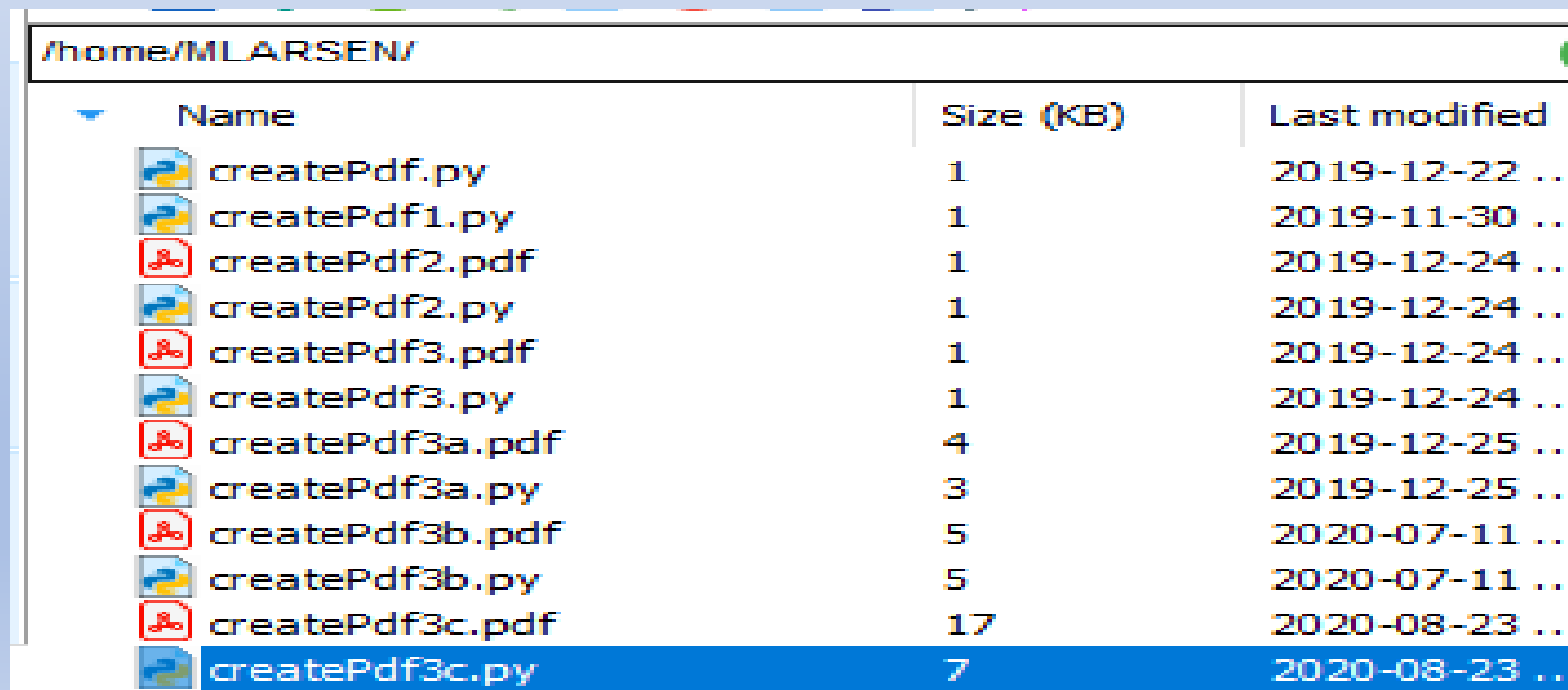- We can execute Python scripts from either CL or RPG

- I'll show how to do it using RPG

# Executing the Python script

- Set up variables to hold the path to the Python executable and the location of the script

```
000152 //----------------------------------------------------
000153 //  setUp subprocedure
000154 //----------------------------------------------------
000155
000156 dcl-proc setUp;
000157
000158    pathToPython = '/QOpenSys/pkgs/bin/python3';
000160    pythonScript = 'createPdf3c.py';
000161
000162 end-proc setUp;
```

# Executing the Python script

- Use Qshell to execute the script

```
000164 //--------------------------------------------------------
000165 //  executePythonScript  subprocedure
000166 //--------------------------------------------------------
000167
000168 dcl-proc executePythonScript;
000169
000170    // - - - -
000171    // Use Qshell to execute the Python script.
000176
000177    CmdStr = 'Qsh Cmd(' + quotes +
000178                              %trim(pathToPython) + ' ' +
000179                              %trim(pythonScript) +
000183                     '''')';
000184
000185    Callp Run(Cmdstr:%Size(CmdStr));
000186
000187 end-proc executePythonScript;
```

# Executing the Python script

- Execute the Python script by calling the RPG program

```
Parameters or command
===> call PYTHON_EX4
```

# Bonus!

# Passing parameters to Python scripts

- Pass four parameters to the script

```
000120
000121 // - - - -
000122 // Workfields
000123
000124 dcl-s firstParameter  char(10) inz('lexie');
000125 dcl-s secondParameter char(10) inz('slayer');
000126 dcl-s thirdParameter  char(10) inz('yankees');
000127 dcl-s fourthParameter char(10) inz('two words');
```

# Passing parameters to Python scripts

- We just need to add the parameters when we execute the script from Qshell

```
000167
000168 dcl-proc executePythonScript;
000169
000170     // Use Qshell to execute the Python script.
000171     //
000172     // Note: Parameters are detected by a space in Python, so the
000173     // the 'fourthParameter' needs to be in quotes since it has two
000174     // words separated by a space.  if I didn't put it in quotes,
000175     // Python would see them as two separate parameters.
000176
000177     CmdStr = 'Qsh Cmd(' + quotes +
000178                          %trim(pathToPython)      + ' ' +
000179                          %trim(pythonScript)      + ' ' +
000180                          %trim(firstParameter)    + ' ' +
000181                          %trim(secondParameter)   + ' ' +
000182                          %trim(thirdParameter)    + ' ' +
000183           '"' + %trim(fourthParameter)    + '"' + ''')';
000184
000185     Callp Run(Cmdstr:%Size(CmdStr));
000186
000187 end-proc executePythonScript;
```

# Passing parameters to Python scripts

- The Python script

- Sys - System-specific parameters and functions

- Argv - The list of command line arguments passed to a Python script

```
13
14  import sys
15
16  print (sys.argv[0]) # prints the name of the Python script
17  print (sys.argv[1]) # prints parameter 1
18  print (sys.argv[2]) # prints parameter 2
19  print (sys.argv[3]) # prints parameter 3
20  print (sys.argv[4]) # prints parameter 4
```

# Passing parameters to Python scripts

- Execute the Python script by calling the RPG program

```
Parameters or command
===> call PYTHON_EX1
```

# Passing parameters to Python scripts

- The parameters are printed to the terminal

```
parameters1.py
lexie
slayer
yankees
two words
Press ENTER to end terminal session.




===>  _____

F3=Exit F4=End of File F6=Print F9=Retrieve F17=
F18=Bottom  F19=Left    F20=Right F21=User Window
```

# Passing parameters to Python scripts

- Why did I enclose the fourth parameter in quotes?
- The separation of parameters is determined by a space
- If I passed 'two words' as a parameter, the script would see it as two different parameters
- If I pass it as ' "two words" ', the script will treat it as one parameter since it's enclosed in quotes

# Additional information

- The article was published by IT Jungle July 2020. https://www.itjungle.com/2020/07/27/guru-creating-pdf-documents-with-python/

- Article showing how to pass parameters to Python https://www.itjungle.com/2019/08/05/guru-passing-parameters-to-python/

- Full source code and slides will be made available.

- FPDF links:
  - http://www.fpdf.org/en/doc/cell.htm
  - https://www.elated.com/create-nice-looking-pdfs-php-fpdf/
  - https://pyfpdf.readthedocs.io/en/latest/Tutorial/index.html

# Questions?

# Thank you!