25th Anniversary

IUG2017

NATIONAL HARBOR | MARYLAND

25TH ANNUAL INNOVATIVE USERS GROUP CONFERENCE

APRIL 2-5, 2017

Automating Reports with Python

Gem Stone-Logan
gem.stone-logan@mountainview.gov
Technology Librarian
Mountain View Public Library
Mountain View, CA

Automating Reports with Python: Agenda

- Installing Python
- Basic Requirements
- Use SQL with Python
- Create an Excel spreadsheet
- Email the report
- Automate with Task Scheduler

NOTE: This script connects to a PostgreSQL database. However, the general concept should be the same for most other databases.

Why Automate?

- Staff time
  - Allow more time for other things
  - Will work while staff is on vacation
- Reduces boring work
- Reports are delivered on a consistent schedule

Manually Create the Report

Task	One Time	Every Time
Report Requirements	x	
Write/Refine SQL	x	
Open pgAdmin, Connect to DB		x
Open SQL session and copy SQL		x
Execute and save data		x
Open Excel and format worksheet		x
Import data and save results		x
Email to staff		x

Automate the Report

Task	One Time	Every Time
Report Requirements	x	
Write/Refine SQL	x	
Open pgAdmin, Connect to DB	x	
Open SQL session and copy SQL	x	
Execute and save data	x	
Open Excel and format worksheet	x	
Import data and save results	x	
Email to staff	x	
Create Batch file and Setup Task Scheduler	x	

What do I want to do?

Pull a list of new items, insert into a nicely formatted spreadsheet, and then email to staff. This requires:

- Connecting to a database
- Querying the database with SQL
- Creating the spreadsheet
- Importing data into the spreadsheet
- Creating an email
- Sending the email
- Automate it

Installing Python

- Python 3.5
  - Better if learning Python for the first time
  - Much better unicode support
- Anaconda Distribution
  - Contains many useful Python packages (add-ins) already
- Pip
  - Used for installing additional packages

Installing Python (notes)

Which version: Started with 2.7, switch to 3.5. 3.5 has better unicode support. Examples here are 3.5.

Distribution: I chose Anaconda.
Anaconda has a large number of useful modules already included.
Download Anaconda: https://www.continuum.io/downloads

Other distribution options: https://wiki.python.org/moin/PythonDistributions

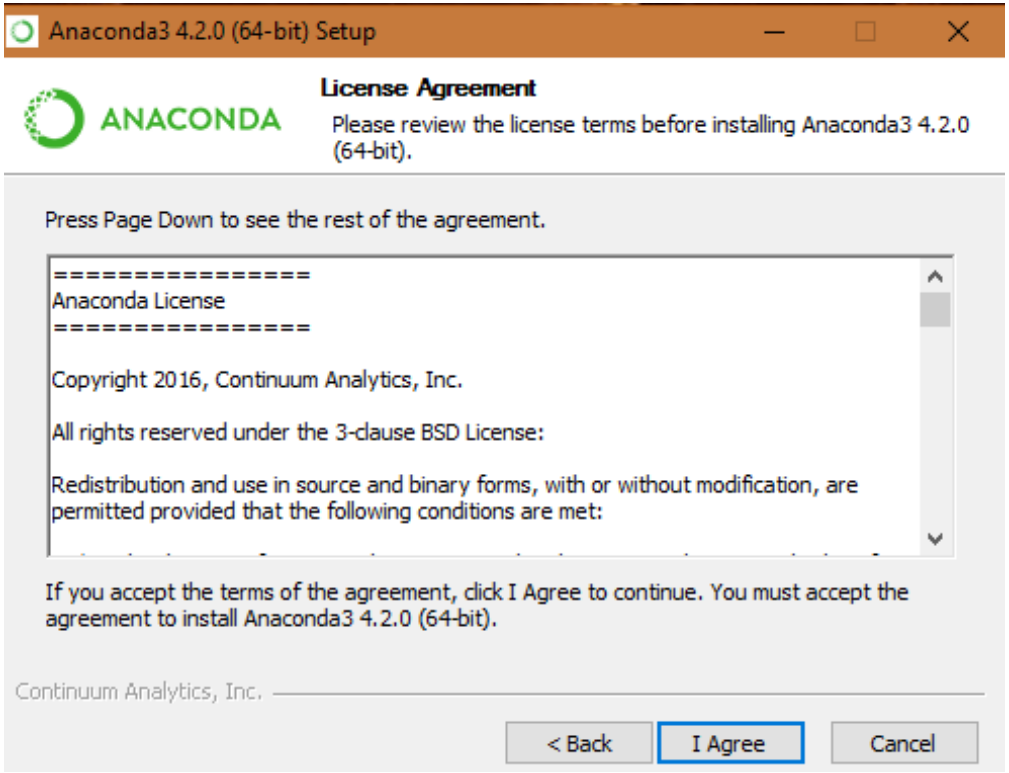
Installing Python

Anaconda Download Link

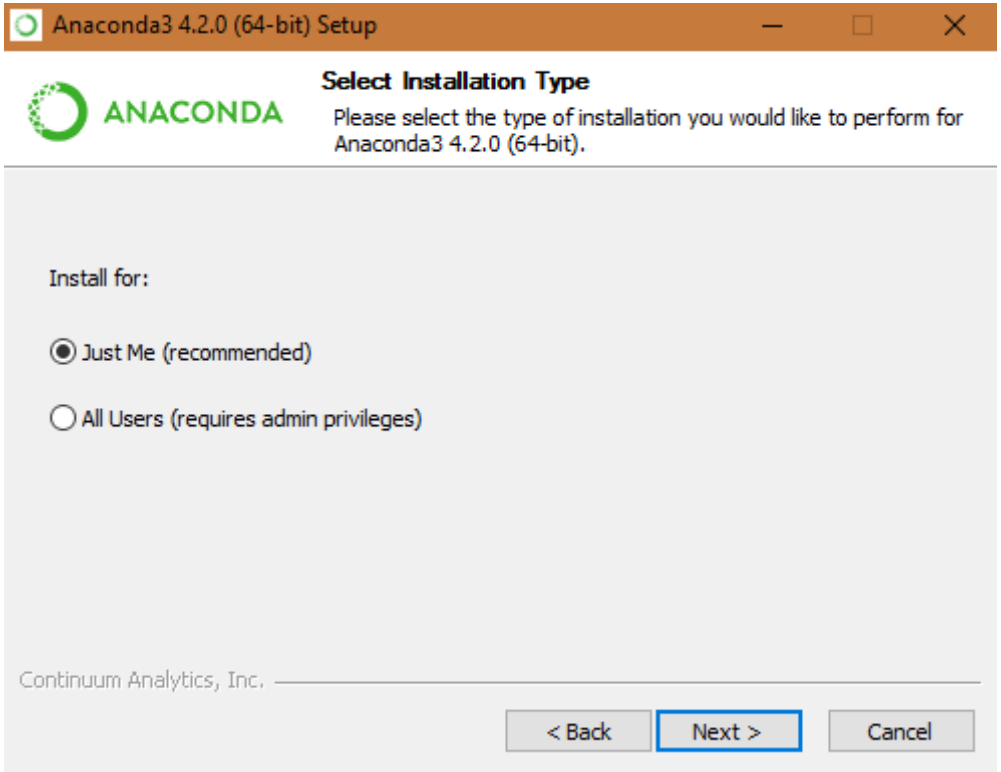
https://www.continuum.io/downloads

Installing Python

Installing Python



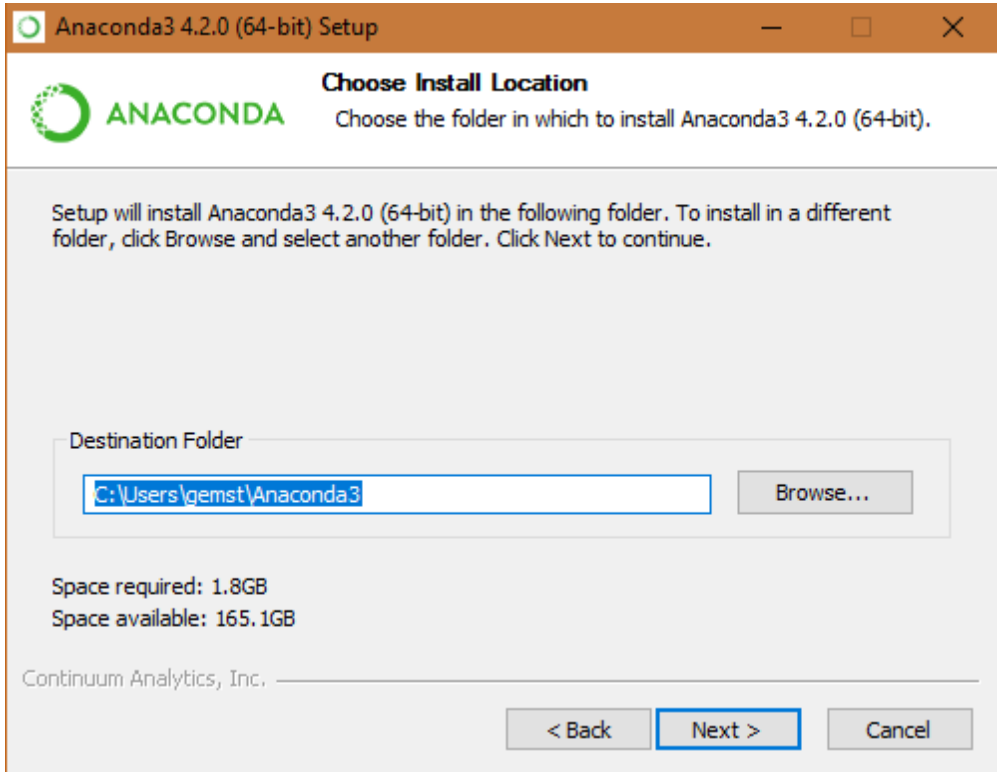
Installing Python



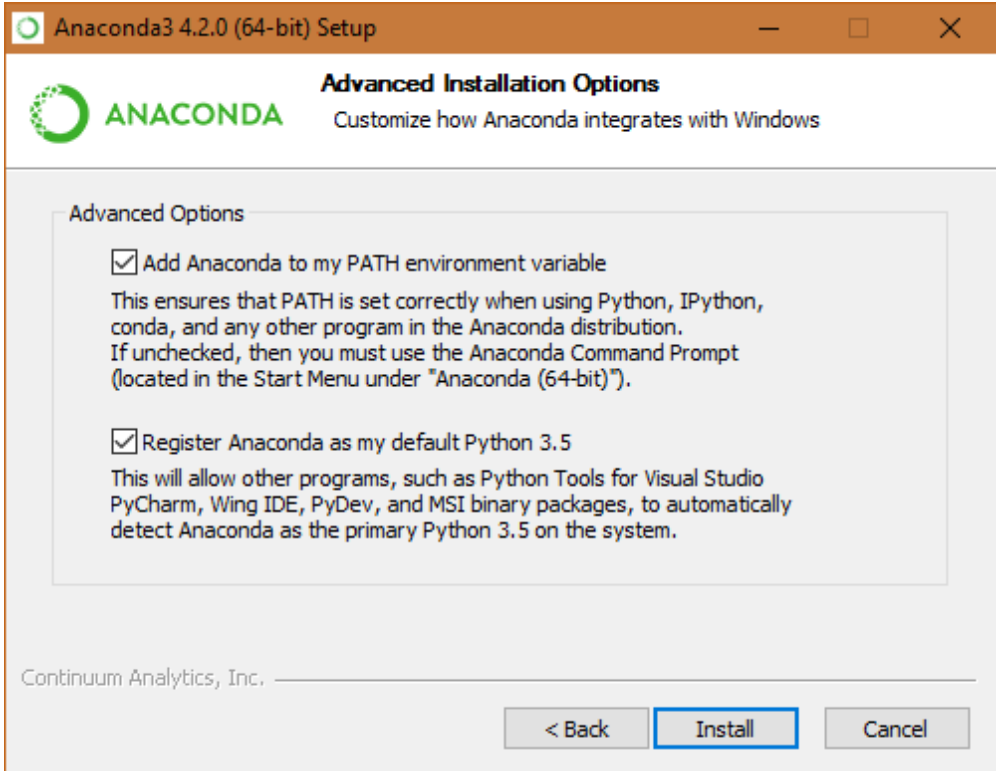
Installing Python (notes)

Make sure to choose “Just Me” if you don't have administrative rights on your Windows computer.

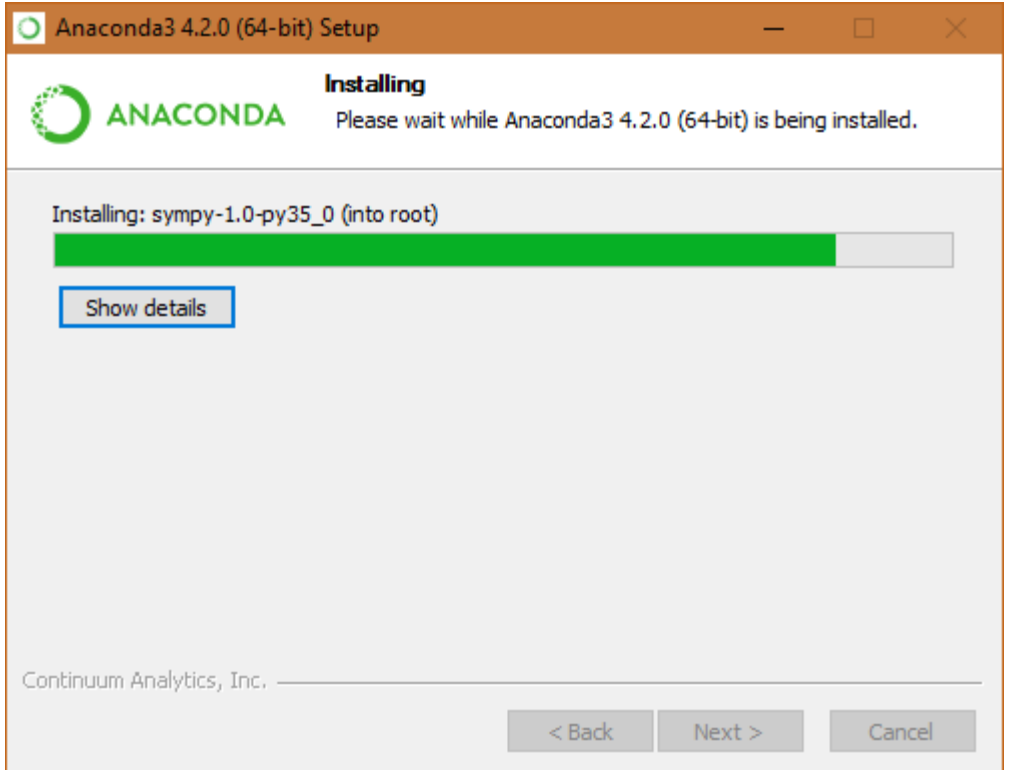
Installing Python



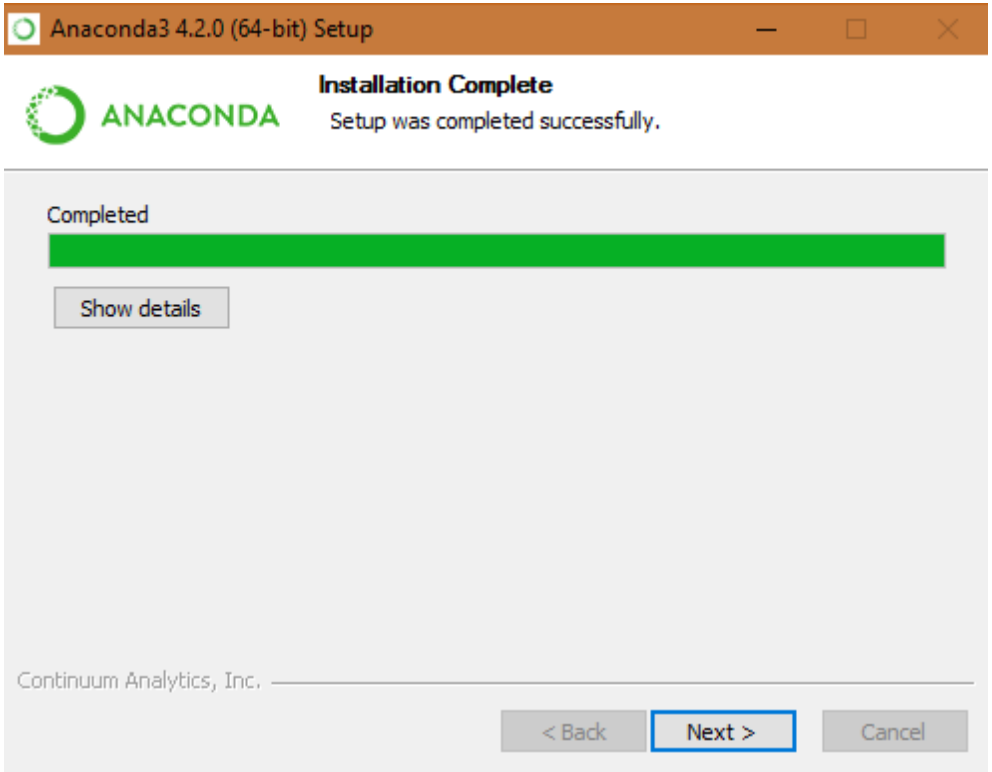
Installing Python



Installing Python



Installing Python



Installing Python

Interactive Shell

- IPython – comes with Anaconda
  - Great for practicing

Editors

Any plain text editor will work:

- Notepad (See `hello.py`)
  - Any Windows computer will have this
  - Simple – probably too simple
- Notepad++
  - Syntax highlighting, substantial improvement over Notepad
- Geany (recommended)
  - Easy to run Python straight from the editor
- gVim
  - Not a good beginning editor.

Python: Shebang Line

- Usually the “shebang” is the first line in a Python Script
- Example: `#!/usr/bin/env python3`
- Provides the reader a hint as to what this file is
- It's not necessary in Windows but is good practice

(See `weeklynew1.py`)

Python: Docstring

- Explains what the script or function does
- Starts and ends with 3 double quotes: “ ” ”

http://www.pythonforbeginners.com/basics/python-docstrings

Python: Comments

- Each comment line starts with a #
- Use comments to explain what your program is doing

Python: Packages

Python: Packages

```
C:\Users\gemst\Documents\IUG2017\WeeklyNew3_5Pres>python
weeklynew1.py
Traceback (most recent call last):
  File "weeklynew1.py", line 10, in <module>
    import psycopg2
ImportError: No module named 'psycopg2'
```

Python: Packages

```
pip install psycopg2
```

OR

```
python -m pip install psycopg2
```

(https://docs.python.org/3/installing/)

OR

```
conda search psycopg2
conda install psycopg2
```

(https://conda.io/docs/using/pkgs.html)

Python: Packages (notes)

On my computers I was able to just type:

```
pip install psycopg2
```

That didn't work for someone who tried it during this conference. We were able to use Conda, which is another option that comes with Anaconda to do the same thing:

```
conda search psycopg2
conda install psycopg2
```

I don't know if it's important to search first or not but that's what we did.

Searching around, it's possible that he could have installed using this command for pip instead but I haven't tried it.

```
python -m pip install psycopg2
```

Python: Packages

```
Collecting psycopg2
  Downloading psycopg2-2.7.1-cp35-cp35m-win_amd64.whl
(939kB)
    100% |################################| 942kB
884kB/s
Installing collected packages: psycopg2
Successfully installed psycopg2-2.7.1
```

Python: SQL Server/MS Server

I have not tried connecting to SQL Server but a module to try is pymssql. To install, use the same syntax as for psycopg2. For example:

```
pip install pymssql
```

Python: Connecting to PostegrSQL

```
conn = psycopg2.connect(“dbname='[database
name]', user='[database user]', host='[host]',
port='[port]',sslmode='[sslmode]')
```

(See `weeklynew2.py`)

Python: SQL

- Open a database session:

```
cursor = conn.cursor()
```

- Querying the database:

```
cursor.execute(open("[sql file]","r".read())
        OR
cursor.execute("[sql statement]")
        OR
cursor.execute([varible with sql])
```

(See `weeklynew3.py` and `weeklynew3ALT.py` and `WeeklyNewItemsRev.sql`)

Python: SQL

- Storing the SQL results for use later:

```
rows = cursor.fetchall()
```

- Closing the session:

```
conn.close()
```

Python: Creating an Excel Workbook

- Module for creating an Excel spreadsheet with Python:

```
import xlsxwriter
```

- Documentation:

```
http://xlsxwriter.readthedocs.io/index.html
```

(See `weeklynew4.py`)

Python: Creating an Excel Workbook

Creating a new Excel workbook

```
workbook = xlsxwriter.Workbook("[file
name]")
```

OR

```
workbook = xlsxwriter.Workbook([variable
with filename])
```

Python: Creating an Excel Worksheet

Creating a new worksheet for our workbook

```
worksheet = workbook.add_worksheet()
```

Python: Excel Formatting

- Change worksheet to landscape view (default is portrait)

```
worksheet.set_landscape()
```

- When printing, print the grid lines

```
worksheet.hide_gridlines(0)
```

(See `weeklynew5.py`)

Python: Add Cell Formatting

- General syntax to specify cell formatting

```
workbook.add_format({})
```

- Make dates appear as mm/dd/yy

```
workbook.add_format({'num_format' :
'mm/dd/yy'})
```

- Make the text within a cell wrap

```
workbook.add_format({'text_wrap' : True})
```

Python: Add Cell Formatting

- Align text at the top

```
workbook.add_format({'valign' : 'top'})
```

- Make the font bold

```
workbook.add_format({'bold' : True})
```

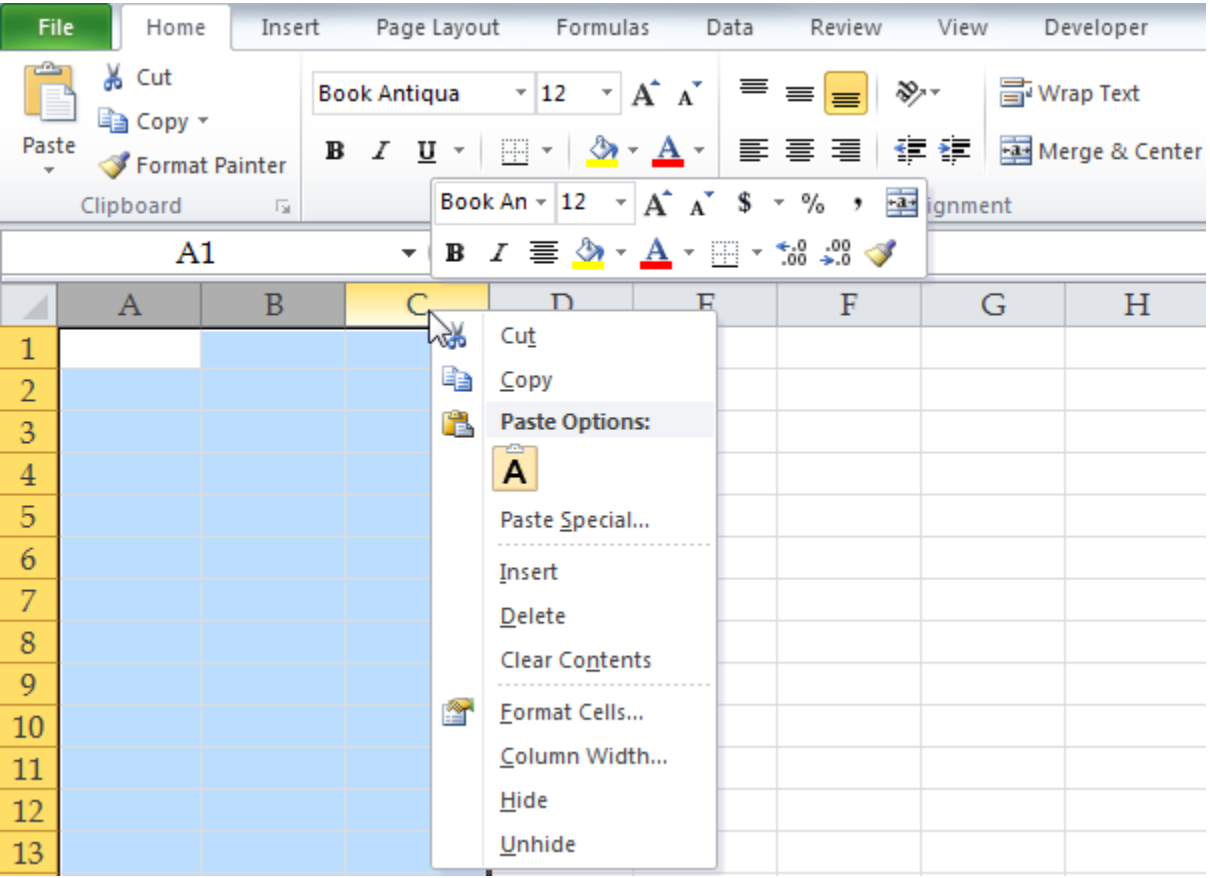
- Combine multiple formats together with a comma

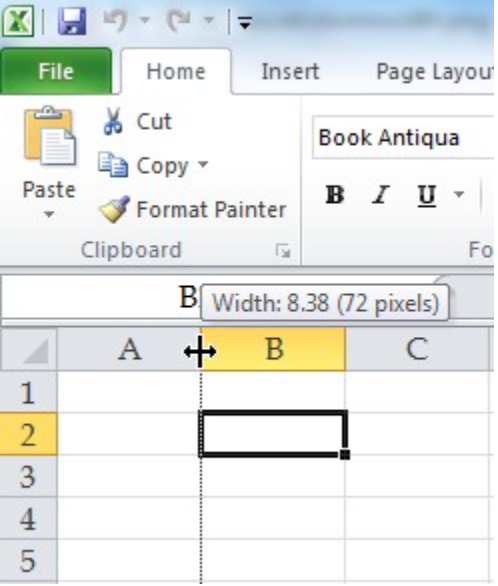
```
workbook.add_format({'valign' : 'top',
'bold' : True})
```

Python: Changing Column Widths

- Specifying a column's width manually

```
worksheet.set_column([first column],[last column],[column width])
```


File
Home
Insert
Page Layout
Formulas
Data
Review
View
Developer
Cut
Copy
Format Painter
Paste
Clipboard
Book Antiqua
12
Wrap Text
Merge & Center
A1
A
B
C
D
F
G
H
1
2
3
4
5
6
7
8
9
10
11
12
13
Cut
Copy
Paste Options:
Paste Special...
Insert
Delete
Clear Contents
Format Cells...
Column Width...
Hide
Unhide


File
Home
Insert
Page Layout
Cut
Copy
Format Painter
Paste
Book Antiqua
Clipboard
Width: 8.38 (72 pixels)
A
B
C
1
2
3
4
5

Python: Changing Column Widths

- For example, if we wanted to set just column C to a width of 12.71, it would look like this.

```
worksheet.set_column(2,2,12.71)
```

- Columns start at 0 which is why Excel column C, the 3rd column, is referred to by a 2.

Python: Worksheet Header

- Inserting a worksheet header

```
worksheet.set_header('[header content]')
```

- Basic Header

```
worksheet.set_header('Weekly New List')
```

- Lots of header customization options
  - http://xlsxwriter.readthedocs.io/page_setup.html#worksheet-set-header
    (See `weeklynew6.py`)

Python: Column Labels

- Adding Labels for Columns

```
worksheet.write([row],[column],[content],
[optional formatting])
```

- An example using the eformatlabel we defined earlier

```
worksheet.write(0, 1, 'Location',
eformatlabel)
```

Python: For Loop

- Allows us to do the same thing repeatedly
- We use a for loop to write every row of data to our spreadsheet.

(See `weeklynew7.py` and `weeklynew8.py`)

Python: For Loop

```
for rownum, row in enumerate(rows):
    [indent and add commands]
```

`for` – python for command
`rownum` – helps us count what row we're on
`row` – a single row of data
`enumerate` – python command starts at the beginning and goes to the end
`rows` – all the rows our SQL query returned
Every command in the for loop must be indented

Python: For Loop (Notes)

Indentation is very, very important in Python. If you copy someone else's code and it isn't working for you, double check that you don't have mixed use of both tabs or spaces. Either tabs or spaces are fine for indentation but choose one and only one.

Python: For Loop

Our example:

```
for rownum, row in enumerate(rows):
  worksheet.write(rownum+1,0,row[0], eformat)
```

- We chose `rownum+1` because we want to start writing our data after the row that has our column labels.
- `0` specifies we're writing to Excel column A
- `row[0]` specifies the first column in our SQL results
- `eformat` is the formatting we specified earlier

Python: Closing the Workbook

- What opens must be closed

```
workbook.close()
```

Python: Setting up Email

Modules for sending email:

```
import smtplib
from email.mime.multipart import
MIMEMultipart
from email.mime.base import MIMEBase
from email.mime.text import MIMEText
from email.utils import formatdate
from email import encoders
```

(See `weeklynew9.py`)

Python: Basic Email

- Email server
- Email server port
- To
- From
- Subject
- Message

Python: Creating the email message

```
msg = MIMEMultipart()
msg['From'] = emailfrom
if type(emailto) is list:
    msg['To'] = ', '.join(emailto)
else:
    msg['To'] = emailto
msg['Date'] = formatdate(localtime = True)
msg['Subject'] = emailsubject
msg.attach (MIMEText(emailmessage))
part = MIMEBase('application', "octet-stream")
part.set_payload(open(excelfile,"rb").read())
encoders.encode_base64(part)
part.add_header('Content-Disposition','attachment; filename=%s'
% excelfile)
msg.attach(part)
```

Python: Sending the Email Message (Basic)

```
smtp = smtplib.SMTP(emailhost, emailport)
smtp.sendmail(emailfrom, emailto,msg.as_string())
smtp.quit()
```

Automating!

- Windows bat files
- Task Scheduler

(See `weeklynew.bat`)

Automating: Windows bat file

- Plain text file that has a bat extension and contains command prompt instructions

- Parts of the bat file for automating a Python script:
  - @echo off
  - Python location
  - Your program's location

- Figure out where Python is installed
  - Open cmd
  - Type `where python`

Automating: Task Scheduler

- Click Windows icon and search for **Task Scheduler**
- Under **Actions** choose **Create Task**
- Provide a name
- Click the **Triggers** tab and **New** to choose the schedule (make sure **Enabled** is checked)
- Click the **Actions** tab and choose **New**
- **Action** is **Start a program**
- Browse to where you **bat** file is stored
- **Start in** should include the path to your program

Automating: Task Scheduler (Notes)

Make sure you automate using the bat file, NOT your program file.

Similar Reports

If you have gotten to this point, here are other really similar reports ideas:

- New Item to Old Item lists
- Items Lost and Paid
- High Demand Holds, the way you want them
- Data cleanup (wrong itypes, wrong icodes, wrong bcodes, etc)
- Weekly Stats
- Items taking too long to be processed
- Items renewed an excessive number of times

Additional Resources and Troubleshooting

- *Python Crash Course* by Eric Matthes
  - My favorite Python book so far

- Stack Overflow
  - https://stackoverflow.com/
  - One of the best places to find answers to programming questions

Questions?

Contact Info: gem.stone-logan@mountainview.gov

Extras

Python: Exceptions

- Useful for troubleshooting
- Your Python program will stop when it finds a problem unless you tell it what to do with the problem.
- try-except blocks are useful for dealing with potential problems.

Python: try-except

```
try:
    [the line(s) you want your program to run]
except:
    [What to do if your program can't run those lines]
```

- try and except both end with a colon
- The lines after try and except must be indented. You can use either spaces or tabs but do not use both, pick one. Most Python people seem to prefer spaces.
  (See `weeklynew2ALT.py`)

Python: except

```
Except pscopyg2.Error as e:
    print (“Unable to connect to database: “ +
str(e))
```

Python: Sending from Gmail with 2FA

Create an app specific password

- myaccount.google.com
- Sign-in & security
- App passwords
- Click the **Select app** dropdown and choose **Mail**
- Click **Select device** dropdown and choose **Other** and enter a good description
- Click **Generate**
- Copy the password given, without spaces, into your Python script (See `weeklynew9ALT.py`)

Python: Sending from Gmail with 2FA

Add additional variables:

```
emailuser (your gmail address)
emailpass (your newly generated password)
Emailport = '587'
```

Python: Sending from Gmail with 2FA

Add to send email:

```
smtp.ehlo()
smtp.starttls()
smtp.login(emailuser,emailpass)
```

Python: Sending the Email Message (Gmail)

```
smtp = smtplib.SMTP(emailhost, emailport)
smtp.ehlo()
smtp.starttls()
smtp.login(emailuser, emailpass)
smtp.sendmail(emailfrom, emailto,msg.as_string())
smtp.quit()
```


25th Anniversary
IUG2017
NATIONAL HARBOR | MARYLAND
25th Anniversary