DATA 520
Lecture 23

The Debugger

Importing - Exporting files

Python is free, but

Many many many books available on learning and extending Python

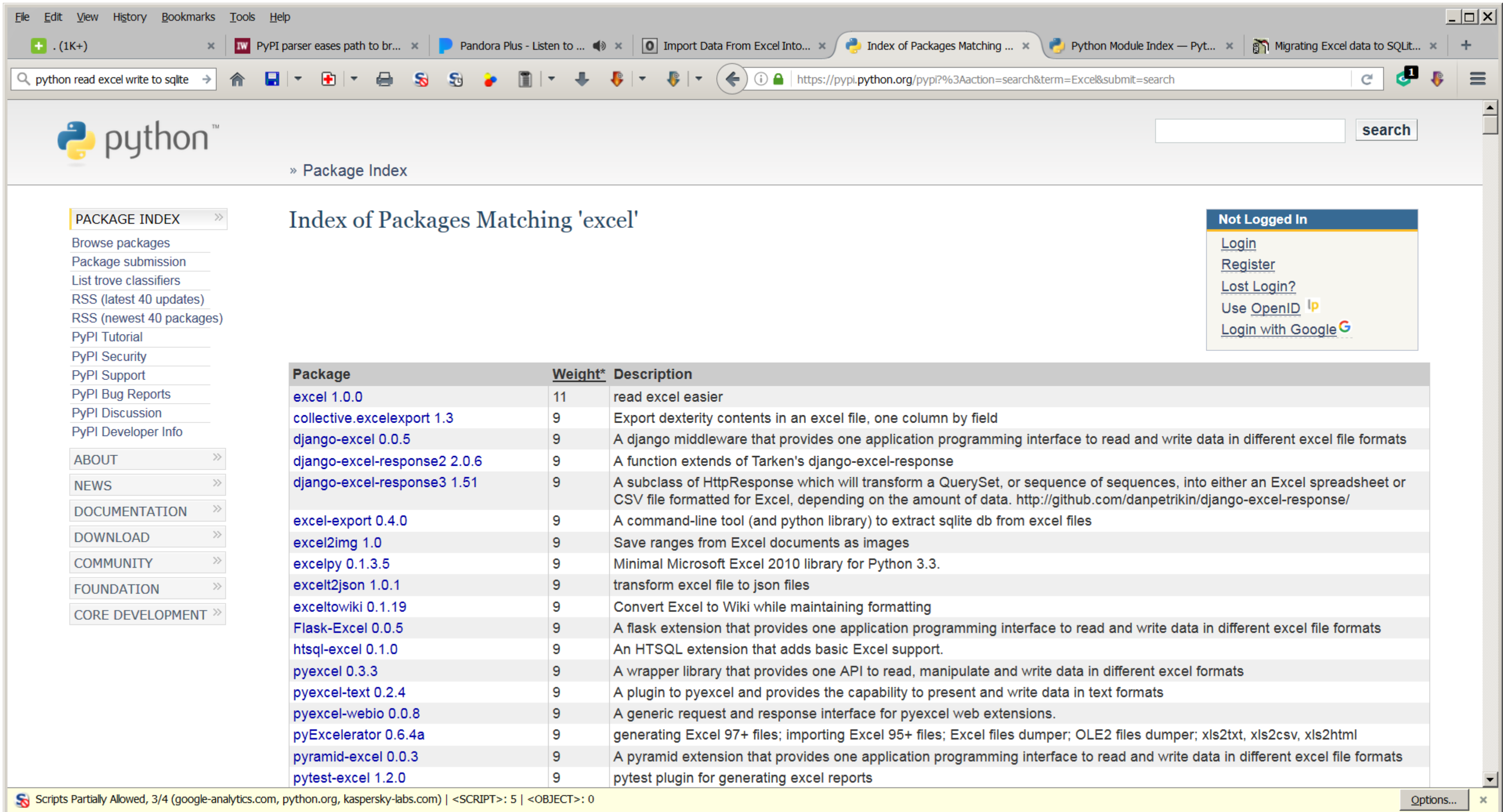
There are special Python distributions for a reasonable cost

Anaconda includes compatible versions of NumPy, etc.

You can find many Python code examples on the web
- some of them even work (it *is* the web!)

More Resources

`http://openpyxl.readthedocs.io/en/default/index.html`
`Modules - Python Package Index: https://pypi.python.org/pypi`
`Over 100 dealing with Excel`


File Edit View History Bookmarks Tools Help
. (1K+)
PyPI parser eases path to br...
Pandora Plus - Listen to ...
Import Data From Excel Into...
Index of Packages Matching ...
Python Module Index — Pyt...
Migrating Excel data to SQLit...
python read excel write to sqlte
https://pypi.python.org/pypi?%3Aaction=search&term=Excel&submit=search
python
search
» Package Index
PACKAGE INDEX
Browse packages
Package submission
List trove classifiers
RSS (latest 40 updates)
RSS (newest 40 packages)
PyPI Tutorial
PyPI Security
PyPI Support
PyPI Bug Reports
PyPI Discussion
PyPI Developer Info
ABOUT
NEWS
DOCUMENTATION
DOWNLOAD
COMMUNITY
FOUNDATION
CORE DEVELOPMENT
Not Logged In
Login
Register
Lost Login?
Use OpenID
Login with Google
Scripts Partially Allowed, 3/4 (google-analytics.com, python.org, kaspersky-labs.com) | <SCRIPT>: 5 | <OBJECT>: 0
Options...

Index of Packages Matching 'excel'

Package	Weight*	Description
excel 1.0.0	11	read excel easier
collective.excelexport 1.3	9	Export dexterity contents in an excel file, one column by field
django-excel 0.0.5	9	A django middleware that provides one application programming interface to read and write data in different excel file formats
django-excel-response2 2.0.6	9	A function extends of Tarken's django-excel-response
django-excel-response3 1.51	9	A subclass of HttpResponse which will transform a QuerySet, or sequence of sequences, into either an Excel spreadsheet or CSV file formatted for Excel, depending on the amount of data. http://github.com/danpetrikin/django-excel-response/
excel-export 0.4.0	9	A command-line tool (and python library) to extract sqlite db from excel files
excel2img 1.0	9	Save ranges from Excel documents as images
excelpy 0.1.3.5	9	Minimal Microsoft Excel 2010 library for Python 3.3.
excelt2json 1.0.1	9	transform excel file to json files
exceltowiki 0.1.19	9	Convert Excel to Wiki while maintaining formatting
Flask-Excel 0.0.5	9	A flask extension that provides one application programming interface to read and write data in different excel file formats
htsql-excel 0.1.0	9	An HTSQL extension that adds basic Excel support.
pyexcel 0.3.3	9	A wrapper library that provides one API to read, manipulate and write data in different excel formats
pyexcel-text 0.2.4	9	A plugin to pyexcel and provides the capability to present and write data in text formats
pyexcel-webio 0.0.8	9	A generic request and response interface for pyexcel web extensions.
pyExcelerator 0.6.4a	9	generating Excel 97+ files; importing Excel 95+ files; Excel files dumper; OLE2 files dumper; xls2txt, xls2csv, xls2html
pyramid-excel 0.0.3	9	A pyramid extension that provides one application programming interface to read and write data in different excel file formats
pytest-excel 1.2.0	9	pytest plugin for generating excel reports

Programming Tips

Break down a complex process into hopefully simple steps

Start simple and small, make sure it works first, then upscale

Simple but flexible solutions are usually the best

Kludges should be temporary

Borrow code when you can, be prepared to modify

Beating your head against the wall is part of the process

Use a debugger!

Using the Debugger

In Python Shell find menu item Debug, click ON Debugger

Click ON Source and Globals (Stack and Locals already checked)

Step - executes one step at a time

Over - steps over a function (executes all steps within function normally)

Go - executes rest of program normally

Out - step out of function (executes all steps within function normally); outside function = Go

Quit - terminates the program

Debugger shows you the line ABOUT to be executed

Using the Debugger

Global variables exist outside functions

- there are many global variables

- in simple programs without functions, your variables may be global

Local variables are assigned within functions or program code

Breakpoints: places to pause execution

In the file editor, right-click the line and choose "Set Breakpoint"

- line gets colored (yellow)

- program will run up to that point

- now, you can click Go to get to that point faster

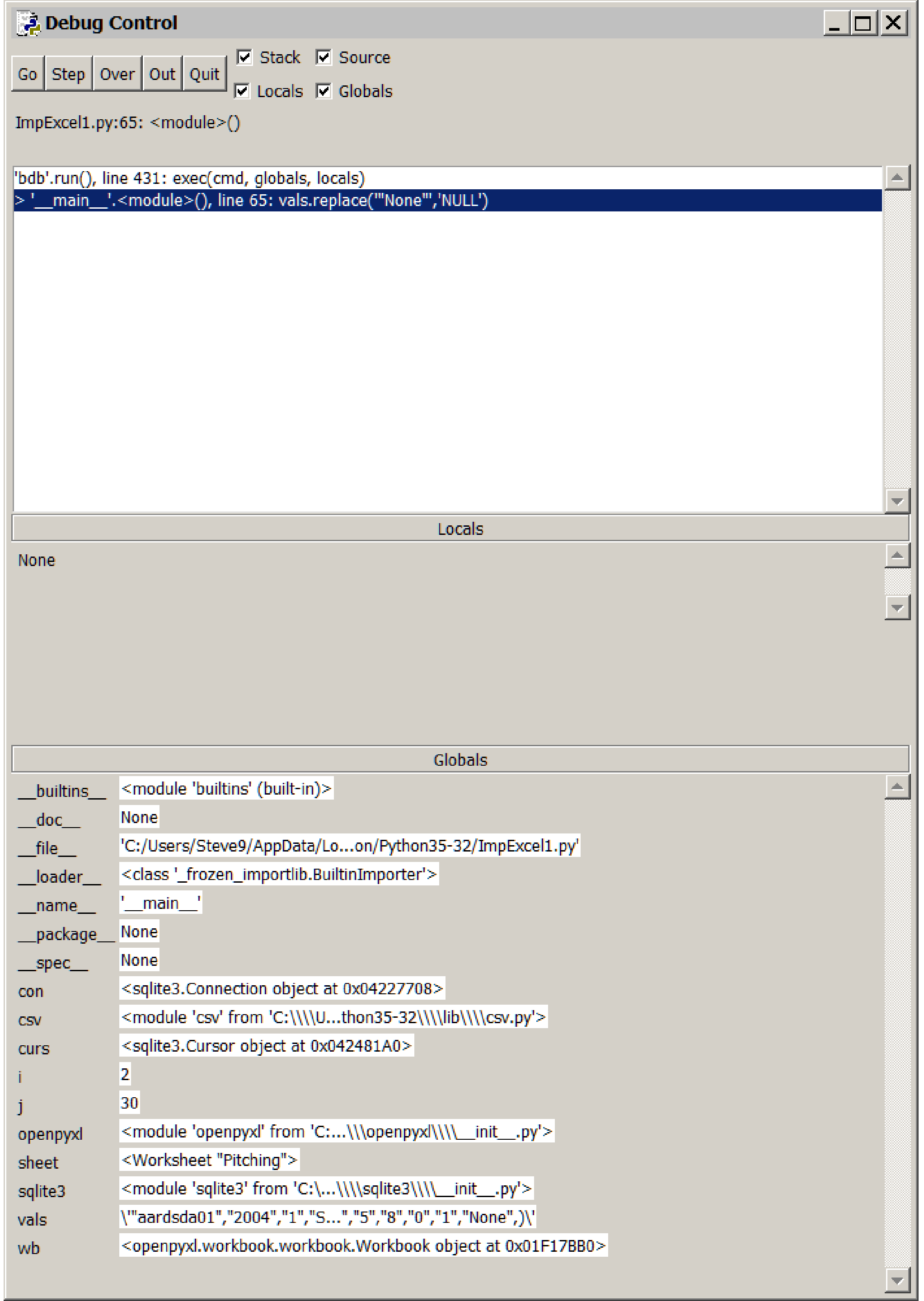
Using the Debugger

Click what you would like to see

(Stack less often)

Hanihara example

Chinese Zodiac



```
ImpExcel1.py:65: <module>()

'bdb'.run(), line 431: exec(cmd, globals, locals)
> '__main__'.<module>(), line 65: vals.replace('"None"','NULL')
```

Locals

```
None
```

Globals

__builtins__	<module 'builtins' (built-in)>
__doc__	None
__file__	'C:/Users/Steve9/AppData/Lo...on/Python35-32/ImpExcel1.py'
__loader__	<class '_frozen_importlib.BuiltinImporter'>
__name__	'__main__'
__package__	None
__spec__	None
con	<sqlite3.Connection object at 0x04227708>
csv	<module 'csv' from 'C:\\\\U...thon35-32\\\\lib\\\\csv.py'>
curs	<sqlite3.Cursor object at 0x042481A0>
i	2
j	30
openpyxl	<module 'openpyxl' from 'C:...\\\\openpyxl\\__init__.py'>
sheet	<Worksheet "Pitching">
sqlite3	<module 'sqlite3' from 'C:\\...\\\\sqlite3\\__init__.py'>
vals	\'"aardsda01","2004","1","S...","5","8","0","1","None",)\'
wb	<openpyxl.workbook.workbook.Workbook object at 0x01F17BB0>

Getting data from Excel into SQLite

Install the openpyxl module

```
Windows: Open command prompt (cmd)
```

On your computer (Windows):

```
cd "C:\Users\sousley\AppData\Local\Programs\Python\Python36-32"
python.exe -m pip install -U pip openpyxl
python.exe -m pip install -U pip pandas
```

On Lab computers:

```
cd "C:\Program Files (x86)\Python36-32\"
# Save file to same directory where you can save python scripts
At the command prompt:
"C:\Program Files (x86)\Python36-32\python.exe" -m pip install -U pip openpyxl -t c:/users/sousley/Documents
"C:\Program Files (x86)\Python36-32\python.exe" -m pip install -U pip pandas -t c:/users/sousley/Documents
In the Python console:
sys.path.append('c:/users/sousley/Documents')
```

Import Data from Excel

```
>>> import openpyxl
>>> help(openpyxl)
Help on package openpyxl:
NAME
    openpyxl - # Copyright (c) 2010-2017 openpyxl

PACKAGE CONTENTS
    cell (package)
    chart (package)
    chartsheet (package)
    comments (package)
    compat (package)
    conftest
    descriptors (package)
    drawing (package)
    formatting (package)
    formula (package)
    packaging (package)
    reader (package)
    styles (package)
    utils (package)
    workbook (package)
    worksheet (package)
    writer (package)
    xml (package)
```

```
DATA
    LXML = False
    NUMPY = True
    PANDAS = False
    __author_email__ = 'eric.gazoni@gmail.com'
    __license__ = 'MIT/Expat'
    __maintainer_email__ = 'openpyxl-users@googlegroups.com'
    __url__ = 'http://openpyxl.readthedocs.org'
    constants = {'__author__': 'See AUTHORS', '__author_email__': 'eric.ga
    here = r'C:\Users\Steve9\AppData\Local\Programs\Python\Python36-32\lib
    src = <_io.TextIOWrapper name='C:\\Users\\Steve9\\AppD...yxl\\.constan
    src_file = r'C:\Users\Steve9\AppData\Local\Programs\Python\Python36-32

VERSION
    2.4.9

AUTHOR
    See AUTHORS

FILE
    c:\users\steve9\appdata\local\programs\python\python36-32\lib\site-
packages\openpyxl\__init__.py
```

Import Data from Excel

```
>>> help (openpyxl.worksheet)
Help on package openpyxl.worksheet in openpyxl:

NAME
    openpyxl.worksheet

PACKAGE CONTENTS
    copier
    datavalidation
    dimensions
    drawing
    filters
    header_footer
    hyperlink
    merge
    page
    pagebreak
    pivot
    properties
    protection
    read_only
    related
    table
    views
    worksheet

DATA
    absolute_import = _Feature((2, 5, 0, 'alpha', 1), (3, 0, 0, 'alpha', 0...
```

Download Excel data (20)

```
# we will import from a web location
>>> import urllib3
>>> http = urllib3.PoolManager()
# retrieve file (GET to download, PUT to upload)
>>> getfile = http.request('GET', 'http://math.mercyhurst.edu/~sousley/STAT_139/data/Pitching20.xlsx')

>>> getfile
<urllib3.response.HTTPResponse object at 0x03AE6210>

# Excel is in BINARY format; save to local file
>>> with open('Pitching20.xlsx', 'wb') as output_file:
    output_file.write(bytearray(getfile.data))
10697

# open file we saved
>>> wb = openpyxl.load_workbook('Pitching20.xlsx')
>>> type(wb)
<class 'openpyxl.workbook.workbook.Workbook'>
```

Import Data from Excel (http, local)

```
# find info about the file
>>> wb.get_sheet_names()
['Pitching']

# find out the default page
>>> sheet = wb.get_active_sheet()
>>> sheet
<Worksheet "Pitching">

# open a specific page
>>> wbsheet = wb["Pitching"]


>>> wbsheet['A1'].value
'playerID'

# another way to refer to columns
>>> wbsheet.cell(row=1, column=2)
<Cell 'Pitching'.B1>

# get column names
>>> for i in range(1, 31):
    print(i, wbsheet.cell(row=1, column=i).value)
```

Export Data from Excel into sqlite

```
>>> import sqlite3
# create database
>>> con = sqlite3.connect("baseball.db")

# create cursor
>>> curs = con.cursor()

# create table
>>>bbsql = """
CREATE TABLE pitching
(playerID TEXT, yearID INTEGER, stint INTEGER, teamID TEXT, lgID TEXT,

W INTEGER, L INTEGER, G INTEGER, GS INTEGER, CG INTEGER,
SHO INTEGER, SV INTEGER, IPouts INTEGER,

H INTEGER, ER INTEGER, HR INTEGER, BB INTEGER, SO INTEGER, BAOpp INTEGER,

ERA REAL,     IBB REAL, WP INTEGER, HBP INTEGER, BK INTEGER,

BFP INTEGER, GF INTEGER, R INTEGER, SH INTEGER,

SF INTEGER, GIDP INTEGER ); """

>>> curs.execute(bbsql)
```

Export Data from Excel into sqlite

```
>>> curs.execute('insert into pitching values
("aardsda01","2015",1,"ATL","NL",1,1,33,0,0,0,0,92,25,16,6,14,35,0.221,4.7,3,1,1,0,129,9,17,0,1,NULL)')

>>>> con.commit()

>>> curs.execute('select * from pitching')

>>> curs.fetchall() # note: NULL values are displayed as None
[('aardsda01', 2015, 1, 'ATL', 'NL', 1, 1, 33, 0, 0, 0, 0, 92, 25, 16, 6, 14, 35, 0.221, 4.7, 3.0, 1, 1, 0,
129, 9, 17, 0, 1, None)]
```

Import Data from Excel

```
# ImpExcel1.py
import sqlite3
import openpyxl

wb = openpyxl.load_workbook('Pitching20.xlsx')
wbsheet = wb["Pitching"]

# Let's read from the excel file in a script and make sure things look correct
vals = ''

for i in range(2, 11): # rows - 2 is first record
    for j in range(1,31): # columns
        print(i, j, wbsheet.cell(row=i, column=j).value)
        vals = vals + '"' + str(wbsheet.cell(row=i, column=j).value) + '",'

    # add right paren
    vals = vals + ')'
    # replace "None" with NULL
    vals.replace('None','NULL')

    # replace last comma
    vals.replace(',)',')')

    print ('insert into pitching values (' + vals)
    vals = ''
```

Import Data from Excel

```
# Ran script, set breakpoint
# database is in memory
# open database
>>> con = sqlite3.connect("baseball.db")

# create cursor
>>> curs = con.cursor()

>>> curs.execute('select * from pitching')
<sqlite3.Cursor object at 0x03A636E0>
>>> curs.fetchall()
[('aardsda01', 2015, 1, 'ATL', 'NL', 1, 1, 33, 0, 0, 0, 0, 92, 25, 16, 6, 14, 35, 0.221, 4.7,
3.0, 1, 1, 0, 129, 9, 17, 0, 1, None)]
>>>
```

Import Data from Excel

```
# ImpExcel2.py : replace would not work no matter what
# added cv
import sqlite3
import openpyxl
wb = openpyxl.load_workbook('Pitching20.xlsx')
wbsheet = wb["Pitching"]
vals = ''
cv = ''
for i in range(2, 11): # rows - 2 is first record
    for j in range(1,31): # columns
        cv = str(wbsheet.cell(row=i, column=j).value)
        print(str(i), str(j),cv)
        if cv.strip() != 'None':
            vals = vals + '"' + str(cv) + '",'
        else: vals = vals + 'NULL,'

    # add right paren
    vals = vals + ')'
    # replace "None" with NULL
    # vals.replace('None','NULL')
    # print(vals)
    # replace last comma - but does not work!
    vals.replace(',)',')')

    print ('insert into pitching values (' + vals)
    print ('insert into pitching values (' + vals[0:len(vals)-2] + ')' ) # a workaround. a kludge?

    vals = ''
```

Import Data from Excel

```
# ImpExcel3.py : refined SQL INSERT code
import sqlite3
import openpyxl
wb = openpyxl.load_workbook('Pitching20.xlsx')
wbsheet = wb["Pitching"]
con = sqlite3.connect("baseball.db")
curs = con.cursor()

vals = vals = ''
cv = ''

for i in range(2, 15): # rows - 2 is first record
    for j in range(1,31): # columns
        cv = str(wbsheet.cell(row=i, column=j).value)
        print(str(i), str(j),cv)
        # replace "None" with NULL
        if cv.strip() != 'None':
            vals = vals + '"' + str(cv) + '",'
        else: vals = vals + 'NULL,'

    # add right paren
    vals = vals + ')'

    print ('insert into pitching values (' + vals[0:len(vals)-2] + ')' )
    curs.execute('insert into pitching values (' + vals[0:len(vals)-2] + ')' )
    vals = ''

con.commit()
```

Download Excel data (ALL)

```
# we will import from a web location
>>> import urllib3
>>> http = urllib3.PoolManager()
# retrieve file (GET to download, PUT to upload)
>>> getfile = http.request('GET', 'http://math.mercyhurst.edu/~sousley/STAT_139/data/Pitching.xlsx')

>>> getfile
<urllib3.response.HTTPResponse object at 0x03AE6210>

# Excel is in BINARY format; save to local file
>>> with open('Pitching.xlsx', 'wb') as output_file:
    output_file.write(bytearray(getfile.data))

6838995

# open file we saved
>>> wb = openpyxl.load_workbook('Pitching20.xlsx')
>>> type(wb)
<class 'openpyxl.workbook.workbook.Workbook'>
```

Import Data from Excel

```
# ImpExcel4.py : run on all records, provide feedback
import sqlite3
import openpyxl
wb = openpyxl.load_workbook('Pitching.xlsx')
wbsheet = wb["Pitching"]
con = sqlite3.connect("baseball.db")
curs = con.cursor()

# then...
```

Import Data from Excel

```
# ImpExcel4.py : run on all records, provide feedback
# insert code from previous slide
vals = ''
cv = ''

for i in range(2, 10001): # rows - 2 is first record
    for j in range(1,31): # columns
        cv = str(wbsheet.cell(row=i, column=j).value)
        #print(str(i), str(j),cv)
        if cv.strip() != 'None':
            vals = vals + '"' + str(cv) + '",'
        else: vals = vals + 'NULL,'

    # add right paren
    vals = vals + ')'
    print(str(i))
    # this actually does it
    # print ('insert into pitching values (' + vals[0:len(vals)-2] + ')' )
    curs.execute('insert into pitching values (' + vals[0:len(vals)-2] + ')' )
    vals = ''

con.commit()
curs.execute ('select count(*) from pitching')
print(curs.fetchall())
```

to be continued...

SQL vs. Excel:

I want to analyze statistics from pitchers, wins and losses

How many total games or wins and losses will be enough to provide meaningful data?

ERA vs. Wins?

ERA vs. winning percentage?

```
curs.execute('select * from pitching where W+L > 30')
```

NumPy

Anaconda