Exploring Python and its Variants

By John Apodaca

August 19th, 2018

First, I knew I wanted to work on two projects: Animation and Parallel processing.

I had Python 3.7 and iPython installed and started using the Idle IDE but wanted something a little more user friendly. Using Youtube.com, several resources appeared with the search 'Python animation'.

AGodboldMath produced a video (https://www.youtube.com/watch?v=j2VjzShr4UY (https://www.youtube.com/watch?v=j2VjzShr4UY)) titled "Introduction to Using the Calico Editor and Shell with Python to Control a Scribbler Robot" in 2013 which showcased both an IDE and a cute robot phython library (Myro) that I thought might be interesting for the OWL students.

The library looked limited but the IDE appeared promising. Any searches for the "Calico project" ended up in a networking specialized python library but a Wiki search produced (http://wiki.roboteducation.org/Calico_Download) (http://wiki.roboteducation.org/Calico_Download)).

The instructions were:

```
To run Calico on Windows:
Install Calico Software
Go to http://myro.roboteducation.org/~dblank/download/?C=N;O=D;P=*-window
s-all.zip
Get the highest-numbered number that ends in "-windows-all.zip"
Unzip it, and put the contents on, say, your desktop
Start Calico
In the folder Calico, run the file calico.bat
```

The file was: Calico-4.0.1-windows-all.zip 19-Oct-2016 13:27 212M

A folder "Calico IDE" was created to copy the contents of this .zip file into.

```
Calico is currently under development. It contains many components:
Choice of languages, which can inter-operate: Python, Jigsaw (a graphical
 language inspired by Scratch), Ruby, Scheme, Java, F# (OCaml and ML), Bo
o, Lisp, Basic, Logo, and more under
development...
Selection of interesting libraries: Myro (robot control, music, sound), P
rocessing (art and animation), Graphics (physics, turtle graphics, and ga
mes), Kinect, and more
Editor - simple, but powerful customizable editor
Shell - integrated languages
Chat - communications framework for talking and sharing
```

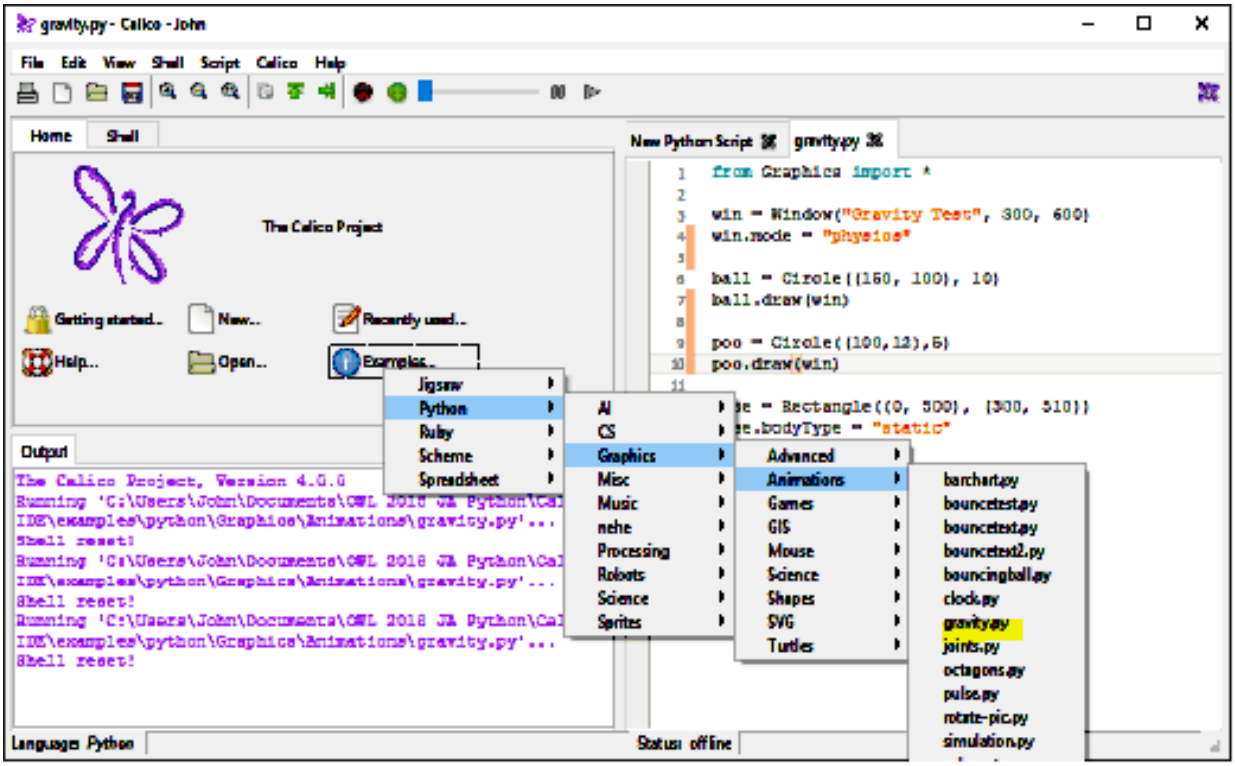
Next, I was curious about Myro (from Myro import *).

To run Calico, run the Calico.bat file, Click on the "Shell" tab (as "Home" is not needed).

Doug Blank published a Youtube.com video , "VPython as a Jupyter kernel" https://www.youtube.com/watch?v=jRAKBJWOAZE (https://www.youtube.com/watch?v=jRAKBJWOAZE) that has an animation in a Jupyter notebook. That is where I'd like to be!!

While playing with Calico I discovered some example programs like "gravity.py" under Shell--> Examples--> Python--> Graphics--> Animations--> gravity.py

In [33]:
```
%%HTML
<br><p>
<img src="Gravitysnapshot.png" alt="Smiley face" height="1200" width="800"> </p><
```



Also, note "Myro supports the Zelle graphics library..."(5) so windows and objects are easier. Well, actually it is not the actual Zelle library but something similar (discovered as some constants and member functions don't seem to actually exist, i.e. 'yellow' and setFill()). The documentation says colors are predefined but they are not. Do something like the following:

```
yellow    = makeColor(255, 255,   0)
win.setBackground(yellow)
```

To include packages in Anaconda as external paths, refer to

https://stackoverflow.com/questions/37006114/anaconda-permanently-include-external-packages-like-in-pythonpath (https://stackoverflow.com/questions/37006114/anaconda-permanently-include-external-packages-like-in-pythonpath)

This video suggests you can load libraries (like Calico) in Anaconda (as a notebook?)

https://www.youtube.com/watch?v=jRAKBJWOAZE (https://www.youtube.com/watch?v=jRAKBJWOAZE)

The intention here is to explore the Python language from an applications perspective and not just within Jupyter. i.e What can I do with it? I begin with looking at scientific plots.

- [x] Graphing Data
- [x] Program Animation (Calico?)
- [x] Show Images
- [x] Equations
- [] Web *Scraping* in Python
- [] Grabbing Scientific Data from Web
- [] Parallel Programming Resources References

Notes: Case sensitive, indents important, no need for variable typing, In iPython up arrow repeats command, Ctrl-D quits interpreter

Graphing Data

The following example is From :Intro to Data Analysis / Visualization with Python, Matplotlib and Pandas | Matplotlib Tutorial https://www.youtube.com/watch?v=a9UrKTVEeZA (https://www.youtube.com/watch?v=a9UrKTVEeZA)

In [1]:
```
import pandas as pd
```

In [2]:
```
from matplotlib import pyplot as plt
```

In [3]:
```
x = [1,2,3]
y = [1,4,9]
z = [10, 5, 0]
plt.plot(x,y)
plt.plot(x,z)
plt.title("Random Shit")
plt.xlabel("x")
plt.ylabel("y and z")
plt.legend(["This is y","This is z"])
plt.show()
```

In [4]:
```
sample_Data = pd.read_csv('sample_data.csv')
```

In [5]:
```
sample_Data
```

Out[5]:

	column_a	column_b	column_c
0	1	1	10
1	2	4	8
2	3	9	6
3	4	16	4
4	5	25	2

In [6]:
```
type(sample_Data)
```

Out[6]:
```
pandas.core.frame.DataFrame
```

In [7]:
```
sample_Data.column_c
```

Out[7]:
```
0    10
1     8
2     6
3     4
4     2
Name: column_c, dtype: int64
```

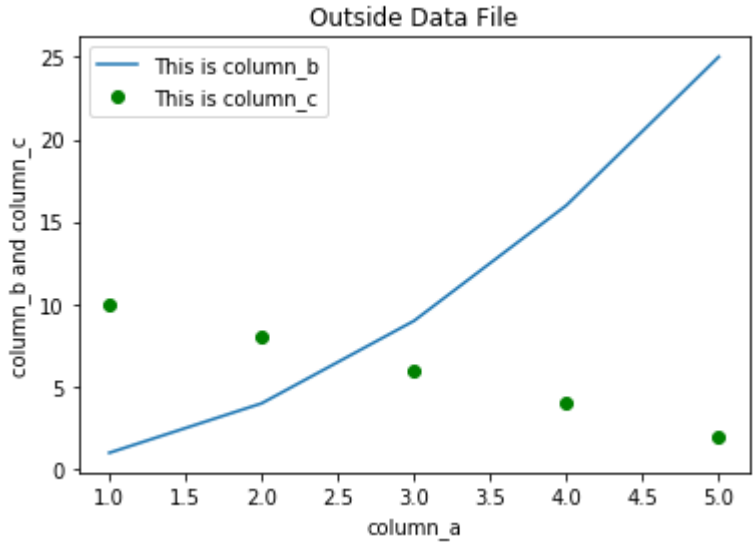
In [8]:
```
type(sample_Data.column_a)
```

Out[8]:
```
pandas.core.series.Series
```

In [9]:
```
sample_Data.column_a.iloc[1]
```

Out[9]:
```
2
```

In [10]:
```
plt.plot(sample_Data.column_a,sample_Data.column_b,'-')
plt.plot(sample_Data.column_a,sample_Data.column_c,'g o')
plt.title("Outside Data File")
plt.xlabel("column_a")
plt.ylabel("column_b and column_c")
plt.legend(["This is column_b","This is column_c"])
plt.show()
```



In [11]:
```
pd.read_csv('countries.csv')
```

Out[11]:

	country	year	population
0	Afghanistan	1952	8425333
1	Afghanistan	1957	9240934
2	Afghanistan	1962	10267083
3	Afghanistan	1967	11537966
4	Afghanistan	1972	13079460
5	Afghanistan	1977	14880372
6	Afghanistan	1982	12881816
7	Afghanistan	1987	13867957
8	Afghanistan	1992	16317921
9	Afghanistan	1997	22227415
10	Afghanistan	2002	25268405
11	Afghanistan	2007	31889923
12	Albania	1952	1282697
13	Albania	1957	1476505
14	Albania	1962	1728137
15	Albania	1967	1984060
16	Albania	1972	2263554
17	Albania	1977	2509048
18	Albania	1982	2780097
19	Albania	1987	3075321
20	Albania	1992	3326498
21	Albania	1997	3428038
22	Albania	2002	3508512
23	Albania	2007	3600523
24	Algeria	1952	9279525
25	Algeria	1957	10270856
26	Algeria	1962	11000948
27	Algeria	1967	12760499
28	Algeria	1972	14760787
29	Algeria	1977	17152804
...	...	...	...
1674	Yemen, Rep.	1982	9657618
1675	Yemen, Rep.	1987	11219340
1676	Yemen, Rep.	1992	13367997
1677	Yemen, Rep.	1997	15826497

	country	year	population
1678	Yemen, Rep.	2002	18701257
1679	Yemen, Rep.	2007	22211743
1680	Zambia	1952	2672000
1681	Zambia	1957	3016000
1682	Zambia	1962	3421000
1683	Zambia	1967	3900000
1684	Zambia	1972	4506497
1685	Zambia	1977	5216550
1686	Zambia	1982	6100407
1687	Zambia	1987	7272406
1688	Zambia	1992	8381163
1689	Zambia	1997	9417789
1690	Zambia	2002	10595811
1691	Zambia	2007	11746035
1692	Zimbabwe	1952	3080907
1693	Zimbabwe	1957	3646340
1694	Zimbabwe	1962	4277736
1695	Zimbabwe	1967	4995432
1696	Zimbabwe	1972	5861135
1697	Zimbabwe	1977	6642107
1698	Zimbabwe	1982	7636524
1699	Zimbabwe	1987	9216418
1700	Zimbabwe	1992	10704340
1701	Zimbabwe	1997	11404948
1702	Zimbabwe	2002	11926563
1703	Zimbabwe	2007	12311143

1704 rows × 3 columns

In [12]:
```
data = pd.read_csv('countries.csv')
```

In [13]:
```
# Compare the population growth in the US and China
```

In [14]:
```
us = data[data.country == 'United States']
```

In [15]:
```
# (Fields are case sensitive)
```

In [16]:
```
us
```

Out[16]:

	country	year	population
1608	United States	1952	157553000
1609	United States	1957	171984000
1610	United States	1962	186538000
1611	United States	1967	198712000
1612	United States	1972	209896000
1613	United States	1977	220239000
1614	United States	1982	232187835
1615	United States	1987	242803533
1616	United States	1992	256894189
1617	United States	1997	272911760
1618	United States	2002	287675526
1619	United States	2007	301139947

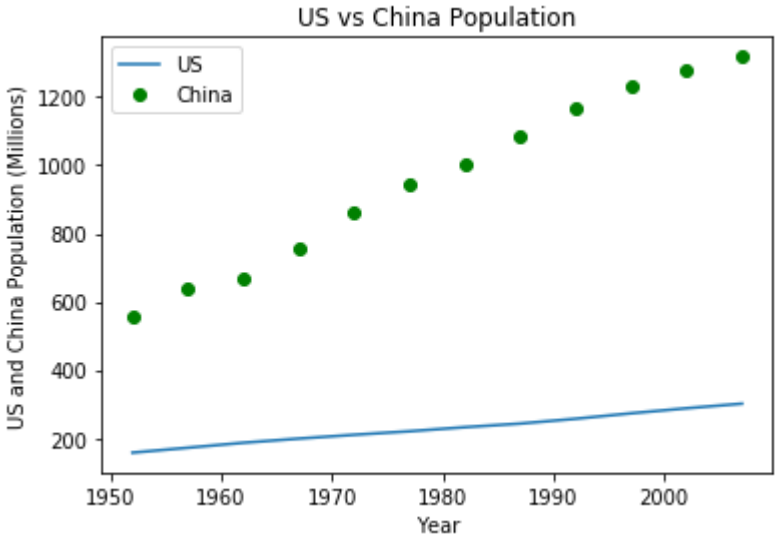
In [17]:
```
china = data[data.country == 'China']
```

In [18]:
```
china
```

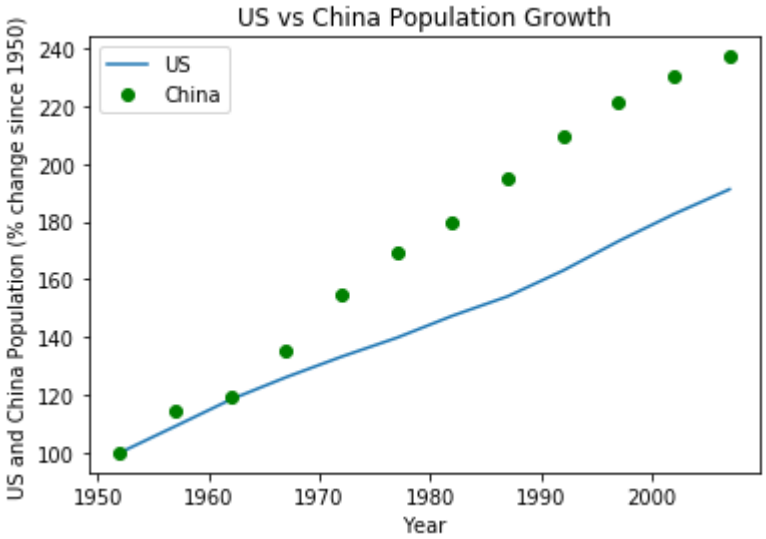
Out[18]:

	country	year	population
288	China	1952	556263527
289	China	1957	637408000
290	China	1962	665770000
291	China	1967	754550000
292	China	1972	862030000
293	China	1977	943455000
294	China	1982	1000281000
295	China	1987	1084035000
296	China	1992	1164970000
297	China	1997	1230075000
298	China	2002	1280400000
299	China	2007	1318683096

In [19]:
```
plt.plot(us.year,us.population / 10**6,'-')
plt.plot(china.year,china.population / 10**6,'g o')
plt.title("US vs China Population")
plt.xlabel("Year")
plt.ylabel("US and China Population (Millions)")
plt.legend(["US","China"])
plt.show()
```



In [20]:
```
plt.plot(us.year,us.population / us.population.iloc[0] * 100,'-')
plt.plot(china.year,china.population / china.population.iloc[0] * 100,'g o')
plt.title("US vs China Population Growth")
plt.xlabel("Year")
plt.ylabel("US and China Population (% change since 1950)")
plt.legend(["US","China"])
plt.show()
```



Program Animation (Calico?)

So, my take on expressing animations in Jupyter, is write the code and execute it outside Jupyter and post it here as a recording (video) with the non-executable code beneath it because Jupyter

ultimately wants html. Unless you're using matplotlib to express a math concept. ;(

create a startup script: http://people.duke.edu/~ccc14/cspy/Customizing_Jupyter.html (http://people.duke.edu/~ccc14/cspy/Customizing_Jupyter.html)

In [ ]:
```
from Calico import *

from Graphics import *

win = Window("Gravity Test", 300, 600)

win.mode = "physics"
yellow    = makeColor(255, 255,   0)
win.setBackground(yellow)


ball = Circle((150, 100), 10)

ball.draw(win)

poo = Circle((100,12),5)
poo.draw(win)



base = Rectangle((0, 500), (300, 510))

base.bodyType = "static"

base.draw(win)



win.run()
```

Show Images

The double % sign serves to run another language command. (You can create a variable and set it equal to the output of the code snippet.)

In [21]:
```
%%HTML
<video width="300" height="632" controls>
  <source src="GravityVideo.mp4" type="video/mp4">
</video>
```

In [22]:
```
mystr1 = "Hello World"
```

Must "run" each line that is being tested. Can do this by keyboard using - Press after the "." to see that object's member functions. **This along with the line counts (In [ ]) is the "i" in "iPython".**

In [23]:
```
mystr1.upper()
```

Out[23]:
```
'HELLO WORLD'
```

Run Shell commands right from the notebook!

In [24]:
```
!ping www.google.com
```

```
Pinging www.google.com [172.217.2.4] with 32 bytes of data:
Reply from 172.217.2.4: bytes=32 time=26ms TTL=56
Reply from 172.217.2.4: bytes=32 time=26ms TTL=56
Reply from 172.217.2.4: bytes=32 time=25ms TTL=56
Reply from 172.217.2.4: bytes=32 time=25ms TTL=56

Ping statistics for 172.217.2.4:
    Packets: Sent = 4, Received = 4, Lost = 0 (0% loss),
Approximate round trip times in milli-seconds:
    Minimum = 25ms, Maximum = 26ms, Average = 25ms
```

In [ ]:
```
!dir
```

Look up GlowScript.org for vPython vector fun! https://www.youtube.com/watch?v=o8OFCsr_Ktc (https://www.youtube.com/watch?v=o8OFCsr_Ktc)

In [5]:
```
from socket import gethostname; print gethostname()
```

```
  File "<ipython-input-5-4ae358ab4b93>", line 1
    from socket import gethostname; print gethostname()
                                                      ^
SyntaxError: invalid syntax
```

By the way, we can insert equations too!

$$\dot{x} = \sigma(y - x)$$
$$\dot{y} = \rho x - y - xz$$
$$\dot{z} = -\beta z + xy$$

In [35]:
```
!calico locate
```

```
'calico' is not recognized as an internal or external command,
operable program or batch file.
```

New Experimental Area

First Attempt to get outside information -- Scraping

In [1]:
```
import requests
```

In [2]:
```
url = "https://www.wikipedia.org/"
```

In [4]:
```
r = requests.get(url)
```

In [5]:
```
text = r.text
```

In [6]:
```
print(text)
```

```
<!DOCTYPE html>
<html lang="mul" class="no-js">
<head>
<meta charset="utf-8">
<title>Wikipedia</title>
<meta name="description" content="Wikipedia is a free online encyclopedia, cr
eated and edited by volunteers around the world and hosted by the Wikimedia F
oundation.">
<![if gt IE 7]>
<script>
document.documentElement.className = document.documentElement.className.repla
ce( /(^|\s)no-js(\s|$)/, "$1js-enabled$2" );
</script>
<![endif]>
<!--[if lt IE 7]><meta http-equiv="imagetoolbar" content="no"><![endif]-->
<meta name="viewport" content="initial-scale=1,user-scalable=yes">
<link rel="apple-touch-icon" href="/static/apple-touch/wikipedia.png">
<link rel="shortcut icon" href="/static/favicon/wikipedia.ico">
<link rel="license" href="//creativecommons.org/licenses/by-sa/3.0/">
```

Second Attempt

from : (https://journalistsresource.org/tip-sheets/research/python-scrape-website-data-criminal-justice (https://journalistsresource.org/tip-sheets/research/python-scrape-website-data-criminal-justice)) Also, (https://www.dataquest.io/blog/web-scraping-tutorial-python/ (https://www.dataquest.io/blog/web-scraping-tutorial-python/))

In [2]:
```
import requests
```

In [3]:
```
from bs4 import BeautifulSoup
```

In [4]:
```
url_to_scrape = 'http://apps2.polkcountyiowa.gov/inmatesontheweb/'
```

In [5]:
```
r = requests.get(url_to_scrape)
```

In [11]:
```
soup = BeautifulSoup(r.text, 'html.parser')
```

In [12]:
```
print(soup.prettify())
```

```
<!DOCTYPE html>
<html>
 <head>
  <meta charset="utf-8"/>
  <meta content="width=device-width, initial-scale=1.0" name="viewport"/>
  <style>
  </style>
  <title>
   Home Page - Polk County Current Inmates
  </title>
  <link href="/PolkCountyInmates/CurrentInmates/Content/bootstrapCss?v=aUM4qc
pvk13whEIs_e_mcrsyGq-OENX_GsGeFMhad1Q1" rel="stylesheet"/>
  <link href="/PolkCountyInmates/CurrentInmates/Content/dataTablesCss?v=NE6kU
2CizulbWAt1ON9toirYl-DcuTMYAlzuOy9fDTg1" rel="stylesheet"/>
  <link href="/PolkCountyInmates/CurrentInmates/Content/siteCss?v=71X3v0eC53J
ynwCheHP_-0AlZjVp1UJIaaYBU0DHJTc1" rel="stylesheet"/>
  <script src="/PolkCountyInmates/CurrentInmates/bundles/modernizr?v=inCVuEFe
6J4Q07A0AcRsbJic_UE5MwpRMNGcOtk94TE1">
  </script>
```

Now that I have the page, time to tease out the data.

In [13]:
```
list(soup.children)
```

Out[13]:
```
['html', '\n', <html>
 <head>
 <meta charset="utf-8"/>
 <meta content="width=device-width, initial-scale=1.0" name="viewport"/>
 <style>
 </style>
 <title>Home Page - Polk County Current Inmates</title>
 <link href="/PolkCountyInmates/CurrentInmates/Content/bootstrapCss?v=aUM4qcp
vk13whEIs_e_mcrsyGq-OENX_GsGeFMhad1Q1" rel="stylesheet"/>
 <link href="/PolkCountyInmates/CurrentInmates/Content/dataTablesCss?v=NE6kU2
CizulbWAt1ON9toirYl-DcuTMYAlzuOy9fDTg1" rel="stylesheet"/>
 <link href="/PolkCountyInmates/CurrentInmates/Content/siteCss?v=71X3v0eC53Jy
nwCheHP_-0AlZjVp1UJIaaYBU0DHJTc1" rel="stylesheet"/>
 <script src="/PolkCountyInmates/CurrentInmates/bundles/modernizr?v=inCVuEFe6
J4Q07A0AcRsbJic_UE5MwpRMNGcOtk94TE1"></script>
 </head>
 <body>
 <div class="navbar navbar-default navbar-static-top ">
 <div class="container-fluid">
```

In [15]:
```
[type(item) for item in list(soup.children)]
```

Out[15]:
```
[bs4.element.Doctype,
 bs4.element.NavigableString,
 bs4.element.Tag,
 bs4.element.NavigableString]
```

In [16]:
```
soup.find_all('p')
```

Out[16]:
```
[<p> <b>Disclaimer:</b>
 Record of an arrest is not an indication of guilt. The Polk County Sheriff’s O
ffice does not provide case disposition. Disposition of cases can be searched <
a href="https://www.iowacourts.state.ia.us/ESAWebApp/DefaultFrame">
     here</a>.
 </p>, <p>
     The Polk County Sheriff’s Office does not expressly or by implication warr
ant that the information or data accessed by the customer is accurate or correc
t. The Sheriff is not liable for any loss, cost, damage or expense arising dire
ctly or indirectly in connection with this access. In no event shall the Sherif
f be liable for any special or consequential damages or for any direct damages
resulting from the customer’s use or application of the information obtained as
a result of using this web site.

     Individuals obtaining information from this web site should verify accurac
y through the arresting agency or <a href="https://www.iowacourts.state.ia.us/E
SAWebApp/DefaultFrame">Iowa Courts Online</a>.
 </p>, <p>
 <b><font color="red"> Information provided should not be relied upon for any t
ype of legal action.</font></b>
 </p>, <p>© 2018 - Polk County Iowa Government</p>]
```

In [20]:
```
soup.find_all('table', class_='dataTable')
```

Out[20]:
```
[<table class="dataTable table table-bordered table-striped " width="100%">
 <thead>
 <tr>
 <th></th>
 <th>Last Name</th>
 <th>First Name</th>
 <th>Age</th>
 <th>Book Date</th>
 </tr>
 </thead>
 <tbody></tbody>
 </table>]
```

Third Attempt...

TV Listings

In [50]:
```
import requests
from bs4 import BeautifulSoup
url_to_scrape = 'https://www.titantv.com/Default.aspx?r=t'
r = requests.get(url_to_scrape)
soup = BeautifulSoup(r.text, 'html.parser')
# print(soup.prettify())
```

In [51]:
```
soup
```

Out[51]:
```
<!DOCTYPE html>

<html xmlns:fb="http://ogp.me/ns/fb#">
<head><title>
        TitanTV – Free Local TV Listings, Program Schedule, Show and Episode
</title><meta content="TitanTV offers fast, customizable TV listings for loca
l broadcasting, cable and satellite lineups. Quickly view program, episode, c
ast credits, and additional airing information." name="description"/>
<meta content="TitanTV, 300 Collins Rd NE, Suite B, Cedar Rapids, Iowa 52402"
name="Author"/>
<meta content="ALL" name="ROBOTS"/>
<meta content="tv guide, tv listings, tvguide, tv guide listings, tv listing,
tvguide listings, local tv listings, tvlistings, tv guide listing, t v guide,
t.v. guide, local tv guide, tv channel guide, tv guide schedule, tv schedule,
tv guide online, tvlisting, t.v guide, television guide, online tv guide, tv
schedules, tv lineup, channel listings, free tv guide, tv channel listings, t
v program guide, tv guides, local tv schedule, tv guide local listings, t.v.
listings, cable tv listings, local tv listing, t v listings, television listi
```

In [58]:
```
bob = soup.find('span', class_='cdt')
print(bob.get.text())
```

```
---------------------------------------------------------------------------
AttributeError                            Traceback (most recent call last)
<ipython-input-58-ec3f241e783e> in <module>()
      1 bob = soup.find('span', class_='cdt')
----> 2 print(bob.get.text())

AttributeError: 'function' object has no attribute 'text'
```

9/ 22/ 2018 Okay, so it has been a week since my last foray into this overwhelming mix of Python variants and Python communities. I'm rusty already. Or, at least, I am having trouble using what I've been exposed to -- notice I did not say "what I've learned" because I have not yet really learned anything I can play with. I've encountered too many errors. For example, importing the "Graphics" library is not the same as importing the "graphics" library and the variants of Python are inconsistent with how either library is treated. And, using either library with the "Calico" compiler is different from using it in either Anaconda or "Idle".

I got ambitious trying to use all three variants at the same time (i.e. Anaconda, Idle, and Calico). I was hoping to do some data mining, graphic programming, and just learning Phython construction techniques and syntax.

I am now stepping back and trying to get a solid understanding of Phython within the confines of Calico. I want to do some fun graphical work with an eye to showing it to the OWL students I work with. I'll return to the other pursuits (data mining, parallel programming, etc) later.

Here are a few notes of what I learned today.

In Calico, use (by the way, use three tildes before and after a "code" snippet)

```
from Graphics import *
```

to do things like

```
win = Window(500,500)
```

which creates a window for your program and subsequent graphics.

Take the statements,

```
circle = Circle((250, 250),50)
circle.fill = Color("blue")
```

This is creating a variable called "circle" (object or a handle to an object...I think) that takes a point (the center) and the first arguement and a radius. Then we can use the "methods" of that object to define some of the parameters of that object. And, this is what I found very interesting. I have to use a function, defined in the graphics library, called "Color" to pass a named color to the "fill" method of the circle object. (I had been struggling with the thought of saying something like circle.fill = "blue" and being surprised that it failed.)

Also, a bit of coolness. In the Python community it seems obvious that to make an object appear in your defined window you have to use the "draw" method thus

```
circle.draw(win)
```

But, wait, there's more! If you want another object to appear within the confines of some other object already in the window (or another window), you do this

```
circle.draw(other_Object)
```

or, more specifically

```
circle.draw(square)
```

Let's talk about using import. This is what I learned today. To keep you code short and simple, invoke import thus

```
from random import *
```

This allows you to write code like

```
x = randint(0, 500)             # Create a random x value
```

else if you write

```
import random
```

then you would need to write

```
x = random.randint(0, 500)             # Create a random x value
```

I'd like to avoid

```
import random

for i in range (10):
    x = random.randint(0, 500)
    print(x)
```

and write

```
from random import *

for i in range (10):
    x = randint(0, 500)
    print(x)
```

It doesn't look like it saves much here but when complicated object calls are included, it begins to make a difference.

Triangle, unlike Rectangle and Circle, does not seem to be a keyword in the Graphics library. instead, use Polygon

```
triangle = Polygon((0,0),(100,100),(100,200))
```

*** Just learned that the Calico documentation sites are ALL GONE.

This was fun.

```
for month in ("Jan", "Feb", "Mar","Apr"):
    for day in range(30):
        print(month, day + 1)
```

In [ ]:

In [ ]:

In [ ]:

Suggested Further Topics for Review

Check out : (in Youtube)

```
    Cellular Automata with Python (Jupyter Notebook)

    EuroSciPy 2017: Interactive 3D Visualization in Jupyter Notebooks

    Time Series Data Visualization Using Matplotlib and Seaborn in Python
- Tutorial 10

    Python animation example.

    Randall J. LeVeque - Writing a Book in Jupyter Notebooks

    Python animations

    Python Matplotlib animation: Planetary orbits

    Load package for Astronomical calculations [ephem] from nbextensions
```

Resources

Discovered along the way... Google python course Support materials and exercises: https://goo.gl/ztTFg (https://goo.gl/ztTFg)

Calico is a multi-language development Environment![7] As such it has SPECIAL libraries like the Graphics importable library that is written in C++. You might be able to add this to your own Python variant but don't depend on it.

I was using it because it had a particularly friendly, easy way to create a window for graphics. (Figures it would be abandoned.)

References

1. Myro Documentation (discontinued as of 9/23/2018) http://wiki.roboteducation.org/Myro_Reference_Manual (http://wiki.roboteducation.org/Myro_Reference_Manual)
2. Python Software Foundation https://docs.python.org/3/ (https://docs.python.org/3/) or www.python.org (http://www.python.org)
3. "Calico project" Wiki. Calico: a multi-programming-language, multicontext framework designed for computer science education http://wiki.roboteducation.org/Calico_Download (http://wiki.roboteducation.org/Calico_Download)
4. Intro to Data Analysis / Visualization with Python, Matplotlib and Pandas | Matplotlib Tutorial https://www.youtube.com/watch?v=a9UrKTVEeZA (https://www.youtube.com/watch?v=a9UrKTVEeZA)
5. To include packages in Anaconda as external paths, refer to https://stackoverflow.com/questions/37006114/anaconda-permanently-include-external-

packages-like-in-pythonpath (https://stackoverflow.com/questions/37006114/anaconda-permanently-include-external-packages-like-in-pythonpath)

6. This video suggests you can load libraries (like Calico) in Anaconda (as a notebook?) https://www.youtube.com/watch?v=jRAKBJWOAZE (https://www.youtube.com/watch?v=jRAKBJWOAZE)
7. More on Calico https://repository.brynmawr.edu/cgi/viewcontent.cgi?referer=&httpsredir=1&article=1031&context=compsci_pubs (https://repository.brynmawr.edu/cgi/viewcontent.cgi?referer=&httpsredir=1&article=1031&context=compsci_pubs)
8. When I finally give up on Calico, due to no documentation, try PyScripter http://code.google.com/p/pyscripter (http://code.google.com/p/pyscripter) and http://openbookproject.net/thinkcs/python/english3e/ (http://openbookproject.net/thinkcs/python/english3e/)