~~Batman's~~ Developer's Utility Belt

By Rajat Goyal

@rajat404

Shell Utilities

Some language agnostic utilities, which all developers can benefit from

To the terminal and beyond...

- Tig
- mycli/pgcli
- s
- howdoi

pip-tools

https://github.com/jazzband/pip-tools

```
$ pip-compile  # To initialize `requirements.in`
$ pip-compile requirements.in
```

Jupyter Extensions

- https://github.com/Jupyter-contrib/jupyter_nbextensions_configurator
- http://jupyter-contrib-nbextensions.readthedocs.io/en/latest/nbextensions.html

hr

In [3]:
```
from hr import hr
hr()
print('it works!')
hr('*')
```

```
################################################################################
####
it works!
********************************************************************************
```

Flower

Celery monitoring

Arrow

Lets you create, manipulate and format Time ~~& Space~~

https://arrow.readthedocs.io/en/latest/

(examples from docs)

In [4]:

```
import arrow
print('arrow.utcnow()            ', arrow.utcnow())
print('arrow.now()               ', arrow.now())
print('arrow.now("Asia/Kolkata") ', arrow.now('Asia/Kolkata'))
print('arrow.now("US/Pacific")   ', arrow.now('US/Pacific'))
```

```
arrow.utcnow()            2017-06-26T08:56:54.529540+00:00
arrow.now()               2017-06-26T14:26:54.529930+05:30
arrow.now("Asia/Kolkata") 2017-06-26T14:26:54.531383+05:30
arrow.now("US/Pacific")   2017-06-26T01:56:54.532811-07:00
```

Read from multiple formats

In [5]:

```
from dateutil import tz
from datetime import datetime
print("arrow.get(1367900664)", '\n', arrow.get(1367900664), '\n')
print("arrow.get('2013-05-05 12:30:45', 'YYYY-MM-DD HH:mm:ss')", '\n', arrow.get('2013-05-05 12:30:45', '
YYYY-MM-DD HH:mm:ss'), '\n')
print("arrow.get('June was born in May 1980', 'MMMM YYYY')", '\n', arrow.get('June was born in May 1980'
, 'MMMM YYYY'), '\n')
# Some ISO-8601 compliant strings are recognized and parsed without a format string:
print("arrow.get('2013-09-30T15:34:00.000-07:00')", '\n', arrow.get('2013-09-30T15:34:00.000-07:00'), '\n')
```

```
arrow.get(1367900664)
 2013-05-07T04:24:24+00:00

arrow.get('2013-05-05 12:30:45', 'YYYY-MM-DD HH:mm:ss')
 2013-05-05T12:30:45+00:00

arrow.get('June was born in May 1980', 'MMMM YYYY')
 1980-05-01T00:00:00+00:00

arrow.get('2013-09-30T15:34:00.000-07:00')
 2013-09-30T15:34:00-07:00
```

Directly create Arrow objects

In [6]:
```
print("arrow.get(2013, 5, 5)", '\n', arrow.get(2013, 5, 5), '\n')
print("arrow.Arrow(2013, 5, 5)", '\n', arrow.Arrow(2013, 5, 5), '\n')
```

```
arrow.get(2013, 5, 5)
 2013-05-05T00:00:00+00:00

arrow.Arrow(2013, 5, 5)
 2013-05-05T00:00:00+00:00
```

Compatibility?

In [7]:
```
a = arrow.now()
print(type(a), a)
print(type(a.datetime), a.datetime)
```

```
<class 'arrow.arrow.Arrow'> 2017-06-26T14:26:54.576306+05:30
<class 'datetime.datetime'> 2017-06-26 14:26:54.576306+05:30
```

In [8]:
```
print('a.timestamp  ', a.timestamp)
print('a.naive      ', a.naive)
print('a.tzinfo     ', a.tzinfo)
print('a.year       ', a.year)
print('a.day        ', a.day)
print('a.date()     ', a.date())
```

```
a.timestamp   1498467414
a.naive       2017-06-26 14:26:54.576306
a.tzinfo      tzlocal()
a.year        2017
a.day         26
a.date()      2017-06-26
```

Manipulation

In [9]:
```
arw = arrow.now()
print(arw)
```

```
2017-06-26T14:26:54.597894+05:30
```

In [10]:
```
print("arw.replace(hour=6)  ", arw.replace(hour=6))     # Replace the hour
print("arw.replace(hours=6) ", arw.replace(hours=6))  # Adds to existing hours
```

```
arw.replace(hour=6)   2017-06-26T06:26:54.597894+05:30
arw.replace(hours=6)  2017-06-26T20:26:54.597894+05:30
```

Excel/CSV

P.S: Uggghh! :(

Tablib

import, export, and manipulate tabular data sets

http://docs.python-tablib.org/

(examples from docs)

In [11]:

```
import tablib
data = tablib.Dataset()

data.headers = ['Name', 'Psychological Disorders']

data.append(['Gollum', 'unhealthy obsession with rings'])
data.append(['Norman Osborn', 'Schizophrenia'])
data.append(['Joker', 'plain crazy'])
```

In [12]:
```
print(data)
```

```
Name         |Psychological Disorders
-------------|----------------------------
Gollum       |unhealthy obsession with rings
Norman Osborn|Schizophrenia
Joker        |plain crazy
```

In [13]:
```
print(data.json)
```

```
[{"Name": "Gollum", "Psychological Disorders": "unhealthy obsession with rings"}, {"Name": "Norman Osbor
n", "Psychological Disorders": "Schizophrenia"}, {"Name": "Joker", "Psychological Disorders": "plain crazy"}]
```

In [14]:
```
print(data.csv)
```

```
Name,Psychological Disorders
Gollum,unhealthy obsession with rings
Norman Osborn,Schizophrenia
Joker,plain crazy
```

In [15]:
```
print(data.html)
```

```
<table>
<thead>
<tr><th>Name</th>
<th>Psychological Disorders</th></tr>
</thead>
<tr><td>Gollum</td>
<td>unhealthy obsession with rings</td></tr>
<tr><td>Norman Osborn</td>
<td>Schizophrenia</td></tr>
<tr><td>Joker</td>
<td>plain crazy</td></tr>
</table>
```

In [16]:
```
from IPython.core.display import display, HTML
display(HTML(data.html))
```

Name	Psychological Disorders
Gollum	unhealthy obsession with rings
Norman Osborn	Schizophrenia
Joker	plain crazy

In [17]:
```
# Simply dump the dataset into a file
with open('data.csv', 'w') as f:
    f.write(data.csv)
```

Importing Data via Tablib

In [18]:
```
import tablib
imported_data = tablib.Dataset().load(open('data.csv').read())
print(imported_data)
```

```
Name         |Psychological Disorders
-------------|----------------------------
Gollum       |unhealthy obsession with rings
Norman Osborn|Schizophrenia
Joker        |plain crazy
```

Pandas

The Backstory...

In [19]:
```
import pandas as pd
```

In [20]:
```
# Get excel sheet with all the names
all_names = pd.read_excel('names.xlsx')
```

In [21]:
```
# logic to tabulate the frequency of people's name by alphabet
def first_letter(row):
    return row[0].lower()

all_names['first'] = all_names['names'].apply(first_letter)
group = all_names.groupby('first')
count_by_letter = group.count()
```

In [22]:
```
# count_by_letter
```

In [23]:
```
count_by_letter.to_excel('count_by_letter.xlsx')
```

Django

- django-import-export
- cookiecutter-django
- django-init

How to make slides programmatically (like this one!)

https://github.com/damianavila/RISE

Last but not the least

- https://pyformat.info
- https://awesome-python.com/
- https://pymotw.com/

That’s all folks!