In [1]:

```
import xmltodict
import json
import pandas as pd
from matplotlib import pyplot as plt
from matplotlib.pyplot import pie, axis, show
from matplotlib import rcParams
from wordcloud import WordCloud, STOPWORDS
import numpy as np
import math
from collections import defaultdict
```

Loading Post file

In [3]:

```
lines = []
with open('./ansh/Posts.json') as file:
    for line in file:
        lines.append(json.loads(line))
df_post = pd.DataFrame(lines)
df_post.head()
```

Out[3]:

	Id	PostTypeId	ParentId	CreationDate	Score	Body	OwnerUserId	LastEdi
0	538	2	535	2008-08-02T18:56:56.460	28	<p>One possibility is Hudson. It's written in...	156	
1	766	1	NaN	2008-08-03T17:44:07.450	35	<p>I can get Python to work with Postgresql bu...	1384652	
2	1484	2	1476	2008-08-04T18:34:45.520	72	<pre><code>&gt;&gt;&gt; print int('01010101111...	2089740	
3	1983	1	NaN	2008-08-05T07:18:55.853	50	<p>In many places, <code>(1,2,3)</code> (a tup...	116	
4	3061	1	NaN	2008-08-06T03:36:08.627	1655	<p>What is the best way to go about calling a ...	121	

5 rows × 21 columns

Loading Tags file

In [4]:

```
lines = []
with open('./ansh/Tags.json') as file:
    for line in file:
        lines.append(json.loads(line))
df_tags = pd.DataFrame(lines)
df_tags.head()
```

Out[4]:

	Id	TagName	Count	ExcerptPostId	WikiPostId
0	1	.net	293379	3624959	3607476
1	2	html	970699	3673183	3673182
2	3	javascript	1955557	3624960	3607052
3	4	css	649436	3644670	3644669
4	5	php	1335050	3624936	3607050

Loading Users file

In [5]:

```
lines = []
with open('./ansh/Users.json') as file:
    for line in file:
        lines.append(json.loads(line))
df_user = pd.DataFrame(lines)
df_user.head()
```

Out[5]:

	Id	Reputation	CreationDate	DisplayName	LastAccessDate	
0	1	58679	2008-07-31T14:22:31.287	Jeff Atwood	2020-02-26T23:04:34.223	http://www.codinghor
1	4	31720	2008-07-31T14:22:31.317	Joel Spolsky	2020-02-29T18:22:56.427	https://joelons
2	13	194621	2008-08-01T04:18:04.943	Chris Jester-Young	2019-12-03T01:13:11.627	http://
3	17	50531	2008-08-01T12:02:21.617	Nick Berardi	2020-02-28T14:38:17.133	http://nic
4	25	31334	2008-08-01T12:15:23.243	CodingWithoutComments	2018-05-03T20:41:05.130	

Loading Votes file

In [7]:

```
lines = []
with open('./ansh/Votes.json') as file:
    for line in file:
        lines.append(json.loads(line))
df_votes = pd.DataFrame(lines)
df_votes.head()
```

Out[7]:

	Id	PostId	VoteTypeId	CreationDate	UserId	BountyAmount
0	2613	972	2	2008-08-04T00:00:00.000	NaN	NaN
1	5292	1829	2	2008-08-05T00:00:00.000	NaN	NaN
2	7197	2982	2	2008-08-06T00:00:00.000	NaN	NaN
3	8354	3117	2	2008-08-06T00:00:00.000	NaN	NaN
4	10940	5102	2	2008-08-07T00:00:00.000	NaN	NaN

Loading Badges File

In [9]:

```
lines = []
with open('./ansh/Badges.json') as file:
    for line in file:
        lines.append(json.loads(line))
df_badges = pd.DataFrame(lines)
df_badges.head()
```

Out[9]:

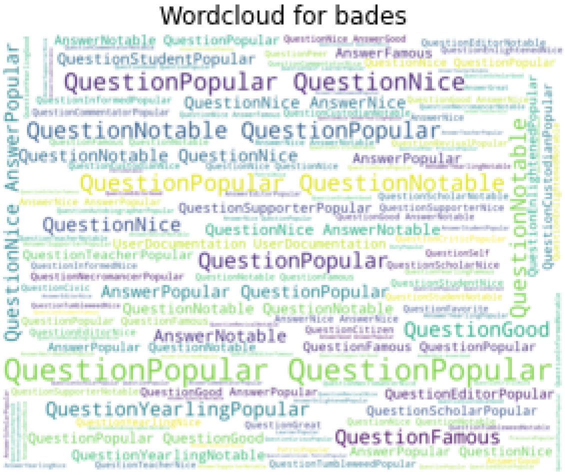
	Id	UserId	Name	Date	Class	TagBased
0	83047	2846	Teacher	2008-09-15T08:55:03.957	3	False
1	83333	2958	Teacher	2008-09-15T08:55:03.957	3	False
2	83430	2354	Teacher	2008-09-15T08:55:03.957	3	False
3	83509	13	Teacher	2008-09-15T08:55:03.970	3	False
4	83609	3149	Teacher	2008-09-15T08:55:03.970	3	False

Badges WordCloud

In [15]:

```
names = ""
for name in df_badges['Name']:
    names = names + name
wordcloud = WordCloud(
    width=1800,
    height=1400,
    max_font_size=300,
    max_words=150,
    background_color='white').generate(names)

plt.figure()
plt.title("Wordcloud for bades")
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```



In [17]:

```
tags_arr = df_post['Tags']
```

Cleaning Tags and making a frequency array for the same

In [21]:

```
d = defaultdict(int)

for tag in tags_arr:
    if type(tag) != str:
        continue
    s = ""
    for c in tag:
        if c == '<':
            continue
        if c == '>':
            d[s] = d[s] + 1
            s = ""
            continue
        s = s + c
l = []
for key,val in d.items():
    l.append((key, val))
```

In [22]:

```

def Sort_Tuple(tup):
    lst = len(tup)
    for i in range(0, lst):

        for j in range(0, lst-i-1):
            if (tup[j][1] < tup[j + 1][1]):
                temp = tup[j]
                tup[j]= tup[j + 1]
                tup[j + 1]= temp
    return tup

print(Sort_Tuple(l))
```

```
[('python', 104506), ('python-3.x', 9593), ('pandas', 9129), ('dja
ngo', 8438), ('numpy', 4769), ('python-2.7', 4737), ('list', 340
3), ('matplotlib', 2905), ('dataframe', 2607), ('dictionary', 234
2), ('tensorflow', 2130), ('regex', 2107), ('tkinter', 1947), ('fl
ask', 1919), ('csv', 1736), ('string', 1701), ('arrays', 1597),
('json', 1467), ('selenium', 1356), ('html', 1265), ('opencv', 124
7), ('beautifulsoup', 1242), ('machine-learning', 1182), ('web-scr
aping', 1072), ('keras', 1062), ('scikit-learn', 1034), ('mysql',
1024), ('scipy', 1022), ('sqlalchemy', 972), ('multithreading', 91
6), ('javascript', 901), ('linux', 887), ('google-app-engine', 88
2), ('loops', 846), ('function', 844), ('pygame', 839), ('pip', 83
2), ('pyqt', 831), ('datetime', 813), ('windows', 807), ('django-m
odels', 792), ('class', 772), ('python-requests', 735), ('scrapy',
732), ('for-loop', 709), ('file', 708), ('xml', 695), ('c++', 68
3), ('algorithm', 649), ('macos', 620), ('sqlite', 616), ('postgre
sql', 612), ('sockets', 591), ('excel', 589), ('sql', 587), ('subp
rocess', 579), ('multiprocessing', 578), ('pyspark', 571), ('pycha
rm', 563), ('plot', 556), ('django-rest-framework', 538), ('sortin
g', 537), ('parsing', 531), ('anaconda', 505), ('performance', 50
```

In [23]:

```
tag_array_all, count_array_all = zip(*l)
tag_array = tag_array_all[:10]
count_array = count_array_all[:10]
print(tag_array, count_array)
```

```
('python', 'python-3.x', 'pandas', 'django', 'numpy', 'python-2.7', 'l
ist', 'matplotlib', 'dataframe', 'dictionary') (104506, 9593, 9129, 84
38, 4769, 4737, 3403, 2905, 2607, 2342)
```

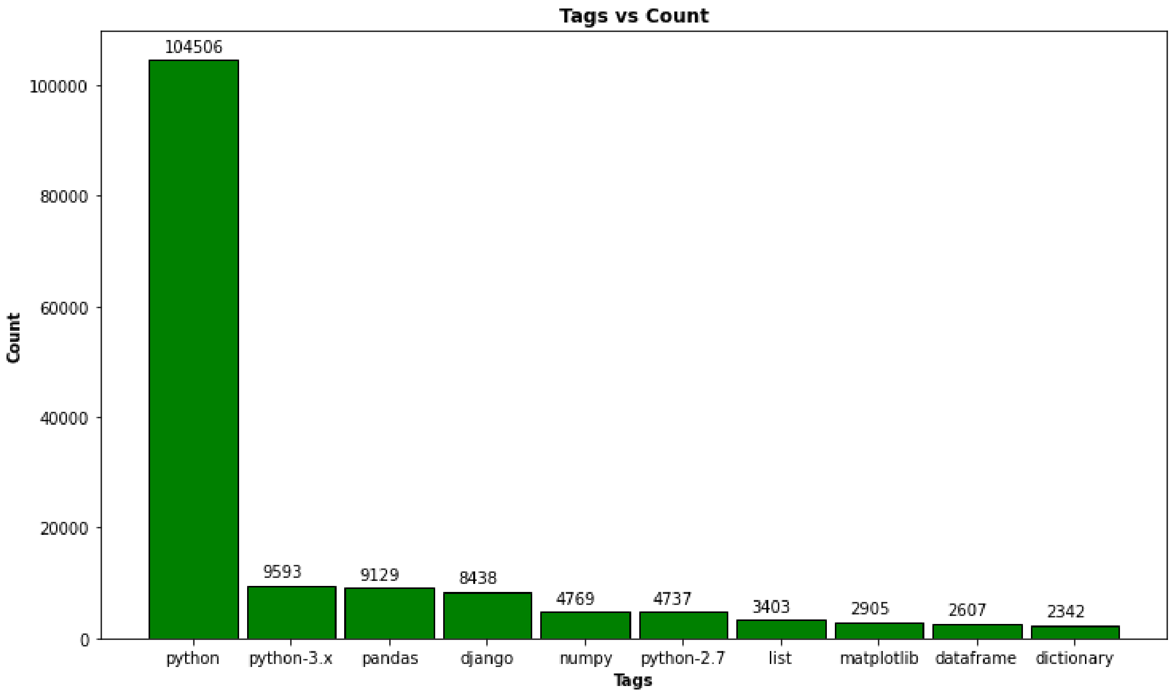
Top 10 tags

In [26]:

```
plt.figure(figsize = (12,7))
plt.bar(tag_array, count_array, width= 0.9, align='center',color='green', edgecolor

# Annotating the bar plot with the values (total death count)
for i in range(len(tag_array)):
    plt.annotate(count_array[i], (-0.3 + i, count_array[i] + 1500))

plt.title("Tags vs Count",fontweight="bold")
plt.xlabel('Tags',fontweight="bold")
plt.ylabel('Count',fontweight="bold")
plt.show()
```

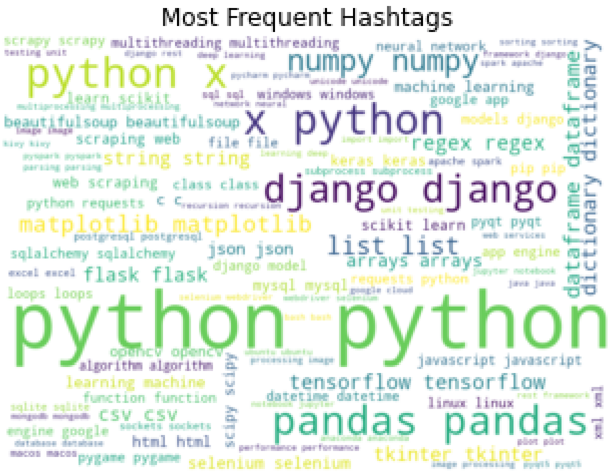


Word Cloud for Tags

In [27]:

```
tags_list = ""
for tag, count in zip(tag_array_all, count_array_all):
    while count > 0:
        tags_list = tags_list + " " + tag
        count = count - 1
wordcloud = WordCloud(
    width=700,
    height=500,
    max_font_size=100,
    max_words=100,
    background_color='white').generate(tags_list)

plt.figure()
plt.title("Hashtags")
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis('off')
plt.show()
```


Most Frequent Hashtags

Common Sampling for Posts is done through python tag

In [20]:

```python
for tag in tags_arr:
    if type(tag) == str:
        print(tag)
```

```
<python><mysql><postgresql><bpgsql>
<python><list><tuples>
<python><object>
<python><doctest>
<python><command-line><packaging>
<python><command-line><command-line-arguments>
<python><http><urllib>
<python><binary><io><buffer>
<python><security>
<python><windows><cross-platform>
<python><multithreading>
<python><class-method>
<javascript><python>
<python><favicon>
<python><gtk><pygtk><glade><gtk2>
<python><svn><dos2unix>
<python><sysadmin><whois>
<python><weak-references>
<python><path><relative-path><absolute-path>
```

In [ ]:

```python

```