

# Lord\_of\_the\_flies\_solution

February 22, 2017

## 0.1 Identify the number of papers in PubMed that has *Drosophila virilis* in the title or abstract

```
In [47]: from Bio import Entrez
import re
```

We construct an esearch request and use the NCBI history function in order to refer to this search in our subsequent efetch call.

```
In [46]: # Always tell NCBI who you are (edit the e-mail below!)
Entrez.email = "your_name@yourmailhost.com"
handle = Entrez.esearch(db="pubmed",
                        term="Drosophila virilis[Title/Abstract]",
                        usehistory="y")
record = Entrez.read(handle)
# generate a Python list with all Pubmed IDs of articles about D. virilis
id_list = record["IdList"]
record["Count"]
```

```
Out[46]: '535'
```

```
In [3]: webenv = record["WebEnv"]
query_key = record["QueryKey"]
```

## 0.2 Retrieve the PubMed entries using our search history

```
In [4]: handle = Entrez.efetch(db="pubmed",
                                rettype="medline",
                                retmode="text",
                                retstart=0,
                                retmax=535, webenv=webenv, query_key=query_key)
```

```
In [5]: out_handle = open("D_virilis_pubs.txt", "w")
data = handle.read()
handle.close()
out_handle.write(data)
out_handle.close()
```

### 0.3 Count the number of contributions per author

We construct a dictionary with all authors as keys and the number of contributions as value.

```
In [6]: with open("D_virilis_pubs.txt") as datafile:
        author_dict = {}
        for line in datafile:
            if re.match("AU", line):
                # capture author
                author = line.split("-", 1)[1]
                # remove leading and trailing whitespace
                author = author.strip()
                # if key is present, add 1
                # if it's not present, initialize at 1
                author_dict[author] = 1 + author_dict.get(author, 0)
```

### 0.4 Find the top five researchers

Dictionaries do not have a natural order but we can sort a dictionary based on the values using the function `sorted`. We retrieve the number of contributions per author from our `author_dict` using `author_dict.get` and use it as value in the sorted function. `sorted` returns a list that can be indexed to return only the top 5 of researchers.

```
In [7]: for author in sorted(author_dict, key = author_dict.get, reverse = True)[:5]:
        print(author, ":", author_dict[author])
```

```
Gruntenko NE : 36
Evgen'ev MB : 29
Raushenbakh IIu : 24
Hoikkala A : 23
Korochkin LI : 22
```