BAD HABITS
IN
ACADEMIC CODE

Python in science

NumPy

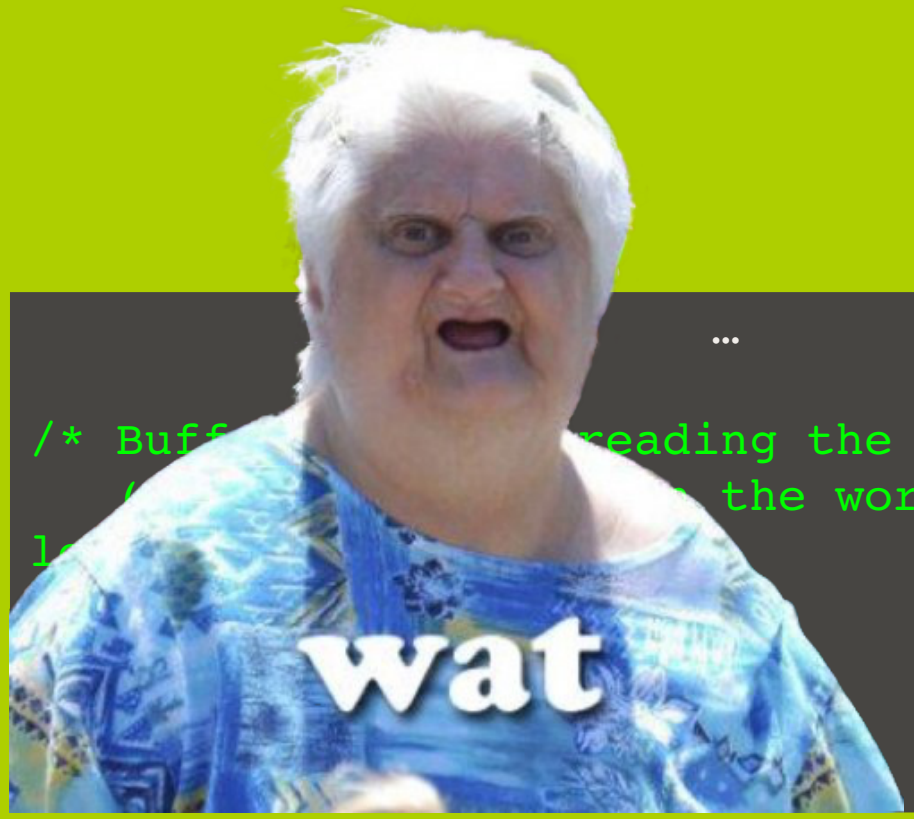
SciPy

NLTK

...

```
                                …

/* Buffer size for reading the wordlist file
   (must be larger than the wordcount in the
longest document!.) */

#define BLOCKSIZE 1000000

                                …
```

```
...

/* Buff                reading the wordlist file
    (                    the wordcount in the
l
```



Python code

Freak show

Logging

```
if c != None:
    ref, sf = c
    if debug:
        sys.stderr.write("[C] {0} : {1}\n".format(ref, sf))
```

Logging

```
if c != None:
    ref, sf = c
    if debug:
        sys.stderr.write("[C] {0} : {1}\n".format(ref, sf))
```

```
import logging

if c != None:
    ref, sf = c
    logging.debug("[C] {0} : {1}".format(ref, sf))
```

Arguments

```
do_ent = (sys.argv[1]=="y")
do_eve = (sys.argv[2]=="y")
sys.stderr.write("entities: {}, events {}\n".format(do_ent, do_eve))
total = 0
n = 0
k = eval(sys.argv[3])
```

Arguments

```
do_ent = (sys.argv[1]=="y")
do_eve = (sys.argv[2]=="y")
sys.stderr.write("entities: {}, events {}\n".format(do_ent, do_eve))
total = 0
n = 0
k = eval(sys.argv[3])
```

```
import argparse

parser.add_argument('ent', type=bool, help='process entities')
parser.add_argument('eve', type=bool, help='process events')
parser.add_argument('k', type=int, help='threshold')
args = parser.parse_args()
```

Read a file

```
fd = open(sys.argv[1], 'r')
parser.parse_tup_lines(fd.readlines())
fd.close()
```

Read a File

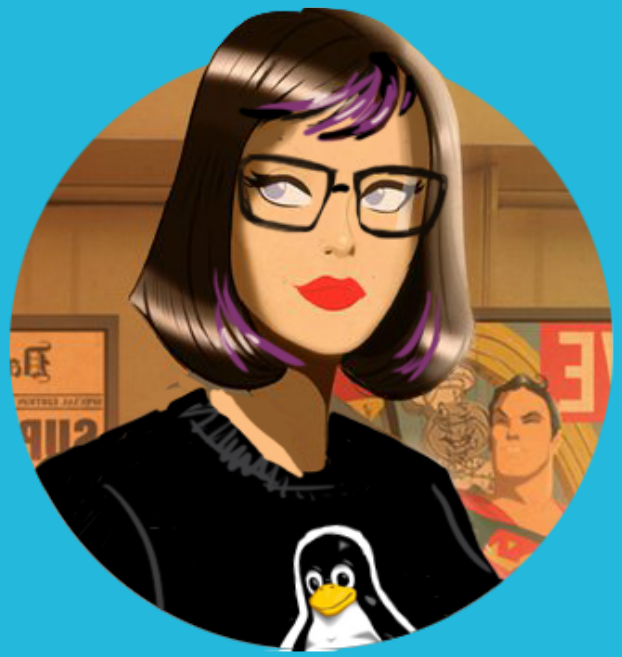
```
fd = open(sys.argv[1], 'r')
parser.parse_tup_lines(fd.readlines())
fd.close()
```

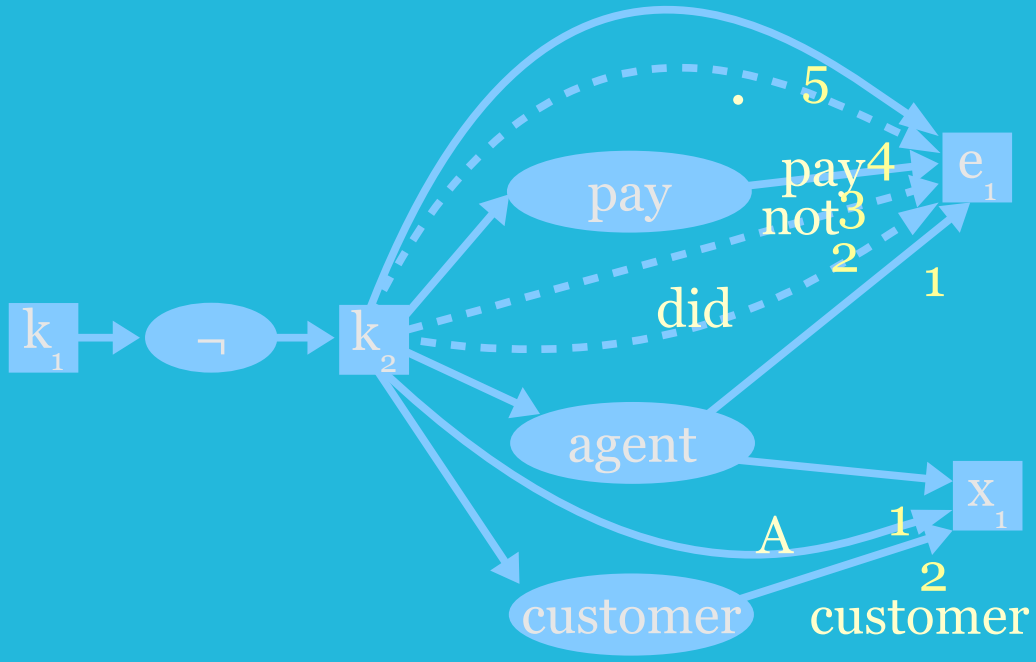
```
with open(sys.argv[1]) as fd:
    parser.parse_tup_lines(fd.readlines())
```

Monolithic

Mysterious

Hard to process

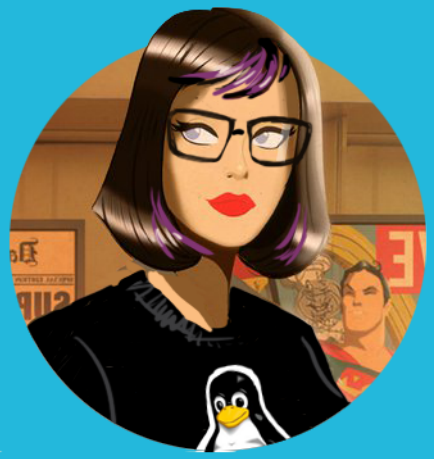





```
1, 0, 1, 0, 0, 0, 0, 1, 0, 1, 1, 0, 0, 0, 0, 1, 1
1, 0, 0, 0, 1, 0, 0, 1, 0, 0, 1, 0, 0, 0, 0, 0, 2
2, 0, 1, 0, 0, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 1, 3
2, 1, 0, 0, 0, 0, 1, 0, 0, 0, 0, 0, 0, 0, 1, 1, 1
2, 0, 1, 0, 1, 0, 0, 1, 1, 0, 1, 0, 0, 0, 0, 0, 2
                                ...
```

readability
maintainability

```
class DRGEdge(GraphEdge):
    """This class represents a specialized edge in a
    Discourse Representation Graph.
    """
    …
    def extractFeatures(self):
        """Convert the edge attributes into a binary vector
        suitable for further elaboration.
        """
        self.surface = toBinaryVector(self.edgeType)
        …
```



efficiency
style

```
# extract the feature vector for the edge type

surface = [int(edge_type == "surface")]
```

results
papers

```
surface = extract_surface(edge, lists)

…

def extract_surface(edge, lists):
    if edge.edge_type == "surface":
        return [1]
    else:
        return [0]
```

results
papers

```
surface = extract_surface(e

…

def extr                                s):
                                surface":

      rn [0]
```

I have no root
and
I want to scream

I have no root

and

I want to scream

Virtualenv + PIP

It's 2014 baby

It's 2014 baby

RESTful APIs

Conclusions (I)

Academia  Python

Conclusions (2)

We need help!

Fin