Teaching Data Science Students to Write Clean Code

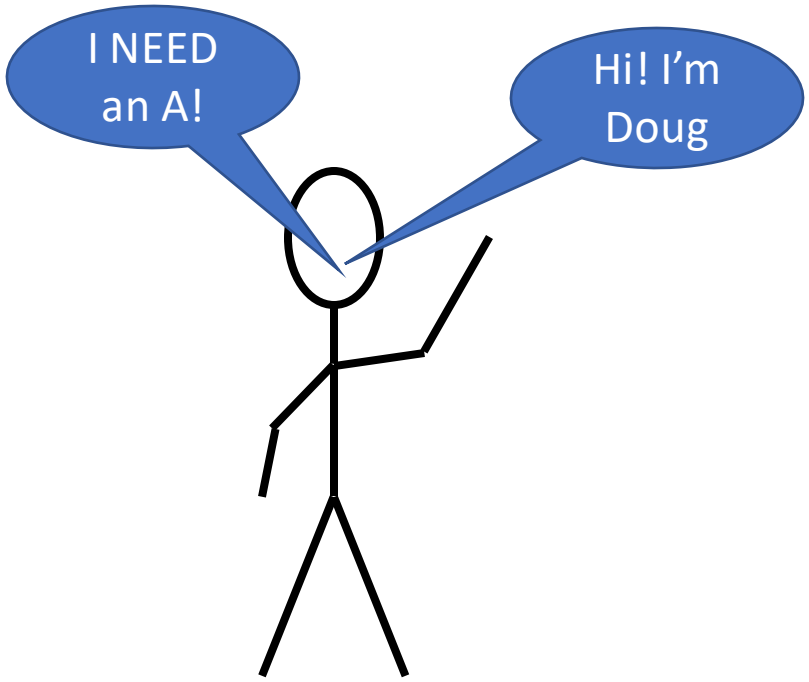
Todd Iverson, Winona State University

Slides/Code: https://bit.ly/2WgFIbI

Episode 37

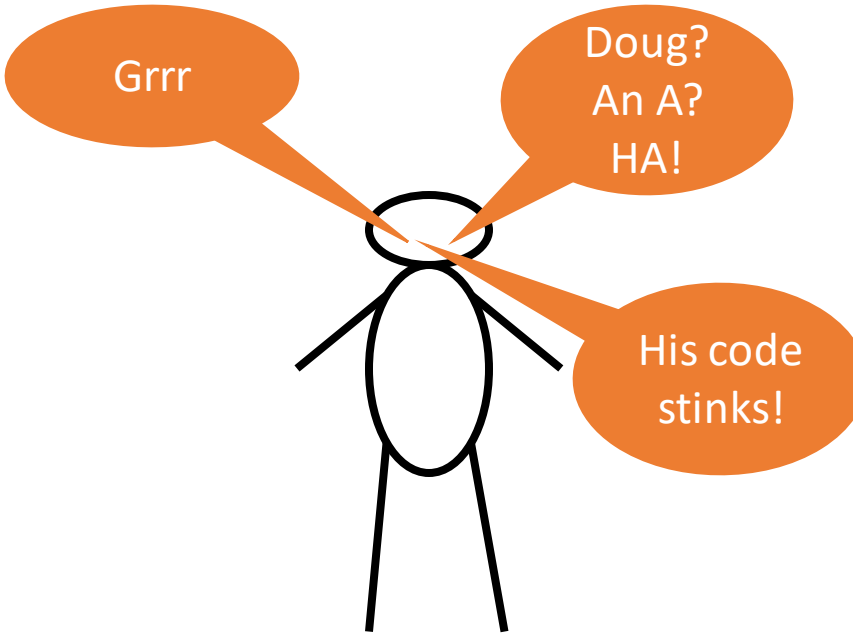
Doug Wants an A

The Hero



Doug "Crazy Legs" Ervison

The Villain

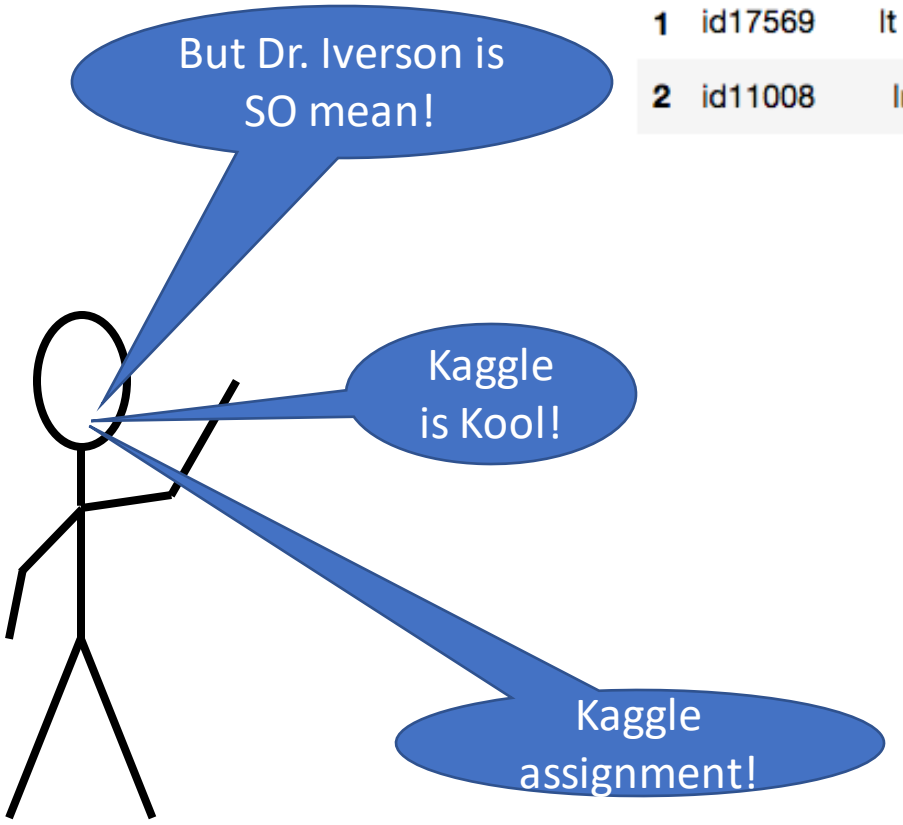


"Mean" Dr. Iverson

Doug will demonstrate

1. **Good names**
2. **Small functions**
3. **Unit tests**
4. **Refactor** code, specifically
    1. Extract functions
    2. Split loops

Opening Scene - The Assignment

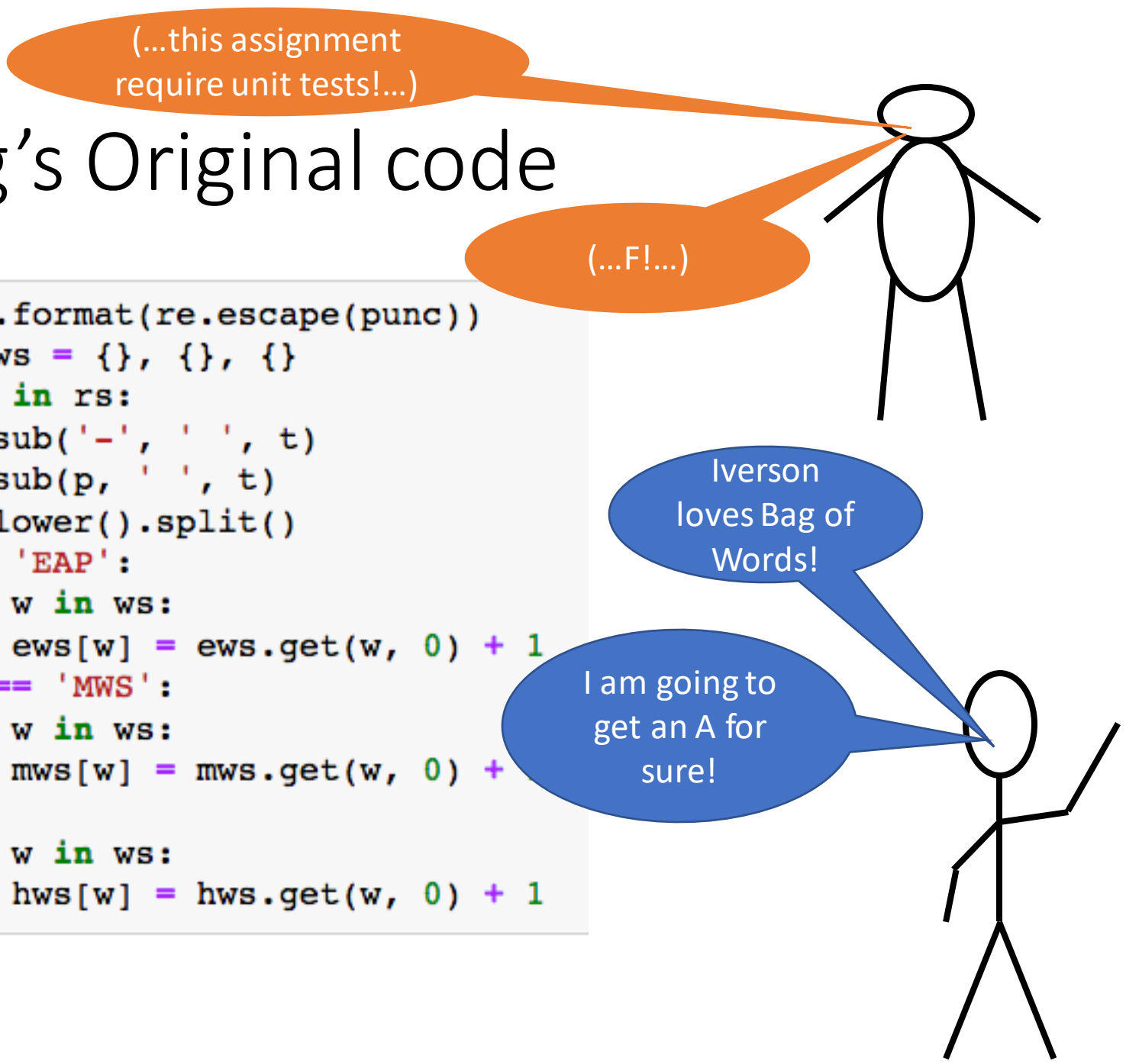


	id	text	author
0	id26305	This process, however, afforded me no means of...	EAP
1	id17569	It never once occurred to me that the fumbling...	HPL
2	id11008	In his left hand was a gold snuff box, from wh...	EAP

https://www.kaggle.com/c/spooky-author-identification/data

Doug's Original code

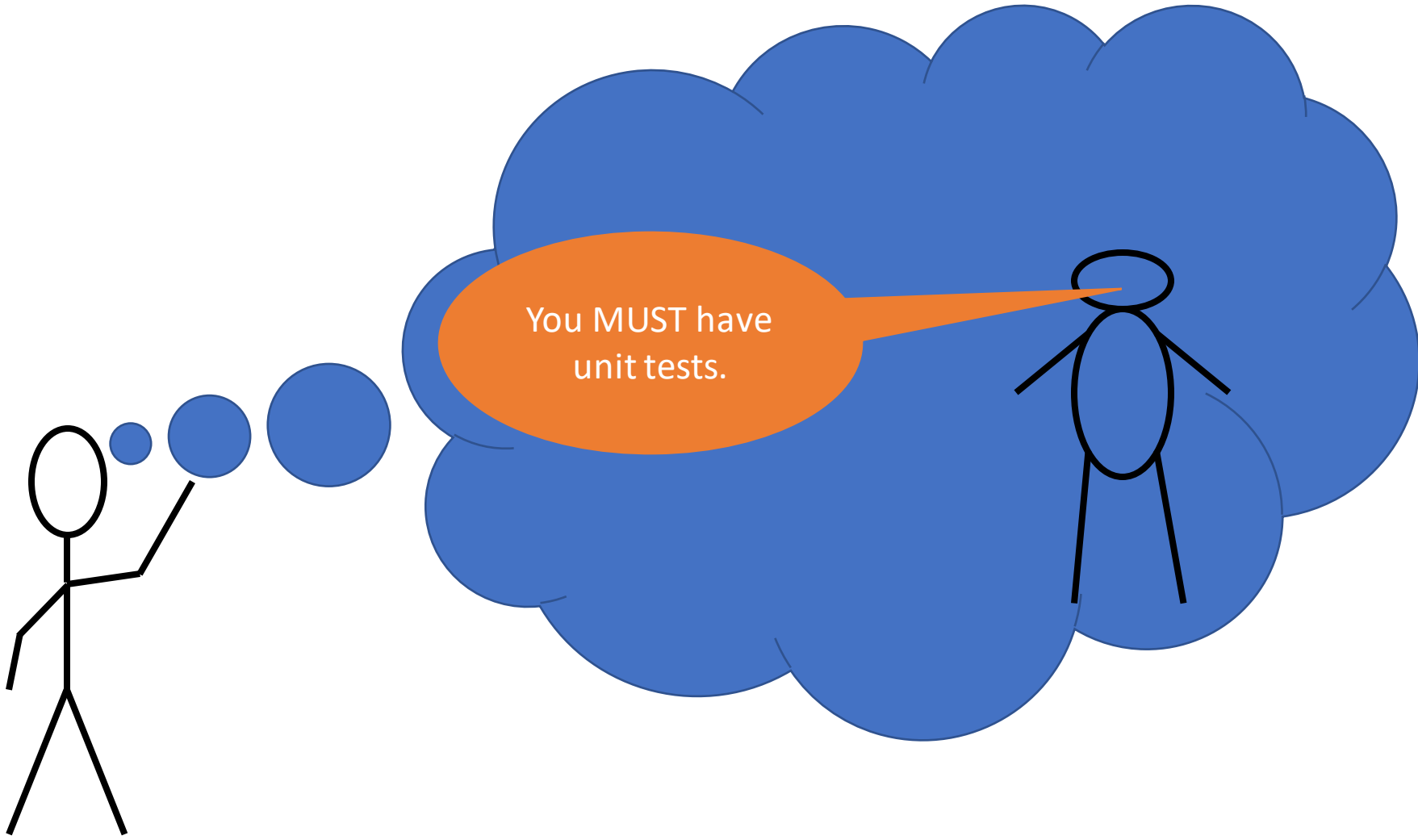
```
p = '[{0}]'.format(re.escape(punc))
ews, mws, hws = {}, {}, {}
for i, t, a in rs:
    t = re.sub('-', ' ', t)
    t = re.sub(p, ' ', t)
    ws = t.lower().split()
    if a == 'EAP':
        for w in ws:
            ews[w] = ews.get(w, 0) + 1
    elif a == 'MWS':
        for w in ws:
            mws[w] = mws.get(w, 0) +
    else:
        for w in ws:
            hws[w] = hws.get(w, 0) + 1
```



It looks like our hero is doomed to an F!

Then just in the nick of time …

… Doug remembers unit tests!



What are unit tests?

- Captures/maintain intended behavior
- Helpful when changing code
- Should be automated

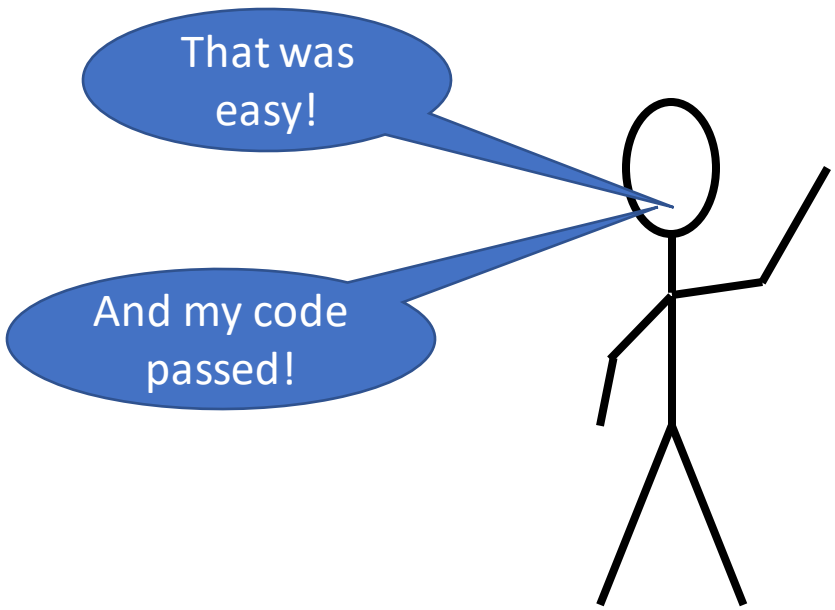
Doug writes some unit tests

```
example_rows = [('1', 'The dog is super-large.', 'HPL'),
                ('2', "The cat's cute, but the cat's mean.", 'MWS'),
                ('3', '"No pets!"', 'EAP')
               ]
example_output = {'EAP':{'no':1, 'pets':1},
                  'MWS':{'the':2, 'cats':2, 'cute':1, 'but':1, 'mean':1},
                  'HPL':{'the':1, 'dog':1, 'is':1, 'super':1, 'large':1}}
```

Original behavior

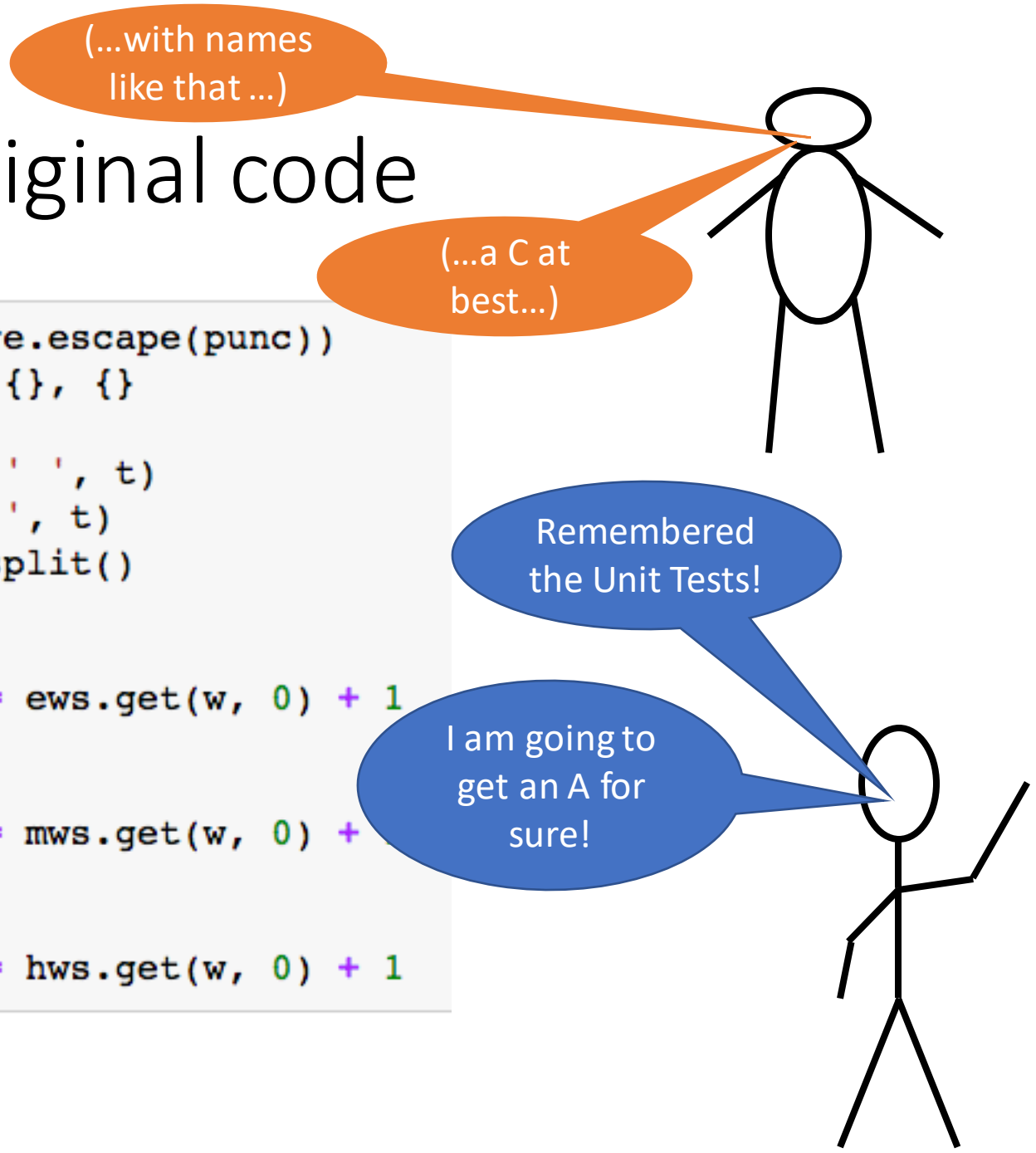
```
def test_main():
    ews, mws, hws = main(example_rows)
    assert ews == example_output['EAP']
    assert mws == example_output['MWS']
    assert hws == example_output['HPL']
```

New behavior

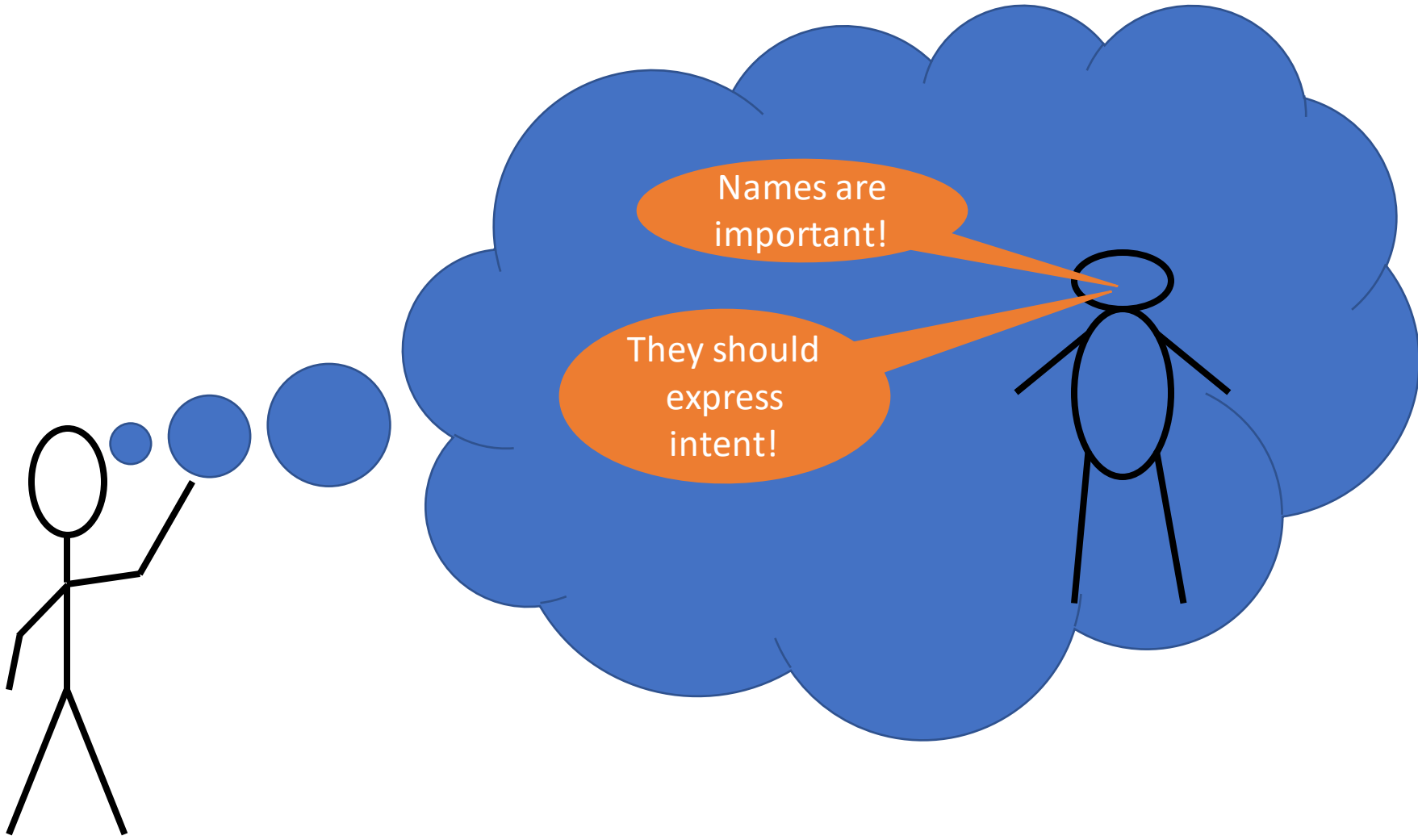


Doug's Original code

```
p = '[{0}]'.format(re.escape(punc))
ews, mws, hws = {}, {}, {}
for i, t, a in rs:
    t = re.sub('-', ' ', t)
    t = re.sub(p, ' ', t)
    ws = t.lower().split()
    if a == 'EAP':
        for w in ws:
            ews[w] = ews.get(w, 0) + 1
    elif a == 'MWS':
        for w in ws:
            mws[w] = mws.get(w, 0) +
    else:
        for w in ws:
            hws[w] = hws.get(w, 0) + 1
```



Luckily, Doug remembers to think about names



Good names…

- **Reveal intent**
- Use the proper parts of speech
- Have the proper length for their scope
- Avoids disinformation and encodings

Data: What is it?

Function: What does it do?

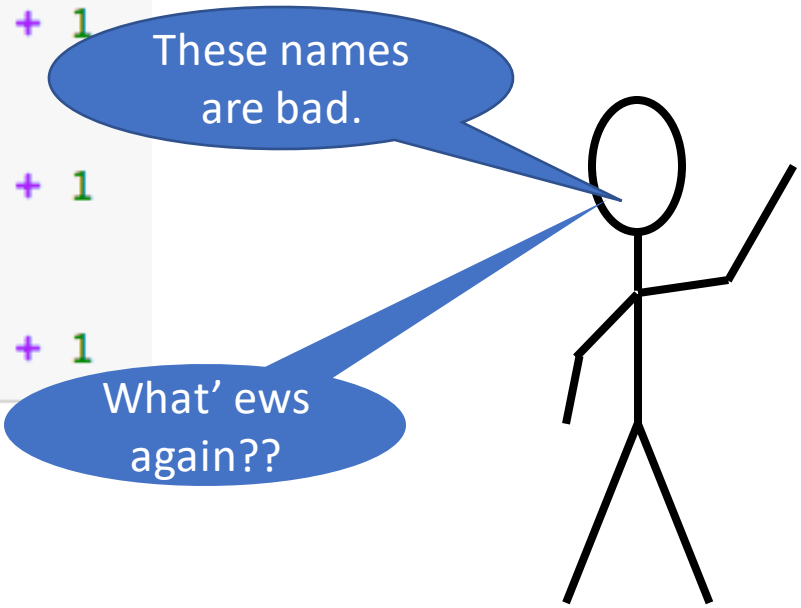
Good names…

- Reveal intent
- **Use the proper parts of speech**
- Have the proper length for their scope
- Avoids disinformation and encodings

Variable: Noun
Function: Verb
Boolean: Predicate

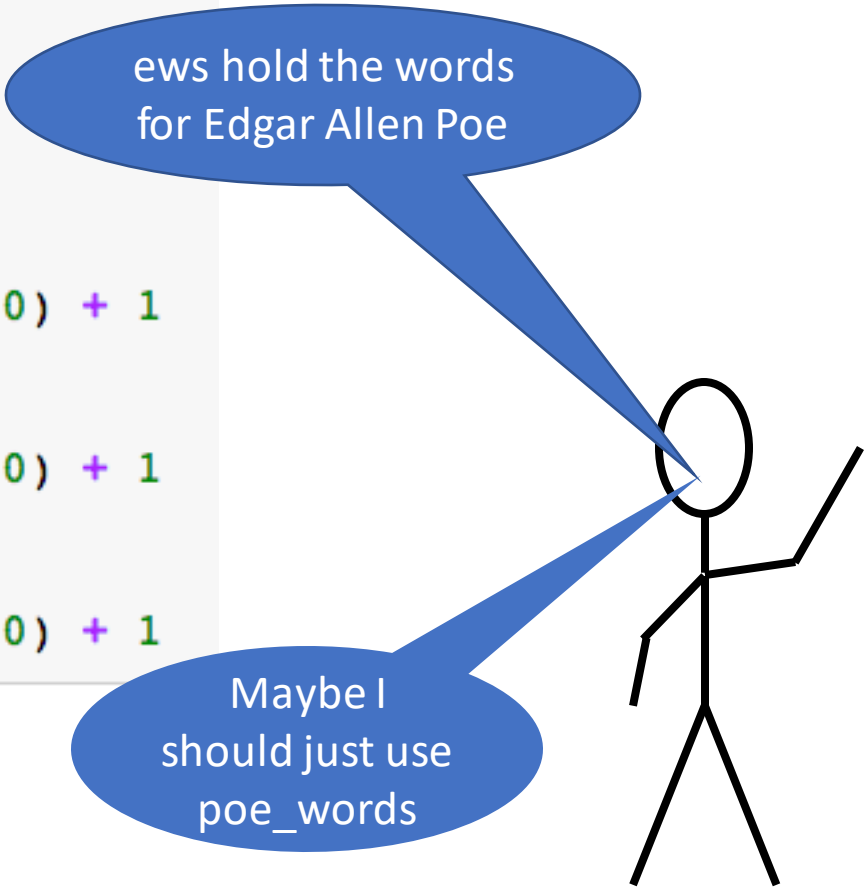
Doug inspects his names

```
p = '[{0}]'.format(re.escape(punc))
ews, mws, hws = {}, {}, {}
for i, t, a in rs:
    t = re.sub('-', ' ', t)
    t = re.sub(p, ' ', t)
    ws = t.lower().split()
    if a == 'EAP':
        for w in ws:
            ews[w] = ews.get(w, 0) + 1
    elif a == 'MWS':
        for w in ws:
            mws[w] = mws.get(w, 0) + 1
    else:
        for w in ws:
            hws[w] = hws.get(w, 0) + 1
```

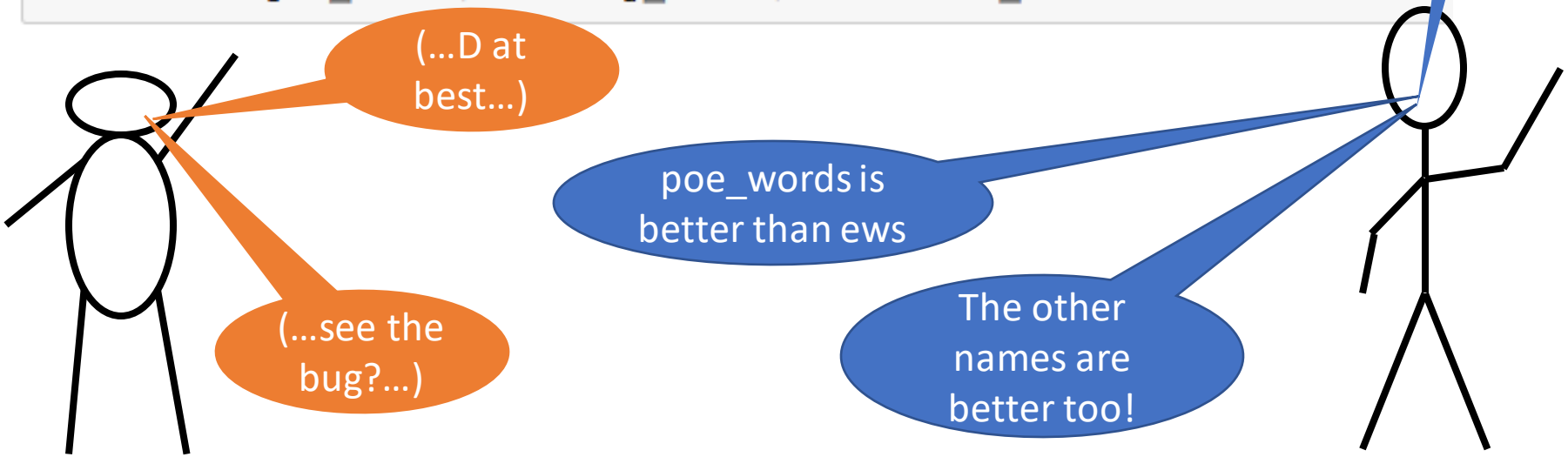


Doug finds some better names

```
p = '[{0}]'.format(re.escape(punc))
ews, mws, hws = {}, {}, {}
for i, t, a in rs:
    t = re.sub('-', ' ', t)
    t = re.sub(p, ' ', t)
    ws = t.lower().split()
    if a == 'EAP':
        for w in ws:
            ews[w] = ews.get(w, 0) + 1
    elif a == 'MWS':
        for w in ws:
            mws[w] = mws.get(w, 0) + 1
    else:
        for w in ws:
            hws[w] = hws.get(w, 0) + 1
```



```
def main(rows):
    punc_pat = '[{0}]'.format(re.escape(punc))
    poe_words, shelley_words, lovecraft_words = {}, {}, {}
    for id, text, author in rows:
        t = re.sub('-', ' ', text)
        t = re.sub(punc_pat, '', t)
        words = t.lower().split()
        if a == 'EAP':
            for w in words:
                poe_words[w] = poe_words.get(w, 0) + 1
        elif a == 'MWS':
            for w in words:
                shelley_words[w] = shelley_words.get(w, 0) + 1
        else:
            for w in words:
                lovecraft_words[w] = lovecraft_words.get(w, 0) + 1
    return poe_words, shelley_words, lovecraft_words
```



It looks like our hero is doomed with D!

Then just in the nick of time …

… Doug remembers to test!

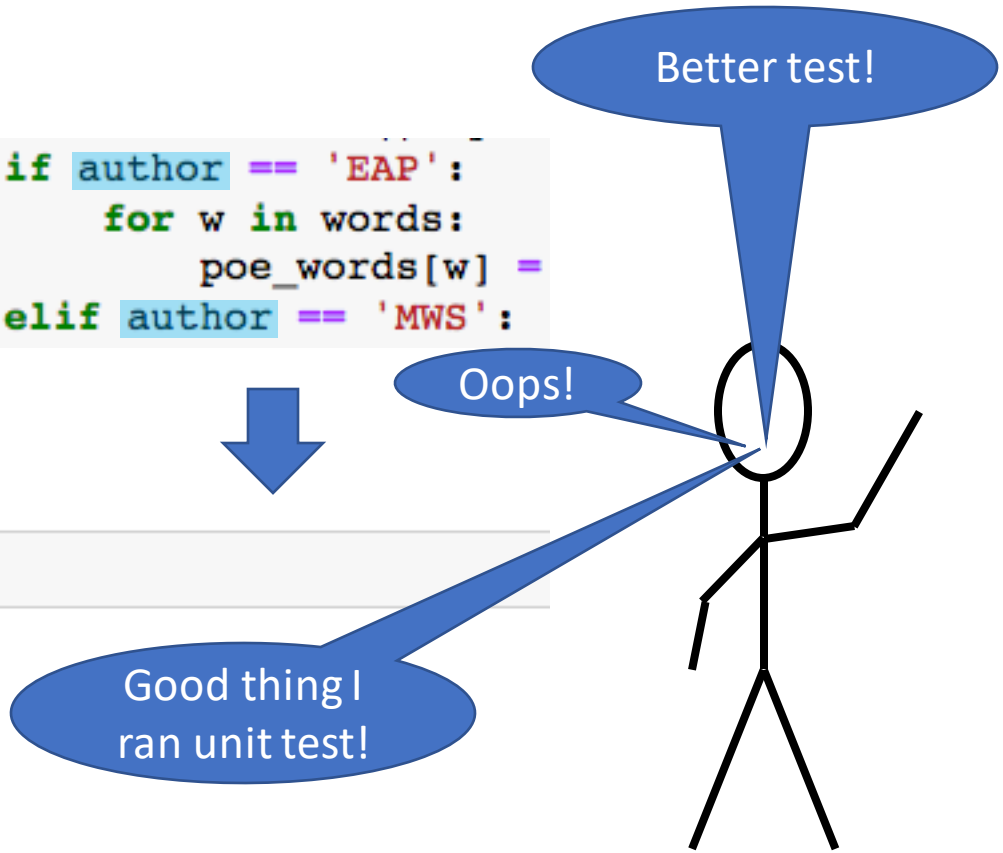
```
test_main()
```

```
AssertionError:
```

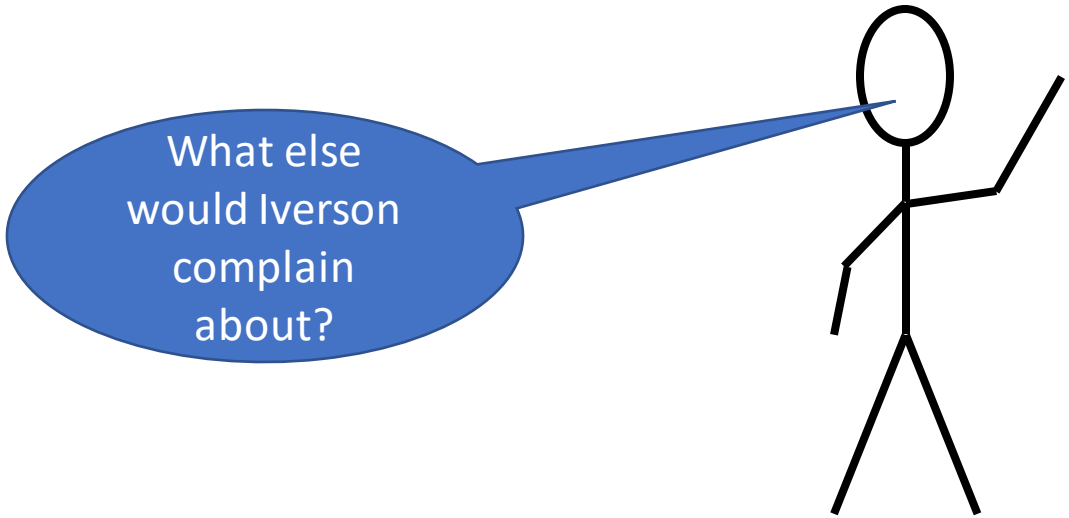
```
if a == 'EAP':
    for w in words:
        poe_words[w]
elif a == 'MWS':
```

```
if author == 'EAP':
    for w in words:
        poe_words[w] =
elif author == 'MWS':
```

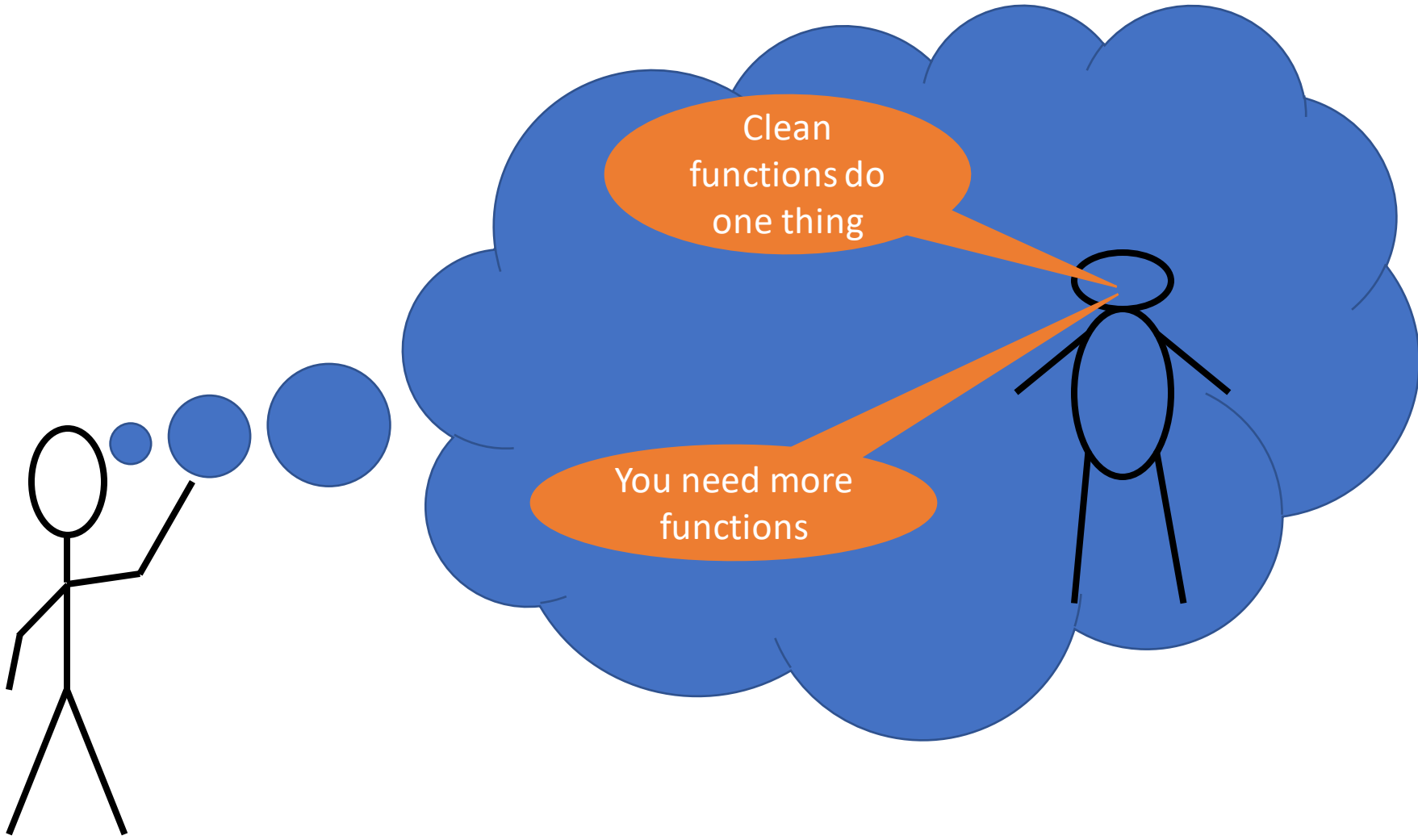
```
test_main()
```



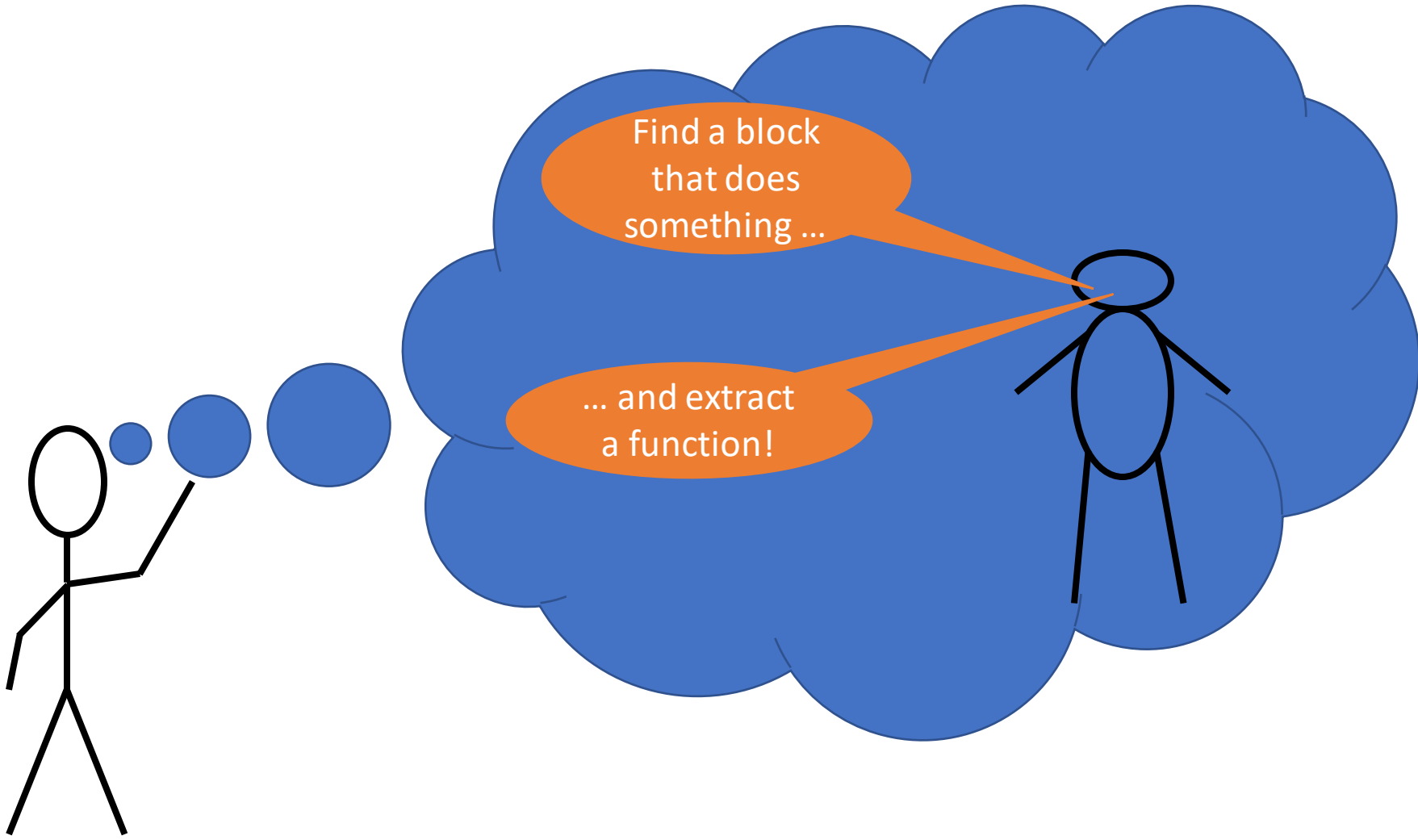
```python
def main(rows):
    punc_pat = '[{0}]'.format(re.escape(punc))
    poe_words, shelley_words, lovecraft_words = {}, {}, {}
    for id, text, author in rows:
        t = re.sub('-', ' ', text)
        t = re.sub(punc_pat, '', t)
        words = t.lower().split()
        if author == 'EAP':
            for w in words:
                poe_words[w] = poe_words.get(w, 0) + 1
        elif author == 'MWS':
            for w in words:
                shelley_words[w] = shelley_words.get(w, 0) + 1
        else:
            for w in words:
                lovecraft_words[w] = lovecraft_words.get(w, 0) + 1
    return poe_words, shelley_words, lovecraft_words
```



Doug imagines Iverson's feedback

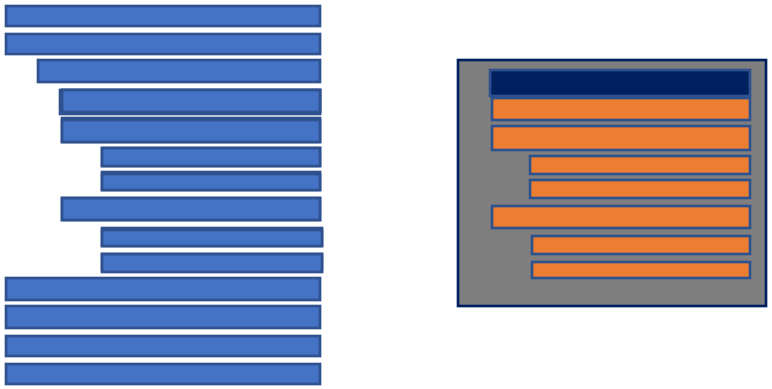


and Doug even remembers refactoring!



Common Refactoring Technique

Extract Function



The DRY principle

- Don’t repeat yourself!
- Find similar code
- Make it exactly the same
- Extract a function!

Extract Functions

Test after each change!

```
test_main()
```

```
p = '[{0}]'.format(re.escape(punc))
ews, mws, hws = {}, {}, {}
for i, t, a in rows:
    t = re.sub('-', ' ', t)
    t = re.sub(p, '', t)
    ws = t.lower().split()
```

Replace hyphens

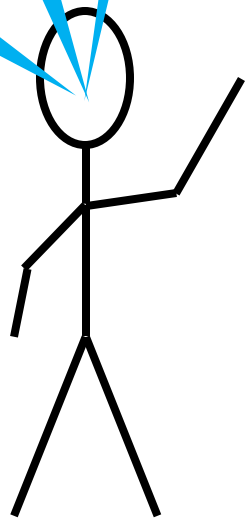
Remove punctuation

```
other_punc = '[{0}]'.format(re.escape(punc))
remove_hypthen = lambda t: re.sub('-', ' ', t)
remove_punc = lambda t: re.sub(other_punc, '', t)
ews, mws, hws = {}, {}, {}
for i, t, a in rows:
    t = remove_hypthen(t)
    t = remove_punc(t)
```

Extract this function!

Better test

What does each part do?



Extract Another Function

Test after each change!

```
test_main()
```

```
    other_punc = '[{0}]'.format(re.escape(punc))
    remove_hypthen = lambda t: re.sub('-', ' ', t)
    remove_punc = lambda t: re.sub(other_punc, '', t)
    ews, mws, hws = {}, {}, {}
    for i, t, a in rows:
        t = remove_hypthen(t)
        t = remove_punc(t)
        ws = t.lower().split()
```

Clean and Split

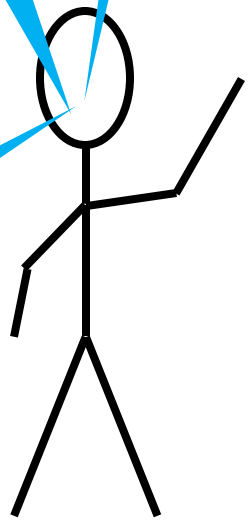
```
def clean_and_split(text):
    other_punc = '[{0}]'.format(re.escape(punc))
    replace_hypthen = lambda t: re.sub('-', ' ', t)
    remove_punc = lambda t: re.sub(other_punc, '', t)
    t = replace_hypthen(text)
    return remove_punc(t).lower().split()

def main(rows):
    ews, mws, hws = {}, {}, {}
    for i, text, a in rows:
        ws = clean_and_split(text)
```

Better test

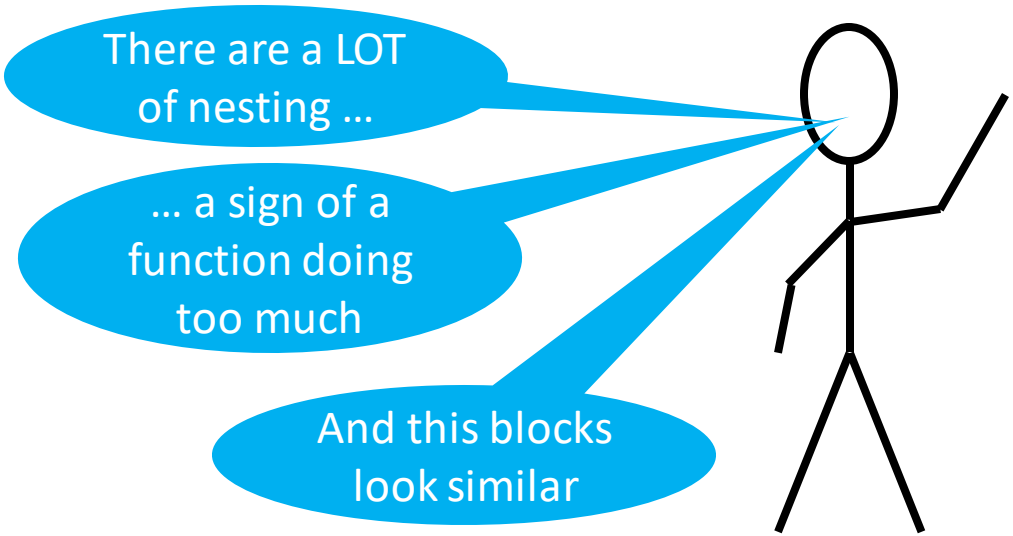
Extract this too!

Any other functions!



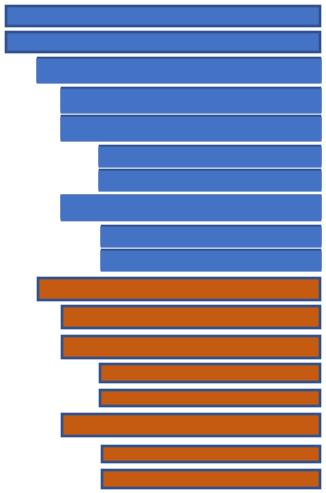
Doug is on a roll now!

```
def main(rows):
    poe_words, shelley_words, lovecraft_words = {}, {}, {}
    for id, text, author in rows:
        words = clean_and_split(text)
        if author == 'EAP':
            for w in words:
                poe_words[w] = poe_words.get(w, 0) + 1
        elif author == 'MWS':
            for w in words:
                shelley_words[w] = shelley_words.get(w, 0) + 1
        else:
            for w in words:
                lovecraft_words[w] = lovecraft_words.get(w, 0) + 1
    return poe_words, shelley_words, lovecraft_words
```



Common Refactoring Technique
Split Loop

1 Loop that
does 2 things

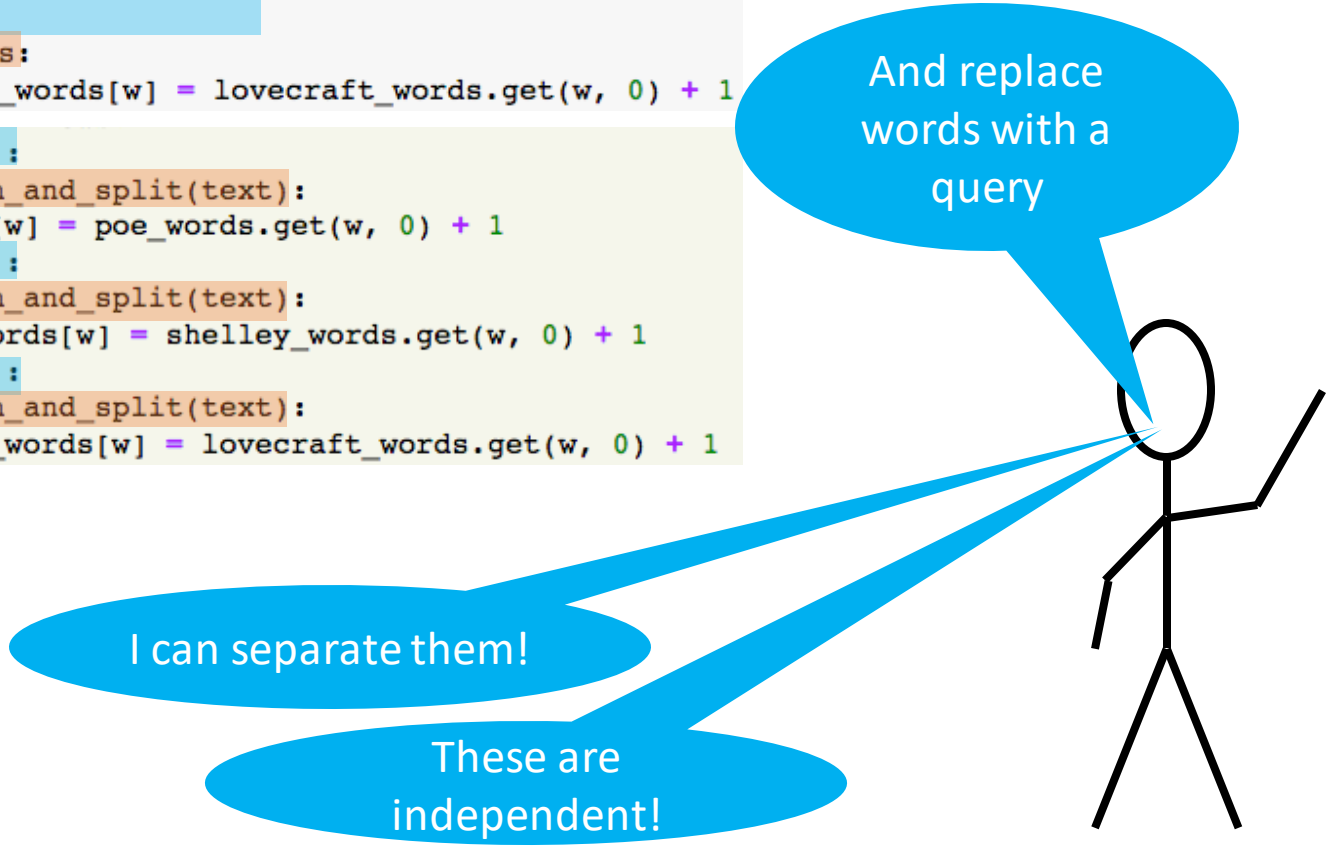


2 Loop that
do 1 thing

Doug makes the blocks identical …

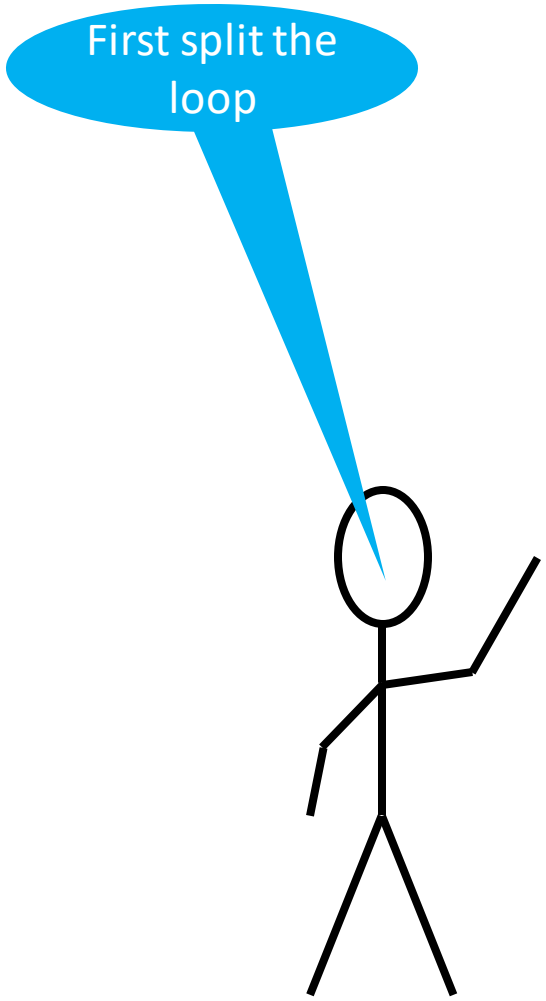
```
words = clean_and_split(text)
if author == 'EAP':
    for w in words:
        poe_words[w] = poe_words.get(w, 0) + 1
elif author == 'MWS':
    for w in words:
        shelley_words[w] = shelley_words.get(w, 0) + 1
else:
    for w in words:
        lovecraft_words[w] = lovecraft_words.get(w, 0) + 1
```

```
if author == 'EAP':
    for w in clean_and_split(text):
        poe_words[w] = poe_words.get(w, 0) + 1
if author == 'MWS':
    for w in clean_and_split(text):
        shelley_words[w] = shelley_words.get(w, 0) + 1
if author == 'HPL':
    for w in clean_and_split(text):
        lovecraft_words[w] = lovecraft_words.get(w, 0) + 1
```



… carefully splits the loop …

```
def main(rows):
    poe_words = {}
    for id, text, author in rows:
        if author == 'EAP':
            for w in clean_and_split(text):
                poe_words[w] = poe_words.get(w, 0) + 1
    shelley_words = {}
    for id, text, author in rows:
        if author == 'MWS':
            for w in clean_and_split(text):
                shelley_words[w] = shelley_words.get(w, 0) + 1
    lovecraft_words = {}
    for id, text, author in rows:
        if author == 'HPL':
            for w in clean_and_split(text):
                lovecraft_words[w] = lovecraft_words.get(w, 0) + 1
    return poe_words, shelley_words, lovecraft_words
```

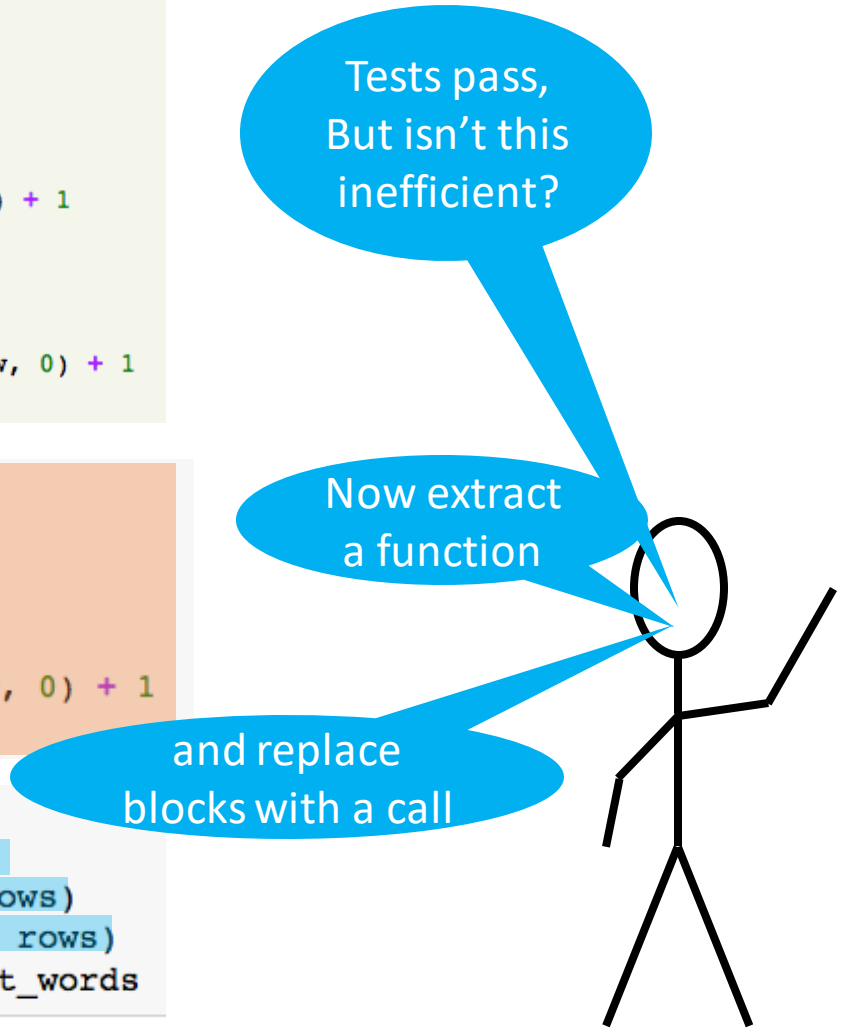


… and extracts a function

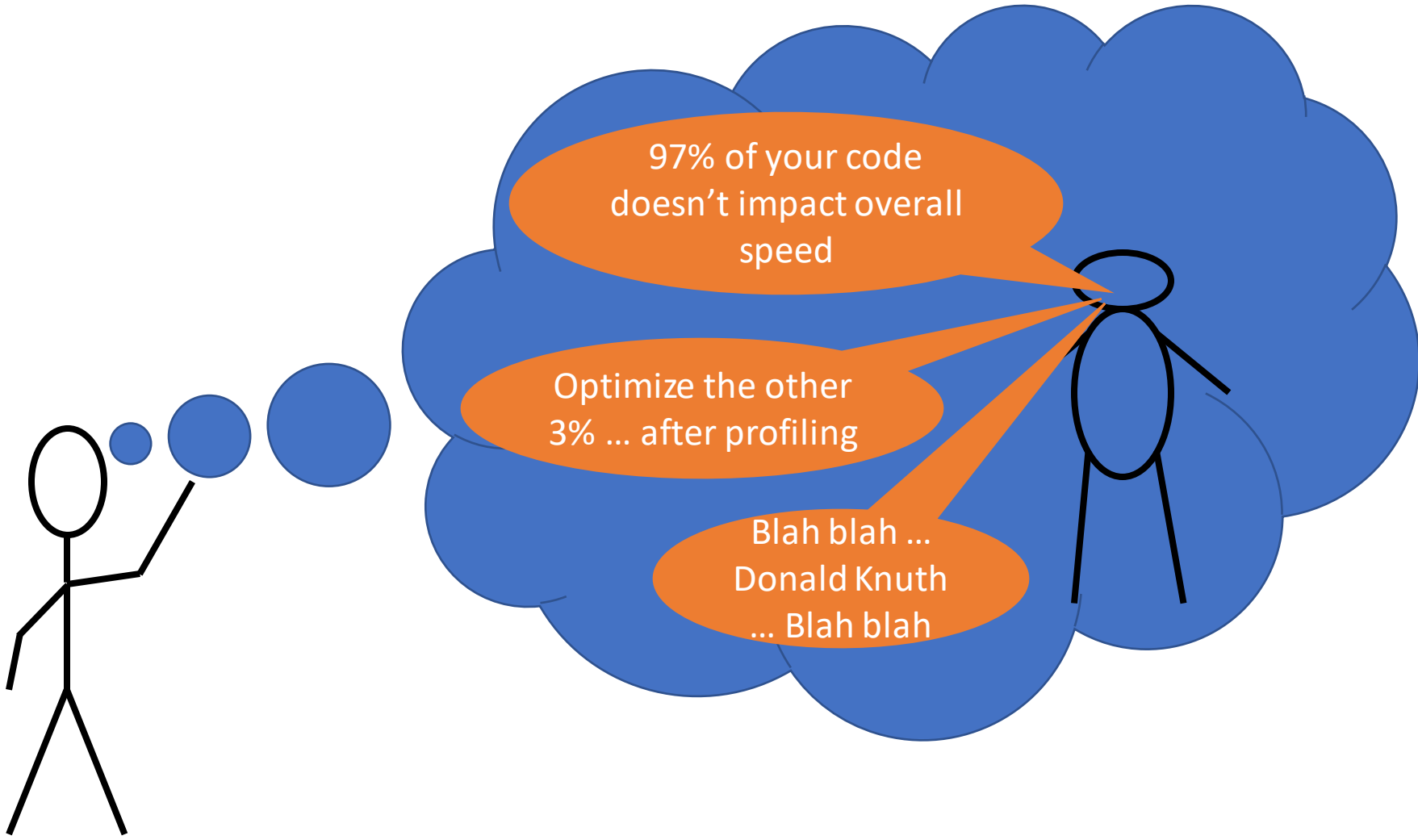
```
def main(rows):
    poe_words = {}
    for id, text, author in rows:
        if author == 'EAP':
            for w in clean_and_split(text):
                poe_words[w] = poe_words.get(w, 0) + 1
    shelley_words = {}
    for id, text, author in rows:
        if author == 'MWS':
            for w in clean_and_split(text):
                shelley_words[w] = shelley_words.get(w, 0) + 1
    lovecraft_words = {}
    for id, text, author in rows:
        if author == 'HPL':
            for w in clean_and_split(text):
                lovecraft_words[w] = lovecraft_words.get(w, 0) + 1
    return poe_words, shelley_words, lovecraft_words
```

```
def get_author_words(auth_code, rows):
    auth_words = {}
    for id, text, author in rows:
        if author == auth_code:
            for w in clean_and_split(text):
                auth_words[w] = auth_words.get(w, 0) + 1
    return auth_words
```

```
def main(rows):
    poe_words = get_author_words('EAP', rows)
    shelley_words = get_author_words('MWS', rows)
    lovecraft_words = get_author_words('HPL', rows)
    return poe_words, shelley_words, lovecraft_words
```

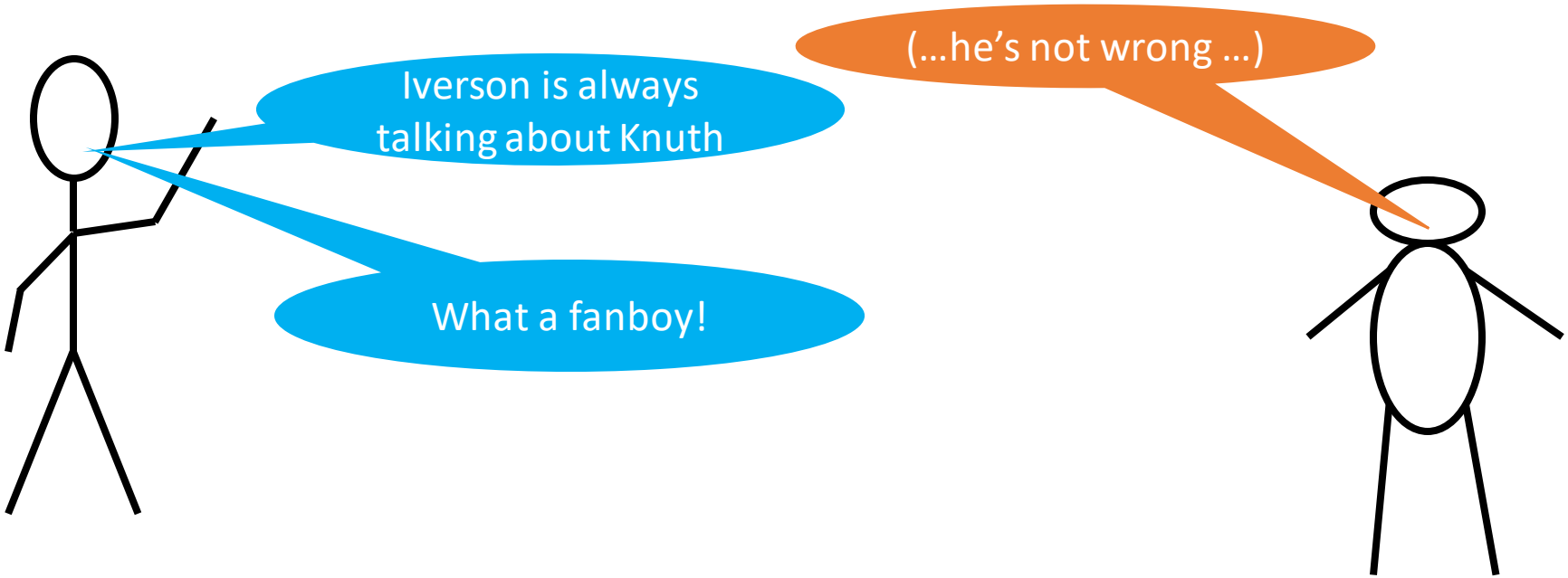


What did Iverson say about efficiency?



The real problem is that programmers have spent far too much time worrying about efficiency in the wrong places and at the wrong times; **premature optimization is the root of all evil (or at least most of it) in programming**.

- Donald Knuth



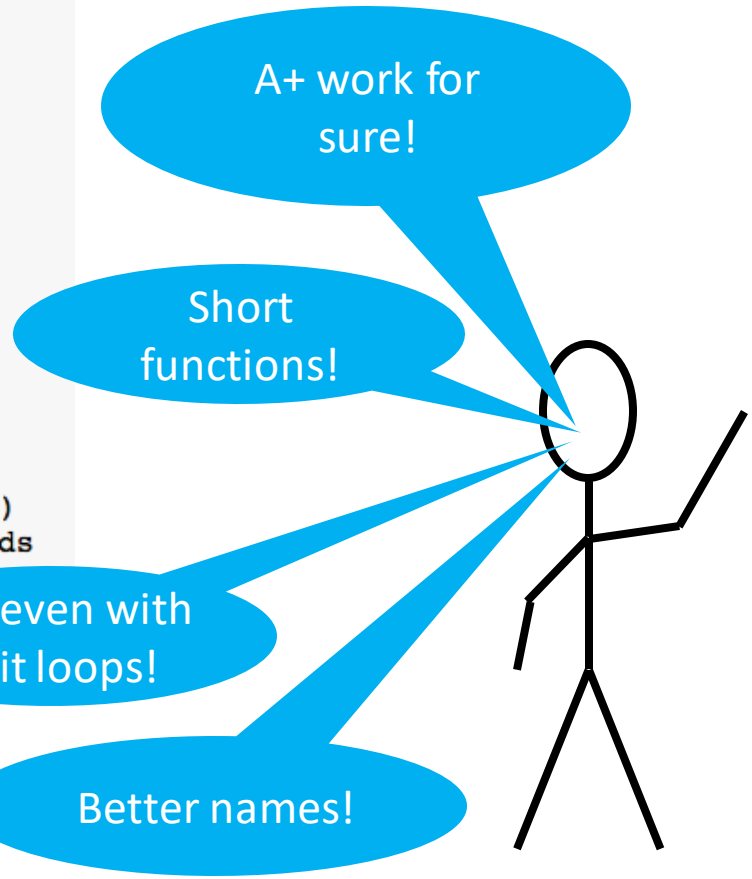
Doug's Final Product

```
def clean_and_split(text):
    other_punc = '[{0}]'.format(re.escape(punc))
    replace_hypthen = lambda t: re.sub('-', ' ', t)
    remove_punc = lambda t: re.sub(other_punc, '', t)
    t = replace_hypthen(text)
    return remove_punc(t).lower().split()

def add_words(word_dict, text):
    for w in clean_and_split(text):
        word_dict[w] = word_dict.get(w, 0) + 1
    return word_dict

def create_word_dict(author, rows):
    word_dict = {}
    for i, text, a in rows:
        if a == author:
            word_dict = add_words(word_dict, text)
    return word_dict

def main(rows):
    poe_words = create_word_dict('EAP', rows)
    shelley_words = create_word_dict('MWS', rows)
    lovecraft_words = create_word_dict('HPL', rows)
    return poe_words, shelley_words, lovecraft_words
test_main()
```



Doug’s code is demonstrably better

He clearly took Iverson’s clean code lectures to heart

His solution consists of many small functions with good names

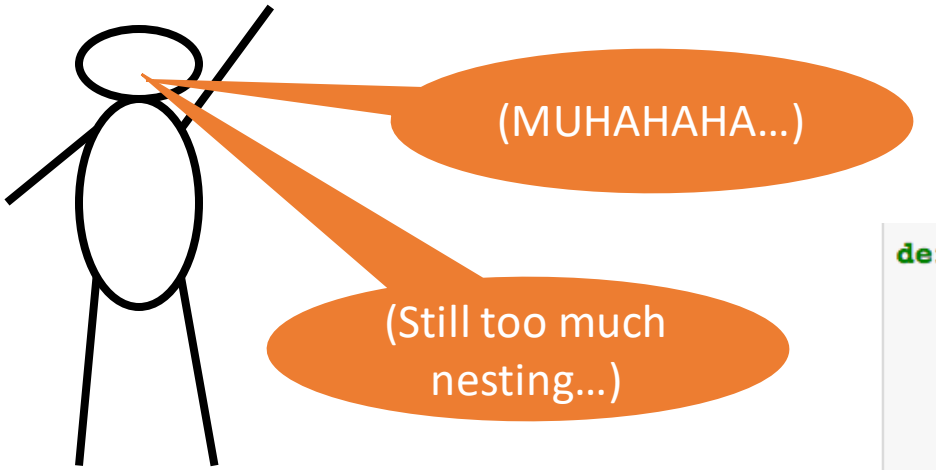
And he even refactored like a pro

So does Doug get the A?

Tune in next week to find out …

.. on the next exciting episode of

Doug Does Data Science




```
def create_word_dict(author, rows):
    word_dict = {}
    for i, text, a in rows:
        if a == author:
            word_dict = add_words(word_dict, text)
    return word_dict
```

Let’s Review

What are unit tests?

- Captures/maintain intended behavior
- Helpful when changing code
- Should be automated

Good names…

- **Reveal intent**
- Use the proper parts of speech
- Have the proper length for their scope
- Avoids disinformation and encodings

Data: What is it?

Function: What does it do?

Good names…

- Reveal intent
- **Use the proper parts of speech**
- Have the proper length for their scope
- Avoids disinformation and encodings

Variable: Noun
Function: Verb
Boolean: Predicate

Refactoring

What is it?

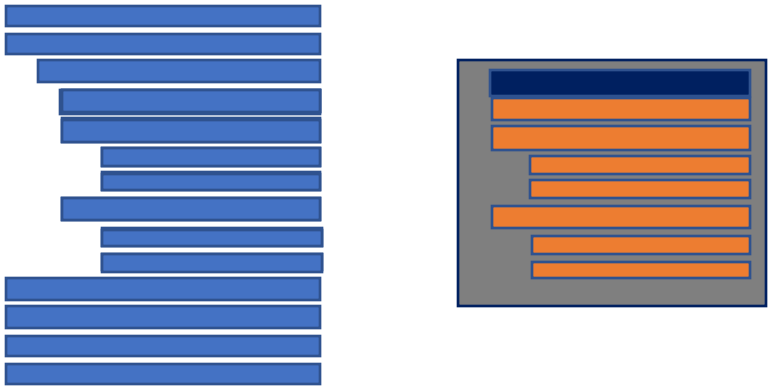
- Reorganize your code
- Break it into different parts
- Change composition

Why use it?

- Understand the code
- Clean the code
- Allow new features

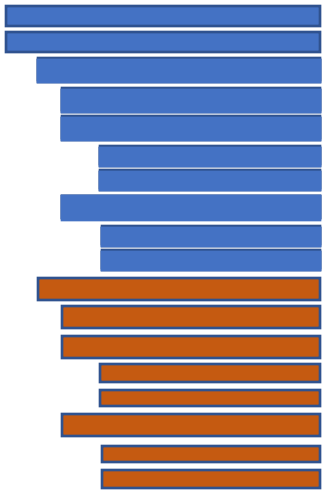
Common Refactoring Technique

Extract Function



Common Refactoring Technique
Split Loop

1 Loop that
does 2 things



2 Loop that
do 1 thing

The DRY principle

- Don’t repeat yourself!
- Find similar code
- Make it exactly the same
- Extract a function!

Advice for teaching clean code

- Require unit tests and good names.
- Don’t just teach it, live it!
- Allow students to see you clean your messy code.
- Teach/reinforce important concepts.
  - DRY
  - Refactoring
  - Efficiency concerns and profiling
- Projects that require 100’s of lines of code.

Clean Code Resources

- These slides: https://bit.ly/2WgFIb
- ***Clean Code***, by Robert Martin
- www.cleancoders.com, videos by Robert Martin and friends
- ***Refactoring Code***, by Martin Fowler