

LECTURE 4.2

Counting words with dictionaries

The idea behind real spam filters

Today's key line of code

```
counts[word] = counts.get(word, 0) + 1
```

Where we are: Lecture 4, part 2

Part 2



Today you will

- 1 Recap: package the spam check as a function
- 2 Split a message into words
- 3 Store word counts in a dictionary
- 4 Count words across many messages
- 5 Build a spam score learned from examples, not typed rules

Recap from 4.1: a spam check as a function

- **def** defines a function: a named, reusable block
- **message** is the input (a parameter)
- **return** hands back the answer and stops
- Call it by name: **is_spam(...)**

Code

```
def is_spam(message):  
    text = message.lower()  
    for word in ["free", "win", "prize"]:  
        if word in text:  
            return True  
    return False  
  
print(is_spam("Win a phone today"))  
print(is_spam("See you at 5"))
```

Output

```
True  
False
```

Split a message into words

- `.split()` cuts text at spaces into a list
- Combine with `.lower()` so WIN and win match
- Punctuation stays stuck: `prize,` is not `prize`
- Cleaning text matters; scikit-learn will help in Lecture 6

Code

```
message = "Win a FREE prize, win now"
words = message.lower().split()
print(words)
print(len(words))
```

Output

```
['win', 'a', 'free', 'prize,', 'win', 'now']
6
```

Dictionaries: look up a value by name

- Curly braces: `{key: value}`
- Look up with `counts["win"]`
- Add or change: `counts["prize"] = 1`
- Like a word book: word → how many times we saw it

Code

```
counts = {"win": 2, "free": 1}
print(counts["win"])
counts["prize"] = 1
counts["win"] = counts["win"] + 1
print(counts)
```

Output

```
2
{'win': 3, 'free': 1, 'prize': 1}
```

Count every word in many messages

`counts.get(word, 0)` means: the current count, or 0 if we have never seen this word.

Code

```
def count_words(messages):
    counts = {}
    for msg in messages:
        for word in msg.lower().split():
            counts[word] = counts.get(word, 0) + 1
    return counts

spam_msgs = ["Win a free prize", "Free cash win now",
             "Claim your free gift"]
spam_counts = count_words(spam_msgs)
print(spam_counts)
```

Output

```
{'win': 2, 'a': 1, 'free': 3, 'prize': 1, 'cash': 1,
'now': 1, 'claim': 1, 'your': 1, 'gift': 1}
```

Tracing count_words, word by word

The first two spam messages. `counts.get(word, 0)` is the count before this word; the dictionary then stores one more.

Code

```
counts = {}
for msg in messages:
    for word in msg.lower().split():
        counts[word] = counts.get(word, 0) + 1
```

message	word	count before	count after
'Win a free prize'	'win'	0	1
'Win a free prize'	'a'	0	1
'Win a free prize'	'free'	0	1
'Win a free prize'	'prize'	0	1
'Free cash win now'	'free'	1	2
'Free cash win now'	'cash'	0	1
'Free cash win now'	'win'	1	2
'Free cash win now'	'now'	0	1

Click to reveal the next row. Predict it first!

Watch it run: count_words

WATCH FOR

- messages is a second name for the list spam
- counts grows by one word at a time
- 'cash' appears twice, so its count reaches 2
- return hands the whole dictionary back

WATCH THE CODE RUN, STEP BY STEP

count_words filling the dictionary

Watch the dictionary counts grow, one word at a time.
This counting is the heart of the spam model.

Click the video to play, click again to pause. Silent: talk the class through it.

Now count normal messages too

- Same function, different examples
- Compare: how often is a word in spam vs. normal?
- "free": 3 in spam, 0 in normal
- "now" appears in both, so it is a weaker clue

Code

```
ham_msgs = ["See you at the library",  
            "Is class at 10 now",  
            "Can you send the notes"]  
ham_counts = count_words(ham_msgs)  
for w in ["free", "now", "you"]:  
    print(w, spam_counts.get(w, 0),  
          ham_counts.get(w, 0))
```

Output

```
free 3 0  
now 1 1  
you 0 2
```

A spam score learned from examples

Add a point for each word seen in spam, subtract one for each word seen in normal messages.

Code

```
def spam_score(message):  
    score = 0  
    for word in message.lower().split():  
        score = score + spam_counts.get(word, 0)  
        score = score - ham_counts.get(word, 0)  
    return score  
  
for msg in ["Free prize for you", "See you in class",  
            "Claim cash now"]:  
    print(spam_score(msg), msg)
```

Output

```
2 Free prize for you  
-4 See you in class  
2 Claim cash now
```

From typed rules to learned counts

Lecture 3: rules we typed

- A person picks the words
- Every new trick needs a new rule
- Yes or no, no sense of strength
- Knows nothing we did not tell it



Lecture 4: counts we learned

- Examples pick the words
- More examples, better counts
- A score: stronger or weaker evidence
- Finds clues we never thought of

Your turn

7 minutes

- 1 Write `word_count(message)` that returns how many words a message has
- 2 Test it on two messages
- 3 Add 2 spam and 2 normal messages of your own to the lists
- 4 Recount, then test `spam_score` on a new message

Code

```
def word_count(message):  
    ...  
    return ...  
  
print(word_count("Win a free prize")) # expect 4
```

Starter code: replace the ... with your own code

Common mistakes with functions and dictionaries

Read the last line of an error first. It names the problem.

What you typed	What Python says	What it means
<pre>print(spam_counts["hello"])</pre>	<code>KeyError: 'hello'</code>	The key does not exist. Use <code>spam_counts.get("hello", 0)</code> .
<pre>def double(x): x * 2 print(double(4))</pre>	<code>None</code>	No error, but <code>None</code> : the function forgot to return.
<pre>print(spam_score())</pre>	<code>TypeError: spam_score() missing 1 required positional argument: 'message'</code>	The function needs an input. Pass a message.

Interactive lab: Word Counting Lab

In the lab you can

1 Split and clean

See `.lower()` and `.split()` turn a message into words.

2 Dictionary builder

Watch `counts.get(word, 0) + 1` fill the dictionary.

3 Spam vs normal

Compare word counts in spam and normal examples.

4 Spam score

Score new messages, then teach the filter more examples.

LECTURE 4.2 · INTERACTIVE LAB

Word Counting Lab

Split messages into words, count them in a dictionary, and build a spam score learned from examples

Introduction to Data Science and Machine Learning
Md Mahbub E Noor · University of Barishal
[All labs](#)

1 Split and clean

2 Dictionary builder

3 Spam vs normal

4 Spam score

Dictionary builder: count every word, one at a time

This is `count_words` from the lecture. Each step handles one word: look up its count (or 0), add one, store it back.

messages (one per line)

Win a free prize
Free cash win now
Claim your free gift

Counting line

`counts[word] = counts.get(word, 0) + 1`

```
1 def count_words(messages):
2     counts = {}
3     for msg in messages:
4         for word in msg.lower().split():
5             counts[word] = counts.get(word, 0) + 1
6     return counts
```

► Next word

► Play

Run to end

↺ Start again

Word 'cash': not in the dictionary yet, so get gives 0.

Stored 1: a new key.

COUNTS (THE DICTIONARY)

key	value
'win'	1
'a'	1
'free'	2
'prize'	1
'cash'	1

5 different words so far.

TRACE

message	word	count before	count after
'Win a free prize'	'win'	0 (new)	1
'Win a free prize'	'a'	0 (new)	1
'Win a free prize'	'free'	0 (new)	1

Open the lab ►

Works offline in any browser
`Labs_open_in_browser/
L4.2_Word_Counting_Lab.html`

Today's toolkit

<code>def f(x):</code>	Define a function with input x
<code>return value</code>	Hand back the answer
<code>text.split()</code>	Cut text into a list of words
<code>d = {"a": 1}</code>	A dictionary: key → value

<code>d["a"]</code>	Look up a key (error if missing)
<code>d.get("a", 0)</code>	Look up, or 0 if missing
<code>d["a"] = 5</code>	Add or change a key
<code>d.get(k, 0) + 1</code>	The word-counting pattern

Homework 4.2

Due before Lecture 5 • [date]

1 From memory

Without looking, write `count_words()` again and test it on 3 sentences. Then compare with the slide.

2 Longest word

Write `longest_word(message)` that returns the longest word. Test: "Claim your fantastic prize" should give "fantastic".

3 Bigger training set

Add 5 spam and 5 normal messages of your own. Recount, then print the spam and ham counts for 5 words you choose.

4 Test the score

Run `spam_score` on 6 new messages. How many does it get right? Explain one mistake.

5 Think ahead

What would happen to `spam_score` if we had 1,000 spam examples but only 10 normal ones? Answer in 3 sentences.

Submit on [LMS / Google Classroom]: your notebook file `HW4_2_<your ID>.ipynb`, or a Colab link set to "Anyone with the link can view".

Recap: Lecture 4.2 in 90 seconds

RECAP

Lecture 4.2 in 90 seconds

A short recap of today's lecture. Pause any time.

Click to play or pause. Watch it again at home to revise.