Regular Expressions

Understanding Regular Expressions

- Very powerful and quite cryptic
- Fun once you understand them
- Regular expressions are a language unto themselves
- A language of “marker characters” - programming with characters
- It is kind of an “old school” language - compact

Example

Give you a string → ” From: test@ntut.edu.tw to Kevin”

How to identify the e-mail ?

Regular Expression Quick Guide

`^`	Matches the beginning of a line
`$`	Matches the end of the line
`.`	Matches any character
`\s`	Matches whitespace
`\S`	Matches any non-whitespace character
`*`	Repeats a character zero or more times
`*?`	Repeats a character zero or more times (non-greedy)
`+`	Repeats a character one or more times
`+?`	Repeats a character one or more times (non-greedy)
`[aeiou]`	Matches a single character in the listed set
`[^XYZ]`	Matches a single character not in the listed set
`[a-z0-9]`	The set of characters can include a range
`(`	Indicates where string extraction is to start
`)`	Indicates where string extraction is to end

The Regular Expression Module

- Before you can use regular expressions in your program, you must import the library using “import re”
- You can use re.search() to see if a string matches a regular expression, similar to using the find() method for strings
- You can use re.findall() to <u>extract portions of a string</u> that match your regular expression, similar to a combination of find() and slicing: var[5:10]

Using re.search() Like find()

```
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if line.find('From:') >= 0:
        print(line)
```

```
import re

hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if re.search('From:', line) :
        print(line)
```

Using re.search() Like startswith()

```
hand = open('mbox-short.txt')
for line in hand:
    line = line.rstrip()
    if line.startswith('From:') :
        print(line)
```

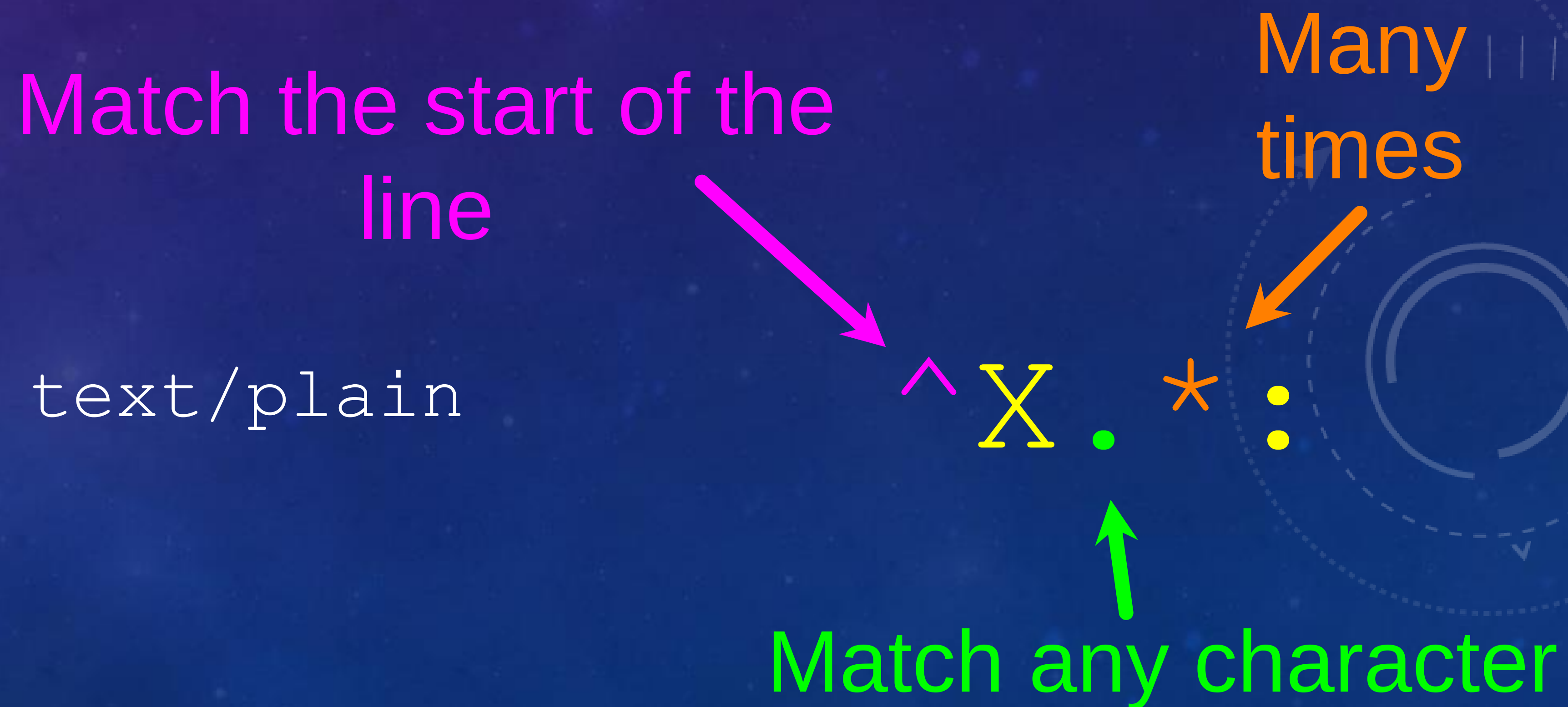
```
import re

hand = open('mbox-short.txt', ‘r’)
for line in hand:
    line = line.rstrip()
    if re.search('^From:', line) :
        print(line)
```

We fine-tune what is matched by adding special characters to the string

Wild-Card Characters

- The dot character matches any character
- If you add the asterisk character, the character is “any number of times”

```
X-Sieve: CMU Sieve 2.3
X-DSPAM-Result: Innocent
X-DSPAM-Confidence: 0.8475
X-Content-Type-Message-Body: text/plain
```



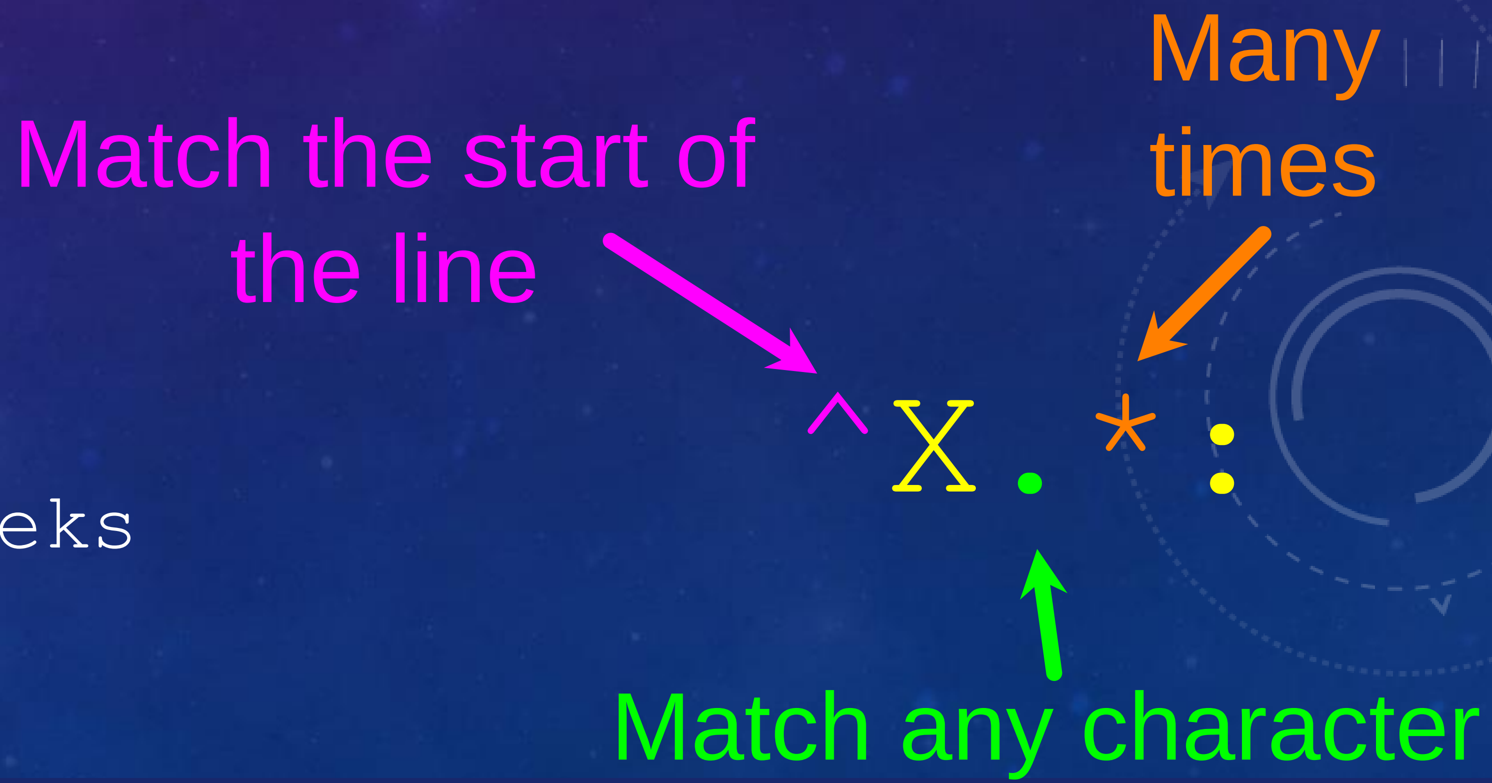
```
x = "X-Sieve: CMU Sieve 2.3"
print(re.findall("^X.*:", x))
```

```
→ ['X-Sieve:']
```

Fine-Tuning Your Match

Depending on how “clean” your data is and the purpose of your application, you may want to narrow your match down a bit

```
X-Sieve: CMU Sieve 2.3
X-DSPAM-Result: Innocent
X-Plane is behind schedule: two weeks
```



Fine-Tuning Your Match

Depending on how “clean” your data is and the purpose of your application, you may want to narrow your match down a bit

```
X-Sieve: CMU Sieve 2.3
X-DSPAM-Result: Innocent
X-Plane is behind schedule: two weeks
```

Match the start of the line

One or more times

```
^X-\S+:
```

Match any non-whitespace character

Matching and Extracting Data

- re.search() returns a True/False depending on whether the string matches the regular expression

- If we actually want the matching strings to be extracted, we use re.findall()

```
[0-9]+
```

$\uparrow$

One or more digits

```
>>> import re
>>> x = 'My 2 favorite numbers are 19 and 42'
>>> y = re.findall('[0-9]+',x)
>>> print(y)
['2', '19', '42']
```

Matching and Extracting Data

When we use re.findall(), it returns a list of zero or more sub-strings that match the regular expression

```
>>> import re
>>> x = ‘My 2 favorite numbers are 19 and 42'
>>> y = re.findall('[0-9]+',x)
>>> print(y)
['2', '19', '42']
>>> y = re.findall('[AEIOUM]+',x)
>>> print(y)
[‘M’]
```

Warning: Greedy Matching

The repeat characters (* and +) push outward in both directions (greedy) to match the largest possible string

```
>>> import re
>>> x = 'From: Using the : character'
>>> y = re.findall('^F.+:', x)
>>> print(y)
['From: Using the :']
```

One or more characters

`^F.+:`

First character in the match is an F

Last character in the match is a :

Why not 'From:' ?

Non-Greedy Matching

Not all regular expression repeat codes are greedy!
If you add a ? character, the + and * chill out a bit...

```
>>> import re
>>> x = 'From: Using the : character'
>>> y = re.findall('^F.+?:', x)
>>> print(y)
['From:']
```

One or more characters but not greedy

```
^F.+?:
```

First character in the match is an F

Last character in the match is a :

Fine-Tuning String Extraction

You can refine the match for re.findall() and separately determine which portion of the match is to be extracted by using parentheses

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
>>> y = re.findall('\S+@\S+',x)
>>> print(y)
['stephen.marquard@uct.ac.za’]
```

`\S+@\S+`

At least one non-whitespace character

Fine-Tuning String Extraction

Parentheses are not part of the match - but they tell where to start and stop what string to extract

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
>>> y = re.findall('\S+@\S+',x)
>>> print(y)
['stephen.marquard@uct.ac.za']
>>> y = re.findall('^From (\S+@\S+)',x)
>>> print(y)
['stephen.marquard@uct.ac.za']
```

```
^From (\S+@\S+)
```

Summary: Use re.findall to perform extraction

```
x = "From: test@ntut.edu.tw to Kevin"
ret = re.findall("^F.+: (\S+@\S+)", x)
print(ret)
```

```
['test@ntut.edu.tw']
```

```
x = "From: Bill to Kevin"
ret = re.findall("^F.+: (\S+@\S+)", x)
print(ret)
```

Another way: Use re.search to perform extraction

```
import re

x = "From: test@ntut.edu.tw to QQQQ"
ret = re.search("^F.+:", x)
if ret:
    print(x[ret.end():])

    x = x[ret.end():]
    ret = re.search("\S+@\S+", x)
    print(ret.group())
```

```
ryanpan@RyanPanPC $ python3 test.py
 test@ntut.edu.tw to QQQQ
test@ntut.edu.tw
```

STRING PARSING EXAMPLES…

```
                     21        31
                     ↓         ↓
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
>>> data = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
>>> atpos = data.find('@')
>>> print(atpos)
21
>>> sppos = data.find(' ',atpos)
>>> print(sppos)
31
>>> host = data[atpos+1 : sppos]
>>> print(host)
uct.ac.za
```

Extracting a host name - using find and string slicing

The Double Split Pattern

Sometimes we split a line one way, and then grab one of the pieces of the line and split that piece again

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
words = line.split()
email = words[1]
pieces = email.split('@')
print(pieces[1])
```

```
stephen.marquard@uct.ac.za
['stephen.marquard', 'uct.ac.za']
'uct.ac.za'
```

The Regex Version

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
import re
line = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('@([^ ]*)',line)
print(y)

['uct.ac.za']
```

`'@([^ ]*)'`

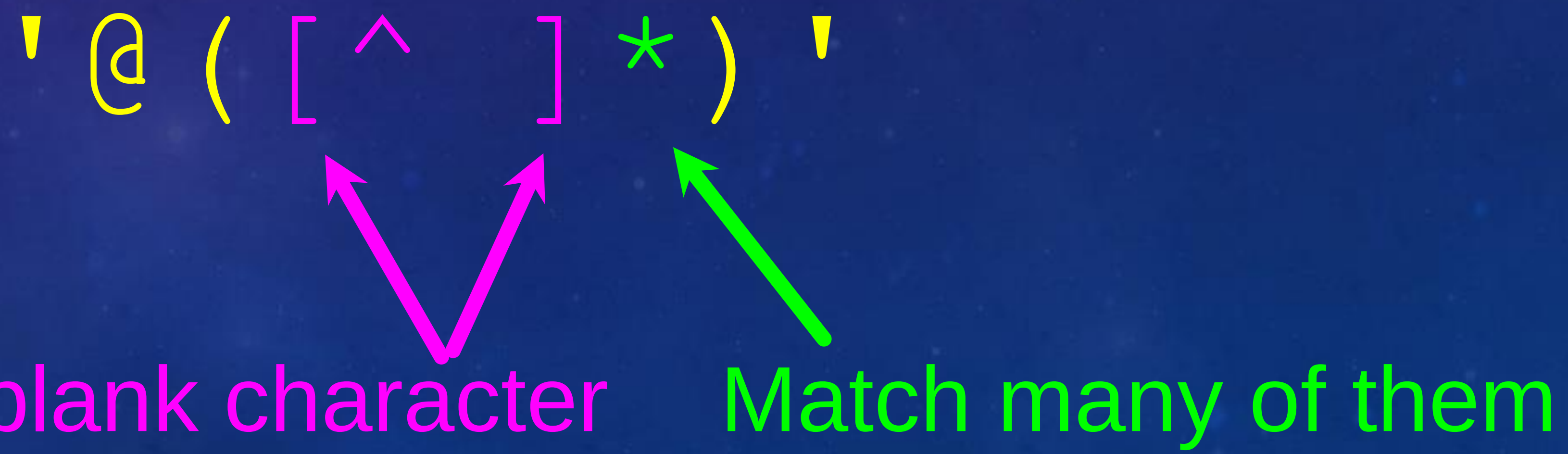
Look through the string until you find an at sign

The Regex Version

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('@([^ ]*)',lin)
print(y)
```

```
['uct.ac.za']
```



The Regex Version

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('@([^ ]*)',lin)
print(y)

['uct.ac.za']
```

```
'@([^ ]*)'
```

Extract the non-blank characters

Even Cooler Regex Version

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('^From .*@([^ ]*)',lin)
print(y)
```

```
['uct.ac.za']
```

```
'^From .*@([^ ]*)'
```

Starting at the beginning of the line, look for the string 'From '

Even Cooler Regex Version

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('^From .*@([^ ]*)',lin)
print(y)

['uct.ac.za']
```

```
'^From .*@([^ ]*)'
```

Skip a bunch of characters, looking for an at sign

Even Cooler Regex Version

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('^From .*@([^ ]*)',lin)
print(y)
```

```
['uct.ac.za']
```

$$\underset{\substack{\uparrow\\ \text{Start extracting}}}{\texttt{'^From .*@}\texttt{(}}\texttt{[^]*)'}$$

Even Cooler Regex Version

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('^From .*@([^ ]*)',lin)
print(y)

['uct.ac.za']
```

```
'^From .*@([^ ]+)'
```

Match non-blank character

Match many of them

Even Cooler Regex Version

```
From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008
```

```
import re
lin = 'From stephen.marquard@uct.ac.za Sat Jan  5 09:14:16 2008'
y = re.findall('^From .*@([^ ]*)',lin)
print(y)
```

```
['uct.ac.za']
```

```
'^From .*@([^ ]+)'
```

Stop extracting

Spam Confidence

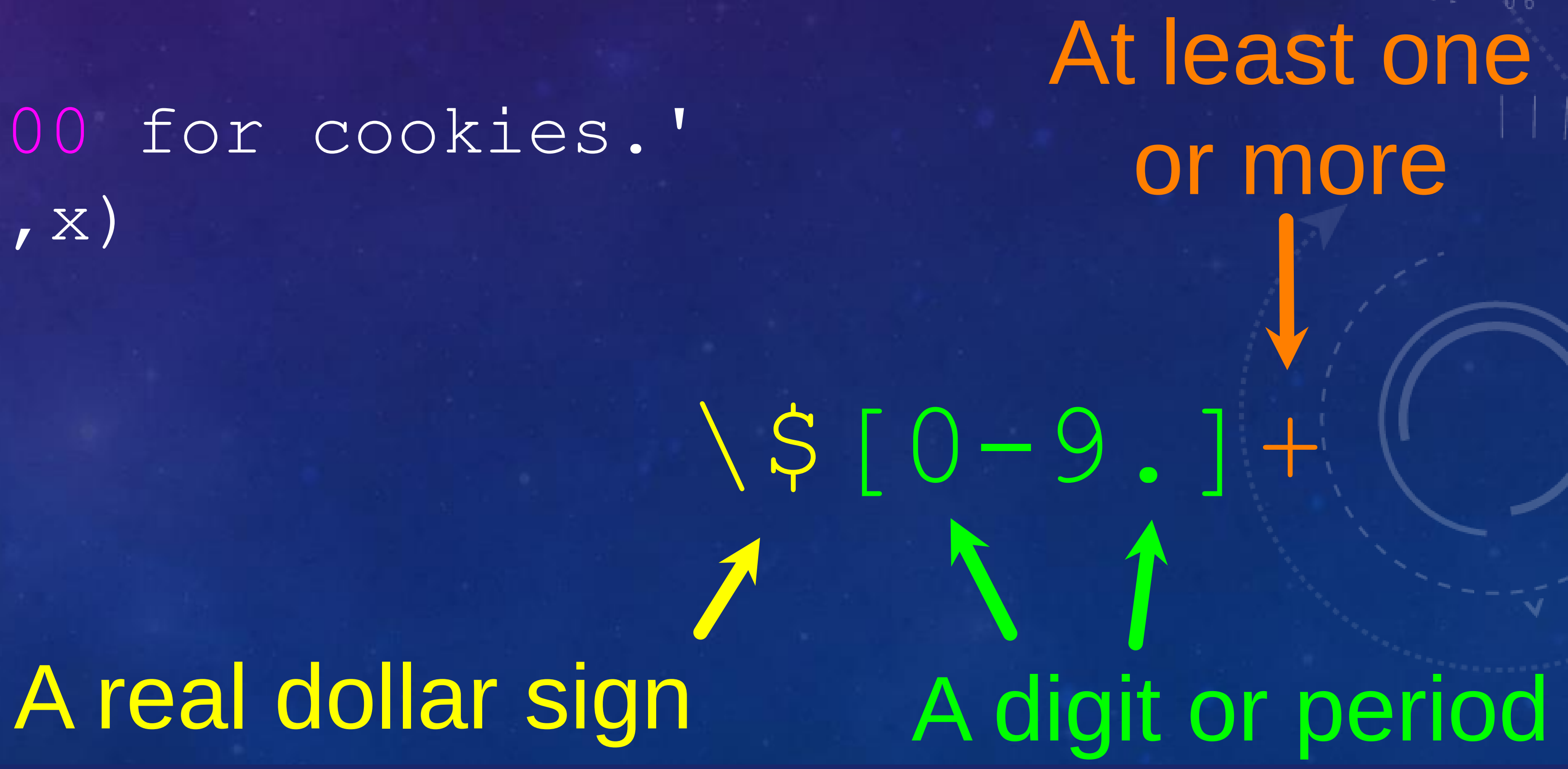
```
X-DSPAM-Confidence: 0.8475
```

```
import re
hand = open('mbox-short.txt')
numlist = list()
for line in hand:
    line = line.rstrip()
    stuff = re.findall('^X-DSPAM-Confidence: ([0-9.]+)', line)
    if len(stuff) != 1 :
        continue
    num = float(stuff[0])
    numlist.append(num)
print('Maximum:', max(numlist))
```

Escape Character

If you want a special regular expression character to just behave normally (most of the time) you prefix it with '\'

```
>>> import re
>>> x = 'We just received $10.00 for cookies.'
>>> y = re.findall('\$[0-9.]+',x)
>>> print(y)
['$10.00']
```



Summary

- Regular expressions are a cryptic but powerful language for matching strings and extracting elements from those strings
- Regular expressions have special characters that indicate intent

HW 1

I will give you a LOG.txt like this

```
=== LOG.txt ===

Hella buys Computer for $734
Alice buys Computer for $548
[VIP] Peter buys Computer for $666
[VIP] Peter buys Book for $973
Alice buys Paper for $545
Alice buys Notebook for $501
Bob buys Paper for $182
[VIP] Sue buys Notebook for $396
[VIP] Sue buys Notebook for $4
Bob buys Book for $850
Bob buys Book for $691
```

Please analyze to a file like this

```
=== Analysis_result.txt ===

[VIP]
Peter buys Computer: 666, Book: 973
Sue buys NoteBook: 400

[Member]
Hella buys Computer: 734
Alice buys Computer: 548, Paper 545, Notebook: 501
Bob buys Paper: 182, Book 1541

Total Computer sales: 1948
Total NoteBook sales: 901
Total Paper sales: 627
Total Book sales: 2514
```

HW2

```
import requests

for i in range(1):
        url = 
f'https://exam.naer.edu.tw/searchResult.php?page={i}&orderBy=lastest&keyword=&selCountry=&selCategory=0&selTech=0&selYear=&selTerm=&selType=&selP
ublisher='
        res = requests.get(url)
        print(res.text)
```

```
縣立國聖國小 六年級 112 下學期 數學領域 /otc/testStoreFile/100286866612ac2ecf28.pdf
縣立國聖國小 六年級 112 下學期 語文領域 /otc/testStoreFile/100286866612ac2c65ce.pdf
縣立內安國小 六年級 112 下學期 數學領域 /otc/testStoreFile/1002958665fcf3eb6f57.pdf
縣立內安國小 六年級 112 下學期 語文領域 /otc/testStoreFile/1002958665fcf3eb47c7.pdf
縣立內安國小 六年級 112 下學期 數學領域 /otc/testStoreFile/1002958665fce9ec1fc8.pdf
縣立內安國小 六年級 112 下學期 語文領域 /otc/testStoreFile/1002958665fce9e5de85.pdf
市立石門國中 八年級 112 下學期 社會領域 /otc/testStoreFile/1002028665d0a8f0b8d3.pdf
市立石門國中 七年級 112 下學期 社會領域 /otc/testStoreFile/1002028665d0a8f0b144.pdf
…………..
```