# Regular Expressions

# Regular Expressions

| ? | ! | ^$ |
| --- | --- | --- |
| Why<br>Regular Expressions? | What<br>are they? | How<br>to use them? |

# Regular Expressions

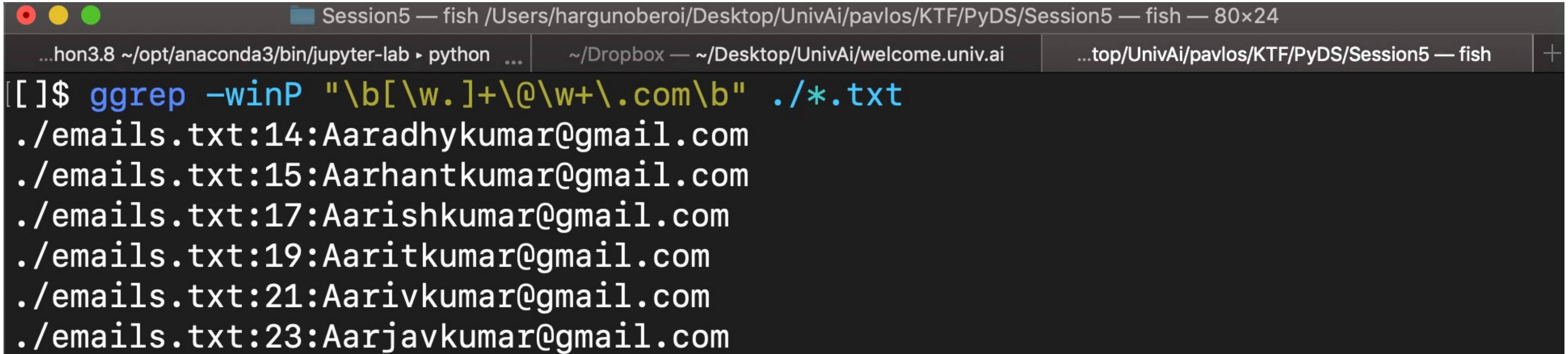
?

Why
Regular Expressions?

# Why Regular Expressions ?

The .* option allows you to search with regular expressions


The *grep* command allows you to search on the command line



# 20th century Data Science with Regular Exp

```
Session5 — fish /Users/hargunoberoi/Desktop/UnivAi/pavlos/KTF/demos/Session5 — fish — 80×24
...hon3.8 ~/opt/anaconda3/bin/jupyter-lab ▸ python ...   ~/Dropbox — ~/Desktop/UnivAi/welcome.univ.ai   ...op/UnivAi/pavlos/KTF/demos/Session5 — fish
[]$ python regex.py inception.txt
2
Top words :Frequency
the       :**************************************************1580
cobb      :********************************1033
you       :***************** 544
and       :************ 394
arthur    :*********** 349
ariadne   :********** 333
his       :********* 285
fischer   :******** 268
saito     :******* 248
int       :******* 225
eames     :****** 220
mal       :****** 199
that      :****** 192
looks     :***** 167
him       :***** 165
into      :**** 145
what      :**** 133
for       :**** 129
they      :*** 126
out       :*** 126
```

# Regular Expresssions


`"Pavlos Protopapas"`

Dr. Pavlos Protopapas is the Scientific Program Director, Institute for Applied Computational Science (IACS) at Harvard University, and leads the Data Science Masters program at Harvard. Pavlos has had a long and distinguished career as a scientist and data science educator, and today teaches the CS109 series for basic and advanced data science, as well as the capstone course (industry-sponsored data science projects) for the IACS masters program at Harvard

# Regular Expressions

```
"([A-Z]\w+\s){0,}[A-Z]\w+"
```

Dr. Pavlos Protopapas is the Scientific Program Director, Institute for Applied Computational Science (IACS) at Harvard University, and leads the Data Science Masters program at Harvard. Pavlos has had a long and distinguished career as a scientist and data science educator, and today teaches the CS109 series for basic and advanced data science, as well as the capstone course (industry-sponsored data science projects) for the IACS masters program at Harvard

# Regular Expressions

*A sequence of characters that define a search pattern, mainly for use in pattern matching with strings, or string matching, i.e. "find and replace"-like operations.*

Pavlos Protopapas

Applied Computational Science

Data Science Masters

Harvard University

```
"([A-Z]\w+\s){0,}[A-Z]\w+"
```

# Regular Expressions

!

What
are they?

'Room (528-491)'

| character class | Represents | Example | Result |
|---|---|---|---|
| \d | Any numeric digit from 0 to 9. | '\d\d\d' | ['528,'491'] |
| \D | Any character that is *not* a numeric digit from 0 to 9. | \D\D\D\D' | ['Room'] |
| \w | Any letter, numeric digit, or the underscore character. (Think of this as matching "word" characters.) | '\w\w\w' | ['Roo', '528', '491'] |
| \W | Any character that is *not* a letter, numeric digit, or the underscore character. | '\W\W\W' | [' ('] |
| \s | Any space, tab, or newline character. (Think of this as matching "space" characters.) | '\s\s' | ['  '] |
| \S | Any character that is *not* a space, tab, or newline. | '\S\S\S' | ['Roo', '(52', '8-4', '91)'] |

# Regular Expressions

| Character | Description | Example | Result |
|---|---|---|---|
| ? | Match zero or one repetitions of preceding | "ab?" | "a" or "ab" |
| * | Match zero or more repetitions of preceding | "ab*" | "a", "ab", "abb", "abbb"… |
| + | Match one or more repetitions of preceding | "ab+" | "ab", "abb", "abbb"… but not "a" |
| {n} | Match n repetitions of preeeding | "ab{2}" " | "abb" |
| {m,n} | Match between m and n repetitions of preceding | "ab{2,3}" | "abb" or "abbb" |

# Regular Expressions
## Making your own character classes

These classes may be limiting, for e.g., if you need to match only letters from the alphabet

```
\d        \w        \s
```

↓

```
[...]
```

# Regular Expressions

^$

How to use them?

# Regular Expressions

```
import re
regex = re.compile(r'\d{10}')
regex.findall('My number is 7775978484')
>>>['7775978484']
```

# Regular Expressions

```
import re
regex = re.compile(r'\d{10}')
regex.findall('My number is 7775978484')
>>>['7775978484']
```

# Regular Expressions

```
import re
regex = re.compile(r'\d{10}')
regex.findall('My number is 7775978484')
>>>['7775978484']
```

# Regular Expressions

```
import re
regex = re.compile(r'\d{10}')
regex.findall('My number is 7775978484')
>>>['7775978484']
```