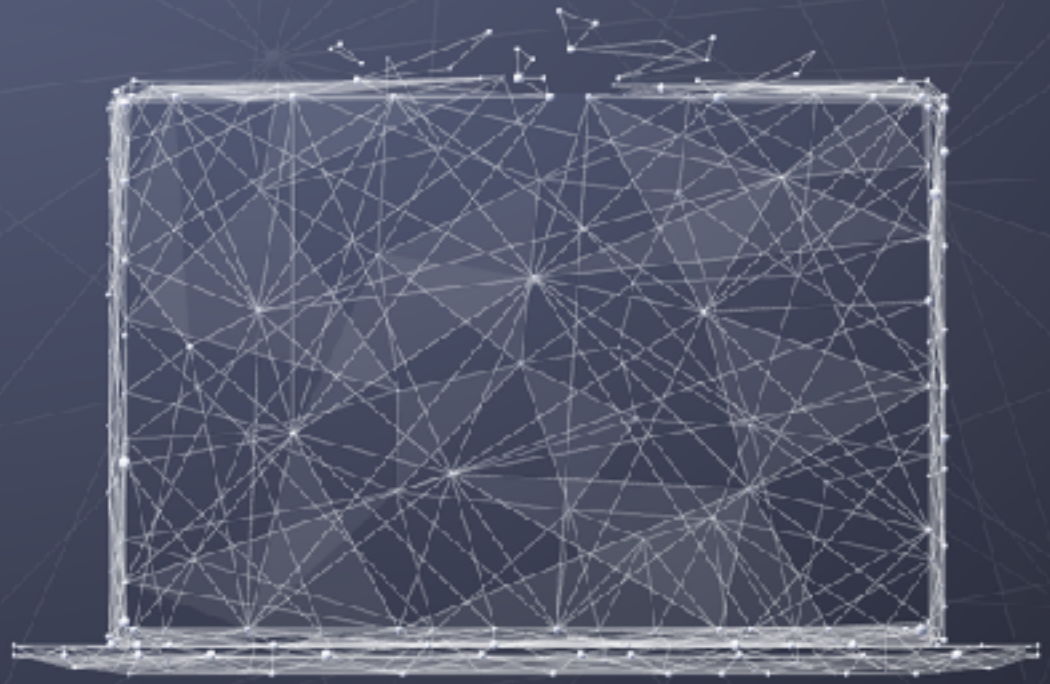
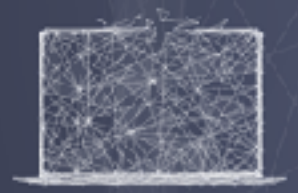
Data Science
Data Engineering I

Regular expressions

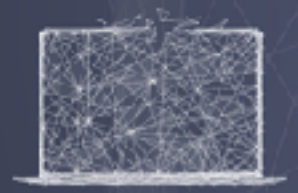


PURDUE UNIVERSITY | College of Science



Regular expressions

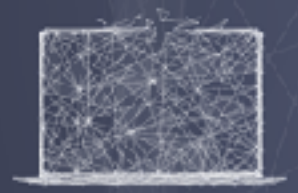
- Often when parsing, scraping, and processing data you need to search for specified elements in the text
- Regular expression are useful for these operations
- A regular expression is a notation to specify a pattern, which is matched by a set of strings.
- The power of regular expressions comes in the ability to match flexibly over possible sequences of characters
- In python the re module provides support for regular expressions



How to specify regex pattern

- The simplest regular expressions are a string of literal characters to match
- The string matches the regular expression if it contains the substring
- A regular expression can match a string in more than one place

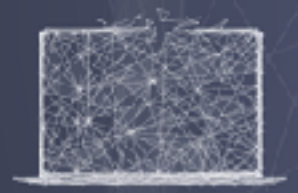
```
Regex: man

Hello world, this is a test file.
Testing the manner in which regular expressions
work in many different situations and match
many text strings at once.
```



How to specify regex pattern

- The “.” regular expression can be used to match any character

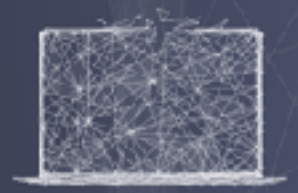
```
Regex: .ions

Hello world, this is a test file.
Testing the manner in which regular expressions
work in many different situations and match
many text strings at once.
```



How to specify regex pattern

- Character classes “[ ]” can be used to match any specific set of characters

```
Regex: [Tt]est

Hello world, this is a test file.
Testing the manner in which regular expressions
work in many different situations and match
many text strings at once.
```

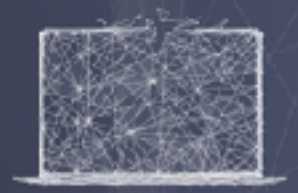


How to specify regex pattern

- Character classes can be negated with the “[^]” syntax

```
Regex: man[^n]

Hello world, this is a test file.
Testing the manner in which regular expressions
work in many different situations and match
many text strings at once.
```

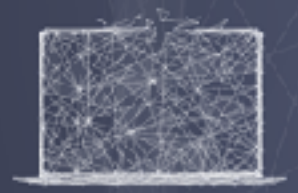
PURDUE UNIVERSITY | College of Science



How to specify regex pattern

- Anchors are used to match at the beginning or end of a line (or both)
- “^” means beginning of the line
- “$” means end of the line

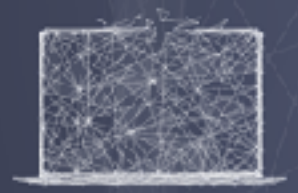
```
Regex: ^many

Hello world, this is a test file.
Testing the manner in which regular expressions
work in many different situations and match
many text strings at once.
```



How to specify regex pattern

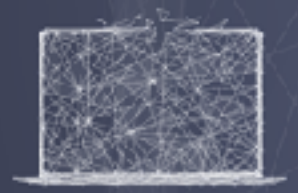
- Anchors are used to match at the beginning or end of a line (or both)
- “^” means beginning of the line
- “$” means end of the line

```
Regex: s$

Hello world, this is a test file.
Testing the manner in which regular expressions
work in many different situations and match
many text strings at once.
```



How to specify regex pattern

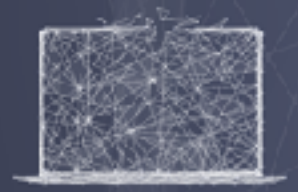
Regex	Matches
a	character ‘a’
.	any character except \n
[]	one character specified inside [], e.g. [aeiou]
[^]	one character NOT specified inside [] after ^, eg. [^abc]
-	one character in range inside [], eg. [0-9] matches any digit
^	beginning of line (when not used in [^])
$	end of line (when not used in [] or [^])
\d	any digit [0-9]
\D	non-digit characters [^0-9]
\s	whitespace character [\t\n\r\f\v]
\S	non-whitespace character [^ \t\n\r\f\v]
\w	alphanumeric character [a-zA-Z0-9]
\W	non-alphanumeric character [^a-zA-Z0-9]



Repetition

- “*” is used to define zero or more occurrences of the single regex preceding it
- “?” is used to define zero or one occurrence of the single regex preceding it
- “+” is used to define one or more occurrences of the single regex preceding it

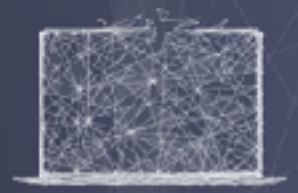
```
Regex: man*

Hello world, this is a test file.
Testing the manner in which regular expressions
work in many different situations and match
many text strings at once.
```



Subexpressions

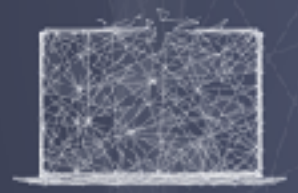
- If you want to group part of an expression so that * or ? or + applies to more than just the previous character, use ( ) notation

```
Regex: "in(gs)* "

Hello world, this is a test file.
Testing the manner in which regular expressions
work in many different situations and match
many text strings at once.
```



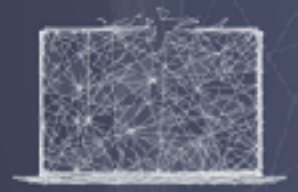
Some regular expression methods in python

- match( )
  Determine if the RE matches at the beginning of the string
- search( )
  Scan through string, look for any location where this RE matches
- findall( )
  Find all substrings where the RE matches, and return as a list
- finditer( )
  Find all substrings where the RE matches, and return as a iterable set of Match objects



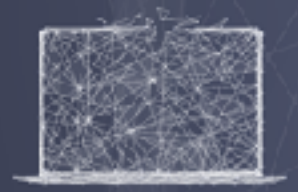
Regular expression in python

```
# regular expression operation library
import re

# compile the regular expression object for reuse
pattern = re.compile(r"test")    #r indicates that pattern is a raw string

str = "This is a test that tests regular expression basics."

# match tests if string starts with expression
re.match(pattern, str)

# group() returns the string matched by the pattern
# span() returns a tuple with the start,end positions of the match
```



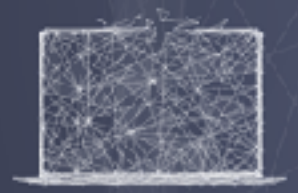
Examples

```
str = "This is a test that tests regular expression basics for numbers
1234 and letters A-Z."

pattern = re.compile(r"test.?")
print(re.findall(pattern, str))
['test ', 'tests']

for match in re.finditer(pattern, str):
    print(match.group(), match.span())
test  (10, 15)
tests (20, 25)
```



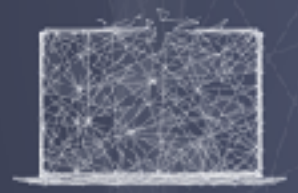
Examples

```
str = "This is a test that tests regular expression basics for numbers
1234 and letters A-Z."

pattern = re.compile(r"\d")
print(re.findall(pattern, str))
['1', '2', '3', '4']

pattern = re.compile(r"\d\s+")
print(re.findall(pattern, str))
['4 ']
```



Using regular expressions in Pandas

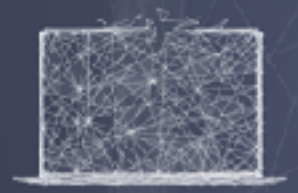
- Regular expressions can be used in many Pandas selection and filtering methods, including:
  - pandas.Series.str.contains
  - pandas.Series.str.extract
  - pandas.Series.str.match
  - pandas.Series.str.replace
  - pandas.DataFrame.filter



Pandas example

```
# select rows uses regular expression
data[data.Name.str.contains('^Kath')]
```

	Index	Year	Age	Name	Movie
5	6	1933	26	Katharine Hepburn	Morning Glory
39	40	1967	60	Katharine Hepburn	Guess Who's Coming to Dinner
40	41	1968	61	Katharine Hepburn	The Lion in Winter
54	55	1982	74	Katharine Hepburn	On Golden Pond
63	64	1991	42	Kathy Bates	Misery



Pandas example

```
# select rows uses regular expression
data[(data.Name.str.contains('C')) & (data.Movie.str.contains('Mo*n'))]
```

	Index	Year	Age	Name	Movie
60	61	1988	41	Cher	Moonstruck
76	77	2004	28	Charlize Theron	Monster