CS 1112: Introduction To Programming

Regular Expressions (RegEx):

Character Classes

Dr. Nada Basit // basit[*at*]Virginia[*dot*]edu

Friendly Reminders

- Your ***safety*** and ***comfort*** is important!
  - If you choose to wear a mask you are welcome to do so
  - *We will interpret wearing a mask as being considerate and caring of others in the classroom (not that you are sick), and realize that some may choose to mask to remain distanced*

- Remember to always be **kind**, **respectful**, **supportive**, **compassionate** and **mindful of others**! ☺

- Be an ***active*** participant in your learning!
  You're welcome and ***encouraged*** to ask questions during class!

- If you feel ***unwell***, or think you are, please stay home
  - *Contact us!* *We will work with you!*
  - Get some rest ☺
  - View the recorded lectures – *please allow 24-48 hours to post*

Announcements

- **Quiz 7** is due by 11:00pm on March 31, 2025 (***Tonight!***)
- **PA06** is due by 11:00pm on April 3, 2025 (***Thursday***)
  - Grace period is Friday, April 4 (must submit prior to **11pm** on Friday!)
  - TA office hours are made available on Thursday and Friday **this week** to assist

Coming up…

- **Exam 2**: Monday, April 7, 2025 *(SDAC accommodations? Book exam time slot on April 7!)*
  - *In-class*
  - *Closed-book/closed-notes/closed-PyCharm/closed-Internet/closed-Computer/closed-everything!*
  - *Duration: 1 hour and 15 minutes*

More on Regular Expressions

CS 1112

Other Notation

- The format of regular expressions are made up of many special characters and notations
- We will introduce some more of them today
- Note, we will not be covering every single special character and notation, but we will cover some useful and common ones
  - Feel free to explore others on your own!

RegEx: use of Metacharacters $

- Matches the end of the string or just before the newline (\n)
- **Spain$** will match the string that ends with “Spain”

```
import re

# Check if the string starts with "The" (notice ^)
# and ends with "Spain” (notice $):
txt = "The rain in Spain"
x = re.search("^The.*Spain$", txt)

if x:
  print("YES! We have a match!")
else:
  print("No match")
```

Remember this?

RegEx: use of Metacharacters $

- Matches the end of the string or just before the newline (\n)
- Spain$ will match the string that ends with “Spain”

```
import re

# Check if the string starts with "The" (notice ^)
# and ends with "Spain” (notice $):
txt = "The rain in Spain"
x = re.search("^The.*Spain$", txt)

# Output if we printed x:
<re.Match object; span=(0, 17), match='The rain in Spain'>
```

RegEx: use of Metacharacters {m}

- Specifies that exactly ***m*** copies of the *previous RE* should be matched

- a{4} will match exactly four 'a' characters, but not 3 or less
- "aaaa"   will be matched by the above regEx
- "aaa"   will NOT be matched by the above regEx

RegEx: use of Metacharacters {m,n}

- Causes the resulting regular expression to match from ***m*** to ***n*** repetitions of the *previous RE,* attempting to match as many repetitions as possible

- a{3,5} will match from 3 to 5 ‘a’ characters
- “aaa” will be matched by the above regEx
- “aaaa” will be matched by the above regEx
- “aaaaa” will be matched by the above regEx

- “aa” will NOT be matched by the above regEx

RegEx: use of Metacharacters {m,n}

- Omitting ***m*** specifies a lower bound of zero, and omitting ***n*** specifies an infinite upper bound

- **a{4,}b** will match four 'a' characters followed by a 'b', or a *thousand* 'a' characters followed by a 'b'
- "aaaab" will be matched by the above regEx
- "aaaaaaaaab" will be matched by the above regEx

- "aaab" will NOT be matched by the above regEx
- "baaaa" will NOT be matched by the above regEx

Character Classes

- There are some additional metacharacters that represent character classes
- These are short-cut versions of some of the more verbose regular expression patterns

Character class: \d

- Matches any decimal digit: `[0-9]`

- **r\dd2** matches the following:

```
txt = "r2d2"
x = re.findall(r"r\dd2", txt)
```

Further explanation of the pattern expressed by the regular expression:
- Matches “r”
- Matches a digit
- Matches “d”
- Matches 2
(So, it would match the string, “r2d2” as above.)

→
```
# Output if we printed x:
['r2d2']
```

Character class: \D

- Matches any character which is NOT a decimal digit: `[^0-9]`
- \D\D\d[a-z] matches the following:

```
txt = “nb3f"
x = re.findall(r“\D\D\d[a-z]", txt)
```

Character class: \s

- Matches any whitespace characters: `[ \t\n…]`

space character

- **aa\sbb** matches the following:

```
txt = “aa bb"
x = re.findall(r“aa\sbb", txt)
```

Character class: \S

- Matches any character which is NOT a whitespace character: `[^ \t\n...]`

- **[0-9][0-9]\Sn** matches the following:

```
txt = “12!n"
x = re.findall(r“[0-9][0-9]\Sn", txt)
```

Character class: \w

- Matches any word characters (alphanumeric characters as well as the underscore): [a-zA-Z0-9_]

Want the character "." so use \.

underscore character

- \w+\@\w+\.edu matches the following:

```
txt = “my_email1@virginia.edu"
x = re.findall(r“\w+\@\w+\.edu", txt)

# Remember:  + matches the RE 1 or more times

# Output if printed x:
['my_email1@virginia.edu']
```

Character class: \W

- Matches any character which is NOT a word character: `[^a-zA-Z0-9_]`

- **[0-9]{2}\W[0-9]{2}pm** matches the following:

```
txt = “10:22pm"
x = re.findall(r“[0-9]{2}\W[0-9]{2}pm", txt)

# Remember:  + matches the RE 1 or more times
```

Character class: \b (not used often)

- A word boundary. Matches if the specified characters are at the beginning or end of a word

- ain\b matches the following:
- "The rain in Spain" matches 'ain' twice in this string

```
import re

txt = "The rain in Spain"
# Check if "ain" is present at the end of a WORD:
x = re.findall(r"ain\b", txt)  # Will find it twice!
# Check if "ain" is present at the beginning of a WORD:
y = re.findall(r"\bain", txt)  # No Match
```

Character class: \b (not used often)

- A word boundary. Matches if the specified characters are at the beginning or end of a word

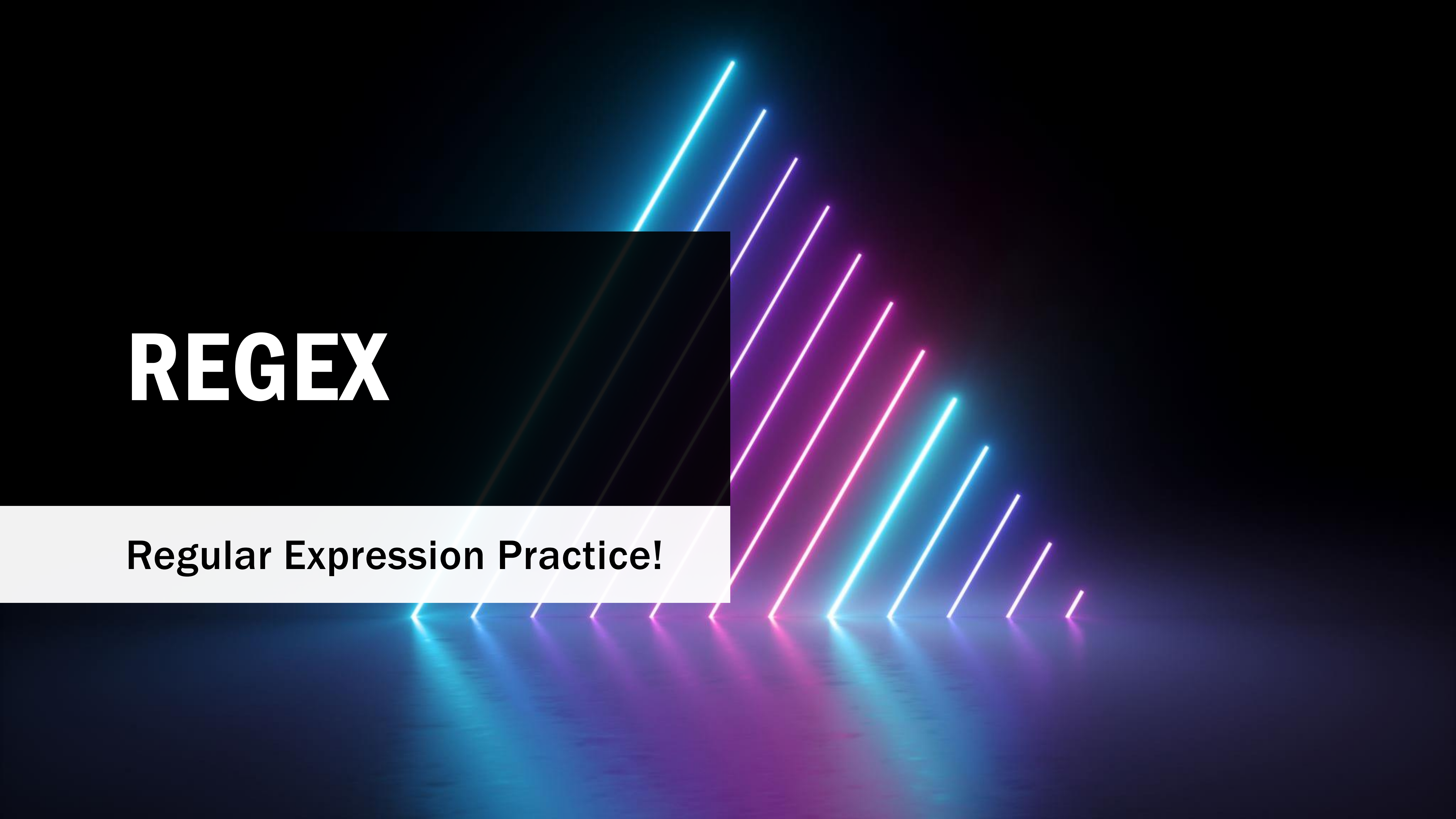
- **ain\b** matches the following:
- "The rain in Spain"   matches 'ain' twice in this string

```
import re

txt = "The rain in Spain"
# Check if "ain" is present at the end of a WORD:
x = re.findall(r"ain\b", txt)  # Will find it twice!
# Output if we printed x:
['ain', 'ain']
```

SUMMARY

- ^ = matches the beginning of a string
- $ = matches the end of a string
- {m} = exactly m copies of the previous RE
- {m,n} = from m to n copies of the previous RE (*both inclusive*)
- \d = matches any decimal digit
- \D = matches any character that is not a decimal digit
- \s = matches any whitespace character
- \S = matches any character that is not a whitespace character
- \w = matches any word character
- \W = matches any character that is not a word character
- \b = word boundary matching RE at beginning or end of a word



REGEX

Regular Expression Practice!

EXAMPLE 1

```
t{3}[0-9]?\sa
```

- **Provide strings that match this regular expression**

EXAMPLE 2

```
[a-zA-Z]\w*\d\d
```

- ➢ Describe the strings that would match this regular expression
- ➢ Minimum length of strings that would match?

EXAMPLE 3

```
p(ab)+\W[lmn]
```

- Provide strings that match this regular expression
- Describe the strings that would match this regular expression

EXAMPLE 4

```
CS\s\d{4}-\w+
```

- ➢ **Will this regular expression match <u>the entire</u> string below?**
  - ➢ `“CS 1112-Intro to Programming”`

IN-CLASS ACTIVITY

Go to Quizzes

Click on “RegEx Quiz ICA” – not an actual quiz (MC practice questions)

In pairs work on the Quiz

Check-in with a TA before leaving class

  Show them that you have completed the quiz

  (Don’t worry about what score you actually earned!)