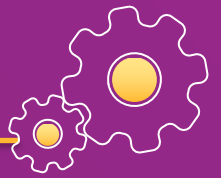
# CSC461
# INTRODUCTION TO DATA SCIENCE

**Dr. Muhammad Sharjeel**

https://muhmmadsharjeel.github.io/


# REGULAR EXPRESSIONS

# REGULAR EXPRESSIONS



- Once we have the data, we can do a lot of stuff with it
  - Search for specific elements within the data
- Regular expressions (regexp or regex) are a text-matching tool embedded in Python
  - Useful in creating string searches and string modifications
  - Documentation: http://docs.python.org/library/re.html

# REGULAR EXPRESSIONS

- Example: Find the first occurrence of the string “data science” within a text

```
import re
text = "This course will introduce the basics of data science"
result = re.search(r"data science", text)
print(result.start())
```

- More examples:
  - Check if start of text matches

```
result = re.match(r"data science", text)
```

  - Iterate over all matches in the text

```
for result in re.finditer(r"data science", text):
    print("found")
```

  - Return all matches

```
all_results = re.findall(r"data science", text)
```

# REGULAR EXPRESSIONS

- match()
  - match the beginning of a string
  - Returns none or a match object
- search()
  - match anywhere in a string
  - Returns none or a match object
- findall()
  - match anywhere in a string
  - Returns a list of strings (or an empty list)
    - Note that it does not return a match object
- match.start()
  - Returns the start index of the matched substring
- match.end()
  - Returns the end index of the matched substring
- match.span()
  - Returns a tuple containing start and end index of the matched part

# REGULAR EXPRESSIONS

- Special characters in regular expressions . ^ $ * + ? { } \ [ ] | ( )
  - To match special characters exactly, escape them using a backslash \
- Regular expressions could be used to match multiple possible sequence of characters
- Example: To match
  - the character 'a' : a (or any other single character)
  - the character 'a', 'b', or 'c' : [abc] (any character provided within the brackets could be matched)
  - any character except 'a', 'b', or 'c' : [^abc]
  - a range to characters 'a' to 'e' : [a-e] (or any range provided)
  - any digit : \d (same as [0-9])
  - any non-digit : \D (opposite of \d)
  - any alpha-numeric : \w (same as [a-zA-Z0-9_])
  - any non-word and non-number : \W (opposite of \w)
  - match whitespace : \s (or to match newslines, tabs etc. [\t \n \r \f \v])
  - match any single character : . (the dot operator)
- Braces are a more detailed way to indicate repeats
  - a{1,3} means at least one and no more than three a's
  - a{4,4} means exactly four a's
- Dollar symbol $ is used to check if a string ends with a certain character
  - Example: a$ will return true for 'formula'

# REGULAR EXPRESSIONS

- Use modifiers to match one or more instances of a character (or set of characters)
- Example: To match
  - match character 'a' exactly once : a
  - match character 'a' zero or one time : a?
  - match character 'a' zero or more times : a*
  - match character 'a' one or more times : a+
  - match character 'a' exactly n times : a{n}
- It is possible to combine regular expressions for multiple character matching
  - Example: To match all instances of “<something> Science” where <something> is an alphanumeric string with at least one character
    - \w+\s+Science
- Use grouping to enclose portions of the regular expression(s) in quotes to “remember” these portions of the match

```
import re
result = re.search(r"(\w+)\s([Ss]cience)", text)
print(result.start(), result.groups())
```

# REGULAR EXPRESSIONS

- Regular expressions also provide a mechanism for replacing some text with other text

```
import re
text = "This course will introduce the basics of data science"
result = re.sub(r"data science", r"computer science", text)
result = re.sub("w","*","which words get replaced?")
```

- strip() and split()
- Strip will remove whitespace at the beginning or end of the input
- Split method splits a string into a list using a separator
  - It takes the separator as an argument
  - Default separator is whitespace

# REGULAR EXPRESSIONS

- Examples of strip() and split()

```
result = "What's your email? ".strip()
if "@" in result:
    print("Valid")
else:
    print("Invalid")
```

---

```
if "@" in result and "." in result:
    print("Valid")
```

---

```
username, domain = result.split("@")
if username and "." in domain:
    print("Valid")
```

---

```
username, domain = result.split("@")
if username and domain.endswith(".edu"):
    print("Valid")
```

# REGULAR EXPRESSIONS

- There is an order of operations in regular expressions
  - abc|def matches the strings “abc” or “def”, not “ab(c or d)ef”
- This could be handled better using parenthesis, e.g., a(bc|de)f

- By default, regular expressions try to capture as much text as possible (greedy matching)
  - <(.*)> applied to <a>text</a> will match the entire expression
- If required to capture the least amount of text possible, use <(.*?)>, it’ll just match the <a> term

# REGULAR EXPRESSIONS

## Practice problems

- Write a regexp that will match any string that starts with “csc” and ends with “001” with any number of characters, including none, in between
  - Hint: consider using a wildcard character
- Write a regexp that will match name and extension of any Python (.py) file
  - There must be at least one character (name of the file) before the “.” (file extension dot)
- Check using a regexp the following string contain a legal Python filename?
  - String: “This contains two files, assignment1.py and ids.py”
- Write a regexp which detects legal Microsoft Word file names in a string and make a list of them
  - File names must end with “.doc” or “.docx”
  - There must be at least one character before the . (file extension dot)
  - Assume there are no spaces in the file names
- Print out the start location of the first such filename you encounter in the following string
  - String: “Please edit the following two MS Word documents (bscs.doc) and (bsse.docx), and share with us”

# THANKS