# Lesson 5.04
# Regular Expressions

# What are Regular Expressions?

- “Regex” for short
- Symbols representing a text pattern
- Used for matching, searching, and replacing text
  - Examples include extracting data that follows a specific pattern or check if columns in Data Frames contain data in the correct format

# Usage Examples

- Test if the phone number has the correct number of digits
- Test if an email address in valid format
- Test if passwords meet complexity criteria
- Search a web page for either “color” or “colour”
- Replace all occurrences of “dollar” or “dollars” with “$”
- Count how many times “course” is preceded by “DSI”, “SEI” or “UXI”

# Usage Examples

- Phone Number Sample Value: 62884876
- Regex Representation: \d\d\d\d\d\d\d\d

# Notation Conventions

- Text String: Hello World
- Regex: /(Hello World)/

IMPORTANT

r'...' denotes raw strings which ignore escape code, i.e., r'\n' is '\'+'n'

# Literal Characters

- Similar to “Find” function in Word Document
- Case Sensitive by default
- /car/ matches “car”
- /car/ matches the first 3 letters of “carnival”

# REGEX Cheat Sheet - 1

### Character classes

| | |
|---|---|
| `.` | any character except newline |
| `\w \d \s` | word, digit, whitespace |
| `\W \D \S` | not word, digit, whitespace |
| `[abc]` | any of a, b, or c |
| `[^abc]` | not a, b, or c |
| `[a-g]` | character between a & g |

### Anchors

| | |
|---|---|
| `^abc$` | start / end of the string |
| `\b \B` | word, not-word boundary |

### Escaped characters

| | |
|---|---|
| `\. \* \\` | escaped special characters |
| `\t \n \r` | tab, linefeed, carriage return |

### Groups & Lookaround

| | |
|---|---|
| `(abc)` | capture group |
| `\1` | backreference to group #1 |
| `(?:abc)` | non-capturing group |
| `(?=abc)` | positive lookahead |
| `(?!abc)` | negative lookahead |

### Quantifiers & Alternation

| | |
|---|---|
| `a* a+ a?` | 0 or more, 1 or more, 0 or 1 |
| `a{5} a{2,}` | exactly five, two or more |
| `a{1,3}` | between one & three |
| `a+? a{2,}?` | match as few as possible |
| `ab\|cd` | match ab or cd |

Read more at [https://regexr.com/](https://regexr.com/)

# REGEX Cheat Sheet - 2

### Special Characters

- `\` escape special characters
- `.` matches any character
- `^` matches beginning of string
- `$` matches end of string
- `[5b-d]` matches any chars '5', 'b', 'c' or 'd'
- `[^a-c6]` matches any char except 'a', 'b', 'c' or '6'
- `R|S` matches either regex R or regex S
- `()` creates a capture group and indicates precedence

### Quantifiers

- `*` 0 or more (append ? for non-greedy)
- `+` 1 or more (append ? for non-greedy)
- `?` 0 or 1 (append ? for non-greedy)
- `{m}` exactly mm occurrences
- `{m, n}` from m to n. m defaults to 0, n to infinity
- `{m, n}?` from m to n, as few as possible

### Special sequences

- `\A` start of string
- `\b` matches empty string at word boundary (between `\w` and `\W`)
- `\B` matches empty string not at word boundary
- `\d` digit
- `\D` non-digit
- `\s` whitespace: `[ \t\n\r\f\v]`
- `\S` non-whitespace
- `\w` alphanumeric: `[0-9a-zA-Z_]`
- `\W` non-alphanumeric
- `\Z` end of string
- `\g<id>` matches a previously defined group

### Extensions

- `(?iLmsux)` Matches empty string, sets re.X flags
- `(?:...)` Non-capturing version of regular parentheses
- `(?P<name>...)` Creates a named capturing group.
- `(?P=name)` Matches whatever matched previously named group
- `(?#...)` A comment; ignored.
- `(?=...)` Lookahead assertion: Matches without consuming
- `(?!...)` Negative lookahead assertion
- `(?<=...)` Lookbehind assertion: Matches if preceded
- `(?<!...)` Negative lookbehind assertion
- `(?(id)yes|no)` Match 'yes' if group 'id' matched, else 'no'

Read more at [http://www.pythex.org/](http://www.pythex.org/)