

The goal of this project is to create a Python script that scans the contents of a text file for phone numbers, email addresses, and website addresses, and then extracts these into separate text files. The script will employ regular expressions to perform the scanning and extraction tasks, and basic file operations to read the source text file and write the results to new text files.

Below are detailed requirements for exactly what the program should accomplish:

## 1. Define Regular Expressions:

Create three separate regular expressions, each designed to match a specific type of data: phone numbers, email addresses, and website addresses. The regexes should be represented as raw strings in Python, which are strings prefixed with an 'r' that treat backslashes as literal characters.

- a. **Phone Number Regex:** This pattern should match a North American phone number that consists of three groups of digits - 3, 3, and 4 digits long, respectively – that are separated by either a dash or a space. For example, **123-456-7890** and **123 456 7890** should both be matches.

Also, the first group of three digits may *optionally* be nested inside of parentheses (*remember: you'll have to escape these!*). This means **(123)-456-7890** should match as well.

- b. **Email Regex:** This pattern should match any sequence of characters consisting of the following:
  - i. One or more alphanumeric characters (OR dots), then a “@” symbol (this is the username)
  - ii. One or more alphanumeric characters, followed by a dot (*remember, you'll have to escape any dot characters in your regex*)
  - iii. **One** of the following: “com”, “net”, or “org” (either-or logic could help here)

Examples of emails that should be matched by your regex include [emailme@cmail.com](mailto:emailme@cmail.com), [number1regexfan@coldmail.net](mailto:number1regexfan@coldmail.net), and [match.me@yasshoo.org](mailto:match.me@yasshoo.org).

- c. **Website Regex:** This pattern should match website addresses that start with one or more alphanumeric characters, followed by a dot, and a domain ending in either “com”, “net”, or “org”. Pretty much just like an email, but without the username (i.e., “yourname@”) piece! And that’s kind of the rub!

We need to be able to distinguish emails from websites, but the problem is that websites basically look like *subsets* of email addresses. So how to get around this?

Well, you *could* simply match a broader pattern that neither specifically includes or excludes the “@” symbol, and then parse the matches out depending on whether they contain “@”...but that wouldn’t be much fun. This section is about regexes after all!

So instead, I want you to design your regex to ensure that a website match starts with a sequence of alphanumeric characters, by requiring the **preceding character** to be either a *whitespace character*, **or** a *start-of-string anchor*. That way, you know for sure that the sequence of alphanumeric characters preceding the “dot” in the website address either comes at the beginning of the string, or after a word in the text -NOT after an “@” sign.

In other words:

- If “mywebsite.com” is at the very beginning of the test string, it should match.
- If it comes after another word in a sentence (i.e., it is preceded by a space), it should also match. For example, “my website is **mywebsite.com**” should produce a match.
- However, “email me at [tlc43@mywebsite.com](mailto:tlc43@mywebsite.com)” should NOT produce a match.

One last (important!) consideration – while the start anchor/whitespace character is technically part of your regex pattern, it should **not** be included in the string you ultimately write to a text file. That means you’ll need to put the rest of the pattern in a group of its own, so it can be captured separately.

Whew! Ok, that’s it on the regex-specific stuff 😊

## 2. Read the Input File:

Use Python to open and read the provided example file (a plaintext file), using the techniques you learned earlier in the course.

Since the focus of this section is on regular expressions, you can keep things simple here by dropping the file into the same directory as your project, so you don’t have to bother with defining an explicit path to the file.

## 3. Find Matches with the Regular Expressions:

Use the **findall** method to find all matches for each regular expression in the contents of the input file. This should be done separately for each of the three regular expressions, and the matches should be stored in separate variables.

## 4. Process the Matches:

For each set of matches (phone numbers, email addresses, and websites), create an empty list. Then loop through the sets of matches, adding each match to its respective list IF that exact match isn’t *already* in the list.

Keep in mind that since some of your regexes will have *groups*, the desired match produced by the `findall` method may be nested within a tuple, that is in turn nested inside a list.

Also, since you'll want each match to appear on a different line in the target text files, you may want to append a newline character to the end of each match before adding it to its respective list.

## **5. Write the Matches to Separate Text Files:**

For each list of matches, open a new text file (with filenames such as 'phone\_numbers.txt', 'emails.txt', and 'websites.txt') and write the contents of the list to the file. There is a method built into Python which lets you write a list of data to a text file in one fell swoop – remember it?