[Skip to main content](https://courses.iitbombayx.in/courses/course-v1:IITBombayX+CS101.2x+2022_T1/courseware/a83378a1759347a8a8eb84e3dfcdefcb/56e137a8d36d4fc1a48c6072273b3e13/?child=first" \l "main)

**[IITBombayX Home Page](http://iitbombayx.in/)**

**IITBombayX: CS101.2xObject-Oriented Programming**

[Courses](http://iitbombayx.in/courses)

[Help](https://iitbombayx.in/contact_us)

[[https://courses.iitbombayx.in/static/images/profiles/default_50.3455a6581573.png](https://courses.iitbombayx.in/dashboard) Dashboard for:Paramhans\_Singh](https://courses.iitbombayx.in/dashboard)

1. [Course, current location](https://courses.iitbombayx.in/courses/course-v1:IITBombayX+CS101.2x+2022_T1/course/)

1. [Discussion](https://courses.iitbombayx.in/courses/course-v1:IITBombayX+CS101.2x+2022_T1/discussion/forum/)

1. [Progress](https://courses.iitbombayx.in/courses/course-v1:IITBombayX+CS101.2x+2022_T1/progress)

1. [Grading Policy](https://courses.iitbombayx.in/courses/course-v1:IITBombayX+CS101.2x+2022_T1/487375f5363a49a492641b63d5e814ec/)

1. [Course Slides](https://courses.iitbombayx.in/courses/course-v1:IITBombayX+CS101.2x+2022_T1/pdfbook/0/)

1. [Practice Programming](https://courses.iitbombayx.in/courses/course-v1:IITBombayX+CS101.2x+2022_T1/b5c2b3240ef9416ebb0ea5b2087cb4ee/)

1. [FAQs](https://courses.iitbombayx.in/courses/course-v1:IITBombayX+CS101.2x+2022_T1/f2e9c32b35ae4ef083ed716b4981ff0d/)

[Course](https://courses.iitbombayx.in/courses/course-v1:IITBombayX+CS101.2x+2022_T1/course/)  [Topic 3: Review of File Processing](https://courses.iitbombayx.in/courses/course-v1:IITBombayX+CS101.2x+2022_T1/course/#block-v1:IITBombayX+CS101.2x+2022_T1+type@chapter+block@a83378a1759347a8a8eb84e3dfcdefcb)  [Session 17: Handling Data in Text Files](https://courses.iitbombayx.in/courses/course-v1:IITBombayX+CS101.2x+2022_T1/course/#block-v1:IITBombayX+CS101.2x+2022_T1+type@sequential+block@56e137a8d36d4fc1a48c6072273b3e13)  Walkthrough

 PreviousNext

1. video Handling Data in Text Files
2. other Slides: Handling Data in Text Files
3. other Walkthrough
4. other Walkthrough

**Walkthrough**

 Bookmark this page

**Regular Expressions**

**Definition**

Regular Expressions (also acronym-ed as regex) is a very powerful tool available for string pattern matching to extract similar strings from a text.

You must be wondering what possibly can it match and how powerful can it be. Well, nothing explains better than an example.

**Example**

**Q.** Extract all valid email address from a given file. A valid email has is allowed to have only following special symbols { .(dot), \_(underscore), -(hyphen) @(at-the-rate) }

Now this will seem almost impossible without regex. One might approach in a way that checks the presence of any @ in the middle of the string, check if the string doesn’t contain any more symbols, doesn’t end with .(dot) and other possible conditions. This is not only inefficient, but also very tiresome. It involves writing many many lines of code ( and debugging).

Now if we use regex, this can be done in a single line! At this point even if you see the answer you won’t be able to make sense out of it. Let’s understand how to approach this problem.

**Syntax**

For answering the above question you need to learn some special characteristics of regex.A regular expression consists of ordinary characters (for example, letters a through z, digits) and special characters, known as *metacharacters*.

**Some Special Characters**

\*(Asterisk)

* + - * Matches preceding character zero or more times
      * Eg. za\* matches z, za, zaa, zaaaa

+(Plus)

* + - * Matches preceding character one or more times
      * Eg. za+ matches za, zaa, zaaaa (but not z)

.(Dot)

* + - * Matches any single character except ’\n'
      * Eg. z. matches za, z@, z- (but not zaa)

^(Carat), $(Dollar)

* + - * ^ Matches the start of a line ( and special meaning when used in brackets .. see below)
      * $ matches the end of the line
      * Eg. matches ^.\*$ matches the whole line

[ ](Square Brackets)

* + - * Anything within these brackets is matched for(or not matched for).
      * This can include a range too
      * ^ is used to select the complementary of things within the brackets
      * Eg. [1-4] matches 1, 2, 3 (but not 5 or 11)
      * Eg. [a-z1-4] matches all lowercase alphabets or numbers from 1-4
      * Eg[^ca] matches everything except a and c.

**Approach**

Now that you have learned many powerful symbols, let’s crack the email question.

Step 1

* For any pattern matching think of absolutely necessary things that are needed to be present.

In our case

1. We know that any email address needs to have @ symbol somewhere in the middle.
2. Also, we need it to end with an alphabet.
3. Also, it can contain only 3 of 4 symbols elsewhere.

Step 2

* Finalize the relative location of things noted in Step 1.

In our case

1. Email will start with some alphabets
2. It may encounter some symbols  along the way
3. Then eventually it will match @
4. Again encounter alphabets and symbols
5. End with alphabets

Step 3

* Now that some clarity is obtained on structure of matching string, think of corner cases and include them in your relative position order

In our case

1. We know after encountering the @ symbol, we want at-least one .(dot)
2. Also after the ending .(dot) the number of alphabets should be more than one i.e (gmail.com has 3 alphabets after dot, yahoo.co.in has 2 alphabets after the last dot)

Step 4

* Verify that all cases have been covered and build the regex step by step according to the relative order

In our case

1. Starts with alphabets ->  [a-zA-Z]+
2. May encounter some symbols -> [a-z.\_-A-Z].
3. Matches @ -> @
4. Some alphabets and symbols (starting with alphabet) -> [a-zA-Z]+[a-z.\_-A-Z].
5. Encounters final dot  -> [.]
6. At Least 2 alphabets after that -> [a-zA-Z]{2,} (Read about use of {} on the web)

Combining all these parts into one gives a template for matching the strings

**Final regex = [a-zA-Z]+ [a-z.\_-A-Z].@[a-zA-Z]+[a-z.\_-A-Z].[.] [a-zA-Z]{2,}**

That’s it. One line to parse all the valid emails from a million character text document.

**Conclusion**

The special characters mentioned is not an exhaustive set. There are a lot more advanced things one can do using regex. Regex is used everywhere in almost all languages even in scanf statements.

Note that different languages support a little bit different syntax for regex.

Read more about syntax [here](https://msdn.microsoft.com/en-us/library/ae5bf541(v=vs.100).aspx)

**Exercise**

Now, in session 16 Professor DB Phatak explained how to read table data from a text file using multiple loops.  Your task is to read the whole file in a single loop using only one sscanf statement per loop.

Hint:  Try to read the whole line in one sscanf using regex.

 PreviousNext

© All Rights Reserved

[IITBombayX Home Page](http://iitbombayx.in/)

1. [About](http://iitbombayx.in/about)
2. [Research](http://iitbombayx.in/research)
3. [FAQs](http://iitbombayx.in/faq)
4. [Contact Us](http://iitbombayx.in/contact_us)

* [MOOCs Copyright](http://iitbombayx.in/mooc-copyright-guidelines)
* [Terms of Service and Honour Code](http://iitbombayx.in/honor)
* [Authorization Document](http://iitbombayx.in/authorization)
* [Privacy Policy](http://iitbombayx.in/privacy)

[Powered by Drupal](https://www.drupal.org/)

[Creative Common License](https://creativecommons.org/)

[](http://open.edx.org/)