15-112 Fundamentals of Programming

Today

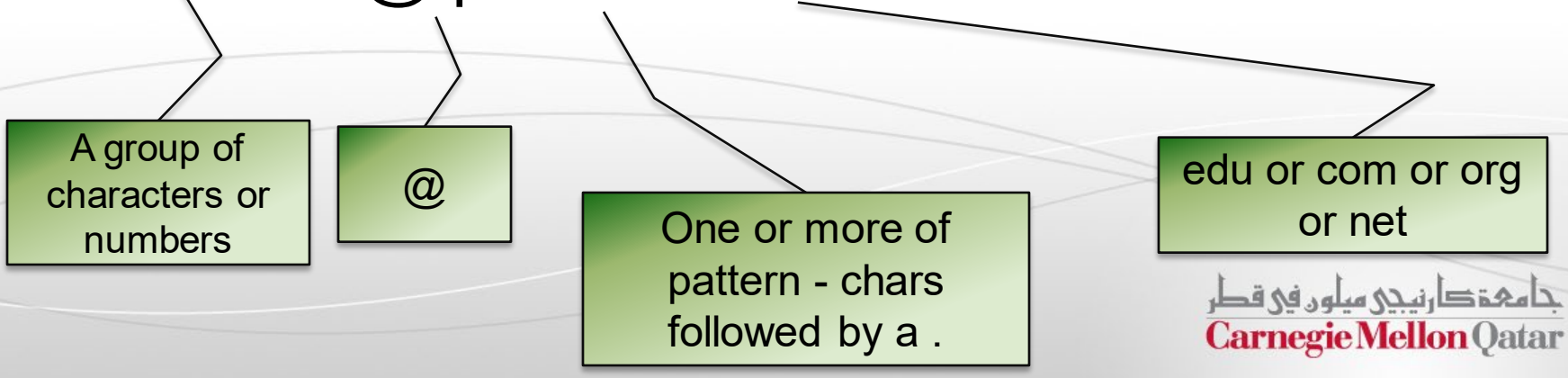
- Regular Expressions

Background

- We have written code where we were looking for specific patterns in a text
- How have we done it so far?
- Go through the string that holds the text and look for patterns
- But there is a better way of doing this

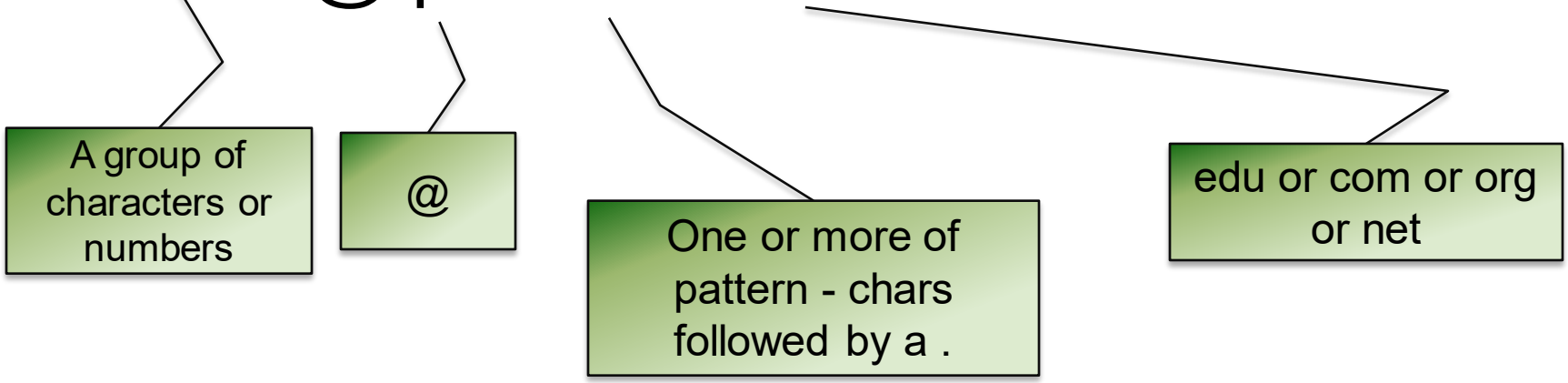
Regular Expressions

- A mechanism to specify a pattern that you are looking for
- For Example:
  - How do we check if an email address is valid
  - srazak@cmu.edu
  - srazak@qatar.cmu.edu



Regular Expressions

- srazak@qatar.cmu.edu



- We should be able to say
  - Make sure that we have a group of chars followed by a single @ followed by one or more of the sequence [chars.] followed by a "com" or "net" or "org" or "edu"

Regular Expressions

- Regular expressions allow us to specify patterns that we want to look for in a string
- import re – to use Regular expressions
- Create a pattern that you want to search
- Run the pattern on the string

A simple Example

- "\d" represents pattern that matches any digit, e.g. "1", "2", "5", etc.
- Example

```
s = "You are all number 1"
pattern = "\d"
result = re.search(pattern,s)
print (result.group())
```

- Group returns None if pattern not found

A simple Example (contd.)

- “\d” represents pattern that matches any digit, e.g. “1”, “2”, “5”, etc.
- Example

```
s = "You are all number 1"
pattern = "\d"
if re.search(pattern,s):
    print (“A number was found”)
```

A simple Example (contd.)

- "\d" represents pattern that matches any digit, e.g. "1", "2", "5", etc.
- Example

```
s = "You are all number 1"
pattern = "\d"
result = re.search(pattern,s)
print result.group()
print result.start()
print result.end()
print result.span()
```

Regular Expressions Syntax

- "\d" represents any digit, e.g. "1", "2", "9", etc.
- "\D" represents any non-digit, e.g. "a", "b", "_"
- "\w" represents any alphanumeric characters, e.g. "a", "1", "z", "0"
- "\W" represents any non-alphanumeric character, "-", "@"

An other Example

```
s = "2B! or not 2B!"
r = re.search("\d",s)
print r.group()

r = re.search("\D",s)
print r.group()

r = re.search("\w",s)
print r.group()

r = re.search("\W",s)
print r.group()
```

- “\d” represents any digit, e.g. “1”, “2”, “9”, etc.
- “\D” represents any non-digit, e.g. “a”, “b”, “-”
- “\w” represents any alphanumeric characters, e.g. “a”, “1”, “z”, “0”
- “\W” represents any non-alphanumeric character, “-”, “@”

Regular Expression Syntax

- "\s" represents whitespace, e.g. space, tab, newline
- "\S" represents non-whitespace
- Most other characters represent themselves, e.g. "a" represents "a", "-" represents "-", "1" represents "1"

Syntax Continued

- Sequence of characters represent sequence of corresponding characters
  - “\d\d” represents two consecutive digits, e.g. “12”, “33”, etc.
  - “abc” represents “abc”
  - “\w\w\s\w” represents two alphanumeric charcters, followed by space, followed by one alphanumeric character, e.g. “ab c”, “12 e” etc.

Syntax Continued

- Any of the specified characters: []
  - "[abc]" represents "a" or "b" or "c"
  - "[\dabc]" represents any digit or "a" or "b" or "c"
  - Use of "–" in "[]"
    - "[a-z]" represents any lower-case alphabet
    - "[A-Z]" represents any upper-case alphabet
    - "[a-zA-Z]" represents any alphabet
    - "[0-9]" represents any digit
    - "[e-yF-Z0-9]" represents e to y or F to Z or 0 to 9

Syntax Continued

- None of the specified characters: [^]
  - "[^abc]" represents any character except "a" or "b" or "c"
  - "[^\dabc]" represents any character except any digit or "a" or "b" or "c"
  - Use of "-" in "[^]":
    - "[^a-z]" represents any character except any lower-case alphabet
    - "[^A-Z]" represents any character except any upper-case alphabet
    - "[^a-zA-Z]" represents any character except any alphabet
    - "[^0-9]" represents any character except any digit
    - "[^e-yF-Z0-9]" represents any character except e to y or F to Z or 0 to 9

Syntax Continued

- Metacharacter: “.”
  - Matches any single character except newline.
  - “a.b” matches “a” followed by anyone character followed by “b”

Syntax Continued

- Metacharacter: “*”
  - “a*” represents zero or more “a”, e.g. “”, “a”, “aa”, “aaa”
  - “b*” represents zero or more “b”, e.g. “”, “b”, “bb”, “bbb”
  - “\d*” represents zero or more digits, e.g. “”, “1”, “2”, “2344”
  - “\D*” represents zero or more non-digits
  - “\w*” represents zero or more alphanumeric characters
  - “\s*” represents zero or more whitespaces
  - “[A-Z]*” represents zero or more upper-case alphabets

Syntax Continued

- Metacharacter: “+”
  - “a+” represents one or more “a”, e.g. “a”, “aa”, “aaa”
  - “b+” represents one or more “b”, e.g. “b”, “bb”, “bbb”
  - “\d+” represents one or more digits, e.g. “1”, “2”, “23”, “23442”, etc.
  - “\D+” represents one or more non-digits
  - “\w+” represents one or more alphanumeric characters
  - “\s+” represents one or more whitespaces
  - “[A-Z]+” represents one or more upper-case alphabets

Alternation

- You can search of alternate regexes by using “|” operator

```
import re
reg = "cat|dog"
s = "the cat ate the mouse"
s2 = "the dog ate the cat"
re.search(reg,s).group()
re.search(reg,s2).group()
```

Groups

- You can specify groups of string matches by using ( )
  - Parentheses group the regex between them. They capture the text matched by the regex inside them into a numbered.

Now try it all out!

- Most phone numbers in us are written in the format:

  (xnn)nnn-nnnn

where n can be any digit and x is any non-zero digit

- Write a function that takes as input a string and returns True if the string represents a valid phone number
- Write a python program that reads a phone number, checks if the number is valid and keeps asking the user for a phone number until a valid format is entered.

Regular Expressions Cheat Sheet

.	Any character except newline
a	The character a
ab	The string ab
a*	0 or more a's
\	Escapes special character
*	0 or more
+	1 or more
?	0 or 1
{2}	Exactly 2
{2, 5}	Between 2 and 5
{2,}	2 or more
(,5}	Up to 5

- [ab-d] One character of: a, b, c, d
- [^ab-d] One character except: a, b, c, d
- \d One digit
- \D One non-digit
- \s One whitespace
- \S One non-whitespace
- \w represents any alphanumeric characters, e.g. “a”, “1”, “z”, “0”
- \W represents any non-alphanumeric character, “-”, “@”