Patterns and regular expressions (regex)

1. Unstructured text, and a specific need.

Unstructured text	**Need**
"Jean Valjean lives in Calais in a house with 4 rooms. He has 2 close friends whose phone numbers are 541 721 7277 and 541-259-9531. Jean's phone number is (653)592 6799. Living across from Jean's house is a woman named Dorothy, whose phone number is 792.648.2152. Dorothy has a daughter whose phone number is (555) 557-5632."	*I want to pull out all of the phone numbers only.*
"Olympic athletes list from Vancouver 2010 Petter Northug Meng Wang Sidney Crosby Steven Holcomb"	*I want to pull out people who have a total of 8 letters in their full name only.*

2. Solution: regular expressions

! Regular expressions: An easy-to-use notation (formula or recipe) to describe any pattern in text, including any combination of letters, numbers, symbols, and spaces.

Regex	Matches any string that
`hello`	contains {hello}
`gray|grey`	contains {gray, grey}
`gr(a|e)y`	contains {gray, grey}
`gr[ae]y`	contains {gray, grey}
`b[aeiou]bble`	contains {babble, bebble, bibble, bobble, bubble}
`z{3,6}`	contains {zzz, zzzz, zzzzz, zzzzzz}
`z{3,}`	contains {zzz, zzzz, zzzzz, ...}

More examples at https://cs.lmu.edu/~ray/notes/regex/

3. Phone numbers to pattern

Area code (any number excepting 0 or 1, followed by any two numbers)	**3 digits**	**4 digits**

1. Square brackets around something means options of "one of these."
[0123456789] means "a digit between 0 and 9."

2. A dash means "all of the characters in between in ASCII."
[0-9] means "a digit between 0 and 9."

3. Curly brackets after a square bracket tells us "how much" of the square brackets we need.
[0-9]{1} means the same thing as [0-9].
[0-9]{2} means "two digits between 0 and 9, such as 12 or 13 or 29 or 93."

Cheat sheets for rules available on Google. ☺

3. Phone numbers to pattern

Area code (any number excepting 0 or 1, followed by any two numbers)	3 digits	4 digits
[2-9][0-9][0-9]	[0-9][0-9][0-9]	[0-9][0-9][0-9][0-9]
([2-9])([0-9]{2})	[0-9]{3}	[0-9]{4}

Simplify…

4. More rules (Sorry! No need to memorize!)

4. Two numbers in {} means “at least” and “at most.”
[0-9]{2,5} means *at least* 2 digits and *at most* 5 digits.

5. Variations on rule number (4):
[0-9]{,5} means *up to and including* 5 digits.
[0-9]{3,} means *at least 3 digits*.

Cheat sheets for rules available on Google. ☺

5. Match the left to the right.

[a-z]{4}	Full names with each component of up to but not including seven letters, and two or three components.
[A-Z]{4}	Four letter upper case words
[A-Z][a-z]{3}	Four letter words with only the first letter capitalized
([A-Z][a-z]{,6}){2,3}	Four letter words

Cheat sheets for rules available on Google. ☺

6. Finally solving the phone numbers.

Cheat sheets for rules available on Google. ☺

7. Last thoughts: language challenges.

1. Working with texts that aren't ASCII-compatible. (What is ASCII, anyways?)

   [1] See the Library of Congress's rules for dealing with Persian characters using transliteration:
   https://www.loc.gov/catdir/cpso/romanization/persian.pdf

   [2] See rules for transliteration of Elder Futhark runes.
   https://www.timenomads.com/elder-futhark-alphabet-cheat-sheet/

2. Complex parsing is *complex*, and hard to debug when you're a human. Probably should use something else (website) to write your regex for you.

RUNE	NAME	TRANSLATION
ᚠ	Fehu	F
ᚢ	Uruz	U
ᚦ	Thurisaz	Th
ᚨ	Ansuz	A
ᚱ	Raido	R
ᚲ	Kaunaz	C/K
ᚷ	Gebo	G
ᚹ	Wunjo	W
ᚺ	Hagalaz	H
ᚾ	Nauthiz	N
ᛁ	Isa	I
ᛃ	Jera	J/Y

7. Last thoughts: Persian example.

A linguist's nightmare… or dream task. I'm not sure.

to match all words in a Persian text that contain the letter گ, which is a unique Persian character not found in the Latin script

```
import re

# Persian text with " "گcharacter
persian_text = ""می‌گوید که گرفتاری‌های زیادی داشته است""

 #Transliterate Persian text to Latin script
latin_text = persian_text.translate
      (str.maketrans(".:,0123456789" ,"۰۱۲۳۴۵۶۷۸۹Aabbcdefghijklmnopqrstuvwxyz؛،آاپبتثجچحخددذرزژسشصضطظعغفقکگلمنوهی"))

# Use regular expression to match words containing "g"
matches = re.findall(r'\b\w*g\w*\b', latin_text)

# Print matches
print(matches)
```