

## OWE 3a: Python II



**Geavanceerde concepten  
in Python en  
programmeren voor  
bio-informatica  
toepassingen**

# Opzet OWE 3a

Les	Onderwerp	Theorie	Opgaven
1	<ul style="list-style-type: none"> <li>Review Python I</li> <li>Pseudocode</li> <li>Flowcharts</li> <li>Documenteren en Testen</li> </ul>	H1..8 SowP*	Afvinkopdracht 1
2	<ul style="list-style-type: none"> <li>Graphs</li> <li>Strings</li> </ul>	<a href="#">Matplotlib tutorial</a> H9 More about Strings	Afvinkopdracht 2
3	<ul style="list-style-type: none"> <li>Datastructuren:</li> <li>Dictionaries</li> <li>Sets</li> </ul>	H10 Dictionaries and Sets	Afvinkopdracht 3
4	<ul style="list-style-type: none"> <li>Text and Language Processing</li> <li><a href="#">Regular Expressions</a></li> </ul>	<a href="#">H7 DiP</a> **	Afvinkopdracht 4
5	<ul style="list-style-type: none"> <li>Object-Oriented Programming</li> </ul>	H11 Classes and Object-Oriented Programming H12 Inheritance	Afvinkopdracht 5
6	<ul style="list-style-type: none"> <li>Recursion</li> </ul>	H13 Recursion	Afvinkopdracht 6
7	<ul style="list-style-type: none"> <li>GUI Programming</li> </ul>	H14 GUI Programming	Voorbeeld thematoets

\*SowP: Starting out with Python

\*\*DiP Dive into Python

# Agenda

- **Strings**
- **Regular Expressions**

## Waarom Strings?

- In de bio-informatica maken we erg veel gebruik van (tekst)zoekopdrachten
- Deze zoekopdrachten zijn gericht op teksten in wetenschappelijke artikelen
- Of zoekopdrachten op patronen in DNA/RNA en eiwitten

## Textmining

- Er is een heel vakgebied wat zich bezig houdt met zoekopdrachten in tekst
- Dit vakgebied – textmining – onderzoekt bijvoorbeeld of het mogelijk is nieuwe verbanden te vinden door teksten te analyseren

# Agenda

- **Strings**
- **Regular Expressions**

# Strings

- **Strings hebben we al veel gebruikt**
- **Onder andere String manipulatie met de slicing methode:**
- **“Hello World!”[2:4]**

# Strings

- **Strings hebben zelf methodes, functies die we met de puntnotatie kunnen aanroepen:**
- **“Hello World!”.upper()**



## String functies

- **str.find(substr)**
  - Zoeken naar een substring
- **str.replace(old, new)**
  - Vervanging van een substring
- **str.split(delim)**
  - Splitsen op basis van een teken
- **str.join(seq)**
  - Plakken van strings
- **str.strip( )**
  - Verwijderen van spaties

## String functies

- **str.rstrip( )**
  - Verwijderen van spaties rechts
- **str.upper( )**
  - Omzetten naar hoofdletters
- **str.isupper( )**
  - Toetsen of een string uit hoofdletters bestaat
- **str.isdigit( )**
  - Toetsen of het cijfers zijn

## Samenvatting

- **Strings zijn geschikt om korte teksten in op te slaan**
- **Strings hebben zelf methodes om met Strings te werken**

## Waarom programmeren?

- <http://tedxtalks.ted.com/video/You-Should-Learn-to-Program-Chr>
- <http://www.tedxsmu.org/talks/christian-genco-you-should-learn-to-program-tedxsmu-salon-2012>

# Agenda

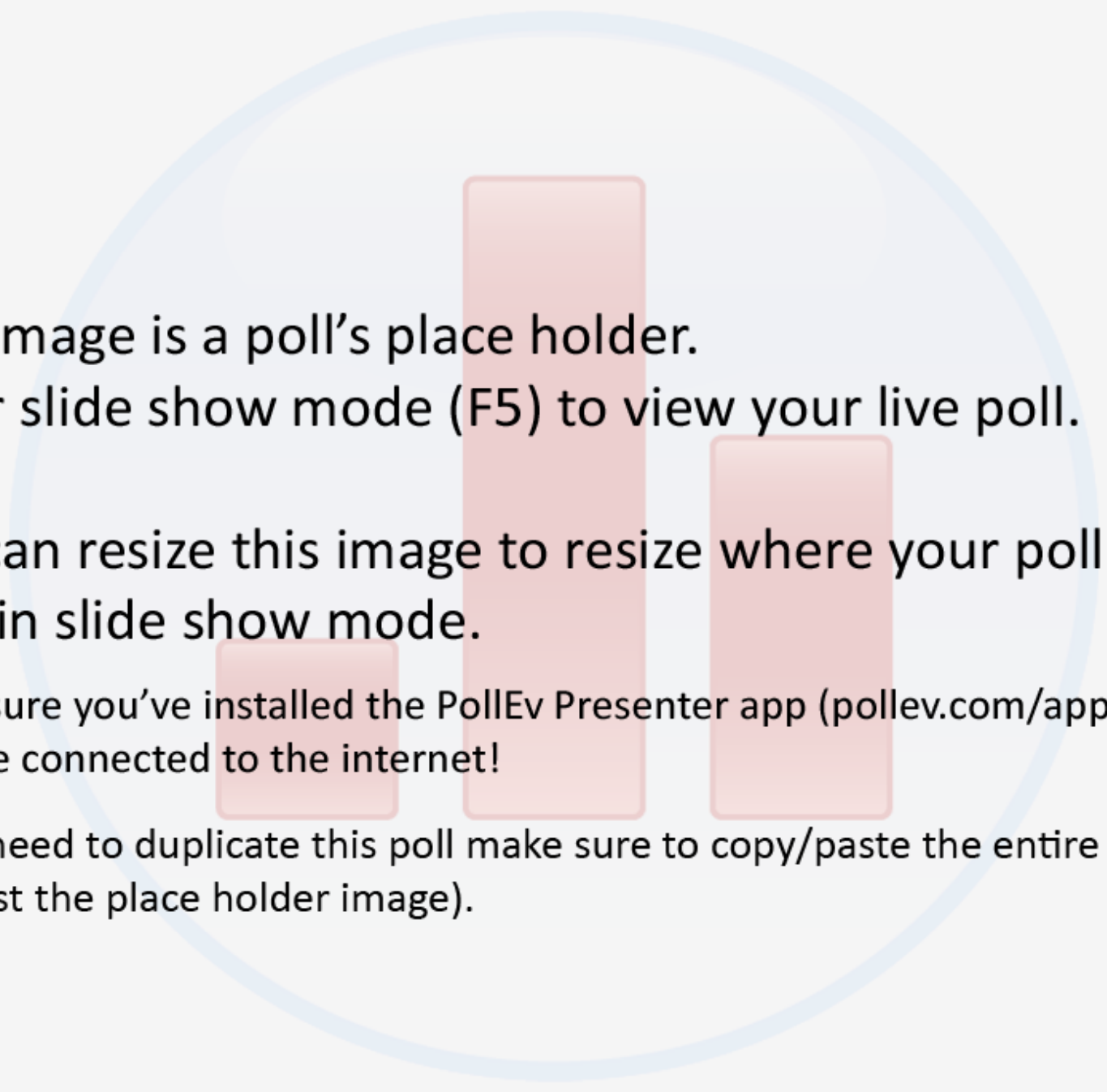
- **Strings**
- **Regular Expressions**
  - Wat zijn regular expressions?
  - Regels voor regular expressions
  - Toepassingen

## Regular Expressions

- Het zoeken van substrings in strings is mogelijk
- Bijvoorbeeld
- “Hello World!”.find(“o”)
- Maar wat als je zowel Hello World! als Hallo Wereld! zoekt?

## Regular Expressions

- Regular Expressions geven je de mogelijkheid om complexe tekstpatronen te zoeken
- In de bio-informatica zijn we heel erg vaak op zoek naar complexe tekstpatronen



This image is a poll's place holder.  
Enter slide show mode (F5) to view your live poll.

You can resize this image to resize where your poll will  
load in slide show mode.

Make sure you've installed the PollEv Presenter app ([pollev.com/app](http://pollev.com/app))  
and are connected to the internet!

If you need to duplicate this poll make sure to copy/paste the entire slide  
(not just the place holder image).



# Hello World!

- Om te zoeken op Hello World! En Hallo Wereld kunnen we een regular expression gebruiken
- H[ea]llo W[oe]re?ld

# Online Regular Expressions

/ H[ea]llo W[eo]re?ld!

## Your test string

```
Dag Wereld!  
Hello World!  
Hi World!  
Hallo Wereld!
```

## Regular expressions

- Vaak wordt “regular expression” afgekort tot “regexp”, “regex”, of “re”
- Regular Expressions vind je terug in databases en op bijvoorbeeld Linux systemen

## Basis

- De basis van een regular expression is het interpreteren van speciale karakters

## Waar?

- **Regular expressions zijn te gebruiken in iedere programmeertaal**
  - Python
  - Java
  - SQL
- **Op ieder system**
  - Linux (UNIX)
  - Windows
  - Mac
- **In heel veel applicaties**

## Speciale karakters (1/2)

- Karakters in regular expressions

Karakter	Betekenis
.	Precies een willekeurig teken
*	0, 1 of meer van het <b>voorgaande</b> teken
+	1 of meer van het <b>voorgaande</b> teken
?	Exact 0 of 1 van het <b>voorgaande</b> teken
^	Geeft aan dat het er mee moet beginnen
\$	Geeft aan dat het er mee moet eindigen

## Speciale karakters (2/2)

- **Karakters in regular expressions**

Karakter	Betekenis
[ ]	Aantal tekens Een ^ aan het begin van [^ ] geeft de inverse
( )	Een groep
\	Escape
{m,n}	Minimaal m en maximaal n van het voorgaande teken

## Toepassingen van Regular Expressions

- Zoeken van een postcode in een tekst
- Zoeken van een voorkomen op Linux
- Vervangen van tekst in Notepad++



# Nederlands postcode

Your regular expression in: Python ▼

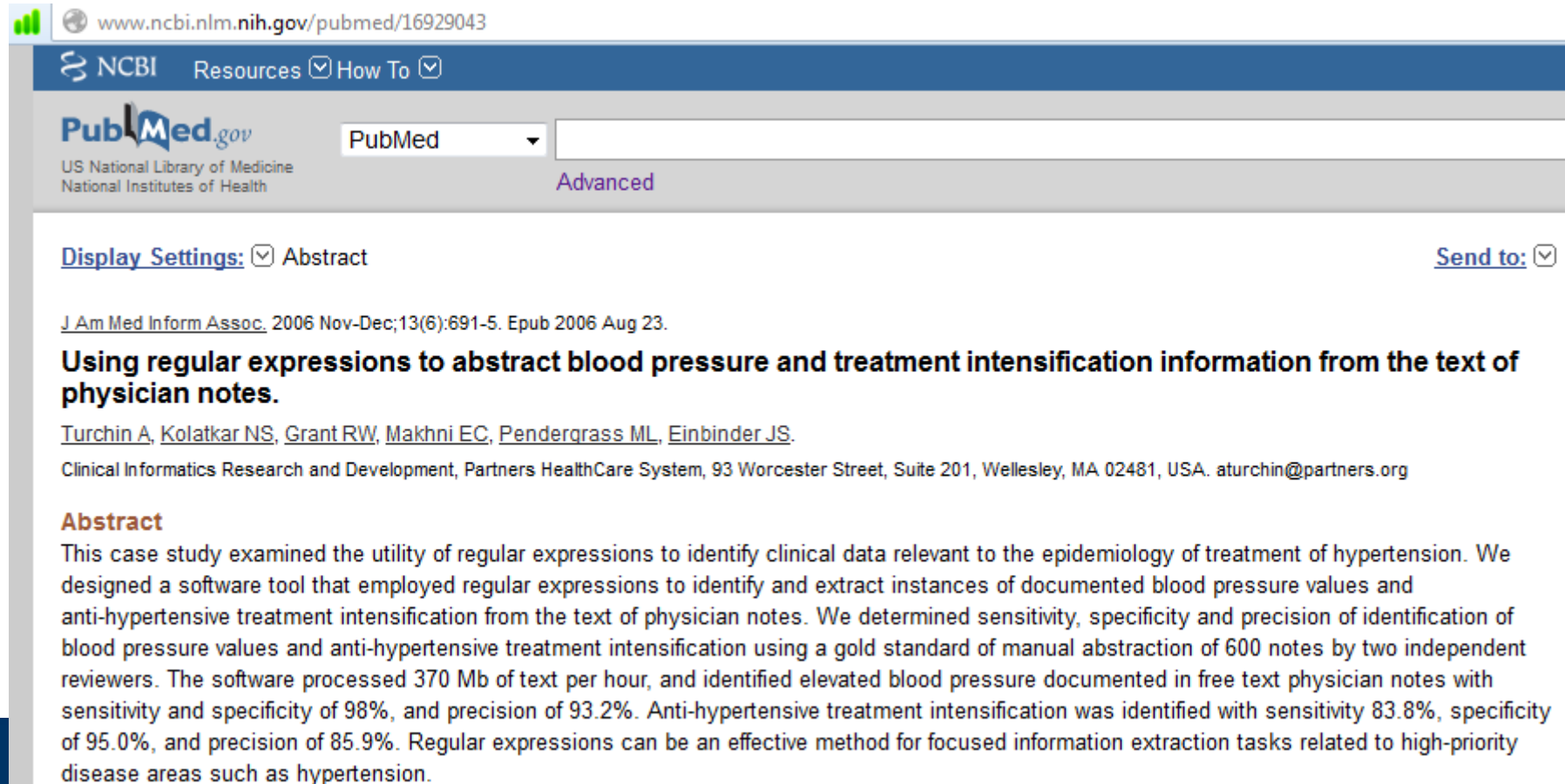
" `[1-9]\d{3}\s?[a-zA-Z]{2}` " g ?

Your test string

[\[-\]](#)

```
1234 AA
0000 AB
2121 12
2176 BB
1120 11
A219 BB
6543 AA
```

# Beschrijvingen doorzoeken



www.ncbi.nlm.nih.gov/pubmed/16929043

NCBI Resources ☒ How To ☒

PubMed.gov

US National Library of Medicine  
National Institutes of Health

[Display Settings:](#) ☒ Abstract [Send to:](#) ☐

*J Am Med Inform Assoc.* 2006 Nov-Dec;13(6):691-5. Epub 2006 Aug 23.

**Using regular expressions to abstract blood pressure and treatment intensification information from the text of physician notes.**

[Turchin A](#), [Kolatkar NS](#), [Grant RW](#), [Makhni EC](#), [Pendergrass ML](#), [Einbinder JS](#).

Clinical Informatics Research and Development, Partners HealthCare System, 93 Worcester Street, Suite 201, Wellesley, MA 02481, USA. [aturchin@partners.org](mailto:aturchin@partners.org)

**Abstract**

This case study examined the utility of regular expressions to identify clinical data relevant to the epidemiology of treatment of hypertension. We designed a software tool that employed regular expressions to identify and extract instances of documented blood pressure values and anti-hypertensive treatment intensification from the text of physician notes. We determined sensitivity, specificity and precision of identification of blood pressure values and anti-hypertensive treatment intensification using a gold standard of manual abstraction of 600 notes by two independent reviewers. The software processed 370 Mb of text per hour, and identified elevated blood pressure documented in free text physician notes with sensitivity and specificity of 98%, and precision of 93.2%. Anti-hypertensive treatment intensification was identified with sensitivity 83.8%, specificity of 95.0%, and precision of 85.9%. Regular expressions can be an effective method for focused information extraction tasks related to high-priority disease areas such as hypertension.

## Voorbeelden

Regular Expression	Voorbeeld hits	Geen hit op
H[ea]llo	Hello en Hallo	Hollo
TATA[AT][AT]	TATAAT, TATATT, TATAAA	TATAGC
....huis (vier puntjes)	Woonhuis, koophuis, leeghuis	Ziekenhuis
.*huis	Ziekenhuis	Ziekenhuisje
.*huis.*	huismus, Verhuis-service	Huizen
.+huis	Thuis	huis

## Voorbeelden

Regular Expression	Voorbeeld hits	Geen hit op
(ATG GTA)	ATG, GTA	ATA
TATA[^AT]	TATAC, TATAG	TATAA
A{2,4}G	AAAAG, AAG	AG
.{2,4}G	CCGCC	AGAA

## Voorbeelden

Regular Expression	Voorbeeld hits	Geen hit op
\?	?	A
\\	\	A
[0-9]{2,3}	218, 29, 42, 263	1, A
[a-z]*	hallo	Hallo

# Regular Expressions in Python



## Overzicht re functies

Functie	Functionaliteit
search	Zoeken naar een overeenkomst
replace	Vervangen van een
split	Splitsen op reguliere expressie

## Search

- **`re.search(pattern, string, flags=0)`**
  - Pattern is het te zoeken patroon: de reguliere expressie
  - String is de tekst waarin je het patroon wilt vinden
  - Flags heeft een default 0, geef je meestal dus niet mee. Hiermee kun je opties meegeven.



## Voorbeeld van search

- `import re`
- `line = "AGGGGGCCACATTAATGATGGA  
GTATAGGAGTA"`
- `matchObj = re.search( r'.*ATG\.*', line)`
- `print (matchObj)`
- Hiermee zien we of er een hit is of niet
  - Bij geen hit → None
  - Bij wel een hit object referentie  
<\_sre.SRE\_Match object at 0xb7408db0>

## Voorbeeld

```
import re
lines = ["Bio-informatica studenten van"
, "de Hogeschool van Arnhem en Nijmegen"
, "blijken zeer gewilde stagiaires te zijn"
, "Dat komt onder andere doordat zij zeer"
, "bedreven zijn in het schrijven van"
, "Regular Expressions" ]

for line in lines:
    if re.search('[A-H]+', line):
        print (line)

>>>
Bio-informatica studenten van
de Hogeschool van Arnhem en Nijmegen
Dat komt onder andere doordat zij zeer
Regular Expressions
>>> |
```

## Replace

- `re.sub(pattern, repl, string, max=0)`

## Splitting

- `re.split(pattern, string, maxsplit=0, flags=0)`

```
import re
sequence = "ATAGGAGATGAGGAGCCAGTAGAGTATGAG" #Sequentie
pattern = '([ATGC]{3})' #Regular expression
print (re.split (pattern, sequence))
```

## Groepen

- Met ronde haken geef je aan dat er een groep is die een match heeft
- Deze groepen kun je er uit halen door `match.group()` uit te vragen

```
def reverse_columns(line):  
    match = re.search(r'^\s*(\d+)\s+(\d+)\s*$', line)  
    if not match:  
        return line  
    return match.group(2) + ' ' + match.group(1)
```

# Regular Expressions in de biologische sequenties



## Splitsen in codons

- `import re`
- `sequence =`  
`“AGAGATGAGGAGGCCAGT”*100`
- `pattern = "([ATGC]{3})“`
- `print (re.split (pattern, sequence))`

# Sequenties voorspellen structuur en functie van genen

- Een van de bekendste genen is p53
- Dit gen kent een geconserveerde regio:  
**MCNSSCMGGMNRR**
- Als deze substring in je eiwit voorkomt is het aannemelijk dat het p53 is of een verwant eiwit



## Zoeken van een string

- **site in sequence – weten we of de substring erin voorkomt**
- **sequence.find(site) – retourneert de positie van de substring**
- **sequence.count(site) – retourneert het aantal voorkomens van de substring**

## Zit p53 in de sequentie?

```
>>> p53 = "MCNSSCMGGMNRR"  
>>> protein = "SEFTTVLYNFMCNSSCMGGMNRRPILTIIS"  
>>> protein.find(p53)  
10  
>>> protein[10:10+len(p53)]  
'MCNSSCMGGMNRR'  
>>>  
,
```

## P53 heeft echter varianten

- Na verder onderzoek ontdek je dat p53 in meerder varianten voorkomt
- Hoe beschrijf je dit patroon?

MCNSSC**M**GGMNRR

of

MCNSSC**V**GGMNRR

# PROSITE

- PROSITE is een database met eiwit patronen
- <http://au.expasy.org/prosite/>
- P53:  
<http://prosite.expasy.org/PDOC00301>
- Signaturen:  
<http://prosite.expasy.org/cgi-bin/prosite/prosite-search-ful?SEARCH=signature>

## Patterns op expasy

- <http://prosite.expasy.org/PDOC00032>
- Antennapedia: pattern

## Antennapedia

- Wild type *Drosophilla melanogaster* links met antennapedia rechts  
(Bron: [http://en.wikipedia.org/wiki/Drosophila\\_melanogaster](http://en.wikipedia.org/wiki/Drosophila_melanogaster) 14-feb-2014)



## Zoek ANTENNAPEDIA

Zie je het patroon [LIVMFE][FY]PWM[KRQTA]?

MDPDCFAMSS	YQFVNSLASC	YPQQMNPQQN	HPGAGNSSAG	GSGGGAGGSG	GVVPSGGTNG
GQGSAGAATP	GANDYFPAAA	AYTPNLYPNT	PQPTTPIRRL	ADREIRIWWT	TRSCSRSDCS
CSSSSNSNSS	NMPMQRQSCC	QQQQQLAQQQ	HPQQQQQQQQ	ANISCKYAND	PVTPGGSGGG
GVSGSNNNNN	SANSNNNNSQ	SLASPQDLST	RDISPKLSPS	SVVESVARSL	NKGVLGGS LA
AAAAAAGLNN	NHSGSGVSGG	PGNVNVPMHS	PGGGDSDSES	DSGNEAGSSQ	NSGNGKKNPP
QIYPWMKRVH	LGTSTVNANG	ETKRQRTSYT	RYQTLELEKE	FHFNRYLTRR	RRIEIAHALC
LTERQIKIWF	QNRMRKWKKE	HKMASMNIVP	YHMGPYGHPY	HQFDIHPSQF	AHL SA

Daarom hebben we computers!

## Regular Expressions in het kort





# Regular Expressions

**bol.com**

Cadeaubon Verkoop Zakelijk bestellen Actiefolder Fotoservice

regular expressions

Alle boeken ▼

ZOEKEN

[BOEKEN](#)
[MUZIEK](#)
[DVD](#)
[GAMES](#)
[SPEELGOED](#)
[BABY](#)
[KOKEN & TAFELN](#)
[MOOI & GEZOND](#)
[COMPUTER](#)
[ELEKTRONICA](#)

 GRATIS RETOURNEREN
  30 DAGEN BEDENKTIJD
  DAG & NACHT BEREIKBAAR
 **Gratis verzending vanaf 20 euro**
 VEILIG BETALEN

[Home](#) > [Boeken](#) > Alle boeken: regular expressions

CATEGORIEËN

[Computer \(20\)](#)  
[Geschiedenis en politiek \(1\)](#)

TAAL

[Engelse boeken \(20\)](#)  
[Duitse boeken \(2\)](#)

PRIJS

[Tot € 10 \(2\)](#)  
[Tot € 20 \(6\)](#)  
[Tot € 30 \(13\)](#)  
[Meer](#)

22 resultaten gevonden voor 'regular expressions' in Alle boe

Sorteer op: Relevantie ▼

Toon res



**Regular Expressions** | Jan Goyvaerts  
 The Complete Tutorial

Engels - Paperback | 2006

This thorough tutorial teaches you the complete regular expression syntax. Detailed examples and descriptions of how regular... [Meer](#)

**bol.com**  
 GEEN AFBEELDING BESCHIKBAAR

**Regular Expressions**

Computing, String (computer science), FormalLanguage, Compiler-compiler, Specification (technical standard), Text Editor, Programming Language, Pattern, Wildcard Character

## Voorbeeld opgave

- Gegeven is '[LG].{4,5}[FYW]' als regular expression. Dit betreft een patroon dat voorkomt in een humaan proteïne. Op welke peptide zal dit patroon een hit leveren?
- a. LALALCY
- b. LFYFLAL
- c. LALALCG
- d. LFYYLAL

## Voorbeeld opgave

- Gegeven is '[^L].{4,5}[FYW]' als regular expression. Dit betreft een patroon dat voorkomt in een humaan proteïne. Op welke peptide zal dit patroon een hit leveren?
- a. LLAALALCY
- b. LAFYFLALW
- c. LAALALCG
- d. LAFYYLAL

## Voorbeeld opgave

- Gegeven is '(ACA|CAF){4}[^C]{2}' als regular expression. Dit betreft een patroon dat voorkomt in een humaan proteïne. Op welke peptide zal dit patroon een hit leveren?
- a. CACAALALCY
- b. CAFYFLALWA
- c. CACACACCGC
- d. ACAFYYLALCA

## Hoeveel zinnen?

```
import re
```

```
lines = ["Bio-informatica studenten van"  
, "de Hogeschool van Arnhem en Nijmegen"  
, "blijken zeer gewilde stagiaires te zijn"  
, "Dat komt onder andere doordat zij zeer"  
, "bedreven zijn in het schrijven van"  
, "Regular Expressions" ]
```

```
for line in lines:
```

```
    if re.search('[x]+', line):  
        print (line)
```

## Hoeveel zinnen?

```
import re
```

```
lines = ["Bio-informatica studenten van"  
, "de Hogeschool van Arnhem en Nijmegen"  
, "blijken zeer gewilde stagiaires te zijn"  
, "Dat komt onder andere doordat zij zeer"  
, "bedreven zijn in het schrijven van"  
, "Regular Expressions" ]
```

```
for line in lines:
```

```
    if re.search('[C-H]+', line):  
        print (line)
```

WHENEVER I LEARN A  
NEW SKILL I CONCOCT  
ELABORATE FANTASY  
SCENARIOS WHERE IT  
LETS ME SAVE THE DAY.

OH NO! THE KILLER  
MUST HAVE FOLLOWED  
HER ON VACATION!



BUT TO FIND THEM WE'D HAVE TO SEARCH  
THROUGH 200 MB OF EMAILS LOOKING FOR  
SOMETHING FORMATTED LIKE AN ADDRESS!



IT'S HOPELESS!

EVERYBODY STAND BACK.



I KNOW REGULAR  
EXPRESSIONS.



## Samengevat

- **Regular Expressions zijn zeer krachtig**
- **Ze voorkomen dat je onnodig complexe programmeeropdrachten geeft**
- **Regular Expressions zijn in veel programma's te gebruiken en in vrijwel alle programmeertalen en databases**



**In deze uitgave is géén auteursrechtelijk beschermd werk  
opgenomen**

**HOGESCHOOL VAN ARNHEM EN  
NIJMEGEN**