9. előadás

Reguláris kifejezések

Programozás (2) előadás

2022. November 14.

Halász Gábor

Debreceni Egyetem

Általános tudnivalók

Ajánlott irodalom:

- Nyékyné G. Judit (szerk): Programozási nyelvek, Kiskapu, 2003.
- Juhász, István: Magas szintű programozási nyelvek 2, elektronikus egyetemi jegyzet, 2009
- Tarczali, Tünde: UML diagramok a gyakorlatban, Typotex Kiadó, 2011.
- Angster, Erzsébet: Objektumorientált tervezés és programozás: JAVA, 4KÖR Bt., 2002, ISBN: 9632165136
- Bird, S., Klein, E., Loper, E.: Natural Language Processing with Python, O'Reilly Media, 2009

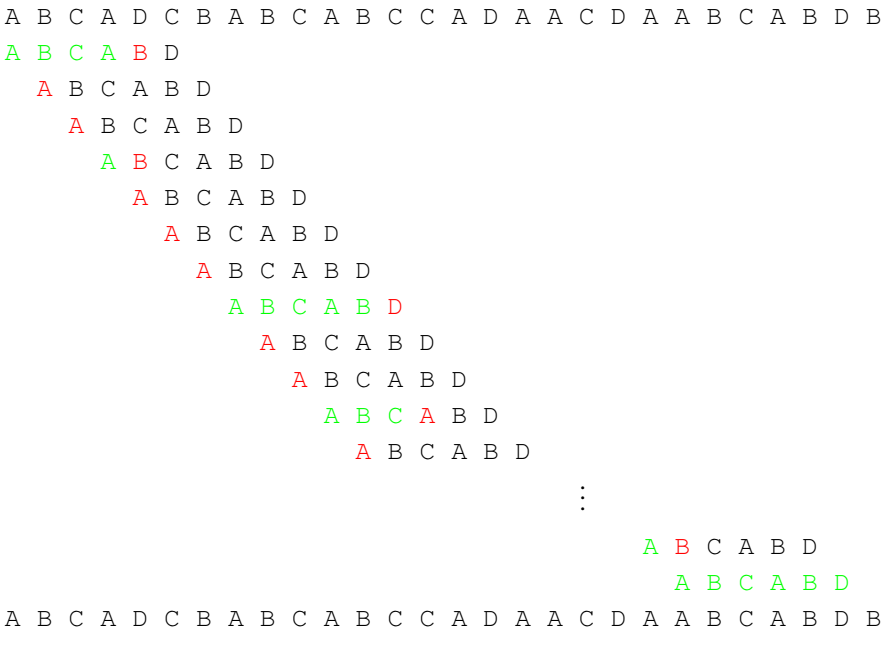
Félév teljesítésének feltételei: jelenlét + 2 gyakorlati + 1 elméleti ZH

Érdemjegy: $1 < 60\% \leq 2 < 70\% \leq 3 < 80\% \leq 4 < 90\% \leq 5$

További részletek: `https://elearning.unideb.hu/`

Mintaillesztés – egyszerű minták

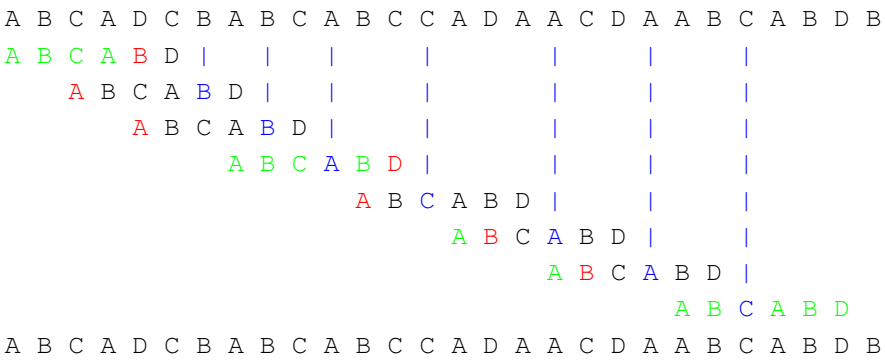
A mezítlábas (brute force) algoritmus



A Knuth–Morris–Pratt-algoritmus

```
A B C A D C B A B C A B C C A D A A C D A A B C A B D B
A B C A B D
      A B C A B D
        A B C A B D
          A B C A B D
            A B C A B D
              A B C A B D
                    A B C A B D
                          A B C A B D
                                A B C A B D
                                  A B C A B D
                                        ⋮
                                            A B C A B D
                                              A B C A B D
A B C A D C B A B C A B C C A D A A C D A A B C A B D B
```

Boyer-Moore mintaillesztés



Rabin–Karp mintaillesztés

d=10 ; q=31 (31, 62, 93, 124, 155, 186, 217, 248, 279)

P="123124" ; m=6 ; A="123143212312331411341123124 2" ; n=28

h=2=d^m mod q (1, 10, 100−93=7, 70−62=8, 80−62=18, 180−155=25, 250−248=2)

p=23 (0, 1, 12, 123−93=30, 301−279=22, 222−217=5, 54−31=23)

a=11 (0, 1, 12, 123−93=30, 301−279=22, 224−217=7, 73−62=11)

1 2 3 1 4 3 2 1 2 3 1 2 3 3 1 4 1 1 3 4 1 1 2 3 1 2 4 2

j=0 | | a=11

j=1 | | a=17=(112−2⋆1)−93

j=2 | | a=12=(171−2⋆2)−155

j=3 |1 2 1 2 3 4| a=23=(122−2⋆3)−93=p

j=4 | | a=14=(233−2⋆1)−217

j=5 | | a=9=(141−2⋆1)−124

j=6 | | a=24=(92−2⋆3)−62

j=7 | | a=22=(243−2⋆2)−217

j=8 | | a=4=(223−2⋆1)−217

1 2 3 1 4 3 2 1 2 3 1 2 3 3 1 4 1 1 3 4 1 1 2 3 1 2 4 2

Rabin–Karp mintaillesztés

d=10 ; q=31 (31, 62, 93, 124, 155, 186, 217, 248, 279)

P="123124" ; m=6 ; A="123143212312331411341123124 2" ; n=28

h=2=d^m mod q (1, 10, 100−93=7, 70−62=8, 80−62=18, 180−155=25, 250−248=2)

p=23 (0, 1, 12, 123−93=30, 301−279=22, 222−217=5, 54−31=23)

a=11 (0, 1, 12, 123−93=30, 301−279=22, 224−217=7, 73−62=11)

```
     1 2 3 1 4 3 2 1 2 3 1 2 3 3 1 4 1 1 3 4 1 1 2 3 1 2 4 2
j=0  |           |  a=11
j=1    |           |  a=17=(112−2⋆1)−93
j=2      |           |  a=12=(171−2⋆2)−155
j=3        |1 2 1 2 3 4|  a=23=(122−2⋆3)−93=p
j=4          |           |  a=14=(233−2⋆1)−217
j=5            |           |  a=9=(141−2⋆1)−124
j=6              |           |  a=24=(92−2⋆3)−62
j=7                |           |  a=22=(243−2⋆2)−217
j=8                  |           |  a=4=(223−2⋆1)−217
     1 2 3 1 4 3 2 1 2 3 1 2 3 3 1 4 1 1 3 4 1 1 2 3 1 2 4 2
```

Rabin–Karp mintaillesztés

d=10 ; q=31 (31, 62, 93, 124, 155, 186, 217, 248, 279)

P="123124" ; m=6 ; A="123143212312331411341123124 2" ; n=28

h=2=d^m mod q (1, 10, 100−93=7, 70−62=8, 80−62=18, 180−155=25, 250−248=2)

p=23 (0, 1, 12, 123−93=30, 301−279=22, 222−217=5, 54−31=23)

a=11 (0, 1, 12, 123−93=30, 301−279=22, 224−217=7, 73−62=11)

```
     1 2 3 1 4 3 2 1 2 3 1 2 3 3 1 4 1 1 3 4 1 1 2 3 1 2 4 2
j=0 |           |  a=11
j=1   |           |  a=17=(112−2*1)−93
j=2     |           |  a=12=(171−2*2)−155
j=3       |1 2 1 2 3 4|  a=23=(122−2*3)−93=p
j=4         |           |  a=14=(233−2*1)−217
j=5           |           |  a=9=(141−2*1)−124
j=6             |           |  a=24=(92−2*3)−62
j=7               |           |  a=22=(243−2*2)−217
j=8                 |           |  a=4=(223−2*1)−217
     1 2 3 1 4 3 2 1 2 3 1 2 3 3 1 4 1 1 3 4 1 1 2 3 1 2 4 2
```

Rabin–Karp mintaillesztés

d=10 ; q=31 (31, 62, 93, 124, 155, 186, 217, 248, 279)

P=”123124” ; m=6 ; A=”123143212312331411341123124 2” ; n=28

h=2=d^m mod q (1, 10, 100−93=7, 70−62=8, 80−62=18, 180−155=25, 250−248=2)

p=23 (0, 1, 12, 123−93=30, 301−279=22, 222−217=5, 54−31=23)

a=11 (0, 1, 12, 123−93=30, 301−279=22, 224−217=7, 73−62=11)

```
     1 2 3 1 4 3 2 1 2 3 1 2 3 3 1 4 1 1 3 4 1 1 2 3 1 2 4 2
j=0 |           |  a=11
j=1   |           |  a=17=(112−2⋆1)−93
j=2     |           |  a=12=(171−2⋆2)−155
j=3       |1 2 1 2 3 4|  a=23=(122−2⋆3)−93=p
j=4         |           |  a=14=(233−2⋆1)−217
j=5           |           |  a=9=(141−2⋆1)−124
j=6             |           |  a=24=(92−2⋆3)−62
j=7               |           |  a=22=(243−2⋆2)−217
j=8                 |           |  a=4=(223−2⋆1)−217
     1 2 3 1 4 3 2 1 2 3 1 2 3 3 1 4 1 1 3 4 1 1 2 3 1 2 4 2
```

Rabin–Karp mintaillesztés

d=10 ; q=31 (31, 62, 93, 124, 155, 186, 217, 248, 279)

P="123124" ; m=6 ; A="123143212312331411341123124 2" ; n=28

h=2=d^m mod q (1, 10, 100−93=7, 70−62=8, 80−62=18, 180−155=25, 250−248=2)

p=23 (0, 1, 12, 123−93=30, 301−279=22, 222−217=5, 54−31=23)

a=11 (0, 1, 12, 123−93=30, 301−279=22, 224−217=7, 73−62=11)

```
     1 2 3 1 4 3 2 1 2 3 1 2 3 3 1 4 1 1 3 4 1 1 2 3 1 2 4 2
j=0 |           |  a=11
j=1   |           |  a=17=(112−2*1)−93
j=2     |           |  a=12=(171−2*2)−155
j=3       |1 2 1 2 3 4|  a=23=(122−2*3)−93=p
j=4         |           |  a=14=(233−2*1)−217
j=5           |           |  a=9=(141−2*1)−124
j=6             |           |  a=24=(92−2*3)−62
j=7               |           |  a=22=(243−2*2)−217
j=8                 |           |  a=4=(223−2*1)−217
     1 2 3 1 4 3 2 1 2 3 1 2 3 3 1 4 1 1 3 4 1 1 2 3 1 2 4 2
```

Rabin–Karp mintaillesztés

d=10 ; q=31 (31, 62, 93, 124, 155, 186, 217, 248, 279)

P=”123124” ; m=6 ; A=”123143212312331411341123124 2” ; n=28

h=2=d^m mod q (1, 10, 100-93=7, 70-62=8, 80-62=18, 180-155=25, 250-248=2)

p=23 (0, 1, 12, 123-93=30, 301-279=22, 222-217=5, 54-31=23)

a=11 (0, 1, 12, 123-93=30, 301-279=22, 224-217=7, 73-62=11)

```
     1 2 3 1 4 3 2 1 2 3 1 2 3 3 1 4 1 1 3 4 1 1 2 3 1 2 4 2
j=0 |           |  a=11
j=1   |           |  a=17=(112-2*1)-93
j=2     |           |  a=12=(171-2*2)-155
j=3       |1 2 1 2 3 4|  a=23=(122-2*3)-93=p
j=4         |           |  a=14=(233-2*1)-217
j=5           |           |  a=9=(141-2*1)-124
j=6             |           |  a=24=(92-2*3)-62
j=7               |           |  a=22=(243-2*2)-217
j=8                 |           |  a=4=(223-2*1)-217
     1 2 3 1 4 3 2 1 2 3 1 2 3 3 1 4 1 1 3 4 1 1 2 3 1 2 4 2
```

Rabin–Karp mintaillesztés

d=10 ; q=31 (31, 62, 93, 124, 155, 186, 217, 248, 279)

P=”123124” ; m=6 ; A=”123143212312331411341123124 2” ; n=28

h=2=d^m mod q (1, 10, 100-93=7, 70-62=8, 80-62=18, 180-155=25, 250-248=2)

p=23 (0, 1, 12, 123-93=30, 301-279=22, 222-217=5, 54-31=23)

a=11 (0, 1, 12, 123-93=30, 301-279=22, 224-217=7, 73-62=11)

```
     1 2 3 1 4 3 2 1 2 3 1 2 3 3 1 4 1 1 3 4 1 1 2 3 1 2 4 2
j=8                  |           |  a=4=(223-2*1)-217
j=9   a=6=(41-2*2)-31 |           |
j=10  a=27=(64-2*3)-31  |           |
j=11  a=21=(271-2*1)-248  |           |
j=12  a=21=(211-2*2)-186    |           |
j=13  a=21=(213-2*3)-186      |           |
j=14  a=22=(214-2*3)-186        |           |
j=15  a=2=(221-2*1)-217           |           |
     1 2 3 1 4 3 2 1 2 3 1 2 3 3 1 4 1 1 3 4 1 1 2 3 1 2 4 2
```

Rabin–Karp mintaillesztés

d=10 ; q=31 (31, 62, 93, 124, 155, 186, 217, 248, 279)

P="123124" ; m=6 ; A="123143212312331411341123124 2" ; n=28

h=2=d^m mod q (1, 10, 100-93=7, 70-62=8, 80-62=18, 180-155=25, 250-248=2)

p=23 (0, 1, 12, 123-93=30, 301-279=22, 222-217=5, 54-31=23)

a=11 (0, 1, 12, 123-93=30, 301-279=22, 224-217=7, 73-62=11)

```
     1 2 3 1 4 3 2 1 2 3 1 2 3 3 1 4 1 1 3 4 1 1 2 3 1 2 4 2
j=15  a=2=(221-2*1)-217            |           |
j=16  a=13=(21-2*4)-0                |           |
j=17  a=6=(132-2*1)-124                |           |
j=18  a=30=(63-2*1)-31                   |           |
j=19  a=16=(301-2*3)-279                   |           |
j=20  a=30=(162-2*4)-124                     |           |
j=21  a=23=(304-2*1)-279=p                     |1 2 1 2 3 4|
j=22  a=13=(232-2*1)-217                       |           |
```

Rabin–Karp mintaillesztés

d=10 ; q=31 (31, 62, 93, 124, 155, 186, 217, 248, 279)

P="123124" ; m=6 ; A="123143212312331411341123124 2" ; n=28

h=2=d^m mod q (1, 10, 100-93=7, 70-62=8, 80-62=18, 180-155=25, 250-248=2)

p=23 (0, 1, 12, 123-93=30, 301-279=22, 222-217=5, 54-31=23)

a=11 (0, 1, 12, 123-93=30, 301-279=22, 224-217=7, 73-62=11)

```
     1 2 3 1 4 3 2 1 2 3 1 2 3 3 1 4 1 1 3 4 1 1 2 3 1 2 4 2
j=15  a=2=(221-2*1)-217            |           |
j=16  a=13=(21-2*4)-0                |           |
j=17  a=6=(132-2*1)-124                |           |
j=18  a=30=(63-2*1)-31                   |           |
j=19  a=16=(301-2*3)-279                   |           |
j=20  a=30=(162-2*4)-124                     |           |
j=21  a=23=(304-2*1)-279=p                     |1 2 1 2 3 4|
j=22  a=13=(232-2*1)-217                       |           |

     1 2 3 1 4 3 2 1 2 3 1 2 3 3 1 4 1 1 3 4 1 1 2 3 1 2 4 2
```

Reguláris kifejezések

Összetett minták / reguláris kifejezések

„ab*c?d.e+f”

- `*`: a megelőző ’karakter’ tetszőleges számú előfordulása
- `?`: a megelőző ’karakter’ nulla vagy egy előfordulása
- `+`: a megelőző ’karakter’ egy vagy több előfordulása
- `.`: tetszőleges karakter

Az „ab*c?d.e+f” minta illeszkedik az alábbi alapsztringekre:

„fgabbbdreefad”, „rtacdhefr”, „sabbbcdeeefh”, „addef”,

és nem illeszkedik az alábbiakra:

„abbckld”, „acdb”, „cvfdcd”

Történelmi áttekintés

- Először az automata elmélet és formális nyelvek elmélete kapcsán merültek fel
  - Ezek az elméletek a számítógép működésének modellezésénél (automaták), illetve ezek osztályozásánál és leírásánál (formális nyelvek) voltak fontosak
- A Ken Thompson által készített QED szövegszerkesztő programban bukkantak fel először
- Ez került a Unix szerkesztőjébe (`ed`) is, ami a reguláris kifejezéseket használó `grep` elkészüléséhez vezetett.

Történelmi áttekintés

- Azóta a reguláris kifejezések széles körben elterjedtek a Unix-szerű rendszerek segédprogramjainál, amilyenek például az `expr`, az `awk`, az `Emacs`, a `vi`, a `lex` és a `Perl`.
- A reguláris kifejezéseket sok szövegszerkesztő, illetve segédprogram használja (`PyCharm` is).
- Több programozási nyelv is használja, illetve támogatja. Például a `Perl`, a `Python` és a `Tcl` is rendelkezik direkt reguláris kifejezések elemzésére szolgáló szintaktikai elemzővel.

Metakarakterek

'if a[3] == "sabbbcdeeefh":'

[ ]	: Egy karaker-halmaz	'a[a-m[]'
\	: Speciális szekvencia ill. speciális karakter	'\[\d\]'
.	: tetszőleges karakter	'e.h'
^	: sztring eleje	'^if'
$	: sztring vége	'\":$'
*	: nulla vagy több előfordulás	'ab*'
+	: egy vagy több előfordulás	'ab+'
?	: nulla vagy egy előfordulás	's?a'
{ }	: adott számú előfordulás	'={2}.*ab{3}cde{2,4}'
( )	: csoportosítás	'(ab)?b'
|	: ez vagy az	'a(\[|b)'

Karakter halmazok

s='if a[3] == "The rain in Spain":'

[arn] : illeszkedik a 3 betű bármelyikére

[a-n] : Az ASCII kódtábla 'a' és 'n' közötti összes katakterére illeszkedik

[a-ck-m] : ≡ [abcklm]

[^arn] : illeszkedik bármire, ami nem 'a', 'r' vagy 'n'

[1230] : Ha felsoroljuk, a sorrend nem érdekes

[0-9] : számjegy

[a-zA-Z] : betű

[0-5][0-9] : '00' és '59' közötti kétjegyű szám

[+] : ≡ '\+' (karaker-halmazokban a '+*.|(){}[' karaktereknek nincs speciális jelentése) (]-)

Speciális karakterek

'if a[3] == "The rain in Spain":'

\d	: számjegy (≡ [0-9])	'\[\d\]'
\D	: NEM számjegy (≡ [^0-9])	'a.\d.*a.\D'
\s	: 'fehér karakter' (≡ [\n\t])	'\sa'
\S	: NEM 'fehér karakter' (≡ [^ \n\t])	'\Sa'
\w	: 'szó karakter' (≡ [a-zA-Z0-9_])	'\wi'
\W	: NEM 'szó karakter' (≡ [^a-zA-Z0-9_])	'\Wi'
\b	: Szó eleje vagy vége	r'\bin.*in\b'
\B	: NEM Szó eleje vagy vége	r'\Bin.*\Bin'
\A	: sztring eleje	'\Aif'
\Z	: sztring vége	'\":\Z'

A Python RegEx modulja

A Python re modulja

```
>>> import re
>>> txt = "The rain in Spain"
>>> x = re.search(r"^The.*\bi", txt)
>>> print(x)
```

<re.Match object; span=(0, 10), match='The rain i'>

```
>>> x = re.search("^The.*\bi", txt)
>>> print(x) # nincs 'r' betű a minta előtt
```

None

```
>>> # Csak a \b és \B használata
>>> # esetén fontos az 'r' a minta elé
```

A `re` modul függvényei

search visszaad egy „Match” objektumot

findall visszad egy listát az összes illeszkedésről

split az illeszkedések mentén darabolja a sztringet

sub az illeszkedő alsztring(ek)et felülírja

A `search` függvény és a Match objektum

```
>>> import re
>>> txt = "The rain in Spain"
>>> x = re.search(r"e\S*\s+.", txt)
>>> print(x)
<re.Match object; span=(2, 5), match='e r'>
>>> print(x.start(), x.span(), x.end())
2 (2, 5) 5
>>> print([x.group(), x.string])
['e r', 'The rain in Spain']
```

A `search` függvény és a Match objektum

```
>>> import re
>>> txt = "The rain in Spain"
>>> x = re.search(r"e\S*\s+.", txt)
>>> print(x)
```

<re.Match object; span=(2, 5), match='e r'>

```
>>> print(x.start(), x.span(), x.end())
```

2 (2, 5) 5

```
>>> print([x.group(), x.string])
```

['e r', 'The rain in Spain']

A `search` függvény és a Match objektum

```
>>> import re
>>> txt = "The rain in Spain"
>>> x = re.search(r"e\S*\s+.", txt)
>>> print(x)
```

<re.Match object; span=(2, 5), match='e r'>

```
>>> print(x.start(), x.span(), x.end())
```

2 (2, 5) 5

```
>>> print([x.group(), x.string])
```

['e r', 'The rain in Spain']

A `findall` függvény

```
>>> import re
>>> txt = 'if a[3] == "The rain in Spain":'
>>> print(re.findall(r"\S\s+.", txt))
['f a', '] =', '= "', 'e r', 'n i', 'n S']
>>> print(re.findall(r"\S{3}\b", txt))
['a[3', 'The', 'ain', 'ain']
>>> print(re.findall(r"\w{3}\b", txt))
['The', 'ain', 'ain']
>>> s = re.search(r"\w{3}\b", txt).end()
>>> print(re.search(r"\w{3}\b", txt[s:]).group())
ain
```

A `findall` függvény

```
>>> import re
>>> txt = 'if a[3] == "The rain in Spain":'
>>> print(re.findall(r"\S\s+.", txt))
```
['f a', '] =', '= "', 'e r', 'n i', 'n S']

```
>>> print(re.findall(r"\S{3}\b", txt))
```
['a[3', 'The', 'ain', 'ain']

```
>>> print(re.findall(r"\w{3}\b", txt))
['The', 'ain', 'ain']
>>> s = re.search(r"\w{3}\b", txt).end()
>>> print(re.search(r"\w{3}\b", txt[s:]).group())
ain
```

A `findall` függvény

```
>>> import re
>>> txt = 'if a[3] == "The rain in Spain":'
>>> print(re.findall(r"\S\s+.", txt))
```
['f a', '] =', '= "', 'e r', 'n i', 'n S']

```
>>> print(re.findall(r"\S{3}\b", txt))
```
['a[3', 'The', 'ain', 'ain']

```
>>> print(re.findall(r"\w{3}\b", txt))
```
['The', 'ain', 'ain']

```
>>> s = re.search(r"\w{3}\b", txt).end()
>>> print(re.search(r"\w{3}\b", txt[s:]).group())
```
ain

A `findall` függvény

```
>>> import re
>>> txt = 'if a[3] == "The rain in Spain":'
>>> print(re.findall(r"\S\s+.", txt))
```
['f a', '] =', '= "', 'e r', 'n i', 'n S']

```
>>> print(re.findall(r"\S{3}\b", txt))
```
['a[3', 'The', 'ain', 'ain']

```
>>> print(re.findall(r"\w{3}\b", txt))
```
['The', 'ain', 'ain']

```
>>> s = re.search(r"\w{3}\b", txt).end()
>>> print(re.search(r"\w{3}\b", txt[s:]).group())
```
ain

A `split` függvény

```
>>> import re
>>> txt = 'if a[3] == "The rain in Spain":'
>>> print(re.split("\s", txt))
['if', 'a[3]', '==', '"The', 'rain', 'in', 'Spain":']
>>> print(re.split("\W", txt))
['if', 'a', '3', '', '', '', '', '', 'The', 'rain', 'in', 'Spain', '', '']
>>> print(re.split("\W+", txt))
['if', 'a', '3', 'The', 'rain', 'in', 'Spain', '']
>>> print(re.split("\W+", txt, 3))
['if', 'a', '3', 'The rain in Spain":']
>>> print(re.split("\d", txt))
['if a[', '] == "The rain in Spain":']
```

A `split` függvény

```
>>> import re
>>> txt = 'if a[3] == "The rain in Spain":'
>>> print(re.split("\s", txt))
```

['if', 'a[3]', '==', '"The', 'rain', 'in', 'Spain":']

```
>>> print(re.split("\W", txt))
```

['if', 'a', '3', '', '', '', '', '', 'The', 'rain', 'in', 'Spain', '', '']

```
>>> print(re.split("\W+", txt))
['if', 'a', '3', 'The', 'rain', 'in', 'Spain', '']
>>> print(re.split("\W+", txt, 3))
['if', 'a', '3', 'The rain in Spain":']
>>> print(re.split("\d", txt))
['if a[', '] == "The rain in Spain":']
```

A `split` függvény

```
>>> import re
>>> txt = 'if a[3] == "The rain in Spain":'
>>> print(re.split("\s", txt))
```

['if', 'a[3]', '==', '"The', 'rain', 'in', 'Spain":']

```
>>> print(re.split("\W", txt))
```

['if', 'a', '3', '', '', '', '', '', 'The', 'rain', 'in', 'Spain', '', '']

```
>>> print(re.split("\W+", txt))
```

['if', 'a', '3', 'The', 'rain', 'in', 'Spain', '']

```
>>> print(re.split("\W+", txt, 3))
```

['if', 'a', '3', 'The rain in Spain":']

```
>>> print(re.split("\d", txt))
```

['if a[', '] == "The rain in Spain":']

A `split` függvény

```
>>> import re
>>> txt = 'if a[3] == "The rain in Spain":'
>>> print(re.split("\s", txt))
```

['if', 'a[3]', '==', '"The', 'rain', 'in', 'Spain":']

```
>>> print(re.split("\W", txt))
```

['if', 'a', '3', '', '', '', '', '', 'The', 'rain', 'in', 'Spain', '', '']

```
>>> print(re.split("\W+", txt))
```

['if', 'a', '3', 'The', 'rain', 'in', 'Spain', '']

```
>>> print(re.split("\W+", txt, 3))
```

['if', 'a', '3', 'The rain in Spain":']

```
>>> print(re.split("\d", txt))
```

['if a[', '] == "The rain in Spain":']

A `split` függvény

```
>>> import re
>>> txt = 'if a[3] == "The rain in Spain":'
>>> print(re.split("\s", txt))
```

['if', 'a[3]', '==', '"The', 'rain', 'in', 'Spain":']

```
>>> print(re.split("\W", txt))
```

['if', 'a', '3', '', '', '', '', '', 'The', 'rain', 'in', 'Spain', '', '']

```
>>> print(re.split("\W+", txt))
```

['if', 'a', '3', 'The', 'rain', 'in', 'Spain', '']

```
>>> print(re.split("\W+", txt, 3))
```

['if', 'a', '3', 'The rain in Spain":']

```
>>> print(re.split("\d", txt))
```

['if a[', '] == "The rain in Spain":']

A sub függvény

```
>>> import re
>>> txt = 'if a[3] == "The rain in Spain":'
>>> print(re.sub("rain","sunshine", txt))
if a[3] == "The sunshine in Spain":
>>> print(re.sub("\d+","11", txt))
if a[11] == "The rain in Spain":
>>> print(re.sub("\s"," - ", txt))
if - a[3] - == - "The - rain - in - Spain":
>>> print(re.sub("\s"," - ", txt, 3))
if - a[3] - == - "The rain in Spain":
```


A sub függvény

```
>>> import re
>>> txt = 'if a[3] == "The rain in Spain":'
>>> print(re.sub("rain","sunshine", txt))
```

if a[3] == "The sunshine in Spain":

```
>>> print(re.sub("\d+","11", txt))
```

if a[11] == "The rain in Spain":

```
>>> print(re.sub("\s"," - ", txt))
```

if - a[3] - == - "The - rain - in - Spain":

```
>>> print(re.sub("\s"," - ", txt, 3))
```

if - a[3] - == - "The rain in Spain":

A sub függvény

```
>>> import re
>>> txt = 'if a[3] == "The rain in Spain":'
>>> print(re.sub("rain","sunshine", txt))
```
if a[3] == "The sunshine in Spain":
```
>>> print(re.sub("\d+","11", txt))
```
if a[11] == "The rain in Spain":
```
>>> print(re.sub("\s"," - ", txt))
```
if - a[3] - == - "The - rain - in - Spain":
```
>>> print(re.sub("\s"," - ", txt, 3))
```
if - a[3] - == - "The rain in Spain":

A sub függvény

```
>>> import re
>>> txt = 'if a[3] == "The rain in Spain":'
>>> print(re.sub("rain","sunshine", txt))
```
if a[3] == "The sunshine in Spain":
```
>>> print(re.sub("\d+","11", txt))
```
if a[11] == "The rain in Spain":
```
>>> print(re.sub("\s"," - ", txt))
```
if - a[3] - == - "The - rain - in - Spain":
```
>>> print(re.sub("\s"," - ", txt, 3))
```
if - a[3] - == - "The rain in Spain":

További variációk

- Csak a reguláris kifejezések python-ban is támogatott tulajdonságait tárgyaltuk
- Más programnyelvek, egyéb alkalmazások további lehetőségeket vezetnek be
  - [:lower:] ....
  - \< szó eleje, \> szó vége
  - (.)\1, vagy (\d)(.)X\2\1
  - sub(/\d/, ”&&”, ”a[3]”)
- awk, nawk, mawk, gawk

Reminder, rules so far

1. Think before you program!
2. A program is a human-readable essay on problem solving that also happens to execute on a computer.
3. The best way to improve your programming and problem solving skills is to practice!
4. A foolish consistency is the hobgoblin of little minds
5. Test your code, often and thoroughly
6. If it was hard to write, it is probably hard to read. Add a comment.
7. All input is evil, unless proven otherwise.
8. A function should do one thing.
9. Make sure your class does the right thing.