



פייתון

Regular Expressions:

הפיכת רצפים לאוסף חוקים

▶ להלן מספר דוגמאות לכתובות קו חוקיות

127.0.0.1 ◦

255.255.100.100 ◦

0.0.10.20 ◦

190.200.0.12 ◦

◦ חישוב- איך אפשר לתאר כתובת קו על ידי תבנית?

• דמיינו שאתם מנסים להסביר לחבר איך למצוא כתובת קו בתוך


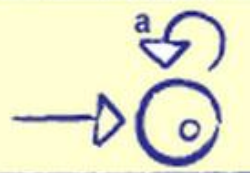

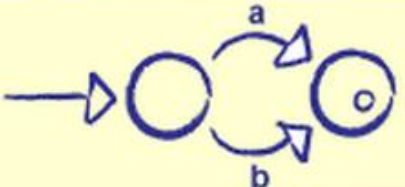
קובץ טקסט

• חפשו תבנית שתתאים לכתובות קו אך תסנן רצפי מספרים אחרים

◦ פתרון אפשרי: "רצף של 4 מספרים, באורך ספרה אחת עד שלוש

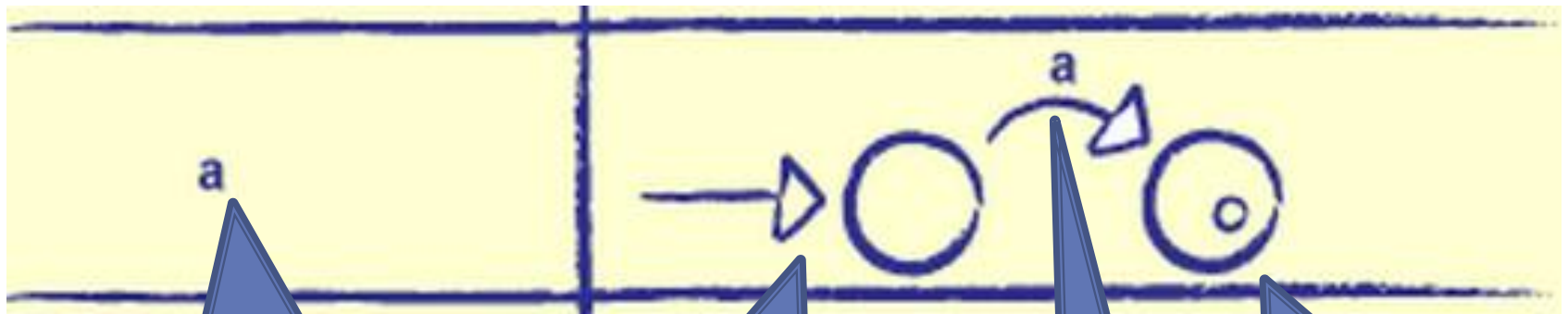
ספרות, אשר מופרדים על ידי נקודה."

ביטויים רגולריים - Regular Expressions

regular expression	finite state machine
a	
a^*	
a^+	
a/b	

- ▶ נתונה מכונת מצבים כלשהי. נרצה לבדוק האם קלט מסויים הוא חוקי:
 - נריץ את הקלט על האוטומט לפי החיצים
 - אם בסוף הקלט הגענו למצב מקבל, הקלט חוקי

ביטויים רגולריים - Regular Expressions



ביטוי רגולרי: מתאר את אוסף כל הקלטים שמתקבלים על ידי מכונת המצבים

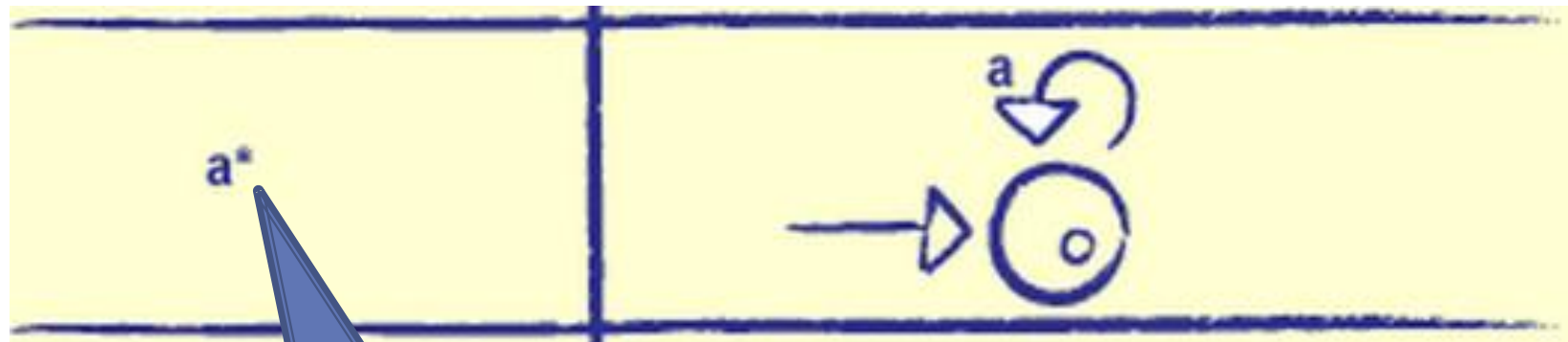
חץ מסמן מצב התחלתי

עיגול מוקף בעיגול מסמן מצב מקבל

קשת מסמנת מעבר למצב הבא, האות מעל הקשת היא הקלט

מכונת המצבים שבאיור מקבלת רק את הקלט 'a'.
הקלט 'aa', לדוגמה, אינו תקין.

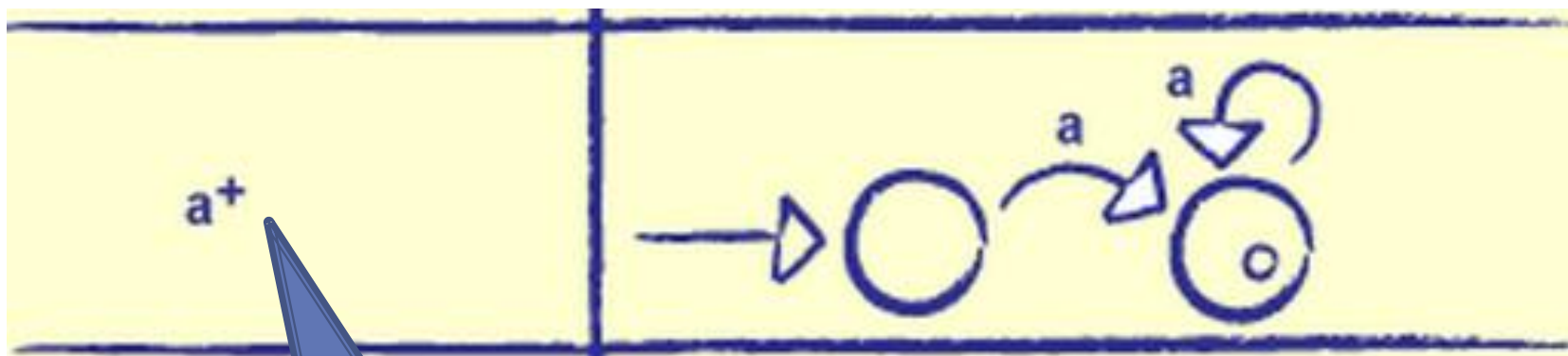
ביטויים רגולריים - Regular Expressions



כוכבית מסמנת
"חזרה בין אפס
לאינסוף פעמים"

מכונת המצבים שבאיור מקבלת
כל קלט שהוא רצף של 'a'.
לדוגמה 'a', 'aa', 'aaa' וכו'.
גם קלט שהוא מילה ריקה
יתקבל (אפס פעמים 'a').

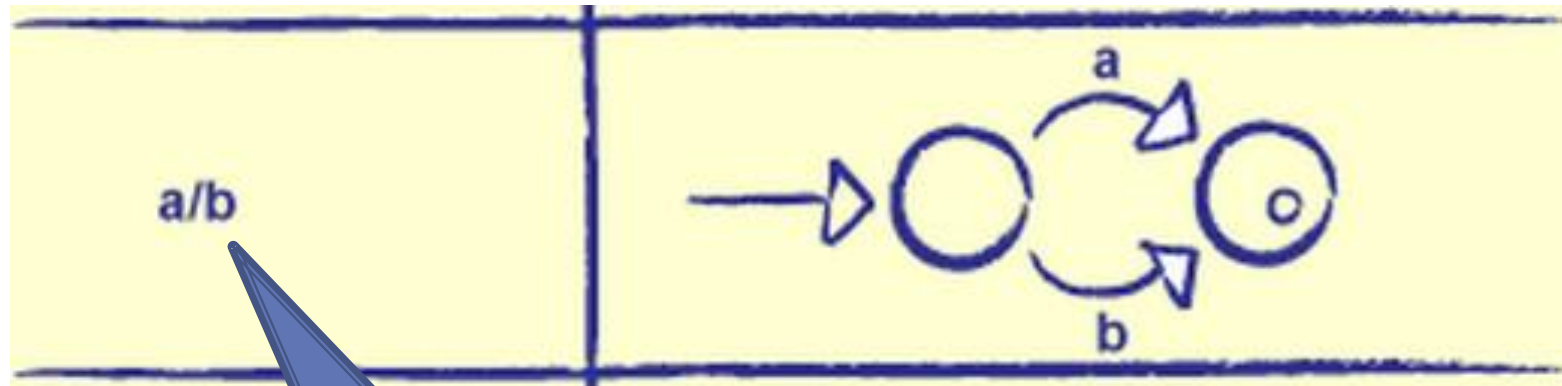
ביטויים רגולריים - Regular Expressions



פלוס מסמן "חזרה
בין פעם אחת
לאינסוף פעמים"

מכונת המצבים שבאיור מקבלת
כל קלט שהוא רצף של 'a'.
לדוגמה 'a', 'aa', 'aaa' וכו'.
קלט שהוא מילה ריקה לא
יתקבל (צריך 'a' אחד לפחות).

ביטויים רגולריים - Regular Expressions



'a' או 'b'

מכונת המצבים שבאיור מקבלת
או את הקלט 'a' או את הקלט
'b' (אך לא 'aa' או 'ab',
לדוגמה).

ביטויים רגולריים - Regular Expressions

איזה קלט יתקבל על ידי מכונת המצבים הבאה? כיתבו בתור ביטוי רגולרי



חזרה אפס פעמים או יותר

$(a/b)^*c(d/e)$

a או b

d או e

פעם אחת c

ביטויים רגולריים בפייתון

- ▶ כעת נראה איך לכתוב ביטויים רגולריים בפייתון
- ▶ יש לבצע `re-import`
- חבילה של מתודות `regular expressions`
- ▶ מתודות שנעשה בהן שימוש:
 - `Search`
 - `findall`



המתודה search

```
>>> import re
>>> match=re.search('ell','hello world')
>>> match.group()
'ell'
>>> match.start()
1
>>> match.end()
4
>>> match.span()
(1, 4)
>>>
```

מקבלת: ▶

◦ רצף לחיפוש

◦ טקסט בתוכו מחפשים

את הרצף

מחזירה אוסף של ▶

נתונים:

◦ איזה רצף נמצא

◦ מיקומים בטקסט

ניתן לשלוף את ▶

הנתונים ע"י מתודות

(בדוגמה)

חיפוש אחרי כל תו

- ▶ מה אם "אכפת לנו" רק מחלק מהתווים?
 - לדוגמה אנחנו מחפשים כל רצף של 4 תווים, שמתחיל ב-W ומסתיים ב-L
- ▶ נשתמש בתו '.' (נקודה) כדי לציין שכל תו מתקבל
 - דוגמאות:

```
>>> match=re.search('w..l','hello world')
>>> match.group()
'worl'
>>> match=re.search('..lo.w','hello world')
>>> match.group()
'ello w'
>>> match=re.search('...ld','hello world')
>>> match.group()
'world'
```

שימוש ב-\ (לוכסן)

▶ מה אם הביטוי שאנחנו מחפשים כולל נקודות? אם נכתוב נקודות סתם כך, כל תו יכול להחליף אותן...

```
>>> match=re.search('..a','11a..a')
>>> match.group()
'11a'
```

▶ הפתרון הוא שימוש בלוכסן, שמסמן שאל התו הבא צריך להתייחס כמו שהוא:

```
>>> match=re.search('\.\\.a','11a..a')
>>> match.group()
'..a'
```

חיפוש תווים מטיפוס מסויים

- ▶ מה אם השימוש ב-'.' הוא רחב מדי לצרכים שלנו?
 - לדוגמה, התו A ואחריו שלושה תווים שהם ספרות או אותיות, ואז התו A
 - ALPHA - תקין
 - A123A - תקין
 - A@BAA - לא תקין (יש באמצע תו שאינו אות או ספרה)
 - AT BA - לא תקין (יש רווח באמצע)

A??A

התאמת \w

▶ הסימון \w מגדיר שאנו מאפשרים לקבל כל תו שהוא אות קטנה, אות גדולה, ספרה או _ (קו תחתי)

```
AttributeError: 'NoneType' object has no attribute 'group'  
>>> match=re.search('ll\w','hello world')  
>>> match.group()  
'llo'  
>>>
```

▶ תו רווח, לדוגמה, אינו נכלל בהתאמת \w

- החיפוש השני מחזיר תוצאה ריקה

```
>>> match=re.search('o.w','hello world')  
>>> match.group()  
'o w'  
>>> match=re.search('o\ww','hello world')  
>>> match  
>>>
```

התאמת \w

▶ מה יחזיר החיפוש הבא?

```
match=re.search('a\\w\\w\\wa','beta gama al!ha al ha alpha')
```


התאמות לסוגי תווים מיוחדים

```
>>> match=re.search('\d\s\d\s\d','1 2 3')
>>> match.group()
'1 2 3'
```

digit - \d ▶

רווח - \s ▶

ועוד התאמות רבות נוספות שלא נעסוק בהן אך כדאי לדעת שהן קיימות:

<https://docs.python.org/2/library/re.html> ◦



התאמות * (כוכבית) ו- + (פלוס)

▶ היזכרו במבוא, אודות משמעות ה- * (כוכבית) בביטויים רגולריים. מה לדעתכם יחזיר הביטוי הבא?

```
match=re.search('a*b', 'aaac b aaaad aaaaabxxx')
```

▶ ומה אם נחליף את הכוכבית בפלוס?

```
match=re.search('a+b', 'aaac b aaaad aaaaabxxx')
```

▶ תשובות:

- 'b' - כיוון שכל כמות של a מתאימה, גם אפס
- 'aaaaab' - הביטוי חייב להתחיל לפחות במופע אחד של a

שילוב `\w` עם `*` או `+`

- ▶ כשם שאפשר להגדיר, לדוגמה, `a*` או `a+` אפשר להגדיר:
 - `\w*` - כל כמות של תווים מסוג אות / ספרה
 - `\w+` - לפחות תו אחד מסוג אות / ספרה

```
>>> match=re.search('\w+c','c aac b aad aabxx')
>>> match.group()
'aac'
>>> match=re.search('\w*c','c aac b aad aabxx')
>>> match.group()
'c'
```

שימוש ב-[] (סוגריים מרובעים)

- ▶ בתוך סוגריים מרובעים אפשר להגדיר אוסף של תווים אפשריים
 - לדוגמה [abc] יתאים ל-a, b או c
 - [\.w] יתאים לכל התווים שהם או מסוג w \ או נקודה
 - אחרי סוגריים מרובעים אפשר לשים * או +, לציון החזרות

- ▶ תרגיל: צרו ביטוי רגולרי (אחד) שמתאים ללכידת כתובות אימייל
 - בידקו אותו על מגוון כתובות אימייל, לצורך הבדיקה הוסיפו לצידן טקסט סתמי כלשהו:

- abc123@yahoo.com
- Humpty.dumpty@gmail.com
- TwistNshout@facebook.com



פתרון התרגיל- מציאת כתובת אימייל

▶ '[\w.]+@[\w.]+'

▶ הסבר מילולי:

- חפש תו אחד לפחות שהוא או מטיפוס \w או נקודה
- לאחר מכן צריך להופיע '@'
- לאחר מכן תו אחד לפחות שהוא או מטיפוס \w או נקודה
- ▶ הפתרון מאפשר גם כתובות אימייל לא חוקיות כגון:
 - john@doe (חסר נקודה משהו לאחר ה-doe)
 - @gmail.com (לפני ה-@ אין אף תו, רק נקודה)
 - שפרו את הפתרון!

שימוש ב- () (סוגריים עגולים)

- ▶ שימוש בסוגריים עגולים מאפשר לנו לשמור ל-tuple חלקים מהביטוי המבוקש
- ▶ נבחן מה אפשר לבצע כאשר אנחנו משנים את הביטוי הרגולרי לאימיילים כך שיכיל סוגריים עגולים:
▶ `'([\w.]+)@([\w.]+)'`

```
>>> match=re.search('([\w.-]+)@([\w.]+)', 'abc@gmail.com')
>>> match.group()
'abc@gmail.com'
>>> match.group(1)
'abc'
>>> match.group(2)
'gmail.com'
```

findall

- ▶ לעיתים קרובות אנחנו רוצים לשלוף לא רק את הביטוי הראשון שמתאים לביטוי הרגולרי, אלא את כולם
- ▶ שימוש ב-findall:

```
>>> match=re.findall('([\w.-]+)@([\w.]+)', 'abc@gmail.com bla 123@gmail.com')
>>> match
[('abc', 'gmail.com'), ('123', 'gmail.com')]
>>> match[0][0]
'abc'
>>> match[0][1]
'gmail.com'
>>> match[1][0]
'123'
>>> match[1][1]
'gmail.com'
>>>
```

תרגיל מסכם- חילוץ כתובות ip

- ▶ כיתבו ביטוי רגולרי שמחלץ כתובות ip מתוך טקסט
 - בידקו על מגוון של כתובות ip
 - הוסיפו מסיחים כדי לבדוק שהם אינם נקלטים
 - תנו לחבריכם לנסות למצוא כתובות ip חוקיות שאינכם מגלים או רצפים לא חוקיים שאתם מחלצים בטעות

