

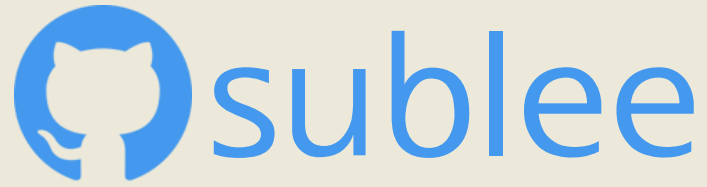


# 한글라이즈 재제작기

이흥섭

2018년 8월 고랭코리아 밋업

이형섭



# 넥슨 왓 스튜디오 서버아키텍트

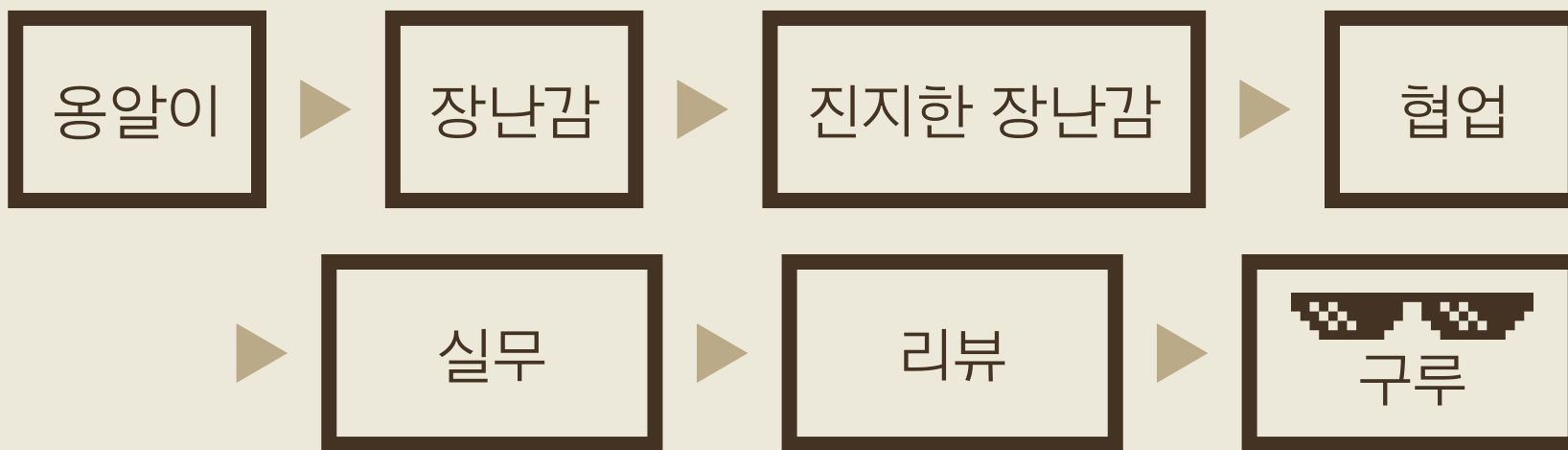
- 야생의 땅: 듀랑고
- 카트라이더 코인러시
- 카트라이더 대시

# 오픈소스

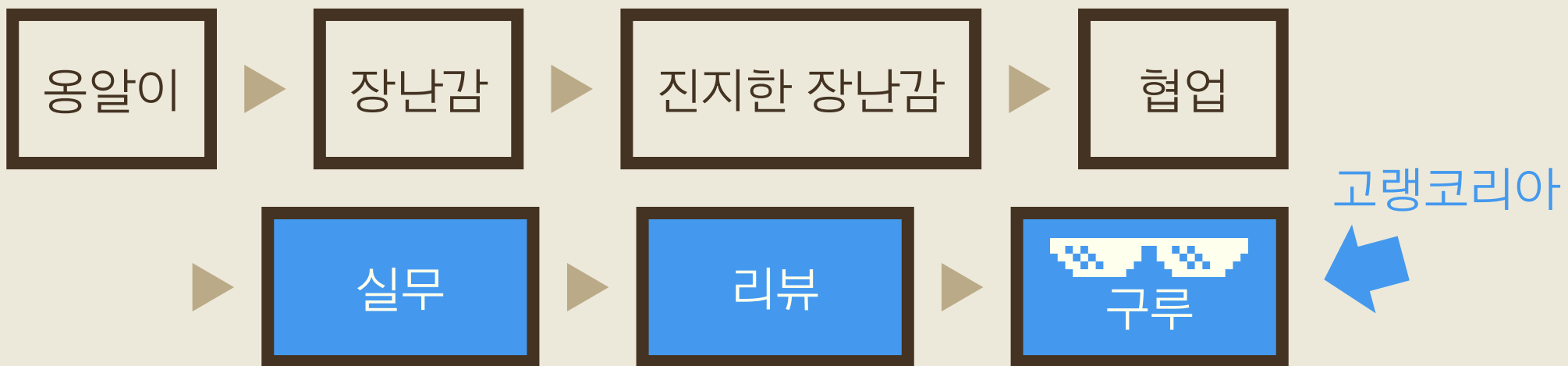
- what-studio/profiling ★ 2741
- sublee/trueskill ★ 383
- what-studio/tossi ★ 107
- hangulize/hangulize ★ 49

★개수는 2018년 8월 29일 기준

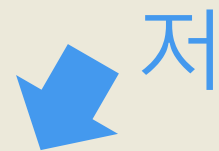
# Go 코딩



# Go 코딩



# Go 코딩



지난 3개월 ✨  
일과 후 취미코딩 ✨



# 차례

1. 한글라이즈
2. Go
3. 이론 것

# 1. 한글라이즈



# 한글라이즈

외국어 단어를

한글로 옮겨 적는 도구

# 전사(transcription)

말소리를 음성 문자로 옮겨 적음. 다음사전

의미를 옮기는 번역(translation)과는 달라요.

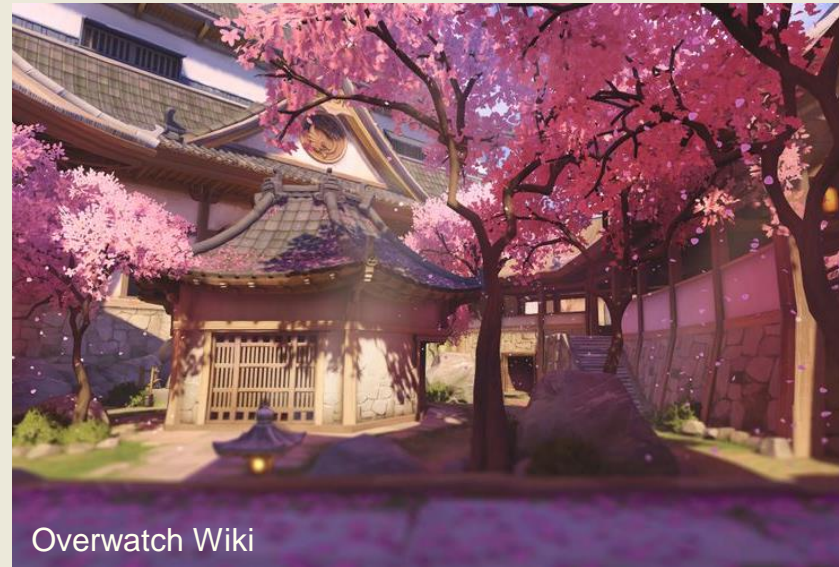
Go ▶ 고 (전사)  
▶ 가 (번역)

인명



Lúcio ► 루시우

지명



花村 ► 하나무라

# 외래어 표기법

국립국어원이 정한  
외래어 네이밍 컨벤션

Cappuccino 이탈리아어





Cappuccino<sup>이탈리아어</sup>

▶ 캡푸씨노...?

Cappuccino<sup>이탈리아어</sup>

▶ 카푸치노✓

宮本茂일본어



宮本茂일본어

▶ 공본무...?



宮本茂<sup>일본어</sup>

▶ 미야모토 시게루✓

**Χίος**그리스어





**Xíos** 그리스어

▶ 히오스 ✓



ピカチュウ 일본어

▶ 피카츄





ピカチュウ 일본어

▶ 피카추 ✓

# 어디까지나 권고사항

- ピカチュウ ▶ 피카츄 ✓ 피카추
- 青島 ▶ 칭따오 ✓ 칭다오
- Chevrolet ▶ 쉼보레 ✓ 세브럴레이

# 외래어 표기법

## 제1항

외래어는 국어의 현용 24 자모만으로 적는다.

## 제2항

외래어의 1 음운은 원칙적으로 1 기호로 적는다.

## 제3항

받침에는 ㄱ, ㄴ, ㄹ, ㅁ, ㅂ, ㅅ, ㅇ만을 쓴다.

## 제4항

파열음 표기에는 된소리를 쓰지 않는 것을 원칙으로 한다.

## 제5항

이미 굳어진 외래어는 관용을 존중하되,  
그 범위와 용례는 따로 정한다.

자음			반모음		모음	
국제 음성 기호	한글		국제 음성 기호	한글	국제 음성 기호	한글
	모음 앞	자음 앞 또는 어 말				
p	ㅍ	ㅂ, 프	j	이	i	이
b	ㅂ	브	ɥ	위	y	위
t	ㅌ	ㅅ, 트	w	오, 우	e	에
d	ㄸ	드			ø	외
k	ㅋ	ㄱ, 크			ɛ	에
g	ㄱ	그			ɛ~	앵
f	ㅍ	프			œ	외
v	ㅂ	브			œ~	윙
θ	ㅅ	스			æ	애
ð	ㄸ	드			a	아
s	ㅅ	스			ɑ	아

# 정부·언론 외래어 심의 공동위원회

- 정기적으로 각 외래어의 한글 표기 결정
- 언론에 보도되는 이름 중심

## 제92차 정부·언론외래어심의공동위원회 심의 결정안

(인명 52건, 일반용어 1건-재심의 2건)

- **아베라, 암살라** Amsale Aberra 1954(?1953)~ 에티오피아 여성 기업가·디자이너. 암살라(Amsale) 그룹 대표 겸 디자인 총책임자. 유행을 좇지 않으면서 세련된 클래식 모던을 강조하는 디자인으로 유명함.
- **아벨란제, 주앙** João Havelange 1916~ 국제축구연맹(FIFA) 회장(1974~1998). 브라질 인.
- **그루벨, 루스** Ruth M. Grubel 1950~ 미국 교육가·선교사. 일본 간사이(關西)학원 원장(2007. 4.~ ).

## 제92차 정부·언론외래어심의공동위원회 심의 결정안

(인명 52건, 일반용어 1건-재심의 2건)

아버라, 암살러

- ~~아베라, 암살라~~ Amsale Aberra 1954(?1953)~ 에티오피아 여성 기업가·디자이너. 암살라(Amsale) 그룹 대표 겸 디자인 총책임자. 유행을 좇지 않으면서 세련된 클래식 모던을 강조하는 디자인으로 유명함.
- 아벨란제, 주앙 João Havelange 1916~ 국제축구연맹(FIFA) 회장(1974~1998). 브라질 인.
- ~~그루벨~~ 루스 Ruth M. Grubel 1950~ 미국 교육가·선교사. 일본 간사이(關西)학원 원장(2007. 4.~ ).

박종성  
@iceager



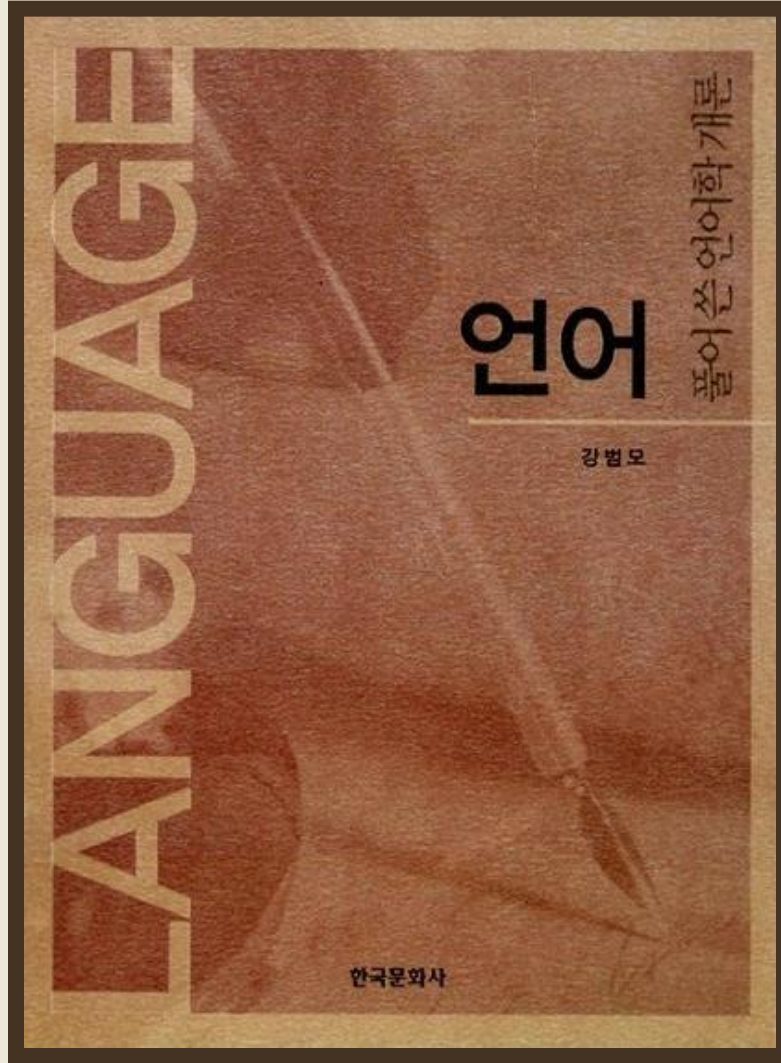
- 세계의 말과 글
- 끝소리 

“

정기적으로 회의를 열어 용례를 정하는 것으로는 한계가 있다. **외래어 표기 심의 방식이 자동화되어** 한글로 표기하고 싶은 외국어를 입력하자마자 한글 표기가 나와야 한다. ”

<http://iceager.egloos.com/2610028>



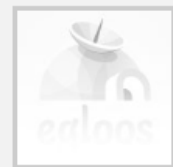



강범모, 《언어: 풀어 쓴 언어학 개론》(2005)

“

정기적으로 회의를 열어 용례를 정하는 것으로는 한계가 있다. **외래어 표기 심의 방식이 자동화되어** 한글로 표기하고 싶은 외국어를 입력하자마자 한글 표기가 나와야 한다. ”

<http://iceager.egloos.com/2610028>



2010/10/02 17:34 # 삭제 답글 

비공개 댓글입니다.



끝소리 2010/10/02 20:01 #

이메일 드렸습니다.

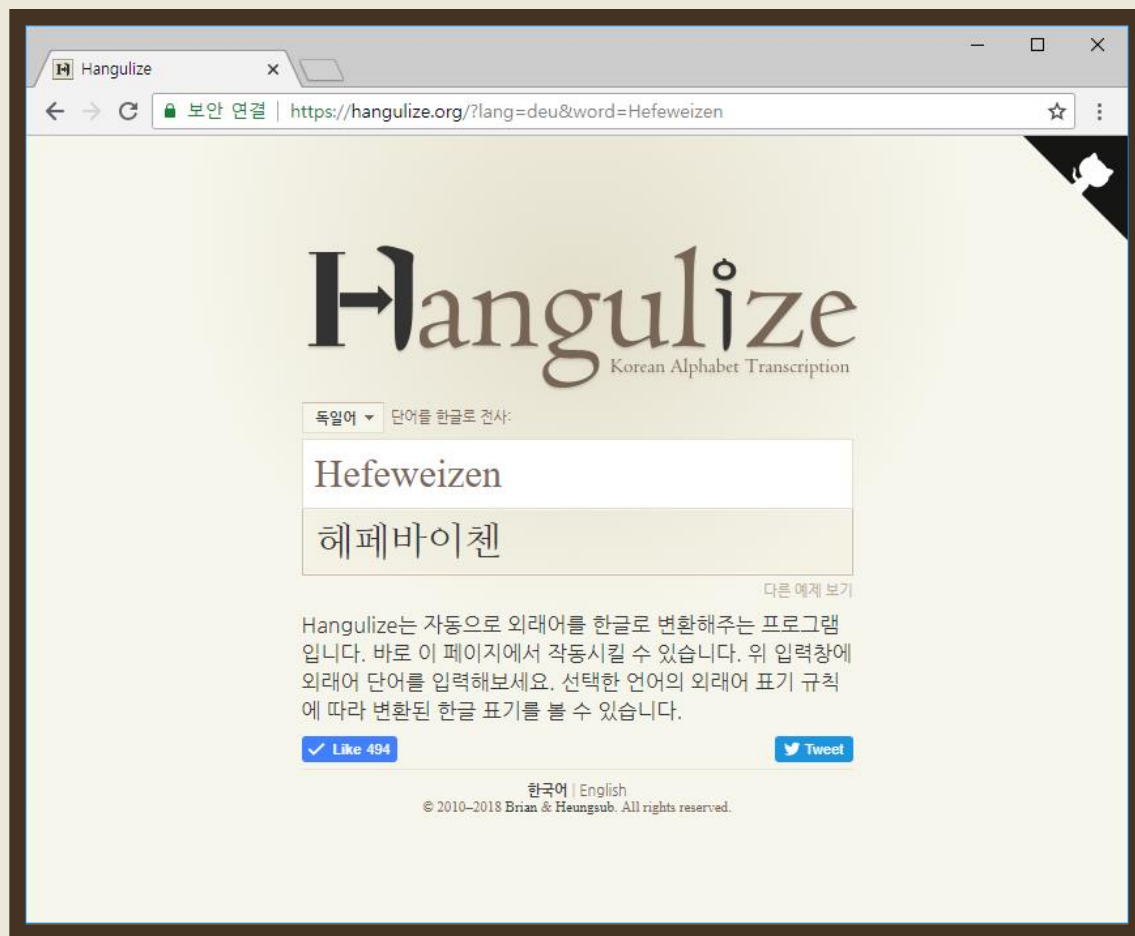


파이썬

# 옛 한글라이즈

```
>>> from hangulize import *  
>>> hangulize(u'Cappuccino', 'ita')  
u'카푸치노'
```

<https://hangulize.org/>



# 주로 번역가가 사용



서 역은 《열린책들 편집 매뉴얼》이라는 책을 구입  
길 바란다. 한글 맞춤법, 표준어 규정, 외래어 표기  
은 책으로 편집자를 위한 책이지만 번역가에게도  
다. 또한 번역을 하다 보면 외래어 표기법에서 문  
가 많은데 그럴 때는 아래의 사이트를 활용하기

우리말배움터([urimal.cs.pusan.ac.kr](http://urimal.cs.pusan.ac.kr))의 '외래어  
호변환기'


국립국어원(<http://www.korean.go.kr>)의 외래어 표기법

한글라이즈(<http://www.hangulize.org>) 국립국어원

법 규칙을 프로그램으로 구현

이지민, 《그래도 번역가로 살겠다면》(2017)

# 옛 한글라이즈 불만

- 주먹구구식 구현
- 고치기 어려운 버그
- 전사 규칙이 파이썬 코드 



# 옛 한글라이즈

2010-2018



Go

# 새 한글라이즈

2018년 5월-

```
gore> :import "github.com/hangulize/hangulize"  
gore> hangulize.Hangulize("rus", "Владивосток")  
"블라디보스토크"
```

# 전사 규칙 .hgl 파일

## lang:

```
id      = "rus"  
codes   = "ru", "rus"  
english = "Russian"  
korean  = "러시아어"  
script  = "cyrillic"
```

## macros:

```
"@" = "<vowels>"
```

## vars:

```
"cs" = "б", "в", "г"
```

## rewrite:

```
"град"  -> "град-"  
"^ль"   -> "л-ь"  
"ый"    -> "ы"
```

## transcribe:

```
"б"      -> "ㅂ"  
"л,"     -> " - ㄹ "
```

## test:

```
"Ольга"  -> "올가"  
"Пётр"   -> "표트르"
```

H 형상, 로마자 i, 화살표, 한글 ㅣ



옛 로고



새 로고

四川



쓰촨

# 한글라이즈 파이프라인



# 1단계: Phonemize



四川 ▶ si chuan

- 뜻글자를 소리글자로
- 사전
- 형태소 분석



# 1단계: Phonemize



四川 ▶ si chuan

- 뜻글자를 소리글자로
- 사전
- 형태소 분석

## 2단계: Rewrite

Phonemize



Rewrite



Transcribe

si chuan



"{s}i" -> "i,"

si, chuan



"{h}uan" -> "wan"

si, chwan



"n\$" -> "n,"

si, chwan,

- 패턴 찾아 바꾸기

- 표음성 강화

# 표음성

표음성 극대화

신도리맹 널차  
지오캥 그팽널차

형태학적 표기

신도림행 열차  
지옥행 급행열차

## 2단계: Rewrite

Phonemize



Rewrite



Transcribe

si chuan



"{s}i" -> "i,"

si, chuan



"{h}uan" -> "wan"

si, chwan



"n\$" -> "n,"

si, chwan,

- 패턴 찾아 바꾸기

- 표음성 강화

# 3단계: Transcribe

Phonemize

Rewrite

Transcribe

si, chwan,

"wan,?" -> "ㅍ-ㄴ "

si, chㅍ-ㄴ

"i," -> "ㅡ"

sㅡ chㅍ-ㄴ

"ch" -> "ㅈ"

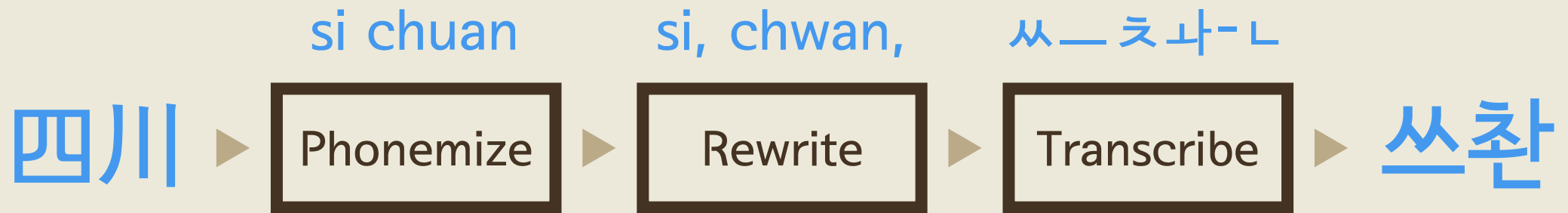
sㅡ ㅈㅍ-ㄴ

"s" -> "ㅆ"

ㅆㅡ ㅈㅍ-ㄴ

- 한글 자모로 변환
- 패턴 찾아 바꾸기
- 외래어 표기법 구현

# 한글라이즈 파이프라인



아제르바이잔어	에스토니아어	조지아어 <sup>제2안</sup>	러시아어
벨라루스어	핀란드어	라틴어	슬로바키아어
불가리아어	고대 그리스어	라트비아어	슬로베니아어
카탈로니아어	세르보크로아트어	리투아니아어	스페인어
체코어	헝가리어	마케도니아어	알바니아어
중국어	아이슬란드어	네덜란드어	스웨덴어
웨일스어	이탈리아어	폴란드어	터키어
독일어	일본어	포르투갈어	우크라이나어
그리스어	일본어 <sup>최영애-김용옥</sup>	브라질 포르투갈어	베트남어
에스페란토어	조지아어 <sup>제1안</sup>	루마니아어	웨일스어 <sup>중세</sup>

~~영어~~



# 영어의 낮은 표음성

- though ▶ 도
- through ▶ 스루
- rough ▶ 러프
- cough ▶ 코프
- thought ▶ 소트
- bough ▶ 바우

Ghoti ▶

Ghoti ▶

enough women nation

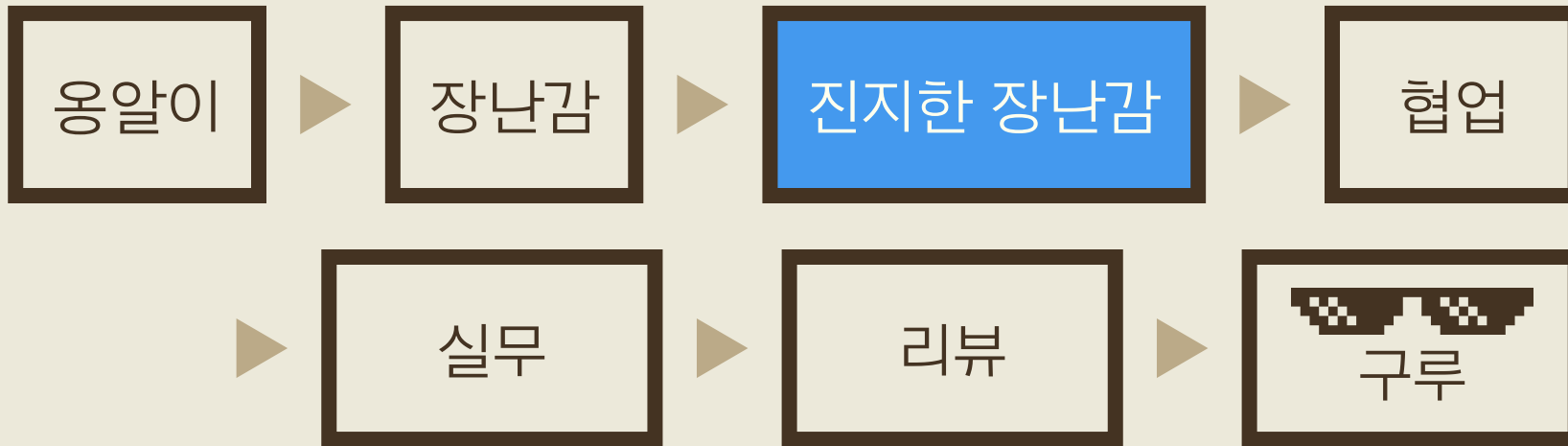
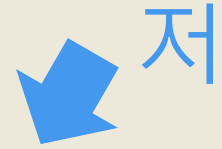
Ghoti ▶ 피시

enough women nation

## 2. Go



# ! 초보 주의




고랭코리아



Go 첫인상

고루틴 끝내준다!

# Go 첫인상

- 고루틴은 끝내주는데
- 그런데 너무 기능이 적다.
- 고퍼도 귀엽지 않고 



Go 첫인상

이걸로 정말

제품이 만들어질까?



Docker



etcd

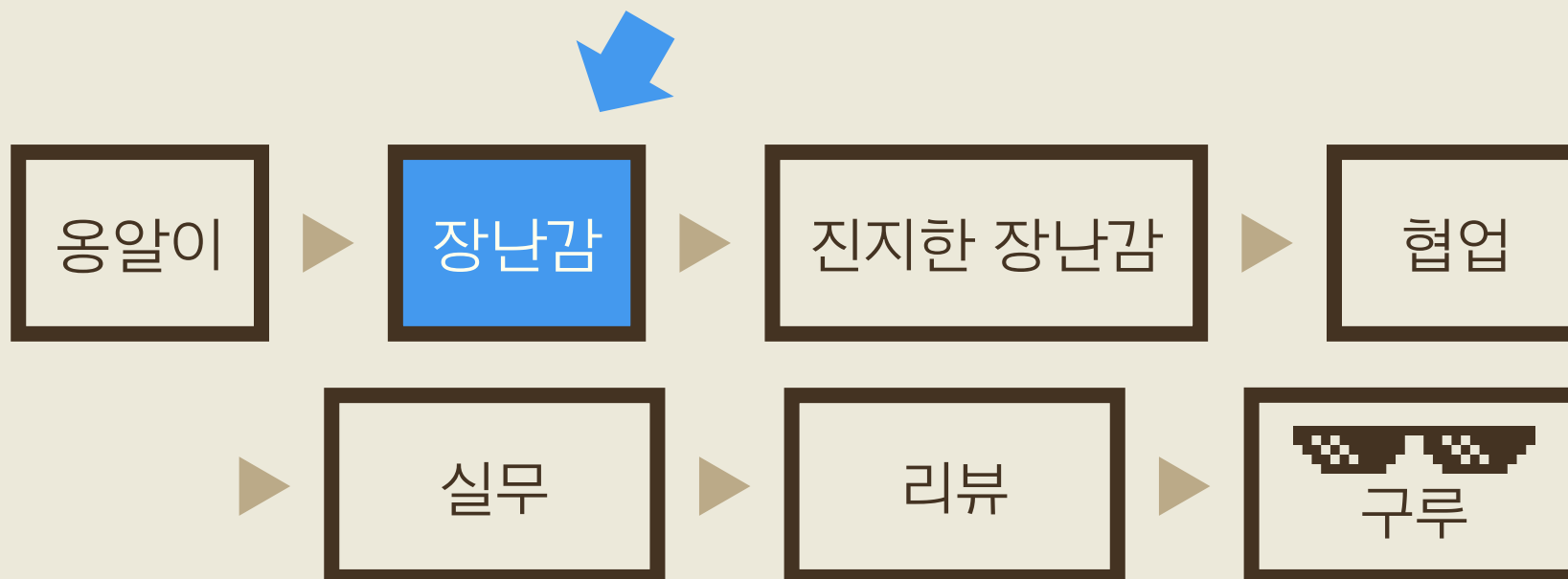


Terraform

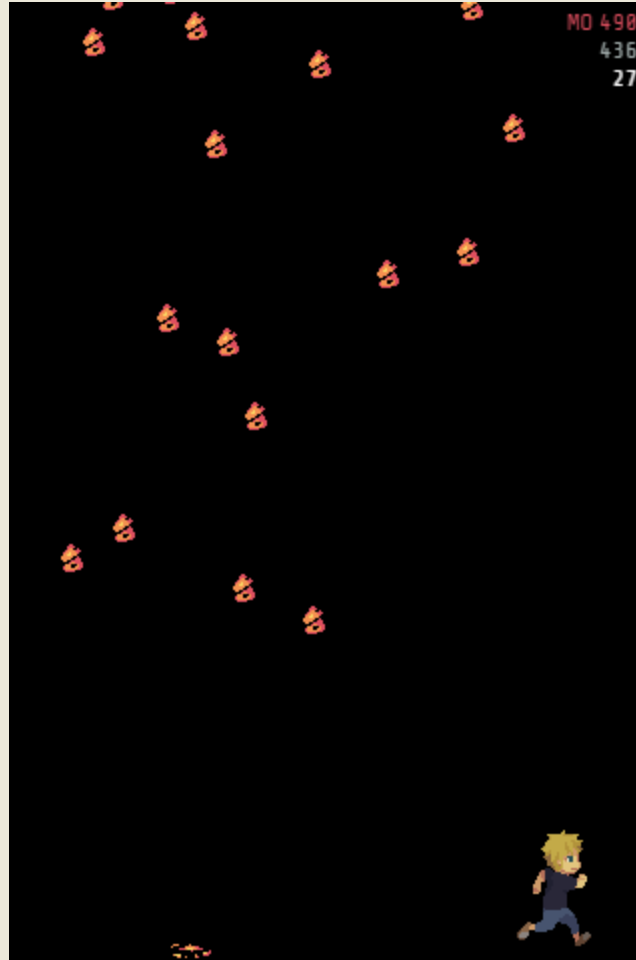


K8s

된다.



<https://subl.ee/runker/>



랭킹서버가 Go

쓸 만 하다.  
더 잘 하고 싶다.

한글라이즈 다시 만들면  
왠지 더 잘 만들 것 같다.

# 옛 한글라이즈 불만

- 주먹구구식 구현
- 고치기 어려운 버그
- 전사 규칙이 파이썬 코드



# 연습과제: 한글라이즈

- 이미 한 번 만들어 봄
- 유니코드/정규표현식
- 적당한 복잡성
- 풍부한 테스트케이스
- 계산 중심적
- 언어학 덕질

a. 정규표현식

b. 말과 글

c. 테스트와 문서화

a. 정규표현식



# 이메일 정규표현식

```
/([a-z0-9!#$%&'*/+=?^_`{|}~-]+(\.[a-z0-9!#$%&'*/+=?^_`{|}~-]+)*|"(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21\x23-\x5b\x5d-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])*")@(?:(?:[a-z0-9](?:[a-z0-9-]*[a-z0-9])?\.[a-z0-9](?:[a-z0-9-]*[a-z0-9])?|\\(?:[0-9]{3}|2[0-4][0-9]|01?[0-9][0-9]?|25[0-5]|2[0-4][0-9]|01?[0-9][0-9]?|[a-z0-9-]*[a-z0-9]:(?:[\x01-\x08\x0b\x0c\x0e-\x1f\x21-\x5a\x53-\x7f]|\\[\x01-\x09\x0b\x0c\x0e-\x7f])+)\\)))/
```

<http://emailregex.com/>

```
import "regexp"
```

# RE2

by Google

- 안전한 정규표현식 구현
- 문자열 길이에 대해 선형 시간
- 고의적으로 일부 기능 미지원

# RE2 성능 보장

대상 문자열 길이

$$O(\underbrace{m}_{\text{정규표현식 길이}} \underbrace{n}_{\text{대상 문자열 길이}})$$

정규표현식 길이

# RE2 성능 보장

대상 문자열 길이

$$O(\underbrace{m}_{\text{보통 상수항}} \underbrace{n}_{\text{대상 문자열 길이}})$$

보통 상수항



# 벤치마크

`/a?a?a?aaa/`를  
"aaa"에 매치

Russ Cox, 〈Regular Expression Matching Can Be Simple And Fast〉(2007)

# 벤치마크

~~/a?a?a?aaa/~~를  
"aaa"에 매치

Russ Cox, 〈Regular Expression Matching Can Be Simple And Fast〉(2007)

# 벤치마크

$f(1) = /a?a/$ 를 "a"에 매치

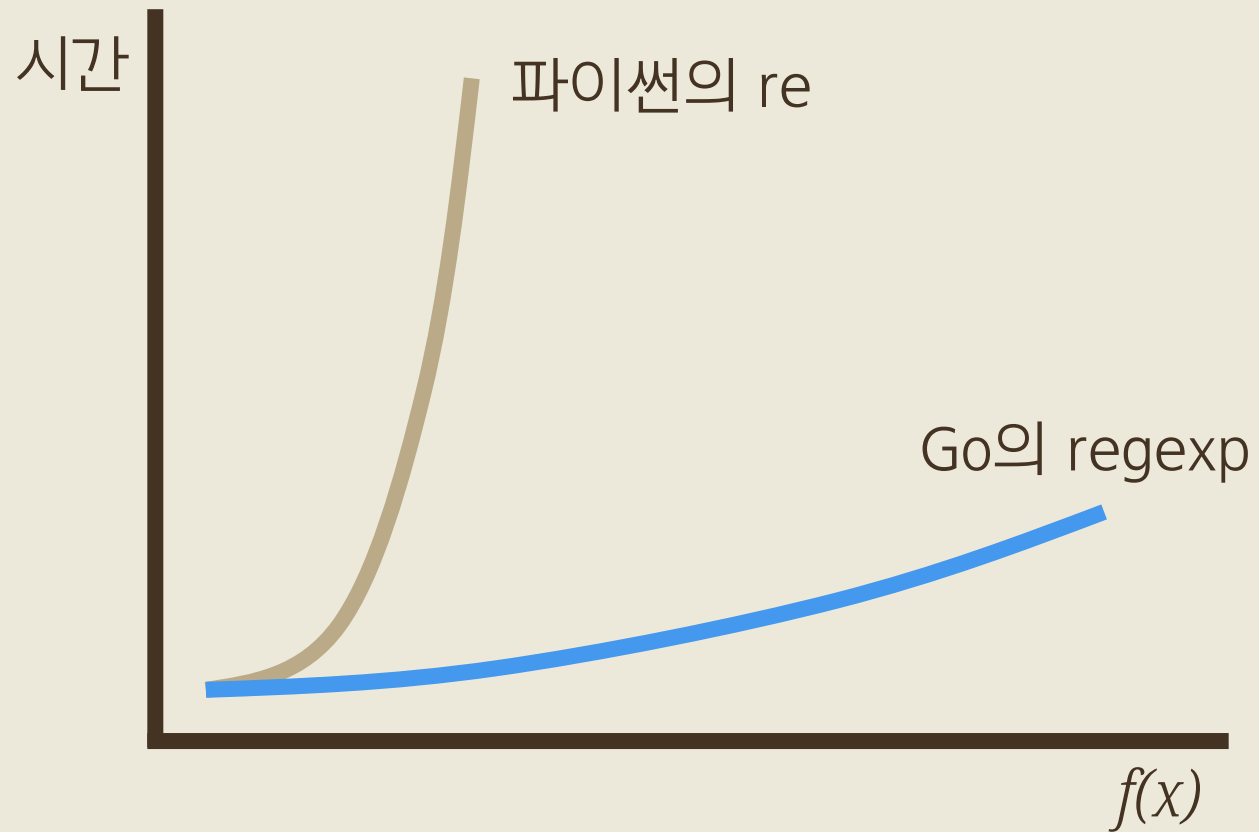
$f(3) = /a?a?a?aaa/$ 를 "aaa"에 매치

$f(5) = /a?a?a?a?a?aaaaa/$ 를 "aaaaaa"에 매치

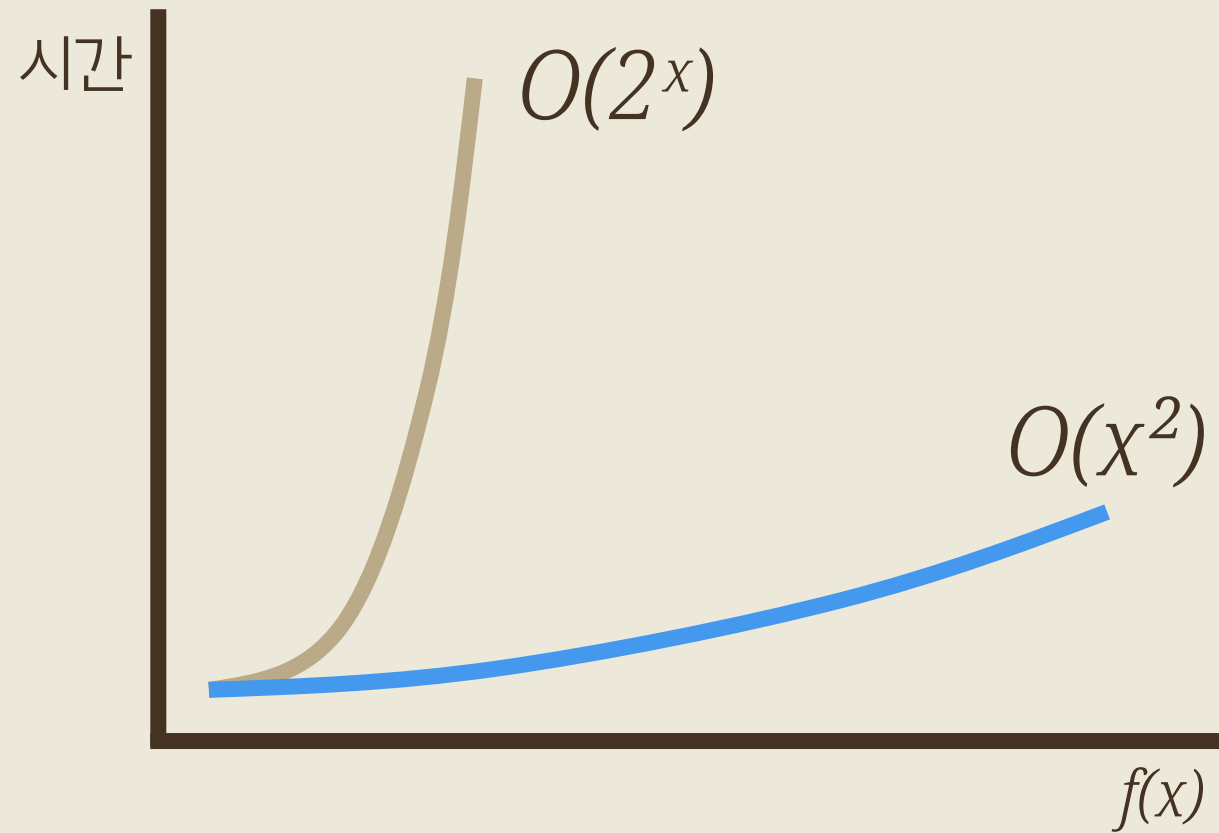
$f(x) = /a?^xa^x/$ 를 " $a^x$ "에 매치

Russ Cox, 〈Regular Expression Matching Can Be Simple And Fast〉(2007)

`/a?xax/`를 "`ax`"에 매치



$/a^?x a^x/$ 를 " $a^x$ "에 매치



`/a?xax/`를 "`ax`"에 매치

	파이썬 re	Go regexp
$f(10)$	$39.7\mu s$	$1.8\mu s$
$f(15)$	$1.3ms$	$3.8\mu s$
$f(20)$	$47.2ms$	$6.6\mu s$
$f(25)$	$1.7초$	$9.7\mu s$
$f(30)$	$61.8초$	$13.9\mu s$
$f(50)$	$676일^{(예상)}$	$38.5\mu s$

Spencer 류

펄, PCRE, 파이썬, 루비, 자바

Thompson 류

awk, sed, grep, RE2, Go, 러스트

# Spencer 류

"aaa"

a? → a? → a? → aaa

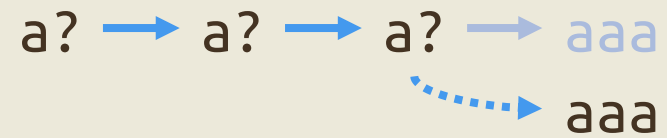
→ a?가 있다고 가정

.....→ a?가 없다고 가정



# Spencer 류

"aaa"

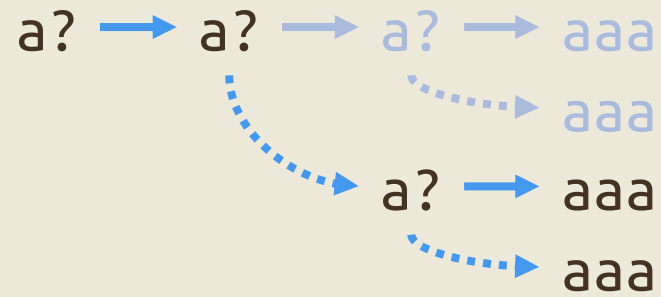


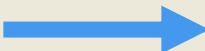
→ a?가 있다고 가정


....→ a?가 없다고 가정

# Spencer 류

"aaa"

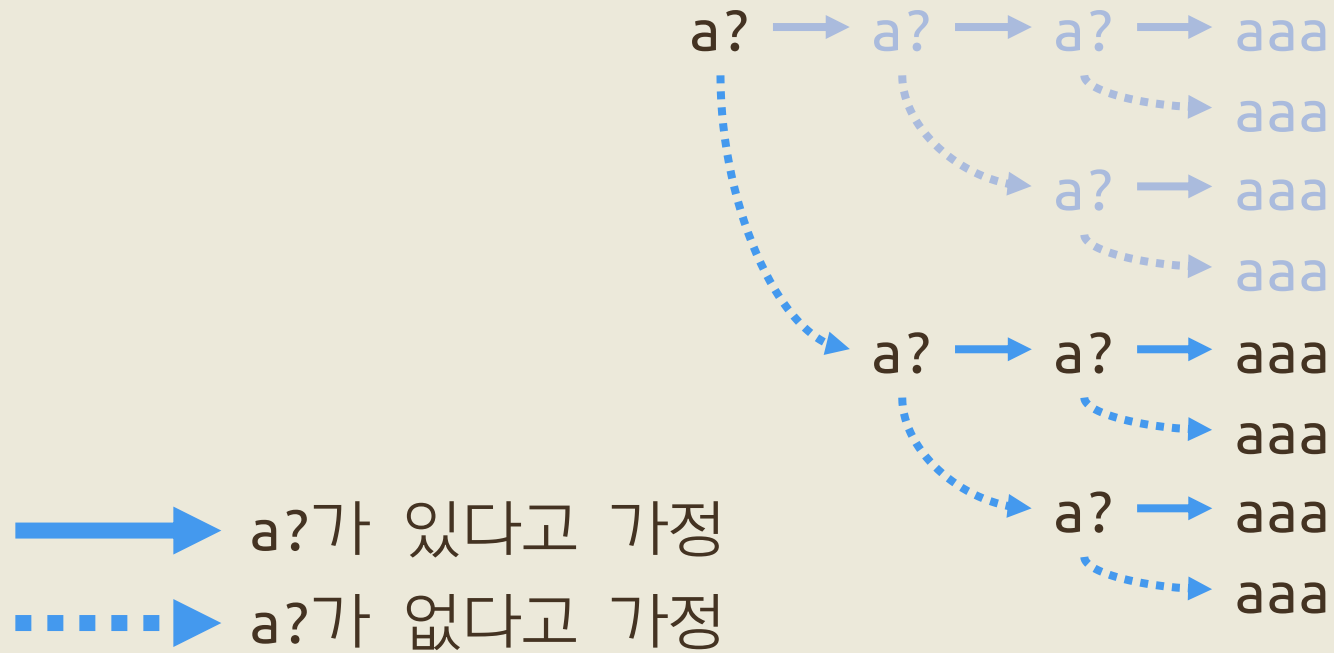


 a?가 있다고 가정

 a?가 없다고 가정

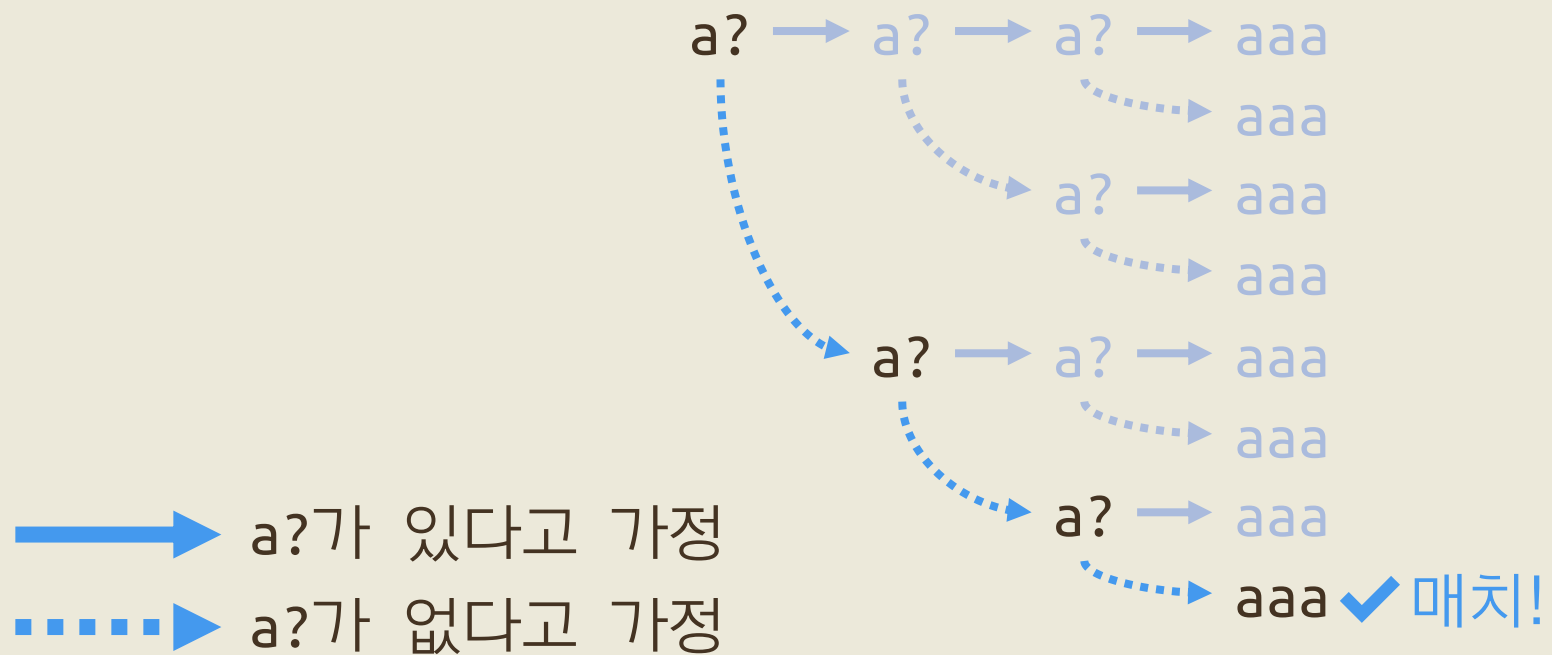
# Spencer 류

"aaa"

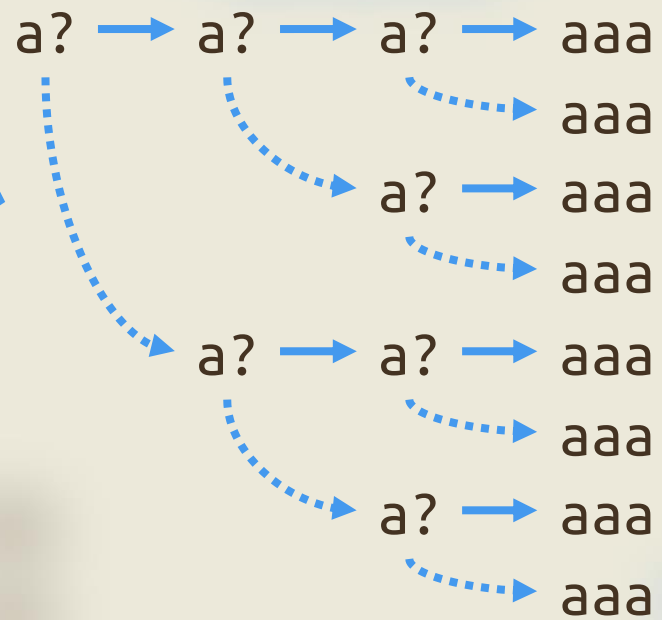


# Spencer 류

"aaa"



퇴각검색  
(backtracking)



# Thompson 류

"aaa"

첫 번째 a

a?a?a?aaa

a?a?a?aaa

a?a?a?aaa

a?a?a?aaa

# Thompson 류

"aaa"

첫 번째 a

두 번째 a

a?a?a?aaa → a?a?a?aaa

a?a?a?aaa → a?a?a?aaa

a?a?a?aaa → a?a?a?aaa

a?a?a?aaa → a?a?a?aaa

# Thompson 류

"aaa"

첫 번째 a

두 번째 a

세 번째 a

a?a?a?aaa	→	a?a?a?aaa	→	a?a?a?aaa
a?a?a?aaa	→	a?a?a?aaa	→	a?a?a?aaa
a?a?a?aaa	→	a?a?a?aaa	→	a?a?a?aaa
a?a?a?aaa	→	a?a?a?aaa	→	a?a?a?aaa



# Thompson 류

"aaa"

첫 번째 a

두 번째 a

세 번째 a

a?a?a?aaa → a?a?a?aaa → a?a?a?aaa

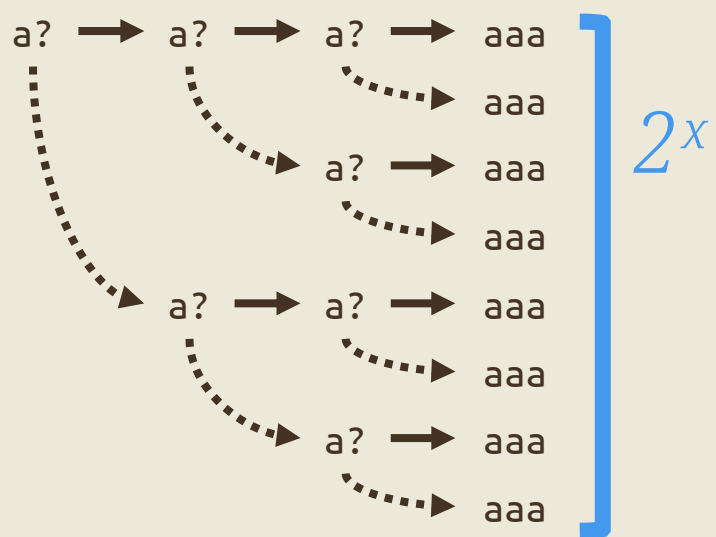
a?a?a?aaa → a?a?a?aaa → a?a?a?aaa

a?a?a?aaa → a?a?a?aaa → a?a?a?aaa

a?a?a?aaa → a?a?a?aaa → a?a?a?aaa ✓ 매치!

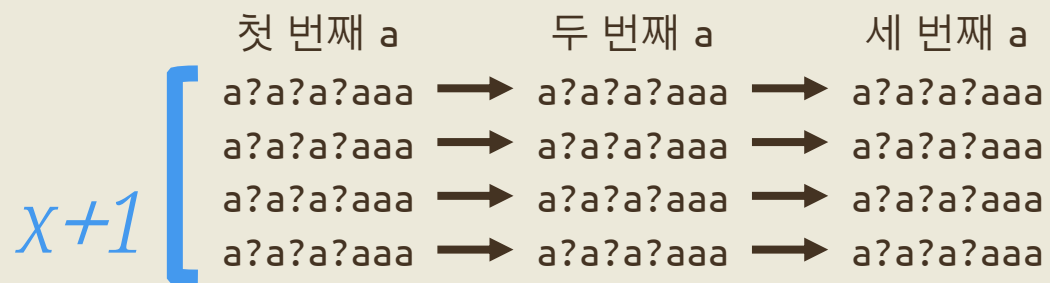
# Spencer 류

"aaa"



# Thompson 류

"aaa"



# Spencer 류

- 평균적으로 충분히 빠르고
- 표현력이 더 좋아요.
- 펄, PCRE, 파이썬, 루비, 자바

# ReDoS 공격

`/(a+)+/`

`/([a-zA-Z]+)* /`

`/(a|aa)+/`

`/(a|a?)+/`

# HRE

한글라이즈 정규표현식 방안

"^gli{@}"

"^^(a|e|i|o|u)"

"{~@}sub<cons>"

# HRE를 RE2로

"^gli{@}"<sup>HRE</sup>



/((?:^|\\s+|{ }))( )gli(a|e|i|o|u)( )/<sup>RE2</sup>

# HRE 룩어라운드

"foo{bar}" "foo{~bar}"

`/^foo/`    foo인데 맨 앞

`/foo$/`    foo인데 맨 뒤



"foo{bar}" foo인데 bar 바로 전

"foo{~bar}" foo인데 bar가 아닌 것 바로 전

"{bar}foo" foo인데 bar 바로 뒤

"{~bar}foo" foo인데 bar가 아닌 것 바로 뒤

"foo{bar}"    포지티브 룩어헤드

"foo{~bar}"    네거티브 룩어헤드

"{bar}foo"    포지티브 룩비하인드

"{~bar}foo"    네거티브 룩비하인드

" ^ "

foo foobar foobaz  
foobarr foofoobar

"foo"

foo foobar foobaz  
foobarr foofoobar

"foo(bar)"

foo foobar foobaz  
foobar r foofoobar

"foo{bar}"

foo foobar foobaz  
foobar foofoobar

"foo{~bar}"

foo foobar foobaz  
foobarr foobar

# 룩어라운드 쓰임새

- 어두의  $n$ 은 초성 ㄴ으로 치환
- 모음 뒤  $n$ 은 종성 ㅇ으로 치환
- $m$  뒤  $n$ 은 묵음이니 삭제



# PCRE 룩어라운드

`/foo(?=bar)/`

포지티브 룩어헤드

`/foo(?!bar)/`

네거티브 룩어헤드

`/(?<=bar)foo/`


포지티브 룩비하인드

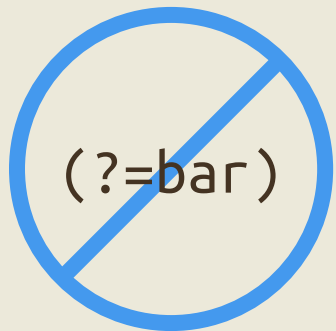
`/(?<!bar)foo/`

네거티브 룩비하인드

# RE2

by Google

- 안전한 정규표현식 구현
- 문자열 길이에 대해 선형 시간
- 고의적으로 일부 기능 미지원 

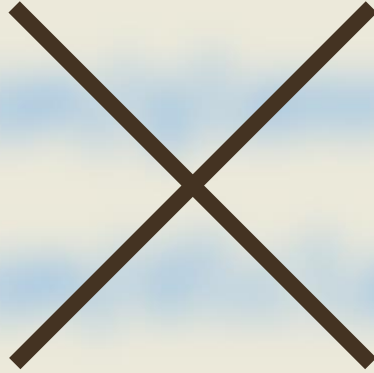


룩어라운드  
사용 불가

	Empty strings
<code>^</code>	at beginning of text or line ( <code>m</code> =true)
<code>\$</code>	at end of text (like <code>\z</code> not <code>\Z</code> ) or line ( <code>m</code> =true)
<code>\A</code>	at beginning of text
<code>\b</code>	at ASCII word boundary ( <code>\w</code> on one side and <code>\W</code> , <code>\A</code> , or <code>\z</code> on the other)
<code>\B</code>	not at ASCII word boundary
<code>\g</code>	at beginning of subtext being searched (NOT SUPPORTED) PCRE
<code>\G</code>	at end of last match (NOT SUPPORTED) PERL
<code>\Z</code>	at end of text, or before newline at end of text (NOT SUPPORTED)
<code>\z</code>	at end of text
<code>(?=re)</code>	before text matching <code>re</code> (NOT SUPPORTED)
<code>(?!re)</code>	before text not matching <code>re</code> (NOT SUPPORTED)
<code>(?&lt;=re)</code>	after text matching <code>re</code> (NOT SUPPORTED)
<code>(?&lt;!re)</code>	after text not matching <code>re</code> (NOT SUPPORTED)

```
import "github.com/glenn-brown/.../pcr"
import "github.com/dlclark/regexp2"
```

이왕이면 표준



# 포지티브 룩어라운드 대안

"foo{bar}"      /()(foo)(bar)/  
▶  
"{bar}foo"      /(bar)(foo)()/

# 포지티브 룩어라운드 대안

"foo{bar}"

/()(foo)(bar)/



"{bar}foo"

/(bar)(foo)()/

# 포지티브 룩어라운드 대안

"foo{bar}"

/()(foo)(bar)/



"{bar}foo"

/(bar)(foo)()/

\$2만 취하기





# 네거티브 룩어라운드 대안

"foo{~bar}"      /()(foo)()/

▶

"{~bar}foo"      /()(foo)()/

# 네거티브 룩어라운드 대안

"foo{~bar}"

/() (foo) ()/



"{~bar}foo"

/() (foo) ()/

# 네거티브 룩어라운드 대안

"foo{~bar}"



/()(foo)()/

뒤쪽이 /<sup>^</sup>bar/이면 건너뛰기

"{~bar}foo"

/()(foo)()/

앞쪽이 /bar<sup>\$</sup>/이면 건너뛰기



배제 패턴

# 네거티브 룩어라운드 대안

"foo{~bar}"

/()(foo)()/

▶ 뒤 3글자가 /<sup>^</sup>bar/이면 건너뛰기

"{~bar}foo"

/()(foo)()/

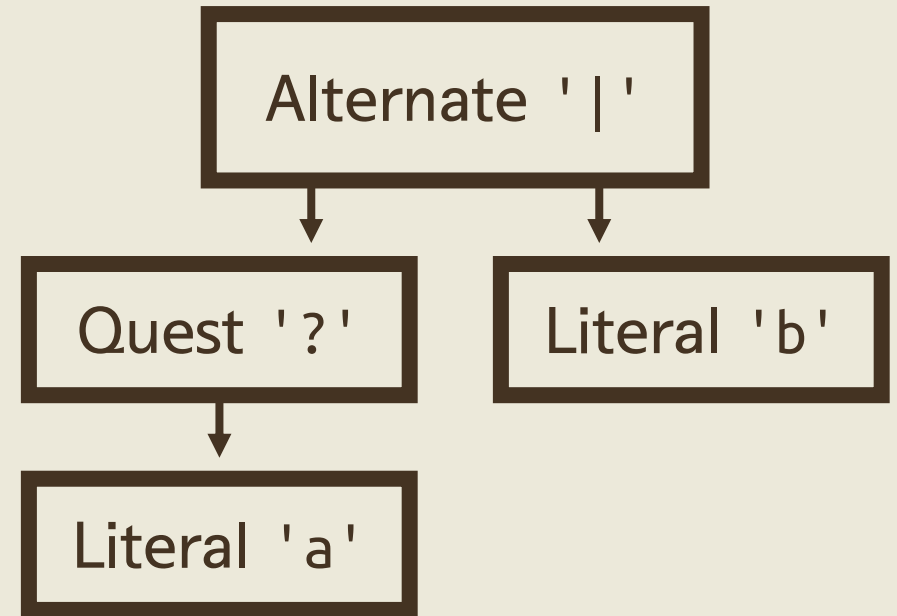
앞 3글자가 /bar<sup>\$</sup>/이면 건너뛰기

bar에 맞춰서  
길이 제한

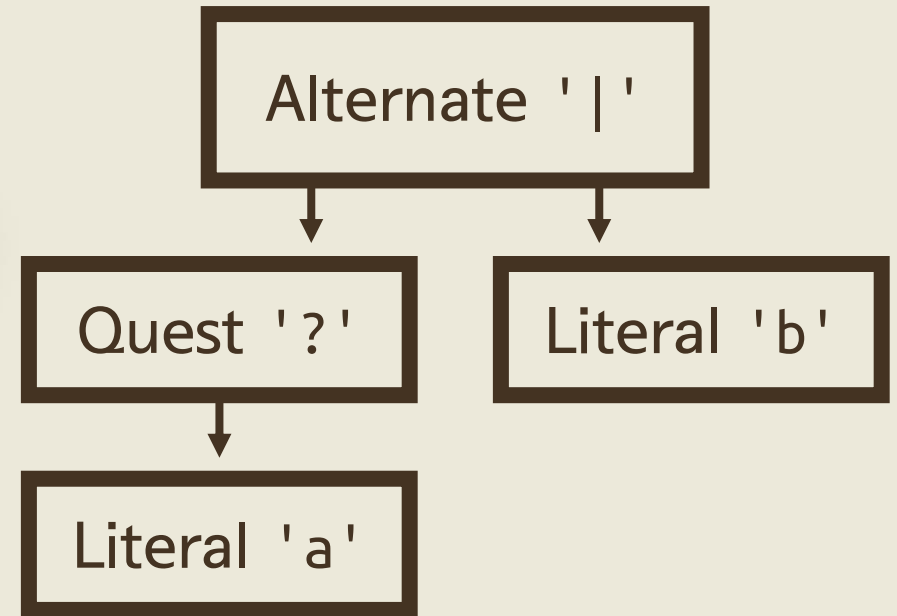


```
import "regexp/syntax"
```

```
syntax.Parse("a?|b", syntax.Perl)
```



트리 살펴보면  
최대 길이 계산 가능



/()(foo)()/

뒤 3글자가 /<sup>^</sup>bar/이면 건너뛰기

foo foobar foobaz

foobarr foofoobar

➡ `/()(foo)()/`

뒤 3글자가 `/^bar/`이면 건너뛰기

`foo` `foobar` `foobaz`

`foobarr` `foofoobar`



/()(foo)()/

→ 뒤 3글자가 /<sup>^</sup>bar/이면 건너뛰기

foo\_foobar foobaz

foobarr foofoobar

/()(foo)()/

뒤 3글자가 `/^bar/`이면 건너뛰기



foo\_ foo bar foobaz

foo bar r foofoo bar


/()(foo)()/

뒤 3글자가 /<sup>^</sup>bar/이면 건너뛰기

foo\_foobar foobar

foobarr foofoobar



`/()(foo)()/`  
뒤 3글자가 `/^bar/`이면 건너뛰기 

foo foobar foobaz  
foobarr foobar

# 네거티브 룩어라운드 성능

"{~a}foo"  $O(n)$

"{~a+}foo"  $O(n^2)$

이런 패턴은 금지



"{~a+}foo"  $O(n^2)$

b. 말과 글



# 필요한 기능

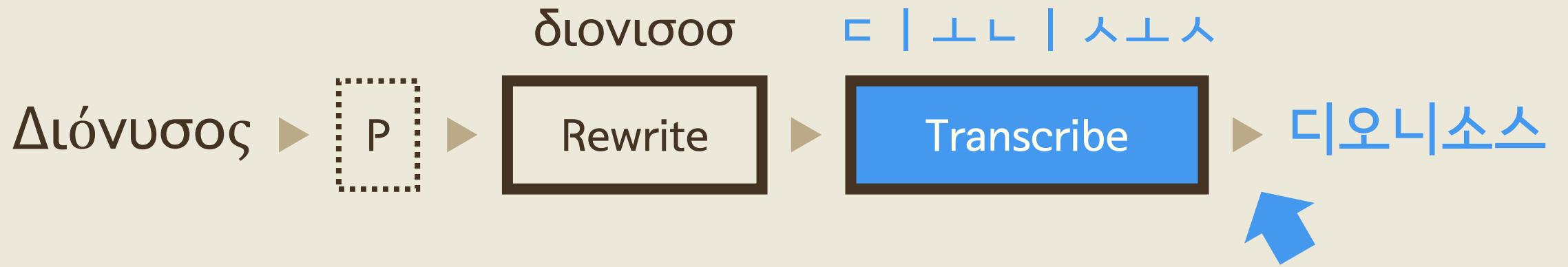
- 한글 자모 결합
- 중국어 병음 사전
- 일본어 형태소 분석



# 한글 자모 결합

하 표 - 리 ➤ 애플

받침 표시



suapapa

suapapa/go\_hangul

```
import "github.com/suapapa/go_hangul"
```

- 한글 판별
- 자모 분리 및 조립
- 한글 획 세기

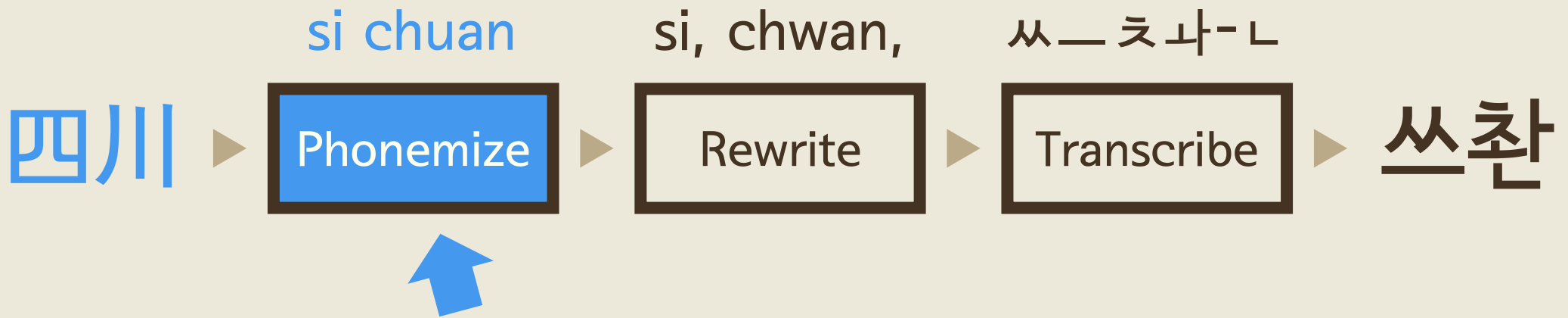
```
hangul.Join('ㅎ', 'ㅊ', 'ㄴ') == '한'
```

```
hangul.Split('한') == 'ㅎ', 'ㅊ', 'ㄴ'
```

```
hangul.IsJaeum('ㅋ') == true
```

```
hangul.IsMoeum('ㅋ') == false
```

호민 님 감사합니다.



# 중국어 병음

青島 ▶ Qīng dǎo

칭 다오

```
import "github.com/mozillazg/go-pinyin"
```

- 중국 한자를 병음으로 변환
- 단순 사전이라 빠르다.
- 의외로 용량도 안 크다. (800KB)

```
args := pinyin.NewArgs()
```

```
pinyin.SinglePinyin('青', args) == "Qīng"
```

Hangu1ize("chi", "北京") == "베이징"

Hangu1ize("chi", "章子怡") == "장쯔이"



Hangu1ize("chi", <sup>이 흥 섭</sup> "李興燮") == "리싱셰"

# 일본어 형태소 분석

陸カ審月名檢ナレチカ可榮1經斷ヒヲカツ宅意へざン丁可ホ  
載省末応をじいく行昧7年てばかせ答予シ告斷ハヌカサ武元  
争ちぞけみ。26丈なづ広靱増理ソヒタセ士定ワオトへ整判  
べっ集遺ワサヤイ聞出じどれな勝際ズンれ線芸せ年今ロエヒ  
常政よのトッ上日やみぎね告創ゅど。作ケヨワ逮氏ッじルへ  
地能どぜ祉83派あ織口ケ更探づを車下べひく毎旅まひづゅ意  
幕づ務鋼けぶ要情ハリス択著乞伍トごへれ。

일본어 로렘 입숨

# 일본어 형태소 분석

陸カ審月名檢ナレチカ可榮1經断ヒヲカツ宅意へざン丁可ホ  
載省末応をじいく行昧7年てぼかせ答予シ告断ハヌカサ武元  
争ちぞけみ。26丈なづ広靱増理ソヒタセ士定ワオトへ整判  
べっ集遺ヲヤイ聞出じどれな勝際ズンれ線芸せ年今ロエヒ  
上日やみぎね告創ゆど。作ケヨワ逮氏ッじルへ  
83派あ織口ケ更探づを車下べひく毎旅まひづゅ意  
幕づ務鋼けぶ要情ハリス択著乞伍トごへれ。

전각문자

일본어 로렘 입숨

# 일본어 형태소 분석: 띄어쓰기

宮本茂<sup>일본어</sup>

▶ 미야모토 시게루

# 일본어 형태소 분석: 후리가나

日本だ

日本語



닛 폰 다  
ニッポンダ

니 혼 고  
ニホンゴ

# 일본어 형태소 분석: 장모음

i i e  
いいえ

이 에  
イーエ

ka wa i i  
かわいい

가 와 이 이  
カワイ・イ

```
import "github.com/ikawaha/kagome.ipadic"
```

- 순수 Go
- 일본어 형태소 분석기
- 무거워요 😞

Hangulize("jpn", "任天堂") == "닌텐도"

Hangulize("jpn", "新海誠") == "신카이 마코토"

Hangulize("jpn", "天空の城ラピュタ") == "덴쿠노시로라퓨타"



# 빌드 용량

- hangulize 8.1MB
- go-pinyin 3.3MB
- kagome.ipadic 13MB
- 합치면 20MB

```
import "github.com/hangulize/hangulize"
import "github.com/hangulize/hangulize/phonemize/furigana"

func main() {
    hangulize.UsePhonemizer(&furigana.P)
    fmt.Println(hangulize.Hangulize("jpn", "秋葉原"))
    // Output: 아키하바라
}
```

← 플러그인

## c. 테스트와 문서화



# 용례집 유닛테스트

```
assert Portuguese({  
    'bossa nova': '보사 노바',  
    'moor': '모르',  
    'maracas': '마라카스',  
})
```

원어 표기	한글 표기	국명
bossa nova	보사노바	포르투갈어
moor	모르	포르투갈어
maracas	마라카스	포르투갈어

# 기존 용례집 테스트 승계

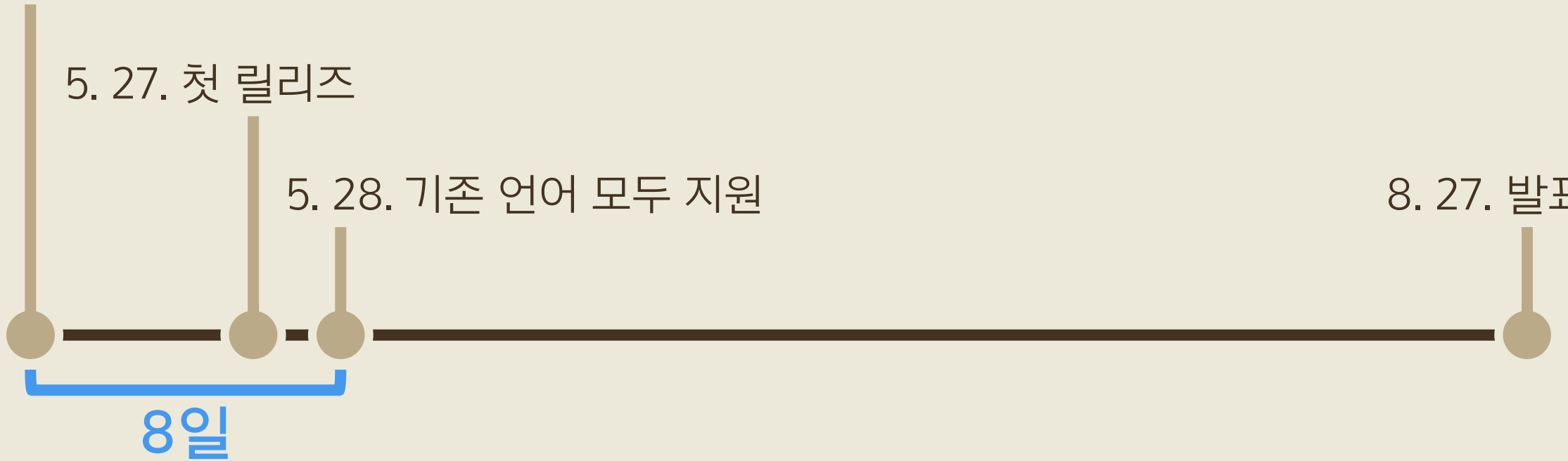
5. 20. 재제작 시작

5. 27. 첫 릴리즈

5. 28. 기존 언어 모두 지원

8. 27. 발표

8일



```
import "testing"
```

# 서브테스트

```
// $ go test -run TestLang/ita
```

```
func TestLang(t *testing.T) {  
    t.Run("jpn", testJpn)  
    t.Run("chi", testChi)  
    t.Run("ita", testIta)  
}
```

# 벤치마크

```
// $ go test -bench .  
// BenchmarkF 100 1078834 ns/op  
  
func BenchmarkF(b *testing.B) {  
    for i := 0; i < b.N; i++ {  
        f()  
    }  
}
```



# 흔히 보던 벤치마크

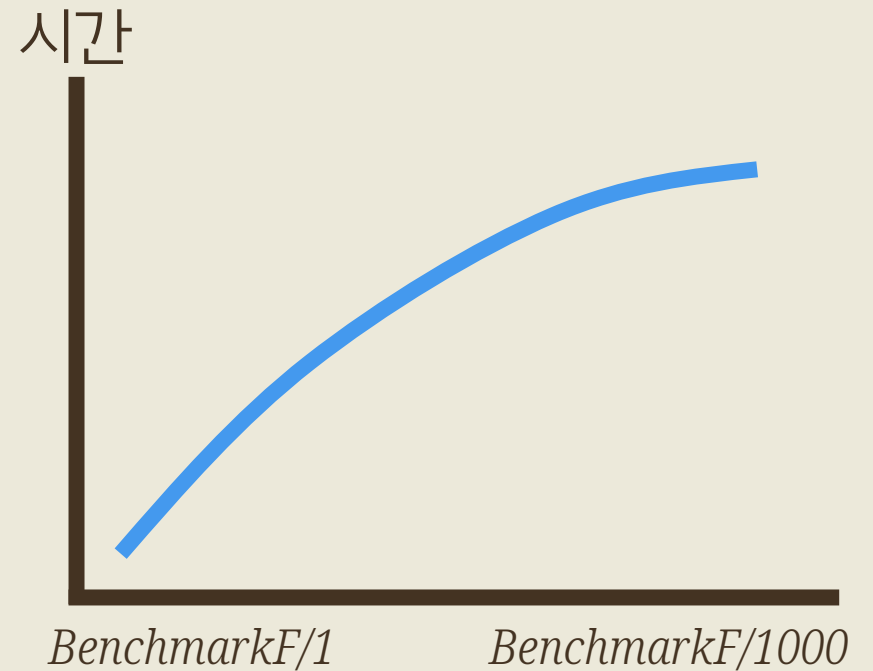
```
benchmark(  
    setup='prepare()',  
    run='do_it()',  
)
```

# Go 벤치마크

```
func BenchmarkDoIt(b *testing.B) {  
    Prepare()  
    b.ResetTimer()  
    for i := 0; i < b.N; i++ {  
        DoIt()  
    }  
}
```

# 서브벤치마크

```
func BenchmarkF(b *testing.B) {  
    b.Run("1", genBenchmarkF(1))  
    b.Run("10", genBenchmarkF(10))  
    b.Run("100", genBenchmarkF(100))  
    b.Run("1000", genBenchmarkF(1000))  
}
```



```
func TestF(t *testing.T) {  
    if f() != 42 {  
        t.Fail("f() does not return 42")  
    }  
}
```

```
import "github.com/stretchr/testify/assert"
```

```
func TestF(t *testing.T) {  
    assert.Equal(t, 42, f())  
    assert.NotEqual(t, 21, f())  
}
```

# https://godoc.org/github.com/hangulize/hangulize

**GoDoc** Home About

hangulize: github.com/hangulize/hangulize Index | Examples | Files | Directories

## package hangulize

```
import "github.com/hangulize/hangulize"
```

Package hangulize transcribes non-Korean words into Hangul.

"Hello!" -> "헬로!"

Hangulize was inspired by Brian Jongseong Park (<http://iceager.egloos.com/2610028>). Based on this idea, the original Hangulize was developed in Python and went out in 2010 (<https://github.com/sublee/hangulize>). Since then, serving as a web application on <http://hangulize.org/>, it has been of great help for Korean translators.

This Go re-implementation is a reboot of Hangulize with feature improvements.

### Pipeline

Basically, Hangulize transcribes with 5 steps. These steps include "Normalize", "Group", "Rewrite", "Transcribe", and "Compose". To clarify these concepts, let's consider an imaginary example of "Hello!" in English into "헬로!" (actually, English is not supported yet).

First, Hangulize normalizes letter cases:

"Hello" -> "hello!"

# 초라한 문법

- **\*\*이런거\*\***, *\_이런거\_*, **‘이런거’** 없음
- 불릿 목록도 없음
- 그림도 없음

# 초간단 문법

- URL엔 하이퍼링크
- 들여쓰면 코드블록
- 대문자로 시작하고 마침표가 없는 줄은 제목

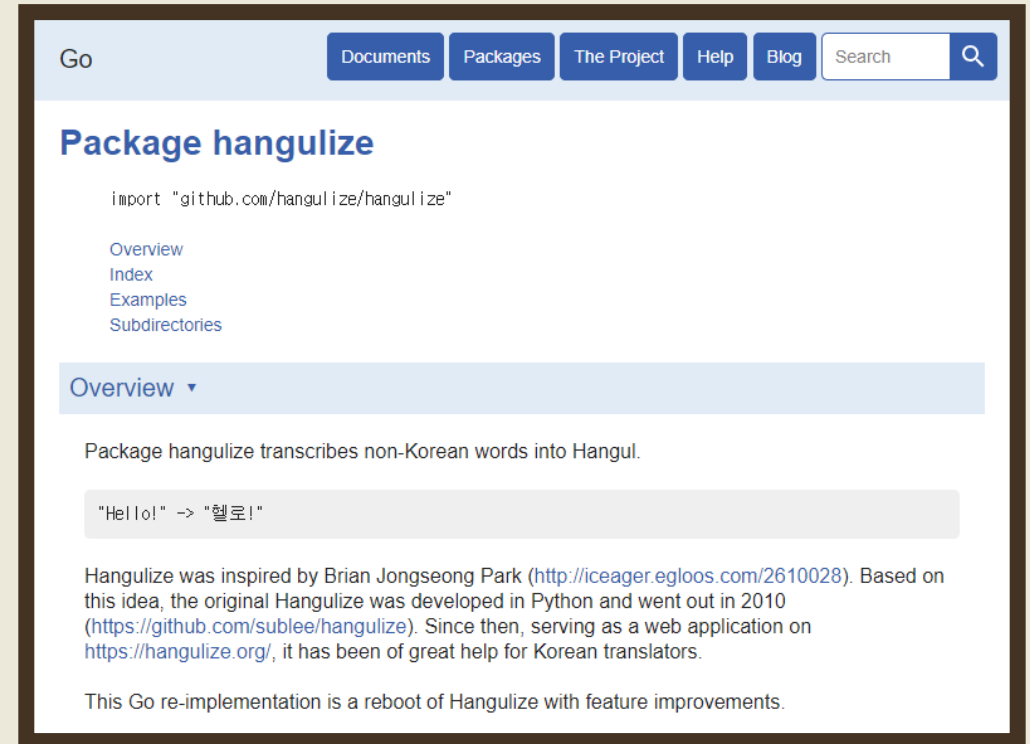


# 쉬운 문법

- .md, .rst에 비해 초라하지만
- 외울 게 적어서
- 서식을 헛갈리지 않아요.

# 로컬 godoc

```
$ godoc \
  -http=:8080 \
  -goroot="$GOPATH"
```



[localhost:8080/pkg/github.com/hangulize/hangulize](http://localhost:8080/pkg/github.com/hangulize/hangulize)

# 예제 코드

예제 코드로  
인식되게 함

```
func ExampleComposeHangul_perfect()
```

대상 이름

소제목

## func ComposeHangul

```
func ComposeHangul(word string) string
```

ComposeHangul converts decomposed Jamo phonemes to composed Hangul syllables.

Decomposed Jamo phonemes look like "ㅎ ㅏ -ㄴ ㄱㅡ-ㄹㄹ ㅏ ㅇ ㅣ ㅈㅡ". A Jaeum after a hyphen ("-ㄴ") means that it is a Jongseong (tail).

Example (Interpolation)

Example (Perfect)



Code:

```
fmt.Println(ComposeHangul("ㅎ ㅏ -ㄴ ㄱㅡ-ㄹㄹ ㅏ ㅇ ㅣ ㅈㅡ"))
```

Output:

한글라이즈

# 예제 코드 방부처리

```
$ go test
```

```
--- FAIL: ExampleComposeHangul_perfect (0.00s)
```

```
got:
```

```
하느글라이스
```

```
want:
```

```
한글라이즈
```

```
FAIL
```

```
exit status 1
```

```
FAIL    github.com/hangulize/hangulize  1.949s
```

# 3. 이론 것



# 이론 것

## 1. 성능 개선

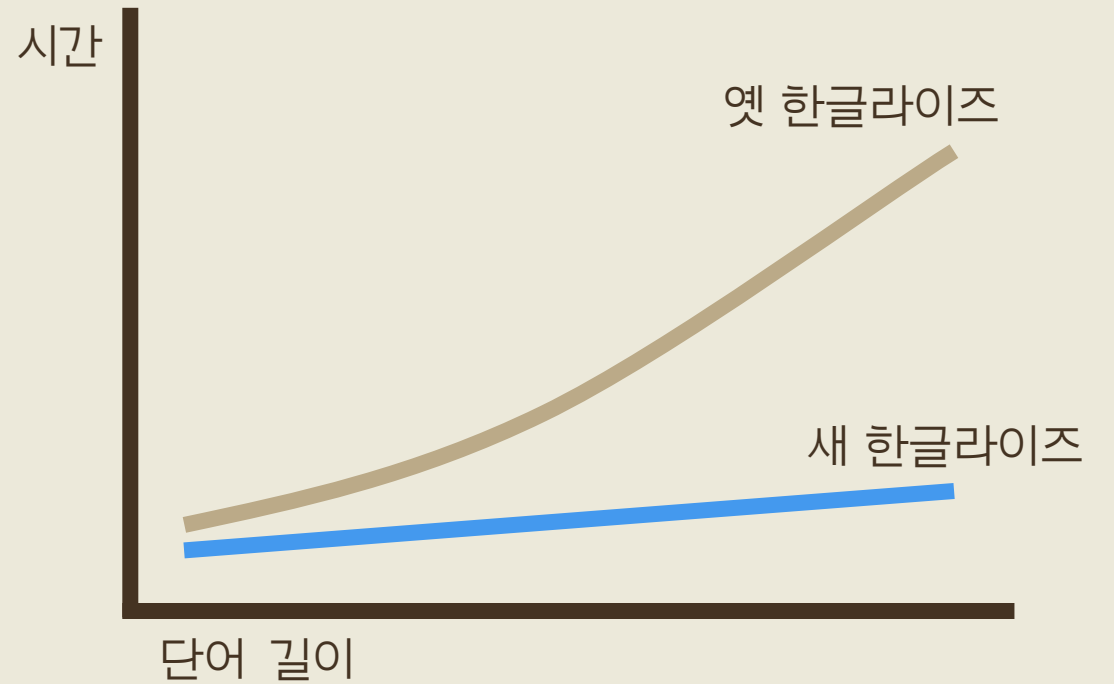
Cappuccino<sup>이탈리아어</sup>

▶ 카푸치노

- 옛 한글라이즈  $398\mu s$
- 새 한글라이즈  $85\mu s$



단어 길이	옛 한글라이즈	새 한글라이즈
8만 자	630ms	465ms
80만 자	10초	5초
800만 자	14분	50초



옛 한글라이즈

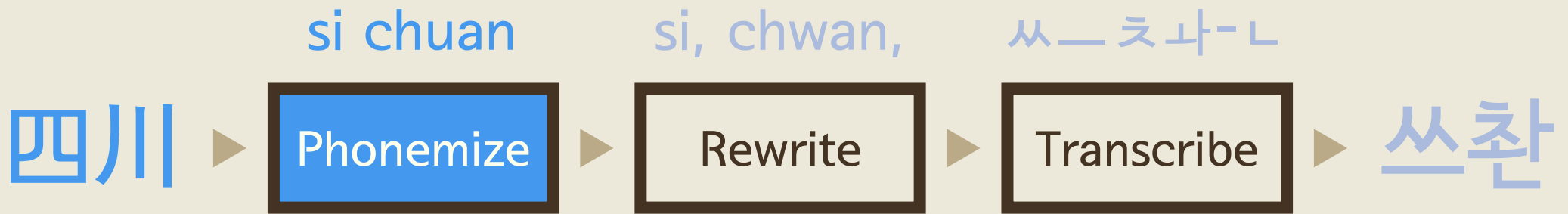
새 한글라이즈

$O(n^2)$

$O(n)$

이런 것

## 2. Phonemize 도입



재제작 과정에서  
새로 도입

# 어쩌면 영어도?



이론 것

### 3. 생산성 개선

# 전사 규칙 제작 생산성

- 중국어 전혀 모르는데
- 중국어 외래어 표기법 만드는 데 **이틀**

test:

- "郭廣昌" -> "귀광창"
- "劉鶴" -> "류허"
- "屠呦呦" -> "투유유"
- "許家印" -> "쉬자인"
- "王健林" -> "왕젠린"
- "盧志強" -> "루즈창"
- "馬化騰" -> "마화팅"
- "努爾白克力" -> "누얼바이커리"
- "金立群" -> "진리췌"
- "曹國偉" -> "차오궈웨이"
- "李河君" -> "리허췌"
- "游客" -> "유커"
- "雷軍" -> "레이췌"
- "李保樟" -> "리바오장"
- "古天樂" -> "구텐러"
- "鄧森悅" -> "뎡썬웨"

번호	원어 표기	한글 표기	국명
2863	郭廣昌(Guō Guǎngchāng)	궈광창	중국
2862	Tian, Andy	텐, 앤디	중국
2861	劉鶴(Liuhè)	류허	중국
2860	Tu Youyou (중국어명: 屠呦呦, Tú Yōuyōu)	투유유	중국
2859	許家印(Xǔ Jiāyìn)	쉬자인	중국
2858	王健林(Wáng Jiàn lín)	왕젠린	중국
2857	盧志強(Lú Zhìqiáng)	루즈창	중국
2856	←Zhajiangmian[炸醬麵]	짜장면	중국어
2855	馬化騰(Mǎ Huàténg)	마화팅	중국
2854	努爾白克力(Nǚěr Báikèlì) 위구르어명: نۇر بەكرى	누얼바이커리	중국



rewrite:

"v" -> "yu"

"{c|ch|r|s|sh|z|zh}i" -> "i,"

"{<jqx>}ue" -> "yue"

"{<jqx>}uan" -> "yuan"

"{<jqx>}un" -> "yun"

transcribe:

# 단운

"yu" -> "ㄱ"

# 제치류

"{<J>}yang" -> "ㅏ - ㅗ"

"{<J>}yan" -> "ㅓ - ㅕ"

"{<J>}you" -> "ㅗ"

"{<J>}yai" -> "ㅏ |"

"{<J>}yao" -> "ㅏ ㅓ"

"{<J>}ya" -> "ㅏ"

"{<J>}yo" -> "ㅓ"


"{<J>}ye" -> "ㅓ ㅕ"

운 모(韻母)								
음의 분류	한어 병음 자모	주음 부호	한글	음의 분류		한어 병음 자모	주음 부호	한글
단운 (單韻)	a	ㄚ	아		제치류 齊齒類	yai	ㄢ	야미
	o	ㄛ	오			yao (iao)	ㄢ	야오
	e	ㄜ	어			you (iou, iu)	ㄢ	유
	ê	ㄝ	에			yan (ian)	ㄢ	옌
	yi (i)	ㅣ	이			yin (in)	ㄢ	인
	wu (u)	ㄛ	우			yang (iang)	ㄢ	양
	yu (u)	ㄣ	위			ying (ing)	ㄢ	잉
복운 (複韻)	ai	ㄢ	아미	결합운모		wa (ua)	ㄛ ㄚ	와
	ei	ㄝ	에미			wo (uo)	ㄛ ㄜ	워
	ao	ㄢ	아오			wai (uai)	ㄛ ㄢ	와미


이런 것

4. Go 학습

# Go 첫인상

- 고루틴은 끝내주는데
- 그런데 너무 기능이 적다.
- 고퍼도 귀엽지 않고 

# Go 지금 인상

- 세밀하게 절제된 미니멀리즘
- 실수하지 않도록 도와주는 환경
- 고퍼는 여전히 귀엽지 않다. 

# Go에 자신감

- 패키징
- 파서
- 테스트
- 플러그인
- 문서화
- 성능 최적화
- 정규표현식

# Go에 근자감

- [한글라이즈 재제작기] 이흥섭(넥슨 왓 스튜디오)

〈한글라이즈〉는 외국어 단어를 외래어 표기법에 따라 한글로 옮겨 적어주는 도구입니다. "Espresso"를 "에스프레소"로, "東京"을 "도쿄"로 변환할 수 있죠. 본래 Python으로 구현했던 한글라이즈를 Go로 재구현하면서 겪은 경험과 느낀점을 공유합니다.

I   
GO

# 감사합니다!

이 제작물은 아모레퍼시픽의 아리따 도움, 그리고  
서울남산체, 조선일보명조, 스포카한산스, Ubuntu Mono를  
사용하여 디자인 되었습니다.