

**Notion**

페이지 링크  
제공!

# 정규 표현식

**Python**으로 훑아보기

이호준 김혜원 김유진 차경림 김진 현지연 정승한



# 정규표현식

## 0. 강의소개

### 1. 정규표현식 sample text

### 2. 환경설정 및 팁

### 3. (실전) 일반 문자열

### 4. (실전) 처음과 끝(Anchors)

### 5. (실전) 모든 문자

### 6. (실전) 탭1

### 7. (실전) 범위

### 8. (실전) 부정

### 9. (실전) 서브패턴

### 10. (실전) 수량자

### 11. (실전) 캐릭터 클래스

### 12. (실전) 이스케이프 문자

### 13. 언어별 실습

#### 13.1 환경설정





#### 13.2 환경별 간단한 실습

#### 13.3 python 메서드 설명 및 실습

### 14. 연습문제

### 15. 주피터 노트북

## 0. 강의소개

-  001. 강의 소개
-  002. 저자 소개
-  003. 영상 강의 소개
-  004. 강의 로드맵



## 001. 강의 소개

### 강의 소개

정규표현식, 언젠가는 공부하겠다고 미뤄두셨나요? 아니면 매번 공부하다 포기하셨나요? 별도 프로그램 설치 없이 웹 환경에서 빠르게 정규표현식을 익혀봅시다. 그리고 Python을 통해 정규표현식을 어떻게 사용할 수 있는지 다양한 문제를 통해 알아봅시다.

정규표현식, 포기하지 말고, 미루지 말고 지금 해보는 것은 어떨까요? 정규표현식을 함께 해독해봅시다.



## 002. 저자 소개

### 이호준

주식회사 유니브와 바울랩이라는 회사를 이끌고 있어요. 청소년에게는 ICT 관련 진로와 직업을 찾을 수 있도록, 청년에게는 ICT 업을 찾아 주는 일을 하고 있습니다.

### 김혜원, 김유진, 차경림

주식회사 유니브의 COO, CTO, CDO입니다.

### 김진, 현지연

주식회사 유니브의 연구원입니다.

### 정승한

주식회사 유니브의 Back-end 개발자입니다.



## 003. 영상 강의 소개



영상 강의는 인프런에서 보실 수 있습니다.

1. 강의는 책에 실린 내용과 좀 더 상세한 설명을 담고 있습니다.
2. 제코베 포트폴리오 템플릿과 해당 책을 인쇄해 볼 수 있는 PDF File을 제공합니다.
3. 함께 보실 수 있는 각종 부록 영상을 포함합니다.

## 1. 정규표현식 sample text

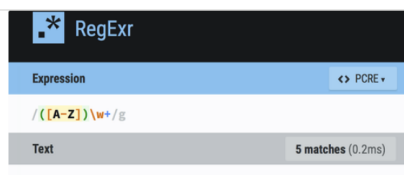
1. google에서 정규표현식 연습사이트 검색해주세요.
2. 2가지 사이트를 추천받을 수 있습니다. (regex101.com과 regexr.com)
3. 그 중 regexr.com에서 실습을 하도록 하겠습니다.

미리 실습환경을 조성해 두었습니다. 해당 샘플은 아래 사이트(<https://regexr.com/5nvc2>)에서 확인 가능합니다.

Regexr: Learn, Build, & Test RegEx

Regular expression tester with syntax highlighting, PHP / PCRE & JS Support, contextual help, cheat sheet, reference, and searchable community patterns.

 <https://regexr.com/5nvc2>



```
hello world
hello world
hello, world
Hello World
```

```
hello world hello
```

```
hello
hallo
hollo
heallo
yellow
```

```
Monday Tuesday Wednesday Thursday Friday Saturday Sunday
```

```
abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ01234567891011121314151617181920
aaabbcaaabbcabcabc
aaa bb c aaa bb c a b c a b c
aaa1bb2c3aaa4bb5c6
```

```
[123456]
123[456]789
abc[def]ghij
```

```
010-9091-5491
010-5043-2901
010-5050-40409
010-49492-3131
010 2913 3132
01019133829
064-721-3213
```

```
paul-korea@naver.com
paul@naver.com
leehojun@gmail.com
hojun.lee@gmail.com
```

```
https://github.com/LiveCoronaDetector/livcod
github.com/LiveCoronaDetector/livcod
https://github.com/LiveCoronaDetector
```

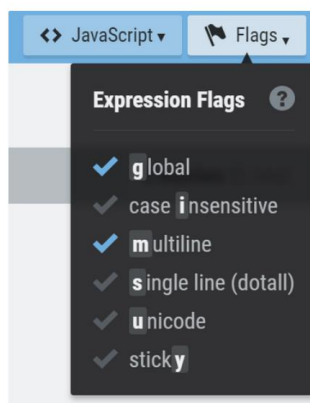
```
I never dreamed about success, I worked for it.
Do not be afraid to give up the good to go for the great.
```

```
hello (hello world) hello
hello \\hello world// hello
^^
:)
```

```
[(name, leehojun), (age, 10), (height, 180), (email, paul-lab@naver.com)]
{name : leehojun, age : 10, height : 180, email : paul-lab@naver.com}
```

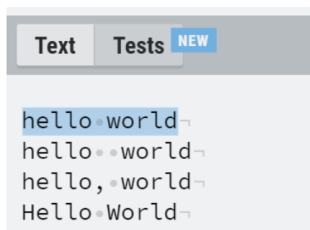
## 2. 환경설정 및 팁

정규표현식을 입력하는 공간입니다. 뒤에 **gm** 이라 붙어있는 것은 **flag** 입니다. **flag** 설정은 오른쪽 상단에서 할 수 있습니다. **g** 는 **global** 로 모든 문자열에서, **m** 은 **multiline** 으로 여러 라인에서 패턴을 찾겠다는 옵션입니다.

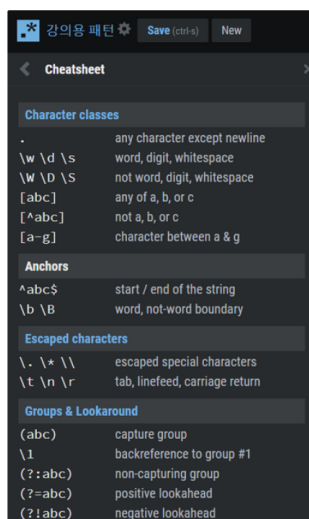
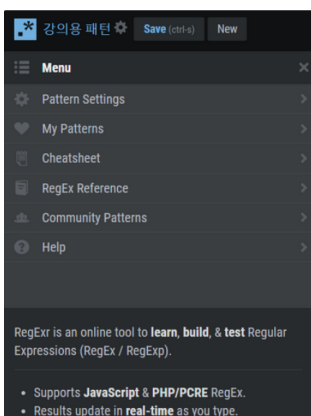




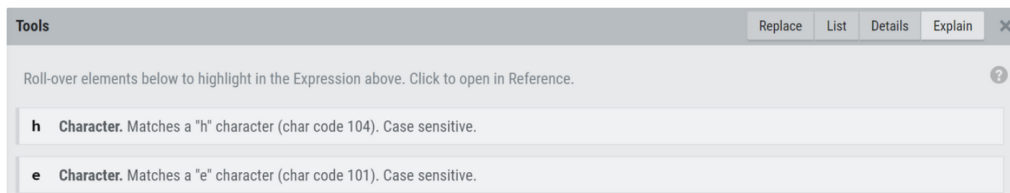
테스트 문자열이 있는 공간입니다. 회원가입을 하시면 여러분만에 별도 테스트 문자열을 만드실 수 있습니다.



메뉴에서는 Cheatsheet 등 다양한 팁을 확인 할 수 있습니다.



어떤 패턴에 매칭이 되는지 상세 설명을 하고 있는 공간입니다.



### 3. (실전) 일반 문자열

아래와 같이 입력해주세요. 정규표현식에서는 대소문자 구분을 하지만 `gim` flag를 주면 대소문자 구분을 하지 않습니다.

```
/hello/gm
/hello world/gm
/Hello/gm
/Hello/gim
```

### 4. (실전) 처음과 끝(Anchors)

문자열의 처음에 나오는 hello, 끝에 나오는 hello를 찾습니다.

```
/^hello/gm : 처음에 hello
/hello$/gm : 끝에 hello
```

### 5. (실전) 모든 문자

`.` (dot)은 모든 문자가 매칭됩니다.

```
./gm : 모든 문자열(*과 같은 역할)
/...../gm : 모든 6개의 문자열
```

👉 만약 `.` 자체의 문자를 사용하고 싶다면 이스케이프 문자를 사용해야 합니다. 이스케이프 문자는 엔터 위에 있어요. `\` 역슬러쉬라고도 부릅니다.

## 6. (실전) 택1

대괄호를 사용하면 대괄호 안에 있는 텍스트 중 택 1합니다. 아래의 경우 hello, hallo, hyllo를 모두 매칭합니다.

```
/h[eay]llo/gm : 대괄호 안에 문자는 문자 1개에 해당!  
/h[ea]l../gm : 총 5개의 문자
```

## 7. (실전) 범위

범위를 지정하여 매칭하고 싶을 때에는 `-` (대쉬)를 사용합니다.

```
/h[a-f]llo/gm  
/[a-zA-Z0-9]/gm : 모든 알파벳과 숫자를 찾음  
/[^a-zA-Z0-9]/gm : 나머지 문자열을 찾음
```

## 8. (실전) 부정

해당 문자열을 제외하고 찾고 싶을 때에는 부정을(`^`) 사용합니다.

```
/h[^ae]llo/gm : 대괄호 안에 있다면 not에 의미
```

## 9. (실전) 서브패턴

서브 패턴은 그룹으로 묶을 수 있습니다. 특히 실무에서 많이 사용됩니다.

### 그룹핑 규칙

Aa 이름	≡ 사용법
(?:abc)	그룹을 사용하지 않음
(?=abc)	(Positive 매칭) 그룹으로 설정(Lookahead)
(?!abc)	(Negative 매칭) 그룹으로 설정(Lookbehind)

```

/(on|ues|rida)/gm : 그룹 1로 3개 중 매칭되는 패턴 찾을
/(?:on|ues)/gm
/(on|ues)|(rida)/gm : 그룹1(on|ues)과 그룹2(rida)로 각각 매칭되는 패턴 찾을
/.(a|e|o)ll./gm
/hello (?!world)/gm : hello 뒤에 world가 오지 않는 것
/hello (?!world)/gm : hello 뒤에 world가 오는 것

```

## 10. (실전) 수량자

수량자는 해당 문자가 몇 개있는지를 명시하여 패턴을 찾는 방식입니다. `*`, `+`, `?`, `{}`를 사용합니다.

```

_* : 앞에 있는 문자가 0개 ~ N개
_+ : 앞에 있는 문자가 1개 ~ N개
_? : 앞에 있는 문자가 0개 ~ 1개

{3} : 3개
{3,} : 3개 이상
{1,3} : 1개 ~ 3개

_* : 앞에 있는 문자가 0개 ~ N개 ({0,})
_+ : 앞에 있는 문자가 1개 ~ N개 ({1,})
_? : 앞에 있는 문자가 0개 ~ 1개 ({0,1})

/[0-9]{3}[-.*][0-9]{4}[-.*][0-9]{4}/gm
/[0-9a-zA-Z]{2,3}[-.*][0-9]{3,4}[-.*][0-9]{4}/gm
/[0-9a-zA-Z]+@[0-9a-zA-Z]+\.[a-zA-Z]+/gm

```

## 11. (실전) 캐릭터 클래스

모든 문자나 숫자 등 자주 사용되는 문자 패턴을 캐릭터 클래스로 제공합니다.

```

/\w/gm : 워드
/\w{5}/gm : 5개의 글자와 스페이스 하나
/\/gm : not 워드
/\/gm : 숫자
/\/gm : not 숫자
/\/gm : 스페이스
/\/gm : not 스페이스

```

## 12. (실전) 이스케이프 문자

백슬러쉬를 사용하여 이미 사용되고 있는 특수 문자를 표현할 때 사용합니다.

```

\[.*\]/gm : 대괄호([ ]) 안에 감싸여진 문자열
/\(.*\)/gm : 소괄호 안에 감싸여진 문자열
/\\.*\//gm : 이미 사용되고 있는 특수문자로 감싸여진 문자열
/-.*/gm : 이스케이프 문자를 사용할 필요가 없는 경우
/\^*/gm : 이스케이프 문자가 필요한 경우
/:\)/gm : 이스케이프 문자가 필요한 경우

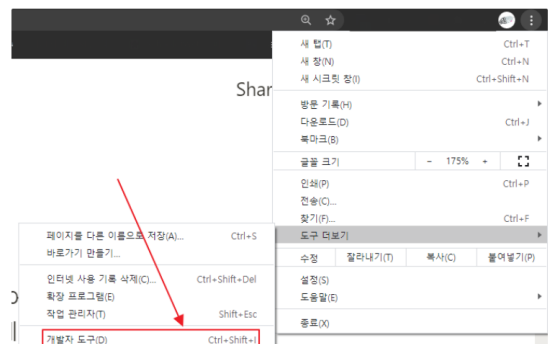
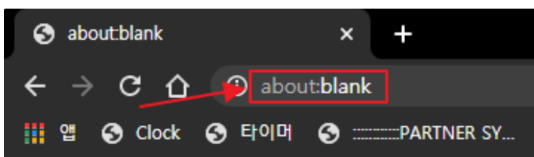
```

## 13. 언어별 실습

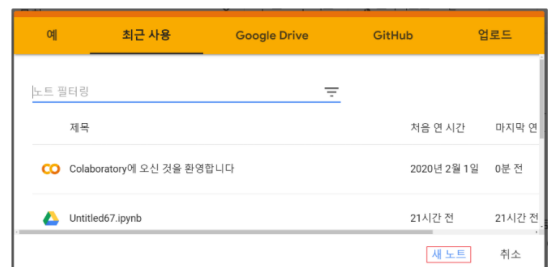
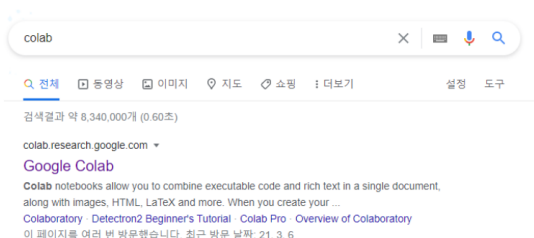
주 실습은 Python으로 할 예정이지만, Javascript와 Python을 실습 할 수 있도록 준비하였습니다. 따로 설치 없이 실습만 해볼 수 있도록 javascript는 크롬 개발자 도구를, python은 colab을 이용하도록 하였습니다.

### 13.1 환경설정

- javascript로 실습을 해보실 경우 크롬에 접속하여 `about:blank` 을 url에 입력하시고 개발자 도구를 열어주세요. window의 경우에는 `ctrl + shift + i` , mac의 경우에는 `cmd + option + i` 입니다. 메뉴에서 개발자 도구를 클릭하셔서 열어도 실행 가능합니다.



- python을 사용하실 경우 google에서 colab을 접속하신 다음 새 노트를 클릭해주세요.



## 13.2 환경별 간단한 실습

- javascript를 사용하시는 분들은 console 창에서 아래와 같이 실습하세요.

```
> let regex = /[a-f]/g;
< undefined

> let s = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ01234567891011121314151617181920';
< undefined

> s.match(regex);
< ▶ (6) ["a", "b", "c", "d", "e", "f"]
```

- python을 사용하시는 분은 아래와 같이 colab에서 실습하세요.

```
[1] import re

regex = re.compile('[a-f]')
s = 'abcdefghijklmnopqrstuvwxyzABCDEFGHIJKLMNOPQRSTUVWXYZ01234567891011121314151617181920'

regex.findall(s)

['a', 'b', 'c', 'd', 'e', 'f']
```

## 13.3 python 메서드 설명 및 실습

### flag(Python의 옵션)

Aa Method	≡ 단축	≡ 설명
<code>re.compile('패턴', re.DOTALL)</code>	<code>re.compile('패턴', re.S)</code>	모든 문자(줄바꿈 포함)
<code>re.compile('패턴', re.IGNORECASE)</code>	<code>re.compile('패턴', re.I)</code>	대소문자 구분 X
<code>re.compile('패턴', re.MULTILINE)</code>	<code>re.compile('패턴', re.M)</code>	모든 라인 매칭
<code>re.compile('패턴', re.VERBOSE)</code>	<code>re.compile('패턴', re.X)</code>	주석 등 편의 기능 사용 가능
<code>re.compile('패턴', re.ASCII)</code>	<code>re.compile('패턴', re.A)</code>	ASCII만을 사용

COUNT 5

## 메서드

Aa Method	≡ 설명
<code>re.compile('패턴').match(문자열)</code>	문자열 처음이 정규식 매칭 여부 판단(있을 경우 object 주소 반환, 없을 경우 None)
<code>re.compile('패턴').search(문자열)</code>	문자열 전체에서 정규식 매칭 여부 판단(있을 경우 object 주소 반환, 없을 경우 None)
<code>re.compile('패턴').findall(문자열)</code>	정규식과 매칭되면 리스트로 반환
<code>re.compile('패턴').finditer(문자열)</code>	정규식과 매칭되면 순회가능 객체로 반환
<code>re.compile('패턴').split(문자열)</code>	패턴대로 문자열 분할
<code>re.compile('패턴').sub(대체문구, 문자열)</code>	패턴을 대체문구로 대체함(replace)
<code>re.compile('패턴').fullmatch(문자열)</code>	문자열 전체가 정규식에 매칭되는지 여부 판단.

매서드는 아래와 같이도 사용할 수 있습니다.

```
re.search(패턴, 문자열, flags=0)
```

또한 위 매서드로 일치한 문자열 정보를 찾았을 때 아래와 같은 매서드를 사용하여 저장된 객체에 접근할 수 있습니다.

## 매치 객체

Aa Method	≡ 설명
<code>group(숫자)</code>	일치한 문자열의 그룹에서 숫자에 해당하는 문자열을 반환한다.
<code>groups(default=None)</code>	일치한 문자열의 모든 그룹을 반환한다.
<code>groupdict(default=None)</code>	일치한 문자열의 패턴을 딕셔너리 값으로 반환한다.
<code>expand(문자열)</code>	일치한 문자열의 그룹에 대해 옵션값에 해당하는 문자열로 반환한다.

COUNT 4

## 1. tutorial 1

```
import re

정규표현식 = r'([a-zA-Z]+) : (\d+)'
문자열 = 'name : leehojun, age : 10, height : 180, email : paul-lab@naver.com'
결과 = re.search(정규표현식, 문자열)

print(f're.search(정규표현식, 문자열) : {결과}')
```

if 결과:

```
    print(f'결과값의 시작과 끝 : {결과.start()}, {결과.end()}')
```

```
    print(f'매칭 그룹핑 : {결과.group(0)}') # 그룹 0 ~ 3
```

else:

```
    print('매칭 결과 없음!')
```

## 2. tutorial 2

```
import re

정규표현식 = r'([a-zA-Z]+) : (\d+)'
문자열 = 'name : leehojun, age : 10, height : 180, email : paul-lab@naver.com'
결과 = re.findall(정규표현식, 문자열)

print(f're.findall(정규표현식, 문자열) : {결과}')
```

```
import re

정규표현식 = r'([a-zA-Z]+) : (\d+)'
문자열 = 'name : leehojun, age : 10, height : 180, email : paul-lab@naver.com'
결과 = re.findall(정규표현식, 문자열)

print(f're.findall(정규표현식, 문자열) : {결과}')
```



```
import re

정규표현식 = r'([a-zA-Z]+) : (\d+)'
문자열 = 'name : leehojun, age : 10, height : 180, email : paul-lab@naver.com'
결과 = re.finditer(정규표현식, 문자열)

print(f're.finditer(정규표현식, 문자열) : {결과}')

for i in 결과:
    print(i)
    print(i.start(), i.end())
    print(i.group())
```

### 3. tutorial 3

```
import re

정규표현식 = r'([a-zA-Z]+) : (\d+)'
문자열 = 'name : leehojun, age : 10, height : 180, email : paul-lab@naver.com'
결과 = re.split(정규표현식, 문자열)
# split(정규표현식, 문자열, [최대분할수])

결과
```

```
import re

정규표현식 = r', '
문자열 = 'name : leehojun, age : 10, height : 180, email : paul-lab@naver.com'
결과 = re.split(정규표현식, 문자열, 2)

결과
```

## 4. tutorial 4

```
import re

정규표현식 = r', '
문자열 = 'name : leehojun, age : 10, height : 180, email : paul-lab@naver.com'

결과 = re.sub(정규표현식, " !!!", 문자열)

결과
```

```
import re

문자열 = 'name : leehojun, age : 10, height : 180, email : paul-lab@naver.com'
결과 = re.compile(',').sub(" !-!", 문자열)

결과
```

## 5. tutorial 5

1. aaabbcccc는 a3b2c3로 압축된다. 압축된 문자열을 정규표현식을 사용하여 다시 풀어보시오.

```
import re

정규표현식 = re.compile(r'([a-z])([1-9])')
문자열 = 'a3b4c2'

결과 = 정규표현식.findall(문자열)
s = ''
for i, j in 결과:
    s += i*int(j)
s
```

## 6. tutorial 6

python에 dict에 items형식을 아래와 같은 형식으로 split하시오.

```
import re

문자열 = '[(name, leehojun), (age, 10), (height, 180), (email, paul-lab@naver.com)]'
```

```
# split 형식
[('name', 'leehojun'),
 ('age', '10'),
 ('height', '180'),
 ('email', 'paul-lab@naver.com')]
```

```
정규표현식 = r'\((.*?)\|, (.*?)\)'
결과 = re.findall(정규표현식, 문자열)

결과
```

## 7. tutorial 7

1. '6746-29301-28391 신한은행'와 같은 문자열이 입력되었을 때 '신한은행 !!! 6746 29301 28391'와 같이 출력해주세요.

```
import re

정규표현식 = r'(?P<one>\d+)-(?P<two>\d+)-(?P<three>\d+) (?P<four>\w+)'
문자열 = '6746-29301-28391 신한은행'
결과 = re.match(정규표현식, 문자열)

결과.expand('\g<four> !!! \g<one> \g<two> \g<three>')
결과.group(1)
결과.group('one')
결과.groups()
결과.groupdict()
```

```
import re

정규표현식 = r'(\d+)-(\d+)-(\d+) (\w+)'
문자열 = '6746-29301-28391 신한은행'
결과 = re.match(정규표현식, 문자열)

결과
결과.group(4)
결과.groups()
결과.groupdict()
```

## 14. 연습문제

1. 전화번호만 찾는 정규표현식을 만들어보세요.
2. 이메일만 찾는 정규표현식을 만들어보세요.
3. URL만 찾는 정규표현식을 만들어보세요.
4. 다음과 가진 패턴을 가진 숫자가 입력되었을 때 다음 패턴에 부합하는지, 부합하지 않는지 가려내는 코드를 작성하세요.
  1. 자릿수는 8자리입니다.
  2. 1부터 9까지의 숫자가 맨 앞에 주어집니다.
  3. 두번째 숫자부터 다섯번째 숫자까지는 1010이 주어집니다.
  4. 여섯번째, 일곱번째 숫자는 0부터 5까지의 숫자가 주어집니다.
  5. 여덟번째 숫자는 0이어야 합니다.
6. 입출력 예시 :
  1. 입력 : 71010330, 출력 : Yes
  2. 입력 : 98101033, 출력 : No
5. 다음과 같은 영어 소설에서 전체문자, 숫자, 띄어쓰기, 대문자, 소문자, 특수문자의 개수를 나타내고 가능하다면 시각화 라이브러리를 사용하여 시각화 하세요.(위대한 게츠비)

www.gutenberg.org

 <http://www.gutenberg.org/files/64317/64317-0.txt>

기본 코드 :

```
import requests

url = 'http://www.gutenberg.org/files/64317/64317-0.txt'
response = requests.get(url)
response.encoding = 'utf-8'
text = response.text

text
```

6. 약자 만들기
  1. ORM, NASA, CS 등 약자를 만드는 프로그램을 작성하세요.
  2. 입출력 예시:
    1. 입력 : Object Relational Mapping, 출력 : ORM

## 7. 주사위 문제

1. 주사위 2개가 있습니다. 1번 주사위는 1~6숫자가 적혀있고, 2번 주사위는 A~D, ! 문자가 적혀 있습니다. (2번 주사위에 하나는 공란입니다. 아무 문자도 적혀있지 않습니다.)
2. 주사위 1번을 던지고 주사위 2번을 던집니다.
3. 2번 주사위는 아래와 같은 규칙을 가집니다.
  1. A : 1번 주사위 점수에 2배
  2. B : 1번 주사위 점수에 3배
  3. C : 1번 주사위 점수에 4배
  4. D : 1번 주사위 점수에 5배
  5. ! : 지금까지 점수의 합의 제공
4. 입출력 예
  1. '2A3A4!'의 경우에는 4+6+4 점수에 지금까지 있었던 점수의 제공이니  $14^2$ 로 196점이 됩니다.
  2. '6!2C2C'의 경우에는 36점에 4+4를 더해야 하므로 44점이 됩니다.

## 15. 주피터 노트북

주피터 노트북을 공유해드립니다. 해당 주피터 노트북 링크는 동영상 강좌 안에 PDF와 함께 게시되어 있습니다.

 정규표현식\_동영상\_촬영.ipynb 381.4KB

Google Colaboratory

 <https://colab.research.google.com/drive/12E0tx4nq465bTcnV5ISQZFwKkN0...>



초판 1쇄 발행 | 2021년 3월  
지은이 | 이호준 김혜원 김유진 차경림 김진 현지연 정승한  
편 집 | 이호준 차경림 김진  
총 괄 | 이호준  
펴낸곳 | 사도출판  
주 소 | 제주특별자치도 제주시 동광로 137 대동빌딩 4층  
표지디자인 | 차경림  
홈페이지 | <http://www.paullab.co.kr>  
E - mail | [paul-lab@naver.com](mailto:paul-lab@naver.com)

---

Copy right © 2021 by. 사도출판

이 책의 저작권은 사도출판에 있습니다. 저작권법에 의해 보호를 받는 저작물이므로 무단 복제 및 무단 전재를 금합니다.

이 책에 대한 의견을 주시거나 오타자 및 잘못된 내용의 수정 정보는 사도출판의 이메일로 연락을 주시기 바랍니다.

# 정규 표현식

Python으로 톺아보기