정규표현식

강사 장철원

Section 0

목차

❑정규 표현식

정규표현식

Section 1. 정규표현식 기초

Section 1-1. 정규 표현식 개념

정규 표현식(regular expressions, regex or regexp)

Sequence of characters that specifies a match pattern in text

텍스트에서 특정 패턴을 매칭할 때 사용하는 기법

정규 표현식(regular expressions, regex or regexp)

이름에서 두번째 글자와, 전화번호에서 가운데 네자리를 마스킹 처리해주세요.

홍길동 010-1234-5678	홍*동 010-****-5678
임꺽정 010-4321-8765	임*정 010-****-8765
성춘향 010-2345-6543	성*향 010-****-6543

정규 표현식없이 파이썬 코드로 구현

[1]:
```
text = """홍길동 010-1234-5678
임꺽정 010-4321-8765
성춘향 010-2345-6543
"""
```

[2]:
```
print(text)
```

```
홍길동 010-1234-5678
임꺽정 010-4321-8765
성춘향 010-2345-6543
```

[3]:
```
result = []
```

[4]:
```
for line in text.splitlines():
    print(line)
    print('------')
```

```
홍길동 010-1234-5678
------
임꺽정 010-4321-8765
------
성춘향 010-2345-6543
------
```

[5]:
```
for line in text.splitlines():
    name, phone = line.split()
    print(name, phone)
    print('--------')
```

```
홍길동 010-1234-5678
--------
임꺽정 010-4321-8765
--------
성춘향 010-2345-6543
--------
```

정규 표현식없이 파이썬 코드로 구현

[6]:
```
for line in text.splitlines():
    name, phone = line.split()
    re_name = name[0] + '*' + name[-1]
    print(re_name, phone)
```

```
홍*동 010-1234-5678
임*정 010-4321-8765
성*향 010-2345-6543
```

[7]:
```
for line in text.splitlines():
    name, phone = line.split()
    re_name = name[0] + '*' + name[-1]
    phone_part = phone.split('-')
    print(phone_part)
```

```
['010', '1234', '5678']
['010', '4321', '8765']
['010', '2345', '6543']
```

[8]:
```
for line in text.splitlines():
    name, phone = line.split()
    re_name = name[0] + '*' + name[-1]
    phone_part = phone.split('-')
    re_phone = phone_part[0]+'-****-'+phone_part[2]
    print(re_phone)
```

```
010-****-5678
010-****-8765
010-****-6543
```

[9]:
```
for line in text.splitlines():
    name, phone = line.split()
    re_name = name[0] + '*' + name[-1]
    phone_part = phone.split('-')
    re_phone = phone_part[0]+'-****-'+phone_part[2]
    new_info = re_name +' '+ re_phone
    print(new_info)
```

```
홍*동 010-****-5678
임*정 010-****-8765
성*향 010-****-6543
```

정규 표현식없이 파이썬 코드로 구현

[10]:
```
for line in text.splitlines():
    name, phone = line.split()
    re_name = name[0] + '*' + name[-1]
    phone_part = phone.split('-')
    re_phone = phone_part[0]+'-****-'+phone_part[2]
    new_info = re_name +' '+ re_phone
    result.append(new_info)
```

[11]:
```
print(result)
```

```
['홍*동 010-****-5678', '임*정 010-****-8765', '성*향 010-****-6543']
```

[12]:
```
for person in result:
    print(person)
```

```
홍*동 010-****-5678
임*정 010-****-8765
성*향 010-****-6543
```

정규 표현식을 활용한 파이썬 코드

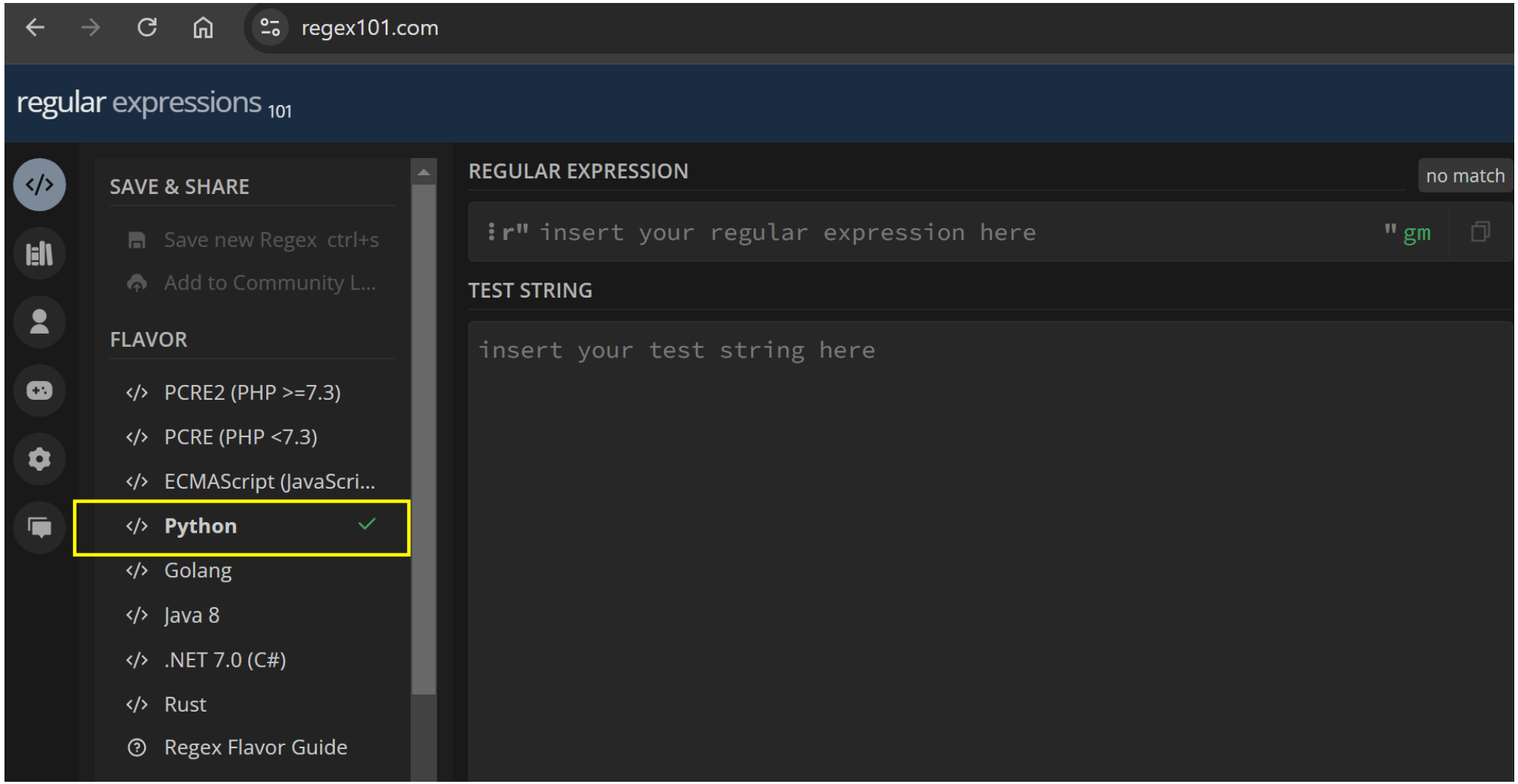
[17]:
```
text = """홍길동 010-1234-5678
임꺽정 010-4321-8765
성춘향 010-2345-6543
"""
```

[18]:
```
import re

pattern = r'(\w)(\w+)(\w)\s(010)-(\d{4})-(\d{4})'
replacement = r'\1*\3 \4-****-\6'
re_text = re.sub(pattern, replacement, text)

# 결과 출력
print(re_text)
```

```
홍*동 010-****-5678
임*정 010-****-8765
성*향 010-****-6543
```

이게 어떻게 가능할까?

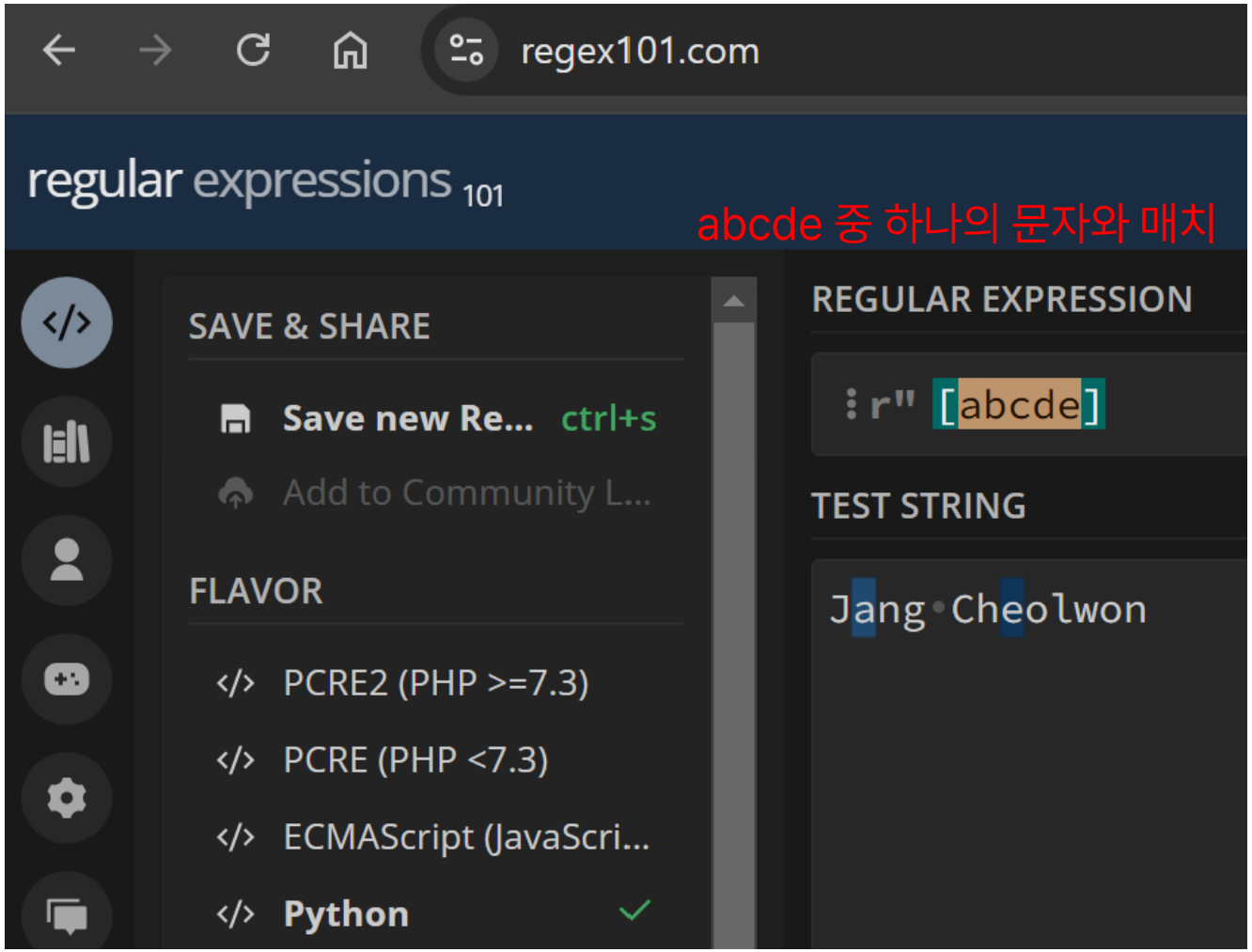
정규 표현식 연습 사이트 https://regex101.com/



메타 문자 [ ]

[와] 사이의 문자들 중 하나와 매치

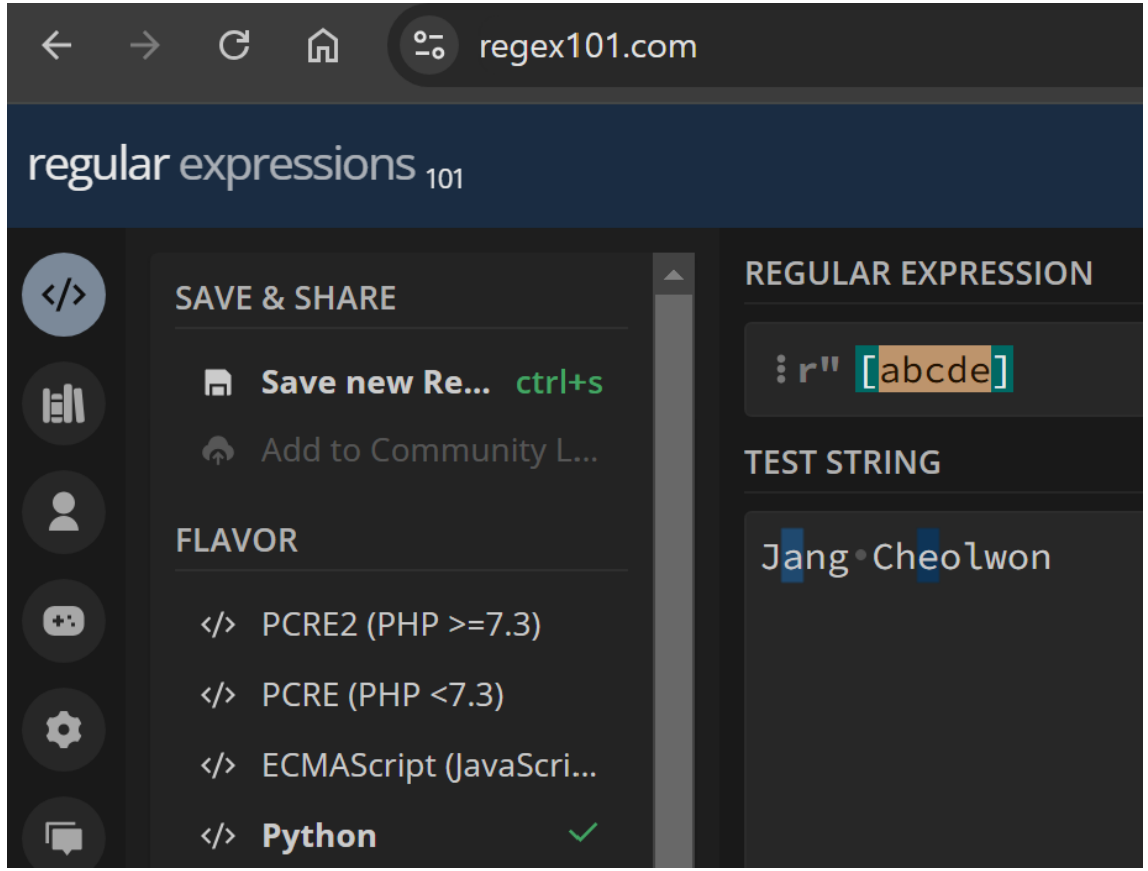
[abc] = abc 중 하나의 글자와 매치



메타 문자 [ ] 실습

[20]:
```
import re

text = 'Jang Cheolwon'
pattern = r'[abcde]'
matches = re.findall(pattern, text)
print(matches)
```

```
['a', 'e']
```



r''과 f''의 차이

r' '(raw string) - 백슬래시(\)를 이스케이프 문자로 사용하지 않고 있는 그대로 사용

[23]:
```
path = "C:\\Users\\gildong\\Documents"
print(path)

path = r"C:\Users\gildong\Documents"
print(path)
```

```
C:\Users\gildong\Documents
C:\Users\gildong\Documents
```

r''과 f''의 차이

f' '(formatted string) - 문자열에 변수를 삽일할 때 사용

[24]:
```
name = "Alice"
age = 30

# f-string 사용
sentence = f"My name is {name} and I am {age} years old."
print(sentence)
```

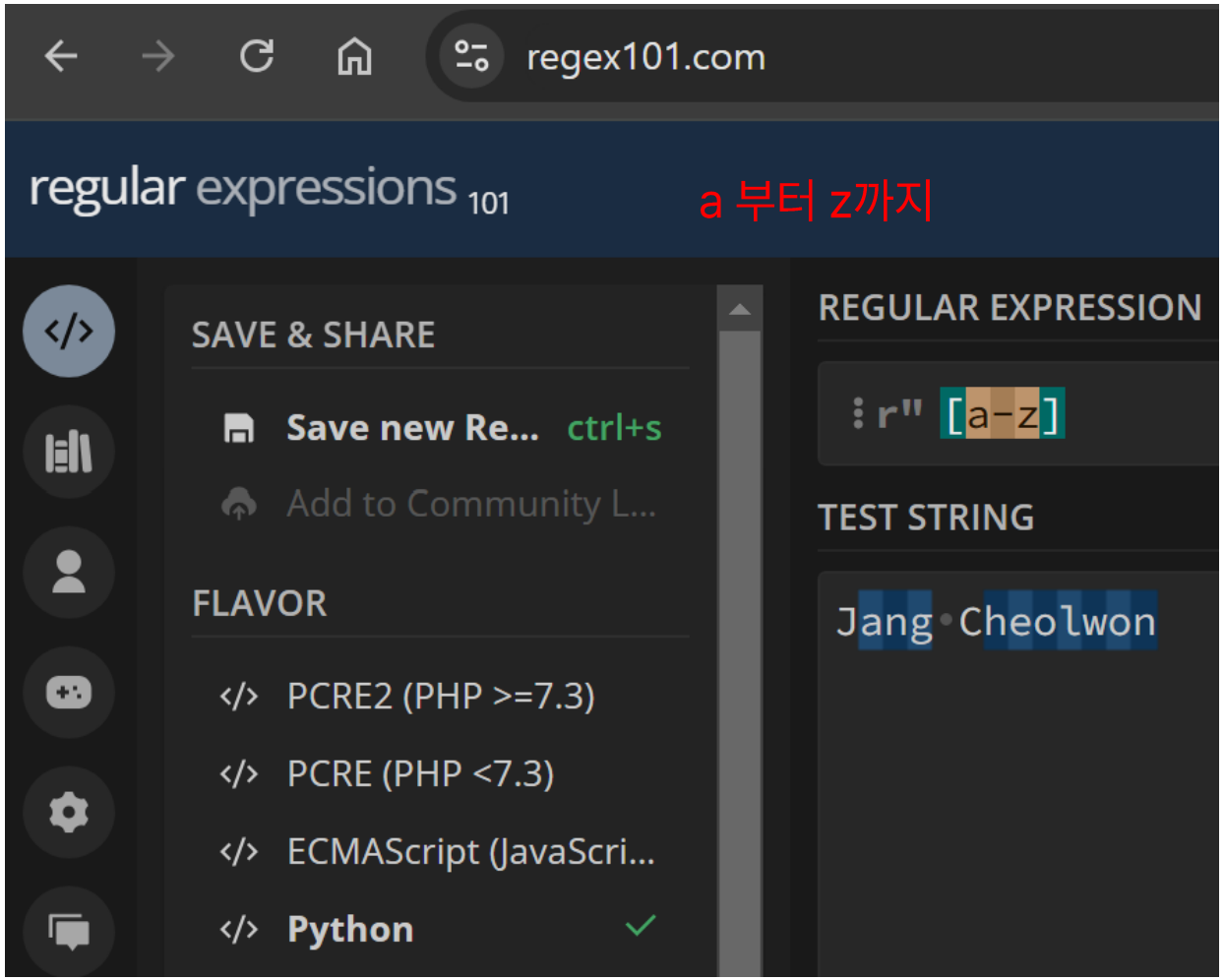
```
My name is Alice and I am 30 years old.
```

Section
정규표현식개념

메타 문자 [ ]

[25]:
```
import re

text = 'Jang Cheolwon'
pattern = r'[a-z]'
matches = re.findall(pattern, text)
print(matches)
```

```
['a', 'n', 'g', 'h', 'e', 'o', 'l', 'w', 'o', 'n']
```



메타 문자 -

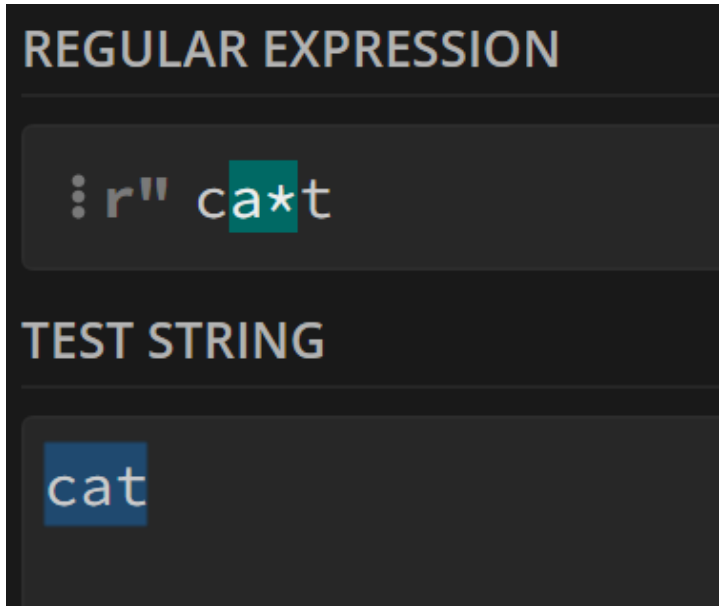
^는 not이라는 의미

[a-z]	모든 소문자	[^a-z]	소문자 제외
[a-zA-Z]	모든 대소문자	[^a-zA-Z]	대소문자 제외
[0-9]	모든 숫자	[^0-9]	숫자 제외

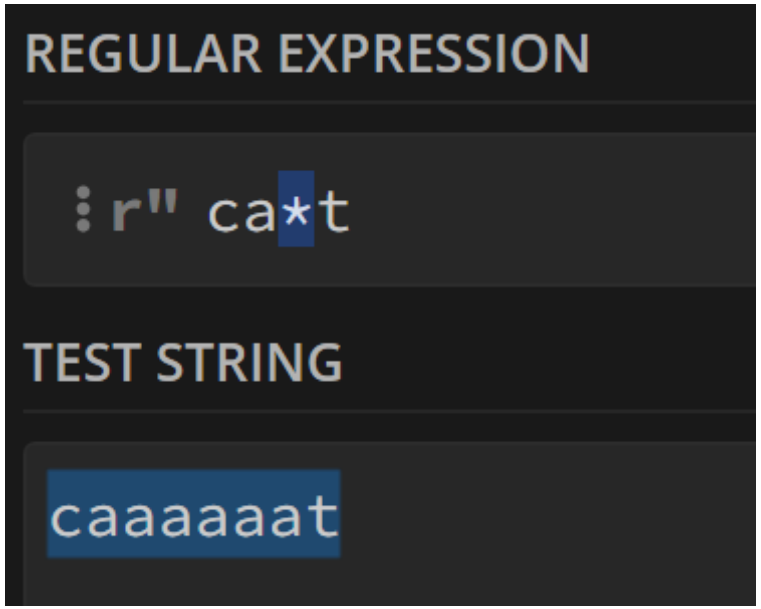
메타 문자 -

문법	의미	동일 문법
\d	숫자와 매치	[0-9]
\D	숫자가 아닌 것과 매치	[^0-9]
\s	공백과 매치	[\t\n\r\f\v]
\S	공백이 아닌 문자와 매치	[^ \t\n\r\f\v]
\w	문자+숫자와 매치	[a-zA-Z0-9_]
\W	문자+숫자가 아닌 것과 매치	[^a-zA-Z0-9_]

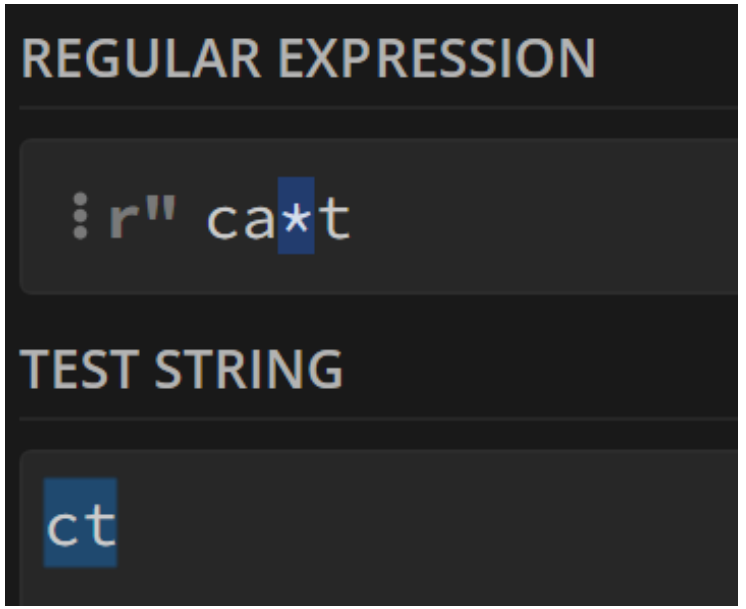
메타 문자 * = 0부터 무한대까지 반복



c와 t사이에 a가 반복

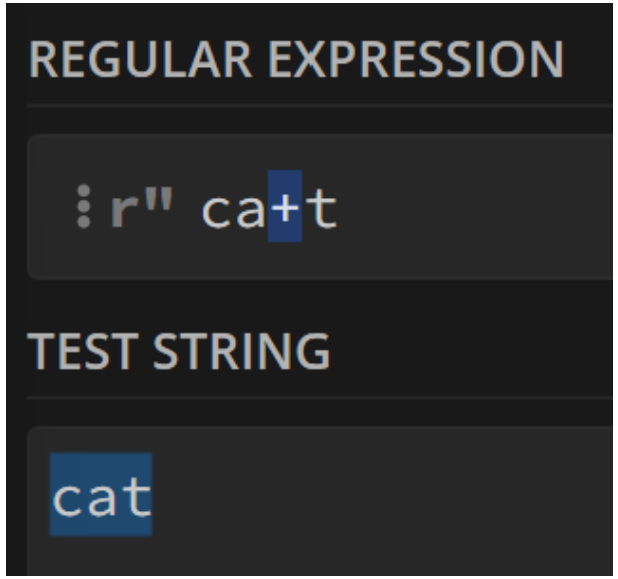


a가 여러번 반복되어도 OK

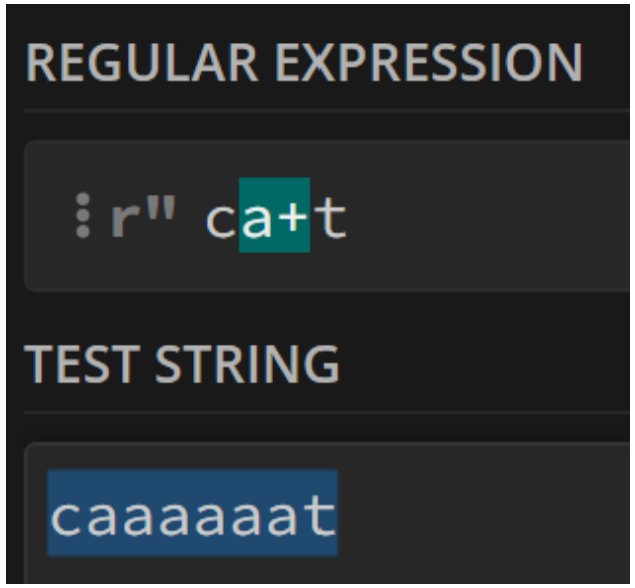


a가 포함되지 않아도 OK
(0번 반복이라고 인식)

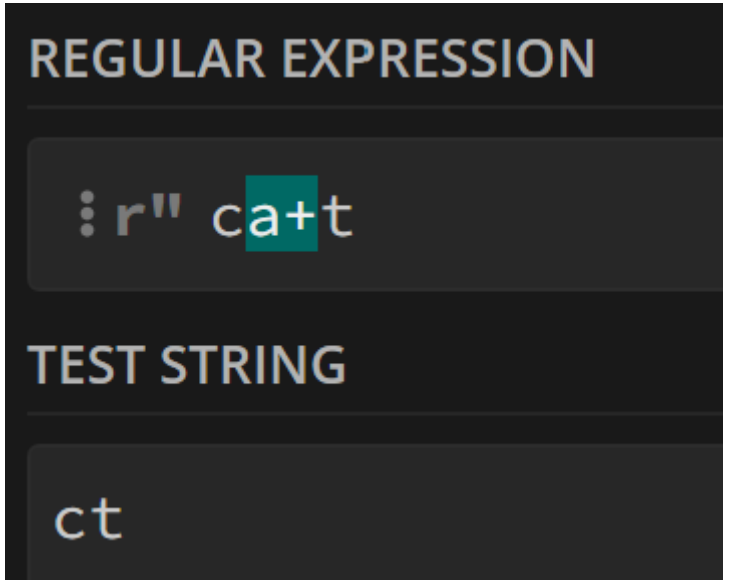
메타 문자 + = 1부터 무한대까지 반복



c와 t사이에 a가 반복

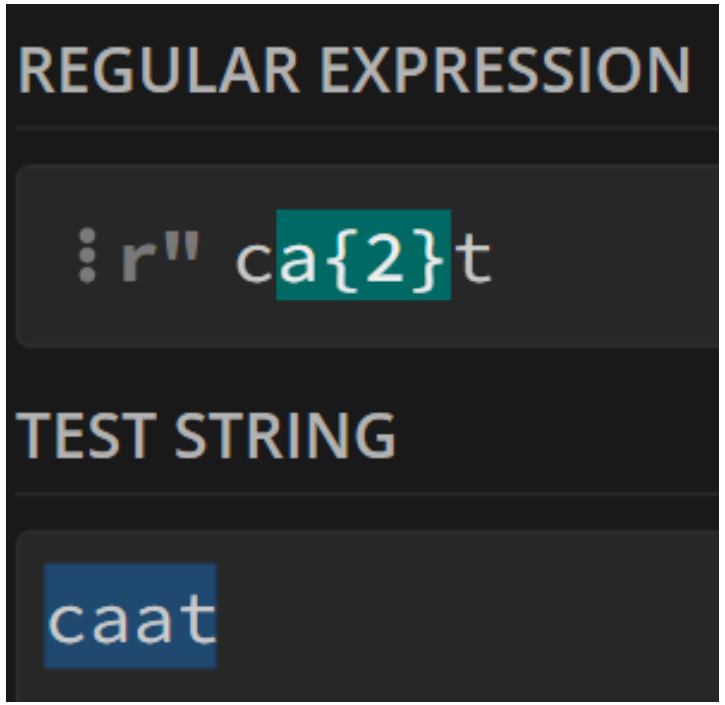


a가 여러번 반복되어도 OK

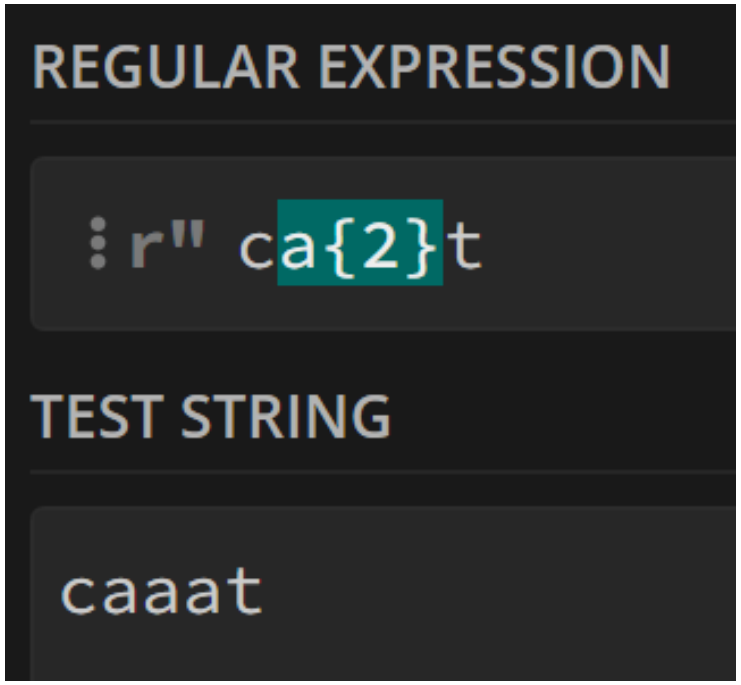


a가 포함되지 않으면 안됨
(*과 +의 차이점)

메타 문자 {m, n} - 반복 횟수가 m이상 n이하

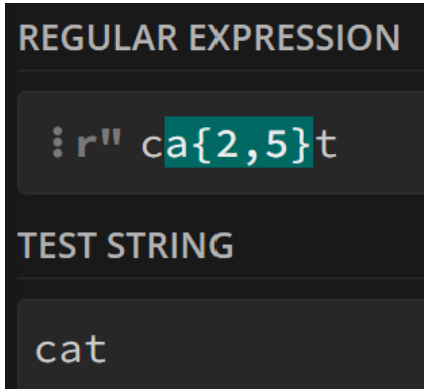


c와 t사이에 a를 정확히 두번 반복

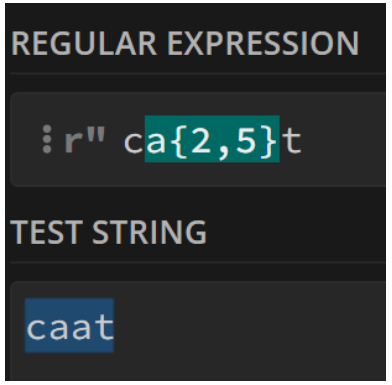


세 번 반복하면 안됨

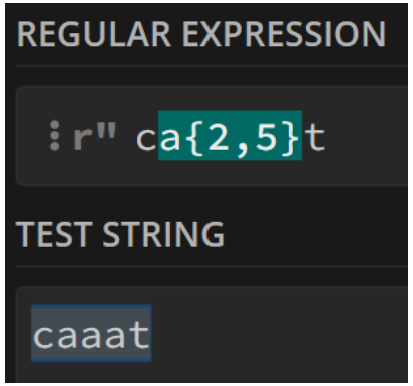
메타 문자 {2, 5} - 반복 횟수가 2이상 5이하



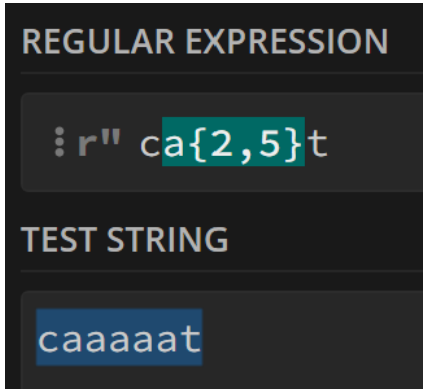
한 번 반복 안됨



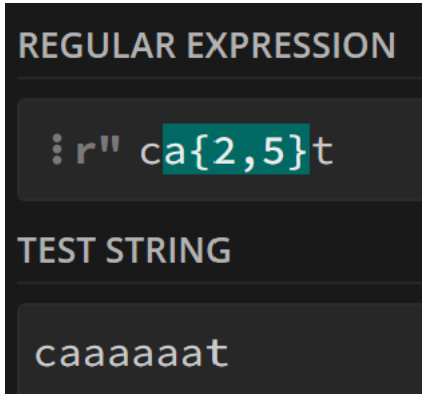
두 번 반복 됨



세 번 반복 됨



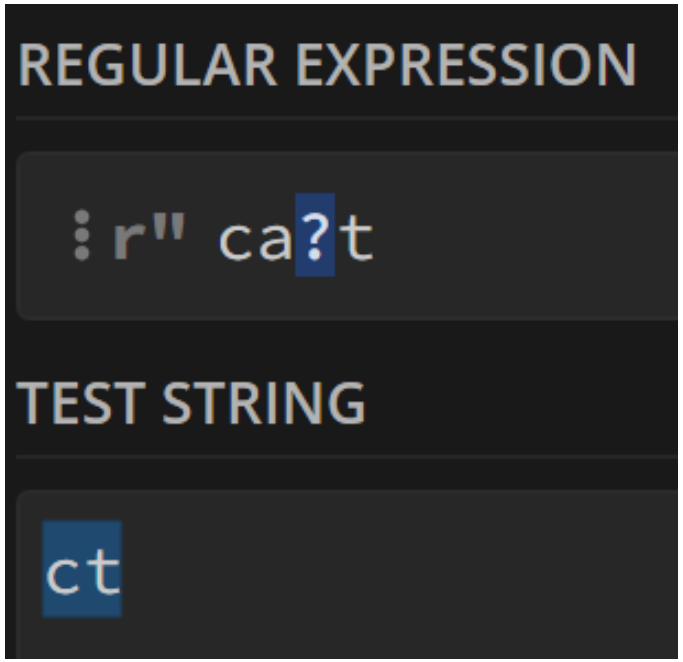
다섯 번 반복 됨



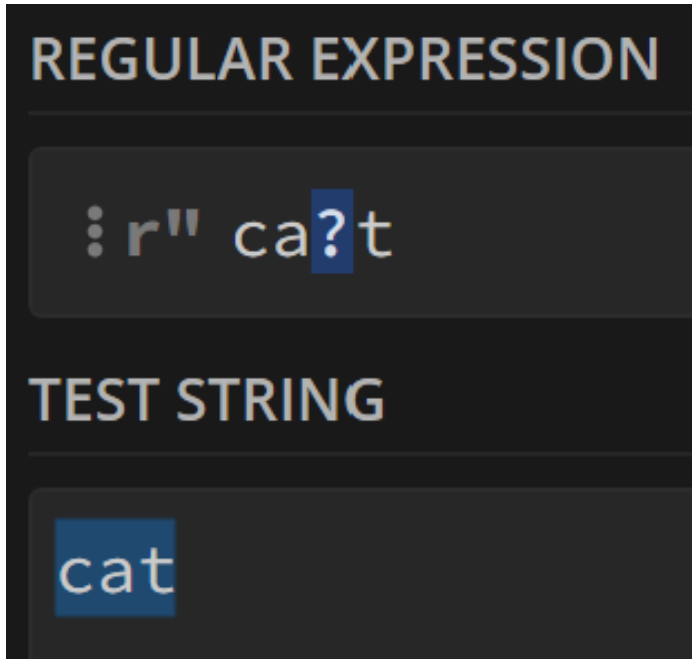
여섯 번 반복 안됨

메타 문자 ? = {0,1}

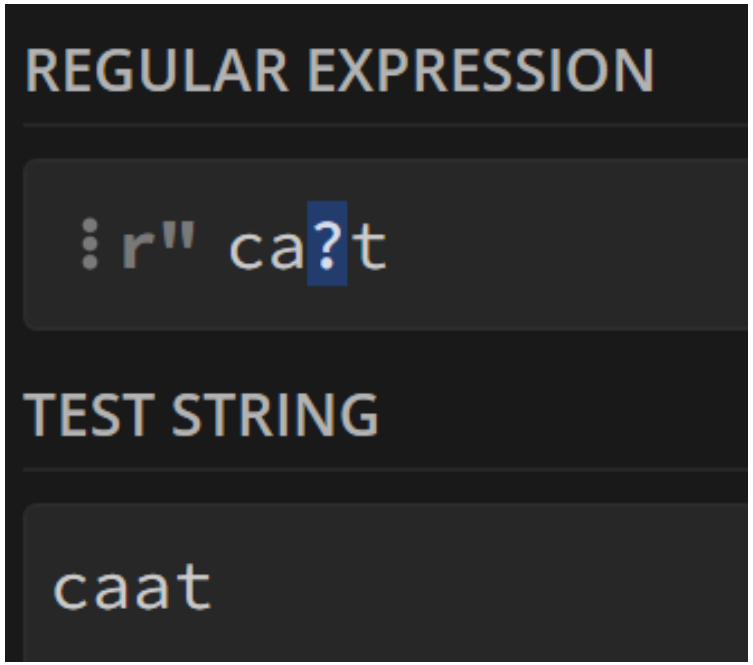
c와 t 사이에 a가 없거나 한 번 있거나



0번 포함 OK



한 번 포함 OK



두 번 포함 NO

메타 문자 ( ) - 그룹화

[38]:
```
import re

text = "example@gmail.com"
pattern = r"\w+\@\w+\.\w+"
match = re.findall(pattern, text)
print(match)
```

```
['example@gmail.com']
```

[39]:
```
import re

text = "example@gmail.com"
pattern = r"(\w+)(\@)(\w+)(\.)(\w+)"
match = re.findall(pattern, text)
print(match)
```

```
[('example', '@', 'gmail', '.', 'com')]
```

[40]:
```
import re

text = "example@gmail.com"
pattern = r"(\w+)\@(\w+)\.(\w+)"
match = re.findall(pattern, text)
print(match)
```

```
[('example', 'gmail', 'com')]
```

정규 표현식을 활용한 파이썬 코드

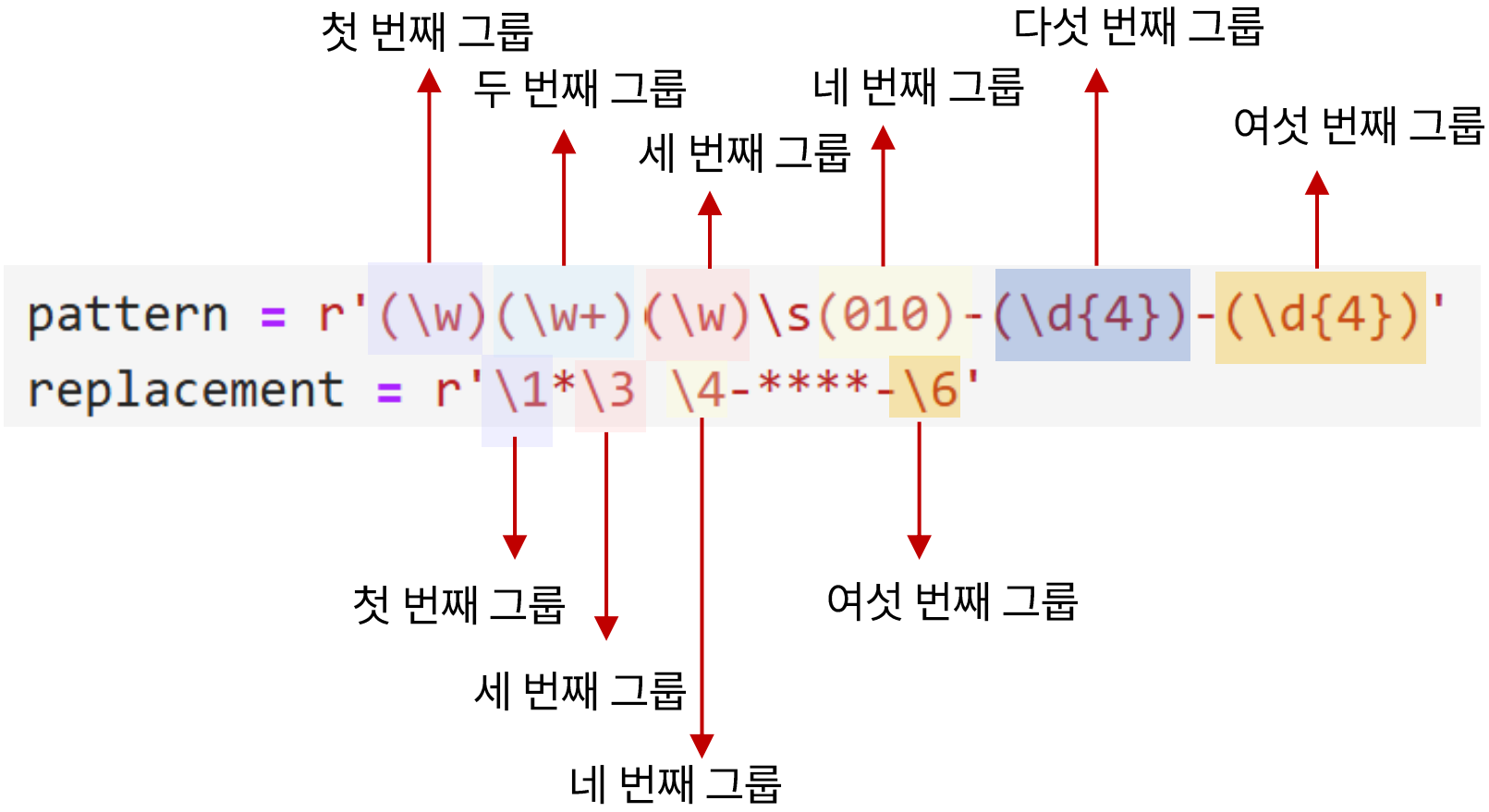
[17]:
```
text = """홍길동 010-1234-5678
임꺽정 010-4321-8765
성춘향 010-2345-6543
"""
```

[18]:
```
import re

pattern = r'(\w)(\w+)(\w)\s(010)-(\d{4})-(\d{4})'
replacement = r'\1*\3 \4-****-\6'
re_text = re.sub(pattern, replacement, text)

# 결과 출력
print(re_text)
```

```
홍*동 010-****-5678
임*정 010-****-8765
성*향 010-****-6543
```

이게 어떻게 가능할까?

정규 표현식을 활용한 파이썬 코드



감사합니다.

Q & A