```
In [ ]: for 분석, 모델링
        데이터 필요함
        데이터 축적/관리/가져오기 => DB, [files, XML, JSON, ..]
        DB -> Data - Relationship => RDB - RDBMS(Sqlite)
        SQL(*) =>
        Business Logic(Service) => 행동
        함수를 정의(굉장히 번거로움) -> Table(Entity)-Attibute=Object
        ORM Technique =>         Object-Property-Method
```

```
In [1]: import re
```

```
In [ ]: re.compile, match, search, findall
        => pattern, ^한개 , 한개   , 여러개
```

```
In [6]: re.match('Hello|Servo', 'CrowHello')
```

```
In [7]: re.search('Hello|Servo', 'CrowHello')

Out[7]: <re.Match object; span=(4, 9), match='Hello'>
```

```
In [8]: re.search('^Life', 'Life asdfasdfasfdalskjjasd')

Out[8]: <re.Match object; span=(0, 4), match='Life'>
```

```
In [26]: True if re.search('^Life', 'asdfasfdasf Life')\
              else False

Out[26]: False
```

```
In [21]: rst.start(), rst.end(), rst.span(), rst.group()

Out[21]: (12, 16, (12, 16), 'Life')
```

```
In [19]: 'asdfasfdasf Life'[12:16]

Out[19]: 'Life'
```

```
In [37]: re.search('(?:ABC)+', 'ABCABCABC OK').group(0)

Out[37]: 'ABCABCABC'
```

```
In [35]: re.search('(ABC)+ (OK)', 'ABCABCABC OK').group(2)

Out[35]: 'OK'
```

```
In [ ]: re.match('\ => [\], \\ => [\], \\\ => [\\]'), search('\')
```

```
In [40]: '\n', r'\n', re.escape('\n'), re.escape(r'\n'),\
         print('\n', r'\\\n', re.escape('\n'), re.escape(r'\n'))

          \n \
          \\n
Out[40]: ('\n', '\\n', '\\\n', '\\\\n', None)
```

```
In [49]: re.search(r'\bclass\b', 'asdfa class asdfasdf'),\
re.search(r'\Bclass\b', 'asdfa subclass asdfasdf'),\
re.search(r'\Bclass\B', 'asdfa subclasss asdfasdf')
```

```
Out[49]: (<re.Match object; span=(6, 11), match='class'>,
          <re.Match object; span=(9, 14), match='class'>,
          <re.Match object; span=(9, 14), match='class'>)
```

```
In [50]: re.search('s.*o', 'stackoverflow')
```

```
Out[50]: <re.Match object; span=(0, 12), match='stackoverflo'>
```

```
In [51]: re.search('s.*?o', 'stackoverflow')
```

```
Out[51]: <re.Match object; span=(0, 6), match='stacko'>
```

```
In [ ]: <div>내용<div>내용</div>내용</div>
```

```
In [67]: data = '아무개 990000-1231231\n개똥이 010101-5010101'

         for line in data.splitlines():
             n = line.split('-')[1]
             if len(n) == 7 and n.isdigit() and \
                n[0] in ['1', '2', '3', '4']:
                 print(line.split('-')[0]+'-'+'*'*7)
             else:
                 print(line, '[X]')
```

```
아무개 990000-*******
개똥이 010101-5010101 [X]
```

```
In [70]: for line in data.splitlines():
             if re.search(r'[1-4]\d{6}', line):
                 print(re.sub(r'[1-4]\d{6}', '*'*7, line))
             else:
                 print(line, '[X]')
         # print(re.sub(r'[1-4]\d{6}', '*'*7, data))
```

```
아무개 990000-*******
개똥이 010101-5010101 [X]
```

```
In [92]: # 전화번호, 이메일 => 수집
         # -> 숫자로 구성된 패턴, -> 숫자+문자 패턴
         # ->                  ,
         # 1. 전화번호
         data = '''
         어쩌고 저쩌고 구매문의 010-1234-1234 주세요
         080-1234-1234
         02-1234-1234
         02-123-1234
         02 123 1234
         010.1234.1234
         010     1231\t1231
         82 010 1234 1234
         +82 010 1234 1234
         +82 10 1234 1234
         8201012341234
         공1공 삼4오6 1234
         '''
```

```
re.findall(r'(?:[+]?(\d{2})[\- .\t]*)?([0-9공일이삼사오육칠팔구]{2,3})[\- .\t]*([0-9공일이
#       ?: => 그룹이긴 하나 캡쳐하지 않음   ^
#  ([0-9공일이삼사오육칠팔구]{2,3})[\- .\t]+(\d{3,4})[\- .\t]+(\d{4})', data)
#          (\d{2,3})[\- .\t]*(\d{3,4})[\- .\t]*(\d{4})', data)
```

```
Out[92]: [('', '010', '1234', '1234'),
          ('', '080', '1234', '1234'),
          ('', '02', '1234', '1234'),
          ('', '02', '123', '1234'),
          ('', '02', '123', '1234'),
          ('', '010', '1234', '1234'),
          ('', '010', '1231', '1231'),
          ('82', '010', '1234', '1234'),
          ('82', '010', '1234', '1234'),
          ('82', '10', '1234', '1234'),
          ('82', '010', '1234', '1234'),
          ('', '공1공', '삼4오6', '1234')]
```

```
In [99]: # 2. 이메일 => JS Validation
         data = '''
         test@test.com
         test@test.co.kr
         test@email.test.co.kr
         test@m.email.test.co.kr로 보내주세요
         '''
         # (.)m
         # .email
         # .test
         # .co
         # .kr

         re.findall(r'([a-z]+)[@]([a-z]+[.][a-z]{3})', data)
         #          영문자1번이상 @ 영문자1번 문자(.) 영문자3번
         re.findall(r'([a-z]+)[@]([.]?[a-z]+)+', data)
         re.findall(r'([a-z]+)[@]((?:[.]?[a-z]+)+)', data)
```

```
Out[99]: [('test', 'test.com'),
          ('test', 'test.co.kr'),
          ('test', 'email.test.co.kr'),
          ('test', 'm.email.test.co.kr')]
```

```
In [110]… data = '''
          http://www.naver.com
          http://www.naver.com/
          https://naver.com
          naver.com/index.nhn
          m.naver.com
          mail.naver.com
          www.naver.com/search/asdfa.index?key=value
          '''
          # /search
          # /asdfa.index?key=value
          re.findall(r'(?:(https?)://)?([a-z]+(?:[.][a-z]+)+)((?:/[a-z.?=]*)*)', data)
```

```
Out[110]: [('http', 'www.naver.com', ''),
           ('http', 'www.naver.com', '/'),
           ('https', 'naver.com', ''),
           ('', 'naver.com', '/index.nhn'),
           ('', 'm.naver.com', ''),
           ('', 'mail.naver.com', ''),
 ('', 'www.naver.com', '/search/asdfa.index?key=value')]
```

In [127]…

```
data = '''
<div>
    <table>
        <tr>
            <td>매출</td>
            <td>증감율</td>
        </tr>
        <tr>
            <td>10원</td>
            <td>+1.4%</td>
        </tr>
        <tr>
            <td>20원</td>
            <td>-1.4%</td>
        </tr>
        <tr><td>30원</td><td>-1.4%</td></tr>
    </table>
</div>
'''
re.findall(r'<tr>\s*<td>(\d+)[^<]+?</td>\s*<td>([+-][0-9.]+)[^<]+?</td>\s*</tr>', data)
```

Out[127]:

```
[('10', '+1.4'), ('20', '-1.4'), ('30', '-1.4')]
```

In [168]…

```
d = '''
1S2D*3T
1D2S#10S
1D2S0T
1S*2T*3S
1D#2S*3S
1T2D3D#
1D2S3T*
'''
squared = {'S':1,'D':2,'T':3}
bonus = {'*':2,'#':-1}

for row in d.splitlines()[1:]:
    rst = re.findall(r'(\d{1,2})([SDT])([*#]?)', row)
    print(rst[0], rst[1], rst[2])
    total = list()
    for r in rst:
        if r[2] and r[2] == '*':
            if len(total) > 0:
                total[len(total)-1] = total[len(total)-1]*bonus[r[2]]
        total.append(int(r[0])**squared[r[1]]*(bonus[r[2]] if r[2] else 1))
    print(sum(total), total)
```

```
('1', 'S', '') ('2', 'D', '*') ('3', 'T', '')
37 [2, 8, 27]
('1', 'D', '') ('2', 'S', '#') ('10', 'S', '')
9 [1, -2, 10]
('1', 'D', '') ('2', 'S', '') ('0', 'T', '')
3 [1, 2, 0]
('1', 'S', '*') ('2', 'T', '*') ('3', 'S', '')
23 [4, 16, 3]
('1', 'D', '#') ('2', 'S', '*') ('3', 'S', '')
5 [-2, 4, 3]
('1', 'T', '') ('2', 'D', '') ('3', 'D', '#')
-4 [1, 4, -9]
```

```
('1', 'D', '') ('2', 'S', '') ('3', 'T', '*')
59 [1, 4, 54]
```