2.17 HTML과 XML 엔티티 처리

[문제] &entity;나 &#code;와 같은 HTML, XML 엔티티를 이에 일치하는 문자로 치환하고 싶음
텍스트 생성시 특정문자를 피하고 싶음
[해결] html.escape() 함수 사용

```
s = 'Elements are written as "<tag>text</tag>".'
import html
print(s) #Elements are written as "<tag>text</tag>".
print(html.escape(s)) #Elements are written as &quot;&lt;tag&gt;text&lt;/tag&gt;&quot;.
# 따옴표는 남겨두도록 지정
print(html.escape(s, quote=False)) #Elements are written as "&lt;tag&gt;text&lt;/tag&gt;".
```

#텍스트를 아스키로 만들고 캐릭터 코드를 아스키가 아닌 문자에 끼워 넣고 싶으면 errors = 'xmlcharrefreplace' 인자를 입출력 관련 함수에 사용

```
s = 'Spicy Jalapeño'
s.encode('ascii', errors='xmlcharrefreplace') #b'Spicy Jalape&#241;o'
```

```
s = 'Spicy &quot;Jalape&#241;o&quot.'
from html.parser import HTMLParser
p = HTMLParser()
p.unescape(s) #'Spicy "Jalapeño".

t = 'The prompt is &gt;&gt;&gt;'
from xml.sax.saxutils import unescape
unescape(t) #'The prompt is >>>'
```

2.18 텍스트 토큰화

[문제] 문자를 파싱해서 토큰화 하고 싶음 (파싱의 기준점이 되는게 토큰?)
[해결] 패턴 매칭 이상의 작업이 필요함.
text = 'foo = 23 +42 *10'
tokens = [('NAME', 'foo'), ('EQ','='), ('NUM', '23'), ('PLUS','+'), ('NUM', '42'), ('TIMES', '*'), ('NUM', 10')]

정규표현식을 사용해야함

```
import re
NAME = r'(?P<NAME>[a-zA-Z_][a-zA-Z_0-9]*)' NUM = r'(?P<NUM>\d+)'
PLUS = r'(?P<PLUS>\+)'
TIMES = r'(?P<TIMES>\*)'
EQ = r'(?P<EQ>=)'
WS = r'(?P<WS>\s+)'

master_pat = re.compile('|'.join([NAME, NUM, PLUS, TIMES, EQ, WS]))

scanner = master_pat.scanner('foo = 42')
scanner.match() #<_sre.SRE_Match object at 0x100677738>
_.lastgroup, _.group() #('NAME', 'foo')
scanner.match() #<_sre.SRE_Match object at 0x100677738>
_.lastgroup, _.group() #('WS', ' ')
scanner.match()#<_sre.SRE_Match object at 0x100677738>
_.lastgroup, _.group()#('EQ', '=')
scanner.match() #<_sre.SRE_Match object at 0x100677738>
_.lastgroup, _.group() #('WS', ' ')
scanner.match() #<_sre.SRE_Match object at 0x100677738>
_.lastgroup, _.group() #('NUM', '42')
```

scanner.match()

간단한 생성자를 만들 수 있음

```
from collections import namedtuple
    Token = namedtuple('Token', ['type','value'])
def generate_tokens(pat, text):
scanner = pat.scanner(text)
for m in iter(scanner.match, None):
yield Token(m.lastgroup, m.group())

#사용 예
for tok in generate_tokens(master_pat, 'foo = 42'): print(tok)
    # Token(type='NAME', value='foo')
    # Token(type='WS', value=' ')
    # Token(type='EQ', value='=')
    # Token(type='WS', value=' ')
    # Token(type='NUM', value='42')

tokens = (tok for tok in generate_tokens(master_pat, text) if tok.type != 'WS')
for tok in tokens: print(tok)
```

[토론]

```
LT = r'(?P<LT><)'
    LE = r'(?P<LE><=)'
    EQ = r'(?P<EQ>=)'
master_pat = re.compile('|'.join([LE, LT, EQ])) # Correct
# master_pat = re.compile('|'.join([LT, LE, EQ])) # Incorrect

PRINT = r'(P<PRINT>print)'
    NAME  = r'(P<NAME>[a-zA-Z_][a-zA-Z_0-9]*)'
master_pat = re.compile('|'.join([PRINT, NAME])) for tok in generate_tokens(master_pat, 'printer'):
print(tok)

    #  Token(type='PRINT', value='print')
    #  Token(type='NAME', value='er')
```

하나도 모르겠음

2.19 간단한 재귀 파서 작성

[문제] 주어진 문법 규칙에 따라 텍스트를 파싱하고 동작을 수행하거나 입력된 텍스트를 추상 신텍스 트리로 나타내야 함. 문법은 간단하지만 프레임 워크 사용하지 않고 파서를 직접 작성하고 싶음

[해결]

```
expr ::= expr + term
    |   expr - term
    | term
term ::= term * factor
    |   term / factor
    | factor
factor ::= ( expr )
    |   NUM

expr ::= term { (+|-) term }*
term ::= factor { (*|/) factor }*
factor ::= ( expr )
      |   NUM
```

expr
expr ::= term { (+|-) term }*
expr ::= factor { (*|/) factor }* { (+|-) term }*
expr ::= NUM { (*|/) factor }* { (+|-) term }*
expr ::= NUM { (+|-) term }*
expr ::= NUM + term { (+|-) term }*
expr ::= NUM + factor { (*|/) factor }* { (+|-) term }*
expr ::= NUM + NUM { (*|/) factor}* { (+|-) term }*
expr ::= NUM + NUM * factor { (*|/) factor }* { (+|-) term }*
expr ::= NUM + NUM * NUM { (*|/) factor }* { (+|-) term }*
expr ::= NUM + NUM * NUM { (+|-) term }*
expr ::= NUM + NUM * NUM

#이해가 안가……코드 그만 치고 글자만 읽을래,,,,,

2.20 바이트 문자열에 텍스트 연산

[문제] 바이트 문자열에 일반적인 텍스트 연산(잘라내기, 검색, 치환 등) 수행
[해결]

```
data = b'Hello World'
data[0:5] #b'Hello
data.startswith(b'Hello') #True
data.split() #공백 기준 쪼개기
data.replace(b'Hello',b'Hello Cruel')

#바이트 배열에도 사용 가능
data = bytearray(b'Hello World')
data[0:5] #bytearray(b'Hello')
data.startswith(b'Hello') #True
data.split() #[bytearray(b'Hello'), bytearray(b'World')]
data.replace(b'Hello', b'Hello Cruel') #bytearray(b'Hello Cruel World')

#바이트 문자열 패턴 매칭에 정규 표현식을 적용할 수 있음 패턴 자체도 바이트로 나타내야함
data = b'FOO:BAR,SPAM'
import re
re.split('[:,]',data)  #오류남
re.split(b'[:,]',data) # 주의 : 패턴도 바이트로 나타냄 [b'FOO', b'BAR', b'SPAM']

#텍스트 문자열에 있는 연산 기능은 바이트 문자열에도 내장되어 있음
#1) 바이트 문자열에 인덱스를 사용하면 개별 문자가 아니라 정수를 가르킴
a = 'Hello World' >>> a[0] #'H'
a[1] #'e'
b = b'Hello World' #바이트 문자열
b[0] #72
b[1] #101
#2) 바이트 문자열은 보기 좋은 표현식을 지원하지 않으며 텍스트 문자열로 변환하지 않으면 깔끔하게 출력할 수 없음
s = b'Hello World'
print(s) #b'Hello World'
print(s.decode('ascii')) #Hello World

#3) 바이트 문자열은 서식화를 지원하지 않음 (format() 사용x)
b'%10s %10d %10.2f' % (b'ACME', 100, 490.1) #오류남
b'{} {} {}'.format(b'ACME', 100, 490.1) #오류남
```

바이트 문자열에 서식화를 적용하고 싶으면 일반 문자열과 인코딩을 사용해야 함
'{:10s} {:10d} {:10.2f}'.format('ACME', 100, 490.1).encode('ascii')
#b'ACME 100 490.10'

#파일이름을 텍스트 문자열이 아니라 바이트 문자열로 제공하면 파일이름을 인코딩디코딩할 수 없다
#텍스트 작업할때는 바이트보다 텍스트를 쓰장

3.1 반올림

[문제] 부동 소수점 값을 10진수로 반올림
[해결] round(value, ndigits)

ndigits
0 : 소수 첫째자리 1 : 소수 둘째자리 2: 소수 셋째자리
-1 : 일의자리 -2: 십의 자리

[토론]
반올림과 서식화는 다른거임
특정 자릿수까지 숫자를 표현하는 것이 목적이면 round()가 아니라 서식화를 위한 자릿수를 명시해야 함(뭐가 다른겨)

```
x = 1.23436
format(x, '0.2f') #1.23 #소수 숫자자리수만큼 출력할거임. 그 뒷자리에서 반올림한 값 반환
format(x, '0.3f') #1.235
format(x,'0.4f') #1.2346

'value is {:0.3f}'.format(x) #1.234
```

문제를 수정하려고 부동 소수점을 반올림하면 x ->decimal모드 사용

```
a = 2.1
b = 4.2
c = a+b
print(c) #6.30000000000000
c = round(c,2) #6.3
```

3.2 정확한 10진수 계산

[문제] 10진수계산 시 부동 소수점을 사용할 때 발생하는 작은 오류를 피하고 싶음
[해결] 부동 소수점 값은 10진수를 정확히 표현하지 못함 : float를 사용해서 그럼 —> 성능 버리고 정확히 하려면 decimal 사용(금융권에서는 무조건)

```
from decimal import Decimal
a = Decimal('4.2')
b = Decimal('2.1')
Decimal('6.3')
print( a + b ) #6.3
(a + b) == Decimal('6.3') ##True
```

숫자를 문자열로 표현하는게 좀 이상하지만 계산 잘함. 문자열 서식화 함수에 사용하거나 출력하면 일반적인 숫자처럼 보임
—> 반올림 자릿수와 같은 계산적 측면 조절 가능

```
from decimal import localcontext
a = Decimal('1.3')
b = Decimal('1.7')
print( a/ b) #0.76470588235294117647058823529
with localcontext() as ctx:
    ctx.prec = 3 ( round랑 같음, 3번째 자리까지 표현, 4번째 자리에서 반올림)
    print(a/b) #0.765
```

```
nums = [1.23e+18, 1, -1.23e+18]
sum(nums) #0.0 #1 계산 못함

import math
math.fsum(nums) #1.0
```

3.3 출력을 위한 숫자 서식화

[문제] 출력을 위해 자릿수, 정렬, 천 단위 구분 등 숫자 서식화가 필요
[해결] format() 사용
- [<>^]?너비[,]?(.자릿수)? (너비나 자릿수는 정수형, ?는 선택사항) (.format() 메소드에도 동일한 서식 사용)

```
x = 1234.5678
#숫자 둘째 자리 정확도
format(x,'0.2f') #1234.57

#소수점 한 자리 정확도로 문자 10개 기준 오른쪽에서 정렬
print (format(x, '>10.1f')) #     1234.6

#소수점 한 자리 정확도로 문자 10개 기준 왼쪽에서 정렬
print (format(x, '<10.1f')) #1234.6     #

#가운데 정렬
print (format(x, '^10.1f'))

#천 단위 구분자 넣기
print ( format(x,',') ) #1,234.5678
print( format(x,'0,.1f') ) #1,234.6 #숫자를 빼줘도 마음대로 넣어줘도 똑같이 실행되는데 뭐지
```

지수 표현법 사용시 f를 e,E로 변경

```
print (format(x,'e')) #1.234568e+03 #일의자리.인듯?
print( format(x,'0.2E')) #1.23E+03 #소수 둘째자리까지(셋째자리에서 반올림)
```

.format()에도 동일하게 작용

```
'The value is {:0,.2f}'.format(x) #The value is 1,234.57 # : 뭐야
```

천단위 구분자는 지역 표기법을 따르지 않음. ?
locale모듈을 사용해야 함.(translate() 사용)

```
print ( 'The value is {:0,.2f}'.format(x)) #The value is 1,234.57
swap_separators = {ord('.'):',' , ord(','):'.'}
print ( format(x,',').translate(swap_separators)) #1.234.5678

숫자를 %연산자로 서식화(format만큼 기능이 많지는 않음)
print ('%0.2f' %x) #1234.57
print('%10.1f' %x ) #    1234.6
print('%-10.1f' %x) #1234.6    #
```

3.4 2진수, 8진수, 16진수 작업

[문제] 숫자를 2/8/16진수로 출력해야함
[해결] bin(), oct(), hex() 사용

```
x = 1234
bin(x) #0b10011010010
oct(x) #'0o2322'
hex(x)  #'0x4d2'

#숫자앞에 문자를 없애고 싶으면 format() 사용
format(x,'b') #'10011010010'
format(x,'o') #'2322'
format(x,'x') #'4d2'

음수시 -만 붙고 동일
```

부호 없는 값 사용 시 최대값을 더해서 비트 길이를 설정해야 함
ex) 32비트 값
x = -1234
format(2**32 + x ,'b') #'11111111111111111111101100101110'
format(2**32 + x , 'x')# 'fffffb2e'

다른 진법의 숫자를 정수형으로 변환하면 int() 사용
int('숫자모양', '숫자모양의 진법')

```
int('4d2',16)
int('10011010010',2)
```

8진법 사용 시 0o을 붙여야 함
import os
#os.chmod('script.py',0755)
os.chmod('script.py',0o755) #sciptpy없다고 오류뜸

3.9 큰 배열 계산

[문제] 배열이나 그리드와 같이 커다란 숫자 데이터(dataset)에 계산을 해야함
[해결] Numpy라이브러리 사용. 리스트 사용보다 효율적

```
#파이썬 리스트
x= [1,2,3,4]
y = [5,6,7,8]
x*2 #[1, 2, 3, 4, 1, 2, 3, 4]
x+10 #오류남 can only concatenate list (not "int") to list
x+y  #[1, 2, 3, 4, 5, 6, 7, 8] 붙여준거

#Numpy배열
import numpy as np
ax = np.array([1,2,3,4])
ay = np.array([5,6,7,8])
ax * 2 #array([2, 4, 6, 8])
ax +10 #array([11, 12, 13, 14])
ax + ay #array([ 6,  8, 10, 12])
ax* ay #array([ 5, 12, 21, 32]) 각 자리에 있는 애들끼리만 곱해줌 (행렬곱 x)


다항식에도 적용 가능
def f(x):
    return 3*x**2 - 2*x +7
f(ax) #array([ 8, 15, 28, 47])
```

numpy는 배열에 사용가능한 일반항수를 제공.(math모듈이 제공하는 함수와 비슷)

```
np.sqrt(ax)
np.cos(ax)
```

웬만하면 numpy의 일반함수를 사용!

NumPy배열은 동일한 데이터 타입을 메모리에 연속으로 나열함—> 파이썬 리스트보다 훨씬 더 큰 배열을 만들 수 있음

```
grid = np.zeros(shape = (10000,10000) , dtype=float )
grid

array([[ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       ...,
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.],
       [ 0.,  0.,  0., ...,  0.,  0.,  0.]])

grid += 10
np.sin(grid)
```

numpy : 다차원 배열의 인덱싱 기능을 확장하고 있음

```
a = np.array( [ [1,2,3,4],[5,6,7,8],[9,10,11,12]])
a
#array([[ 1,  2,  3,  4],
       [ 5,  6,  7,  8],
       [ 9, 10, 11, 12]])

a[1]
a[:,1] #array([ 2,  6, 10])

a [1:3, 1:3]
# array([[ 6,  7],
#        [10, 11]])

a [1:3, 1:3] + 10
#array([[16, 17],
#       [20, 21]])
a [1:3, 1:3] += 10  #전체에서 해당 부분만 더하기
# array([[ 1,  2,  3,  4],
#        [ 5, 26, 27,  8],
#        [ 9, 30, 31, 12]])

#행 벡터를 모든 행 연산에 적용
a + [100,101,102,103]
# array([[101, 103, 105, 107],
#        [105, 127, 129, 111],
#        [109, 131, 133, 115]])
a # += 10만 되어 있음 행연산은 왜 반영안되는거임?

#조건이 있는 할당
np.where(a<10, a, 10)  #10보다 작은거는 a, 나머지는 10
```

3.10 행렬과 선형 대수 계산

[문제] 행렬 곱셈, 행렬식 찾기, 선형 방적식 풀기 등 행렬이나 선형대수 계산이 필요
[해결] NumPy -> matrix : array랑은 다르게 선형대수 계산법을 따름

```
import numpy as np
m = np.matrix( [[1,-2,3],[0,4,5],[7,8,-9]])
m
# matrix([[ 1, -2,  3],
#         [ 0,  4,  5],
#         [ 7,  8, -9]])
#전치행렬(transpose)
m.T
# matrix([[ 1,  0,  7],
#         [-2,  4,  8],
#         [ 3,  5, -9]])
#역행렬(inverse)
m.I
# matrix([[ 0.33043478, -0.02608696,  0.09565217],
#         [-0.15217391,  0.13043478,  0.02173913],
#         [ 0.12173913,  0.09565217, -0.0173913 ]])
#벡터 만들고 곱하기
v = np.matrix( [[2],[3],[4]])
v
# matrix([[2],
#         [3],
#         [4]])
m*v
# matrix([[ 8],
#         [32],
#         [ 2]])

#numpy.linalg 서브 패키지에 더 많은 연산이 있음
import numpy.linalg
#Determiant
numpy.linalg.det(m) #-229.99999999999983

#Eignevalues
numpy.linalg.eigvals(m) #array([-13.11474312,   2.75956154,   6.35518158])

#mx = v에서 x 풀기
x = numpy.linalg.solve(m,v)
```