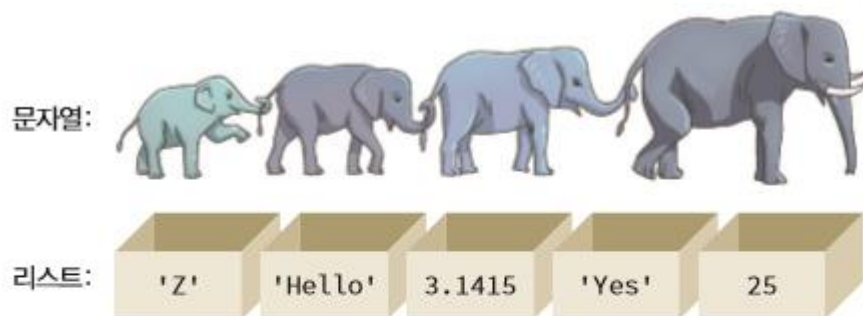


# 7장 튜플, 세트, 딕셔너리, 문자열

# 자료구조란?

- 프로그램에서 자료들을 저장하는 여러 가지 구조들이 있다. 이를 자료 구조(data structure)라 부른다.



# 시퀀스

- 시퀀스에 속하는 자료 구조들은 동일한 연산을 지원한다.
  - 인덱싱(indexing), 슬라이싱(slicing), 덧셈 연산(adding), 곱셈 연산(multiplying)
- 리스트는 앞에서 자세하게 살펴본바 있다. 여기서는 나머지 시퀀스들을 탐구해보자.

# 튜플

- 튜플(tuple)은 변경될 수 없는 리스트

전체적인 구조



튜플 = ( 항목1 , 항목2 , ... , 항목n )

```
>>> colors = ("red", "green", "blue")
```

```
>>> colors
```

```
('red', 'green', 'blue')
```

```
>>> numbers = (1, 2, 3, 4, 5)
```

```
>>> numbers
```

```
(1, 2, 3, 4, 5)
```

# 튜플은 변경할 수 없다

```
>>> t1 = (1, 2, 3, 4, 5);
>>> t1[0] = 100;
Traceback (most recent call last):
  File "<pyshell#11>", line 1, in <module>
    t1[0]=100
TypeError: 'tuple' object does not support item assignment
```

```
>>> numbers = ( 1, 2, 3, 4, 5 )
>>> colors = ("red", "green", "blue")
>>> t = numbers + colors
>>> t
(1, 2, 3, 4, 5, 'red', 'green', 'blue')
```

# 기본적인 튜플 연산들

파이썬 수식	결과	설명
<code>len((1, 2, 3))</code>	3	튜플의 길이
<code>(1, 2, 3) + (4, 5, 6)</code>	<code>(1, 2, 3, 4, 5, 6)</code>	접합
<code>('Hi!') * 4</code>	<code>('Hi!', 'Hi!', 'Hi!', 'Hi!')</code>	반복
<code>3 in (1, 2, 3)</code>	True	멤버십
<code>for x in (1, 2, 3): print x,</code>	1 2 3	반복

함수	설명
<code>cmp(t1, t2)</code>	2개의 튜플을 비교한다.
<code>len(t)</code>	튜플의 길이를 반환한다.
<code>max(t)</code>	튜플에 저장된 최대값을 반환한다.
<code>min(t)</code>	튜플에 저장된 최소값을 반환한다.
<code>tuple(seq)</code>	리스트를 튜플로 변환한다.

# 튜플 대입 연산

```
>>> student1 = ("철수", 19, "CS")
>>> (name, age, major) = student1
>>> name
'철수'
>>> age
19
>>> major
'CS'
```

# Lab: 함수의 튜플 반환 예제

- 원의 넓이와 둘레를 동시에 반환하는 함수를 작성, 테스트해보자.

원의 반지름을 입력하시오: 10

원의 넓이는 314.1592653589793이고 원의 둘레는 62.83185307179586이다  
.



# Solution

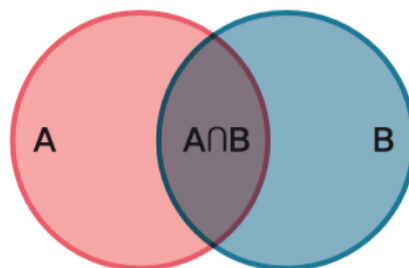
```
import math

def calCircle(r):
    # 반지름이 r인 원의 넓이와 둘레를 동시에 반환하는 함수 (area, circum)
    area = math.pi * r * r
    circum = 2 * math.pi * r
    return (area, circum)

radius = float(input("원의 반지름을 입력하시오: "))
(a, c) = calCircle(radius)
print("원의 넓이는 "+str(a)+"이고 원의 둘레는"+str(c)+"이다.")
```

# 세트(Set)

- 세트(set)는 우리가 수학에서 배웠던 집합이다.
- 세트는 중복되지 않은 항목들이 모인 것
- 세트의 항목 간에는 순서가 없다.



전체적인 구조



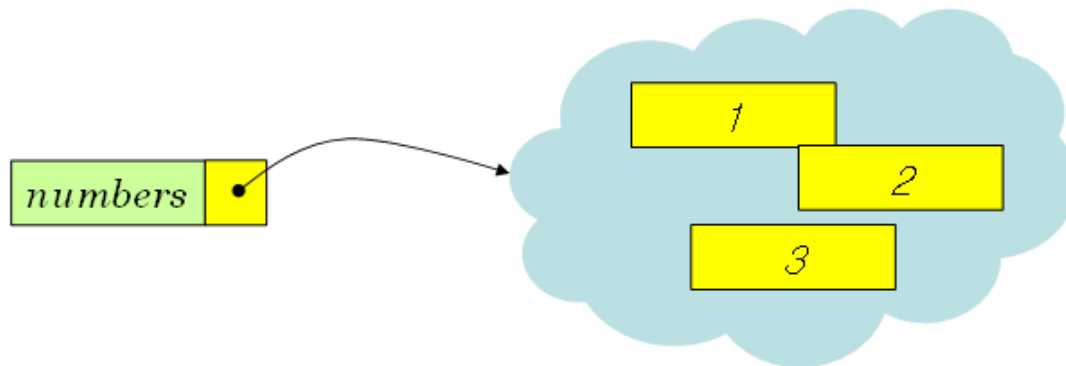
세트 = { 항목1 , 항목2 , ... , 항목n }

# 예제

```
>>> numbers = {2, 1, 3}
>>> numbers
{1, 2, 3}

>>> len(numbers)
3

>>> fruits = { "Apple", "Banana", "Pineapple" }
>>> mySet = { 1.0, 2.0, "Hello World", (1, 2, 3) }
```



# in 연산자

```
>>> numbers = {2, 1, 3}
>>> if 1 in numbers:
    print("집합 안에 1이 있습니다.")
```

집합 안에 1이 있습니다.

```
>>> numbers = {2, 1, 3}
>>> for x in numbers:
    print(x, end=" ")
```

1 2 3

# 세트에 요소 추가하기

```
>>> numbers = { 2, 1, 3 }
>>> numbers[0]
...
TypeError: 'set' object does not support indexing

>>> numbers.add(4)
>>> numbers
{1, 2, 3, 4}
```

# 부분 집합 연산

```
>>> A = {1, 2, 3}
```

```
>>> B = {1, 2, 3}
```

```
>>> A == B
```

```
True
```

```
>>> A = {1, 2, 3, 4, 5}
```

```
>>> B = {1, 2, 3}
```

```
>>> B < A
```

```
True
```

```
>>> A = {1, 2, 3, 4, 5}
```

```
>>> B = {1, 2, 3}
```

```
>>> B.issubset(A)
```

```
True
```

# 집합 연산

```
>>> A = {1, 2, 3}
```

```
>>> B = {3, 4, 5}
```

```
>>> A | B
```

```
{1, 2, 3, 4, 5}
```

```
>>> A & B
```

```
{3}
```

```
>>> A - B
```

```
{1, 2}
```

# Lab: 파티 동시 참석자 알아내기

- 파티에 참석한 사람들의 명단이 세트 A와 B에 각각 저장되어 있다. 2개 파티에 모두 참석한 사람들의 명단을 출력하려면 어떻게 해야 할까?

2개의 파티에 모두 참석한 사람은 다음과 같습니다.  
{ 'Park' }





# Solution

```
partyA = set(["Park", "Kim", "Lee"])
partyB = set(["Park", "Choi"])

print("2개의 파티에 모두 참석한 사람은 다음과 같습니다. ")
print ( partyA.intersection(partyB))
```

# Lab: 파일에서 중복되지 않은 단어의 개수

- 텍스트 파일을 읽어서 단어를 얼마나 다양하게 사용하여 문서를 작성하였는지를 계산하는 프로그램을 작성해보자.

입력 파일 이름: proverbs.txt

사용된 단어의 개수= 18

```
{'travels', 'half', 'that', 'news', 'alls', 'well', 'fast',  
'feather', 'flock', 'bad', 'together', 'ends', 'is', 'a', 'done',  
'begun', 'birds', 'of'}
```

# Solution

```
# 단어에서 구두점을 제거하고 소문자로 만든다.
def process(w):
    output = ""
    for ch in w:
        if( ch.isalpha() ):
            output += ch
    return output.lower()
words = set()

# 파일을 연다.
fname = input("입력 파일 이름: ")
file = open(fname, "r")

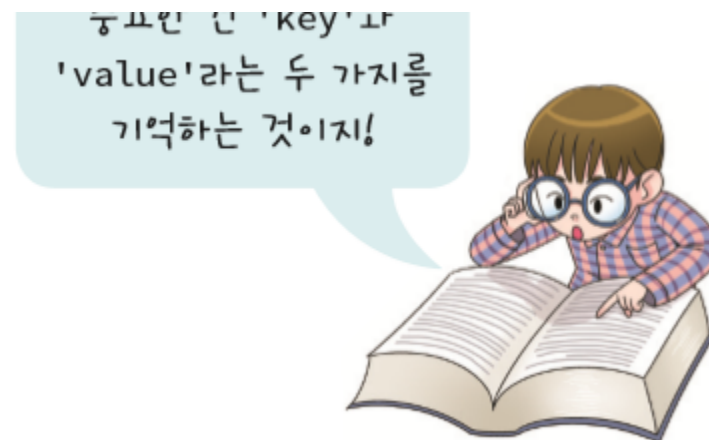
# 파일의 모든 줄에 대하여 반복한다.
for line in file:
    lineWords = line.split()
    for word in lineWords:
        words.add(process(word))        # 단어를 세트에 추가한다.

print("사용된 단어의 개수=", len(words))
print(words)
```

# 딕셔너리

- 딕셔너리는 키(key)와 값(value)의 쌍을 저장할 수 있는 객체

키(key)	값(value)
"Kim"	"01012345678"
"Park"	"01012345679"
"Lee"	"01012345680"



# 딕셔너리 생성

전체적인 구조



딕셔너리 = { 키1 : 값1 , 키2 : 값2 , ... }

```
>>> contacts = {'Kim':'01012345678', 'Park':'01012345679',  
'Lee':'01012345680' }
```

```
>>> contacts  
{'Kim': '01012345678', 'Lee': '01012345680', 'Park': '01012345679'}
```

# 항목 접근하기

```
>>> contacts = {'Kim':'01012345678', 'Park':'01012345679',  
'Lee':'01012345680' }
```

```
>>> contacts['Kim']  
'01012345678'
```

```
>>> contacts.get('Kim')  
'01012345678'
```

```
>>> if "Kim" in contacts:  
    print("키가 딕셔너리에 있음")
```

# 항목 추가 & 삭제하기

```
>>> contacts['Choi'] = '01056781234'
>>> contacts
{'Kim': '01012345678', 'Choi': '01056781234', 'Lee': '01012345680',
'Park': '01012345679'}
```

```
>>> contacts = {'Kim': '01012345678', 'Park': '01012345679',
'Lee': '01012345680' }

>>> contacts.pop("Kim")
'01012345678'

>>> contacts
{'Lee': '01012345680', 'Park': '01012345679'}
```

# 항목 순회하기

```
>>> scores = { 'Korean': 80, 'Math': 90, 'English': 80}  
>>> for item in scores.items():  
    print(item)  
  
('Math', 90)  
('English', 80)  
('Korean', 80)  
>>>
```



# Lab: 영한 사전 만들기

- 우리는 영한 사전을 구현하여 보자. 어떻게 하면 좋은가?  
공백 딕셔너리를 생성하고 여기에 영어 단어를 키로 하고  
설명을 값으로 하여 저장하면 될 것이다.

단어를 입력하시오: one  
하나

단어를 입력하시오: python  
없음

# Solution

```
english_dict = dict()

english_dict['one'] = '하나'
english_dict['two'] = '둘'
english_dict['three'] = '셋'

word = input("단어를 입력하시오: ");
print (english_dict.get(word, "없음"))
```

# Lab: 단어 카운터

- 사용자가 지정하는 파일을 읽어서 파일에 저장된 각각의 단어가 몇 번이나 나오는지 계산하는 프로그램을 작성하여 보자.

파일 이름: proverbs.txt

```
{'a': 1, 'done.': 1, 'that': 1, 'well.': 1, 'ends': 1, 'Well': 1, 'flock': 1, 'feather': 1, "All's": 1, 'Birds': 1, 'together.': 1, 'of': 1, 'fast.': 1, 'begun': 1, 'half': 1, 'well': 1, 'travels': 1, 'news': 1, 'is': 1, 'Bad': 1}
```

# Solution

```
fname = input("파일 이름: ")
file = open(fname, "r")

table = dict()
for line in file:
    words = line.split()
    for word in words:
        if word not in table:
            table[word] = 1
        else:
            table[word] += 1

print(table)
```

# Lab: 축약어 풀어쓰기

- 현대인들은 축약어를 많이 사용한다. 예를 들어서 "B4(Before)" "TX(Thanks)" "BBL(Be Back Later)" "BCNU(Be Seeing You)" "HAND(Have A Nice Day)"와 같은 축약어들이 있다. 축약어를 풀어서 일반적인 문장으로 변환하는 프로그램을 작성하여 보자.

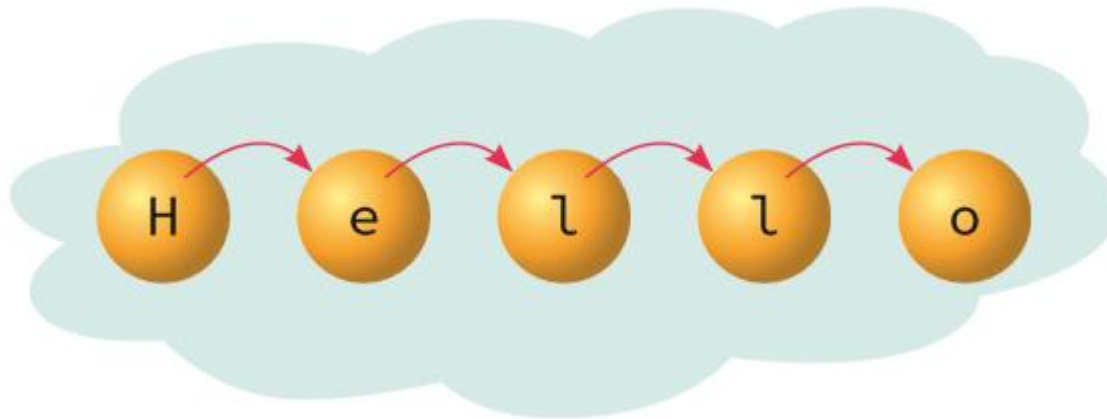
번역할 문장을 입력하시오: TX Mr. Park!  
Thanks Mr.Park!

# Solution

```
table = { "B4": "Before",  
          "TX": "Thanks",  
          "BBL": "Be Back Later",  
          "BCNU": "Be Seeing You",  
          "HAND": "Have A Nice Day" }  
  
message = input('번역할 문장을 입력하시오: ')  
words = message.split()  
result = ""  
for word in words:  
    if word in table:  
        result += table[word] + " "  
    else:  
        result += word  
  
print(result)
```

# 문자열

- 문자열은 문자들의 시퀀스로 정의된다. 글자들이 실 (string)로 묶여 있는 것이 문자열이라고 생각하면 된다.



# 문자열의 예

```
s1 = str("Hello")  
s2 = "Hello"  
  
s1 = "Hello"  
s2 = "World"  
s3 = "Hello"+"World"
```



# 개별 문자 접근하기

```
>>> word = 'abcdef'
>>> word[0]
'a'
>>> word[5]
'f'
```

a	b	c	d	e	f
0	1	2	3	4	5
-6	-5	-4	-3	-2	-1

음수 인덱스도 굉장히 편리  
하답니다. 꼭 기억해두세요!



# 슬라이싱

```
>>> word = 'Python'  
>>> word[0:2]  
'Py'  
>>> word[2:5]  
'tho'
```



슬라이싱은 문자열의 일부를  
추출하는 기능입니다.



# 슬라이싱의 예

```
>>> word = 'Python'
>>> word[0:2]
'Py'
>>> word[2:5]
'tho'
```

# in 연산자와 not in 연산자

```
>>> s="Love will find a way."  
>>> "Love" in s  
True  
>>> "love" in s  
False
```

```
s = input("문자열을 입력하시오")  
if 'c' in s:  
    print('c가 포함되어 있음')  
else:  
    print('c가 포함되어 있지 않음')
```

# 문자열 비교하기

```
a = input("문자열을 입력하시오: ")
b = input("문자열을 입력하시오: ")
if( a < b ):
    print(a, "가 앞에 있음")
else:
    print(b, "가 앞에 있음")
```

```
문자열을 입력하시오: apple
문자열을 입력하시오: orange
apple 가 앞에 있음
```

# 문자열에서 단어 분리

```
>>> s = 'Never put off till tomorrow what you can do today.'  
>>> s.split()  
['Never', 'put', 'off', 'till', 'tomorrow', 'what', 'you', 'can', 'do',  
'today.']
```

# Lab: 회문 검사하기

- 회문(palindrome)은 앞으로 읽으나 뒤로 읽으나 동일한 문장이다. 예를 들어서 "mom", "civic", "dad" 등이 회문의 예이다. 사용자로부터 문자열을 입력받고 회문인지를 검사하는 프로그램을 작성하여 보자.

문자열을 입력하시오: dad  
회문입니다.

# Solution

```
def check_pal(s):
    low = 0
    high = len(s) - 1

    while True:
        if low > high:
            return True;

        a = s[low]
        b = s[high]

        if a != b:
            return False
        low += 1
        high -= 1

s = input("문자열을 입력하시오: ")
s = s.replace(" ", "")

if( check_pal(s)==True):
    print("회문입니다.")
else:
    print("회문이 아닙니다.")
```



# Lab: 머리 글자어 만들기

- 머리 글자어(acronym)은 NATO(North Atlantic Treaty Organization)처럼 각 단어의 첫글자를 모아서 만든 문자열이다. 사용자가 문장을 입력하면 해당되는 머리 글자어를 출력하는 프로그램을 작성하여 보자.

문자열을 입력하시오: North Atlantic Treaty Organization  
NATO

# Solution

```
phrase = input("문자열을 입력하시오: ")

acronym = ""
for word in phrase.upper().split():
    acronym += word[0]

print( acronym )
```

# Lab: CSV 파일 분석

- CSV 파일은 `split()` 메소드를 이용하여 파싱될 수 있다.  
`open()`, `readlines()`, `strip()` 메소드를 사용하여서 다음과 같은 CSV 파일에서 데이터를 읽는 프로그램을 작성하여 보자.

```
홍길동,2018,생활관A,312,3.18
홍길동
2018
생활관A
312
3.18
김철수,2017,생활관B,102,3.25
김철수
2017
생활관B
102
3.25
...
```

# Solution

```
# 파일을 오픈한다.  
  
f = open("E:\\students.txt", "r")  
  
# 파일의 각 줄에 대하여 반복한다.  
for line in f.readlines():  
  
    # 공백 문자를 제거한다.  
    line = line.strip()  
  
    # 줄을 출력한다.  
    print(line)  
  
    # 줄을 단어로 분리한다.  
    words = line.split(",")  
  
    # 줄의 단어를 출력한다.  
    for word in words:  
        print("    ", word)
```

# Lab: 문자열 분석

- 문자열 안에 있는 문자의 개수, 숫자의 개수, 공백의 개수를 계산하는 프로그램을 작성하여 보자.

문자열을 입력하시오: *A picture is worth a thousand words.*  
{'digits': 0, 'spaces': 6, 'alphas': 29}

# Solution

```
sentence = input("문자열을 입력하시오: ")

table = { "alphas": 0, "digits":0, "spaces": 0 }

for i in sentence:
    if i.isalpha():
        table["alphas"] += 1
    if i.isdigit():
        table["digits"] += 1
    if i.isspace():
        table["spaces"] += 1

print(table)
```

# 핵심 정리

- 튜플은 리스트와 유사하지만 변경할 수 없는 객체이다.
- 세트는 집합으로 요소들은 중복되지 않아야 한다.
- 딕셔너리는 사전으로 키와 값의 쌍으로 저장된다. 키를 이용하여 값을 찾을 수 있다.

# Q & A

