데이터처리프로그래밍

Data Structure

강원대학교 교육혁신원 송혜정

<hjsong@kangwon.ac.kr>

Data Structure

✓ 학습목표
- 데이터구조를 이해하고 활용한다.

✓ 학습내용
- Stack
- Queue
- Set
- Dictionary

강의에 앞서서..

- 본 강의자료는 아래의 자료들을 참고하여 만들어 졌음을 알립니다

1. 데이터과학을 위한 파이썬 프로그래밍, 최성철, 한빛아카데미,2019
2. Python (https://docs.python.org)
3. 점프 투 파이썬 (https://wikidocs.net/book/1)
4. 파이썬 for Beginner, 우재남, 한빛아카데미
5. wikipedia.org

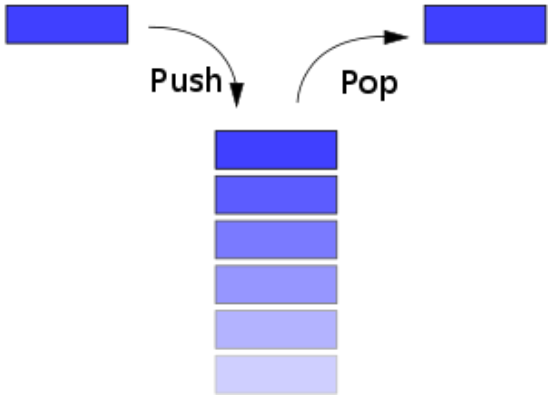
자료구조

- 자료구조(Data Structure)
  - 데이터를 효율적으로 관리하기 위한 다양한 구조 제공
  - 데이터 저장, 추출, 수정 등의 처리를 제공

자료구조명	특징
스택(stack)	나중에 들어온 값을 먼저 나갈 수 있도록 해 주는 자료구조(last in first out)
큐(queue)	먼저 들어온 값을 먼저 나갈 수 있도록 해 주는 자료구조(first in first out)
튜플(tuple)	리스트와 같지만, 데이터의 변경을 허용하지 않는 자료구조
세트(set)	데이터의 중복을 허용하지 않고, 수학의 집합 연산을 지원하는 자료구조
딕셔너리(dictionary)	전화번호부와 같이 키(key)와 값(value) 형태의 데이터를 저장하는 자료구조, 여기서 키값은 다른 데이터와 중복을 허용하지 않음
collections 모듈	위에 열거된 여러 자료구조를 효율적으로 사용할 수 있도록 지원하는 파이썬 내장(built-in) 모듈

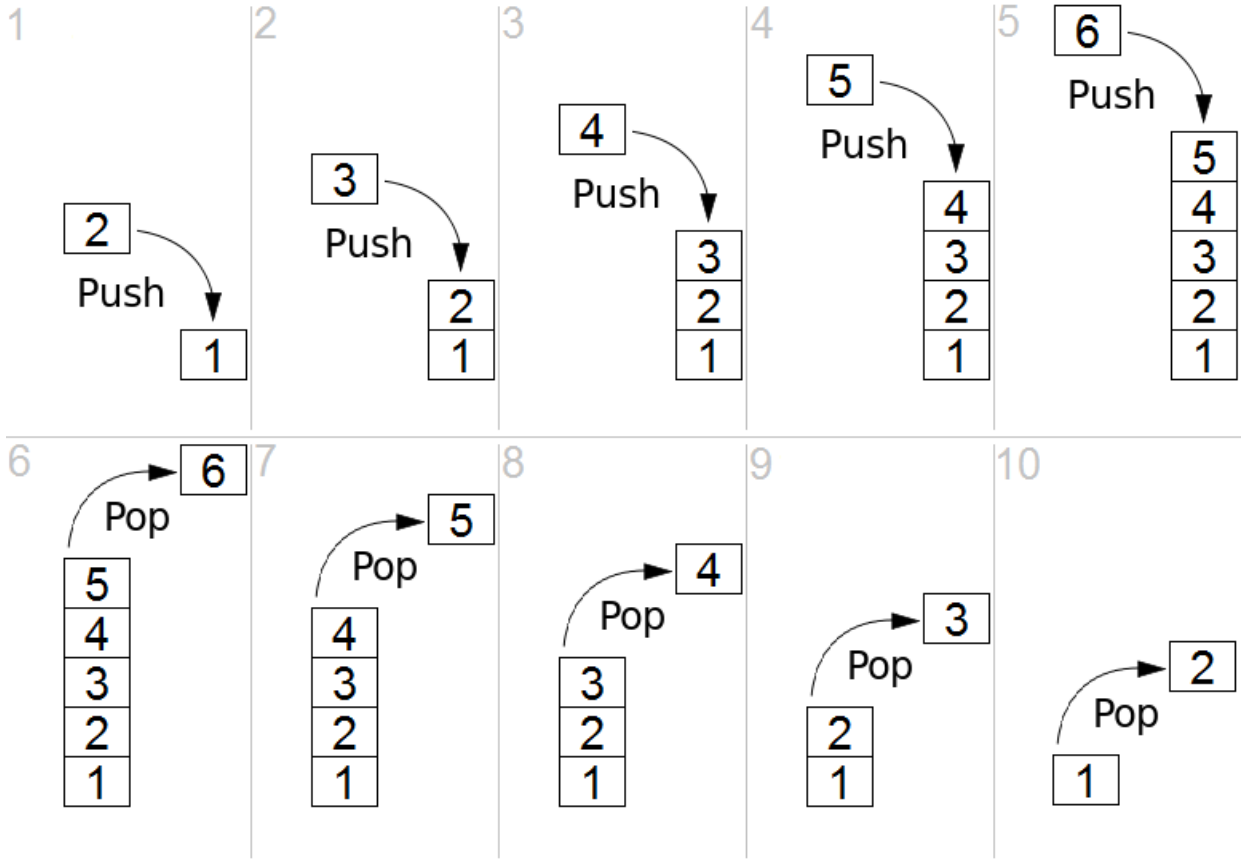
Stack

- 스택(stack)
  - 접근이 한쪽으로 제한적인 리스트 구조
  - LIFO (Last In First Out)구조
  - 마지막에 들어간 데이터가 가장 먼저 나오는 형태로 데이터의 저장 공간을 구현
  - 연산
    - Push : 스택에 데이터를 추가 (insert)
    - Pop : 스택의 맨 위 데이터 추출 (delete)
  - Overflow : 스택 공간이 차서 더 이상 push를 할 수 없는 상태



Stack

- Stack push, pop



Stack

- 파이썬의 리스트를 사용하여 stack 구현

```
#stack push, pop
s=[1,5, 3]
print('s=',s)
s.append(10)    #push  처리
s.append(20)
print('s=',s)
p = s.pop()    #pop 처리
print('s=',s)
print('p=',p)
```

```
s= [1, 5, 3]
s= [1, 5, 3, 10, 20]
s= [1, 5, 3, 10]
p= 20
```

```
#stack pop을 이용한 reverse
s=[1,5, 3, 9, 2]
r=[]
l = range(len(s))    #s의 길이만큼 반복
for i in l:
    p = s.pop()
    r.append(p)
print('s=',s)
print('r=',r)
```

```
s= []
r= [2, 9, 3, 5, 1]
```

Stack

- 예제1. 스택 형태의 주차공간의 주차 및 출차 처리 문제
  - 주차공간 : 스택 구조의 리스트 최대 주차 가능 대수 : 5
  - 입력 : 주차, 출차 메뉴를 선택
  - 처리 : 메뉴를 입력하여 반복 처리
    - 주차 : 차량번호로 스택 주차장에 추가 , 주차가능공간이 없으면 메시지출력
    - 출차 : 차량번호로 스택 주차장에서 추출 ,
      차량번호가 있으면 해당차량을 꺼내기 위해
      주차차량을 반복처리로 꺼내어 임시공간(stack)에 저장했다가
      다시 주차공간으로 추가

<출력예시>

```
다음을 선택하세요. (0.종료 1.주차  2.출차 3.주차공간확인)  ? 1
차량번호를 입력하세요 ? 1010
차량번호  1010  주차완료!  주차대수 :  1
다음을 선택하세요. (0.종료 1.주차  2.출차 3.주차공간확인)  ? 1
차량번호를 입력하세요 ? 2020
차량번호  2020  주차완료!  주차대수 :  2
다음을 선택하세요. (0.종료 1.주차  2.출차 3.주차공간확인)  ? 1
차량번호를 입력하세요 ? 3030
차량번호  3030  주차완료!  주차대수 :  3
다음을 선택하세요. (0.종료 1.주차  2.출차 3.주차공간확인)  ? 4
입력값은 0, 1, 2, 3 만 가능합니다.
```

```
다음을 선택하세요. (0.종료 1.주차  2.출차 3.주차공간확인)  ? 3
 * 주차공간 현황 *
주차가능 공간 :  0
주차차량 대수 :  5
주차차량 :
1010
2020
3030
4040
5050
다음을 선택하세요. (0.종료 1.주차  2.출차 3.주차공간확인)  ? 2
차량번호를 입력하세요 ? 2020
차량번호  2020  출차합니다.
3 대 차량 재주차 완료!
```

Stack

- 예제1. 스택 형태의 주차공간의 주차 및 출차 처리 문제

```
parking = []    #주차 공간 (stack)
m = 5           #최대 주차 대수
c = 0           #현재 주차 대수

while True:
    s = input("다음을 선택하세요. (0.종료 1.주차  2.출차 3.주차공간확인)  ? ")
    if (s == '0'):
        print("종료합니다.")
        break
    elif (s == '1'):  #주차 처리
        if (c == m):
            print("주차공간이 없습니다.")
        else:
            n = input("차량번호를 입력하세요 ? ")
            parking.append(n)  #주차공간에 추가
            c += 1             #주차대수 증가
            print("차량번호 ", n, " 주차완료!", " 주차대수 : ", c)
    elif (s == '2'):           #출차 처리
        if (c == 0):
            print("주차 차량이 없습니다.")
```

Stack

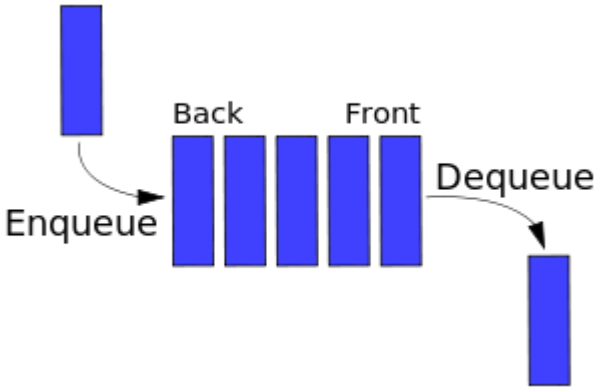
- 예제1. 스택 형태의 주차공간의 주차 및 출차 처리 문제

```
        else:       #출차 처리
            n = input("차량번호를 입력하세요 ? ")
            tmp = []
            if (n in parking):          #차량번호가 parking안에 있다면
                p = parking.pop()       #주차공간에서 차량 1대 추출
                c -= 1                  #주차대수 감소
                while (p != n):         #차량번호가 같을때까지 반복
                    tmp.append(p)       #꺼낸 차량을 임시 공간에 저장
                    p = parking.pop()   #주차공간에서 다음차량 추출
                    c -= 1              #주차대수 감소
                print("차량번호 ", n, " 출차합니다.")
                #임시공간에 저장된 차량 재주차, 마지막꺼낸 차량을 먼저 주자 (임시공간도 stack)
                l = len(tmp)
                for i in range(l):
                    parking.append(tmp.pop())  #temp 임시공간 차량을 꺼내어 주차공간에 저장
                    c += 1
                print(l, "대 차량 재주차 완료!")
            else:                       #차량번호가 parking안에 없다면
                print("차량번호", n, "은 주차 차량이 아닙니다.")

    elif s == '3':
        print(" * 주차공간 현황 *")
        print("주차가능 공간 : ", m - c)
        print("주차차량 대수 : ", c)
        print("주차차량 : ")
        for t in parking:
            print(t)
    else:
        print("입력값은 0, 1, 2, 3 만 가능합니다.")
```

Queue

- 큐(queue)
  - 먼저 입력된 데이터가 먼저 나오는 구조
  - FIFO (First In First Out)구조
  - 연산
    - Put : 큐의 rear에 데이터 추가 (insert)
    - Get : 큐의 front에서 데이터 추출 (delete)
  - 큐의 구조
    - front(head) : 데이터를 get할 수 있는 위치
    - rear(back, tail) : 데이터를 put할 수 있는 위치



Queue

- Queue put, get

PUT(A)	PUT(B)	PUT(C)	GET( )	PUT(D)	GET( )
				D	D
		C	C	C	C
	B	B	B	B	
A	A	A			

- 파이썬의 리스트를 사용하여 queue 구현

```
#queue put, get
s=[]
print('s=',s)
s.append('A')   #put 처리
s.append('B')   #put 처리
s.append('C')   #put 처리
print('s=',s)
p = s.pop(0)   #get 처리, pop(0) : 맨앞의 데이터 추출후 삭제
print('s=',s)
print('p=',p)
s.append('D')   #put 처리
print('s=',s)
p = s.pop(0)   #get 처리, pop(0) : 맨앞의 데이터 추출후 삭제
print('s=',s)
print('p=',p)
```

```
s= []
s= ['A', 'B', 'C']
s= ['B', 'C']
p= A
s= ['B', 'C', 'D']
s= ['C', 'D']
p= B
```

Set

- 값을 순서 없이 저장하면서 중복을 허용하지 않는 자료구조
- 삭제, 변경이 가능하며, 다양한 집합 연산을 제공
- set( ) 함수를 사용하여 리스트나 튜플의 데이터를 입력하여 생성

```
#리스트, 튜플을  이용하여 세트 생성
s1=set([1,2,5,4,2])  #리스트의 중복을 제거하고 세트가 생성
print('s1=',s1)
s2=set((3,7,2,4,2))  #튜플의 중복을 제거하고 세트가 생성
print('s2=',s2)  #
```

```
s1= {1, 2, 4, 5}
s2= {2, 3, 4, 7}
```

Set

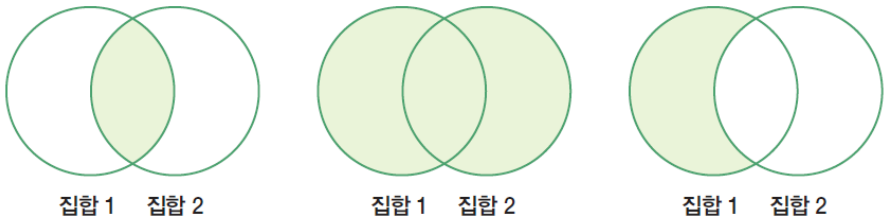
- set 함수
  - add( ) : 데이터 추가
  - remove( ) 또는 discard( ) : 데이터 제거
  - update( ) : 새로운 리스트를 그대로 추가
  - clear( ) : 모든 데이터를 삭제

```
# set 함수
s1.add(10)          #추가
print('s1=',s1)
s1.remove(10)       #삭제
print('s1=',s1)
s1.update(s2)       #수정, set s2 추가, 중복제거
print('s1=',s1)
s1.update([5,9])    #수정, list 추가, 중복제거
print('s1=',s1)
s1.clear()          #모두 삭제
print('s1=',s1)
```

```
s1= {1, 2, 4, 5, 10}
s1= {1, 2, 4, 5}
s1= {1, 2, 3, 4, 5, 7}
s1= {1, 2, 3, 4, 5, 7, 9}
s1= set()
```

Set

- 집합 연산
  - Union : 합집합, 두 집합을 하나로 결합
  - Intersection : 교집합, 두 집합 중 공통적인 부분만 추출
  - Difference : 차집합, 집합1에서 집합 2의 내용을 제거



(a) 교집합 (b) 합집합 (c) 차집합

연산	함수	기호	예시
합집합	union	|	s1.union(s2), s1 | s2
교집합	intersection	&	s1.intersection(s2), s1 & s2
차집합	difference	–	s1.difference(s2), s1 – s2

Set

- 집합 연산

```
#집합연산
s1=set([1,2,3])  #리스트로 세트 생성
s2=set([3, 4,5])  #리스트로 세트 생성
u = s1 | s2   #합집합, union
i = s1 & s2   #교집합, intersection
d = s1 - s2   #차집합, difference
print('s1=',s1)
print('s2=',s2)
print('union=',u)
print('intersection=',i)
print('difference=',d)
```

```
s1= {1, 2, 3}
s2= {3, 4, 5}
union= {1, 2, 3, 4, 5}
intersection= {3}
difference= {1, 2}
```