Lecture 1 : 키와 값의 쌍인 딕셔너리

1. 딕셔너리의 이해

- 사전과 같이 키(key)와 값(value)의 쌍인 항목의 모임(collection)
- 키는 수정될 수 없고, 값은 수정 가능
- 항목의 순서는 의미 없고, 키는 중복 불가

> { key-1: value-1, key-2: value-2, ..., key-n: value-n }

- 키(key)는 수정 불가능한(immutable) 객체만 가능

> dict.keys()

- 값(value)은 모든 자료형 가능

> dict.values()

딕셔너리 생성과 참조 / LAB-1

번호	사용 예	출력(결과)	설 명
1	my_car = {}		
	print(my_car)	{}	빈 딕셔너리 생성
2	my_car = {'brand': '현대', 'model': '제네시스'}		
	print(my_car)	{'brand': '현대', 'model': '제네시스'}	초기화 딕셔너리 생성
3	lecture = {}		
	lecture['강좌명'] = '파이썬'		'강좌명'-'파이썬' 항목 추가
	print(lecture)	{'강좌명': '파이썬'}	
	lecture['학점시수'] = (3, 3)		'학점시수명'-'(3, 3)' 항목 추가
	print(lecture)	{'강좌명': '파이썬', '학점시수': (3, 3)}	
4	my_car = {'brand': '현대', 'model': '제네시스'}		
	print(my_car)	{'brand': '현대', 'model': '제네시스'}	초기화 딕셔너리 생성
	my_car['year'] = 2025		'year'-2025 항목 추가
	print(my_car)	{'brand': '현대', 'model': '제네시스', 'year': 2025}	

번호	사용 예	출력(결과)	설 명
	my_car['model'] = '그랜저'		'model' 값을 '그랜저'로 수정
	print(my_car)	{'brand': '현대', 'model': '그랜저', 'year': 2025}	
5	data = {}		
	data[(1, 2)] = ('JAN', 'FEB')		튜플 (1, 2)는 수정 불가 개체, 키 가능
	print(data)	{(1, 2): ('JAN', 'FEB')}	
6	data = {}		
	data[[1, 2]] = ('JAN', 'FEB')		리스트 [1, 2]는 수정 가능 개체, 키 불가능
	print(data)	unhashable type: 'list	
7	my_car = dict((['brand', '현대'], ['model', '제네시스']))		
	print(my_car)	{'brand': '현대', 'model': '제네시스'}	초기화 딕셔너리 생성

LAB-1 딕셔너리 생성과 참조

In [ ]:
```
my_car = {}
print(my_car)
```

In [ ]:
```
my_car = {'brand': '현대', 'model': '제네시스'}
print(my_car)
```

In [ ]:
```
lecture = {}
lecture['강좌명'] = '파이썬'
print(lecture)
lecture['학점시수'] = (3, 3)
print(lecture)
```

In [ ]:
```
my_car = {'brand': '현대', 'model': '제네시스'}
print(my_car)
my_car['year'] = 2025
print(my_car)
my_car['model'] = '그랜저'
print(my_car)
```

In [ ]:
```
data = {}
data[(1, 2)] = ('JAN', 'FEB')
print(data)
```

In [ ]:
```
data = {}
data[[1, 2]] = ('JAN', 'FEB')
print(data)
```

In [ ]:
```
my_car = dict(brand='현대', model='제네시스')
print(my_car)
```

In [ ]:
```
my_car = dict((['brand', '현대'], ['model', '제네시스']))
print(my_car)
```

2. 딕셔너리 아이템(키, 값) 참조 메서드 / LAB-2

번호	사용 예	출력(결과)	설 명
1	city = dict(경기='수원', 전북='전주')		dict() 함수로 생성
	city	{'경기': '수원', '전북': '전주'}	
	for key in city.keys() :		dict() 키값 추출
	print(key, end=' ')	경기 전북	키값 출력
2	city = dict(경기='수원', 전북='전주')		
	for key in city :		dict() 키값 추출
	print(key, end=' ')	경기 전북	키값 출력
3	city = dict(경기='수원', 전북='전주')		
	for value in city.values() :		dict() 값 추출
	print(value, end=' ')	수원 전주	값 출력
4	city = dict(경기='수원', 전북='전주')		
	for item in city.items() :		dict() 값 추출
	print(f'{item})	('경기', '수원') ('전북', '전주')	값 출력

LAB-2 딕셔너리 아이템(키, 값) 참조 메서드

In [ ]:
```
city = dict(경기='수원', 전북='전주')
print(city)
for key in city.keys() :
    print(key, end=' ')
```

In [ ]:
```
city = dict(경기='수원', 전북='전주')
for key in city :
    print(key, end=' ')
```

In [ ]:
```
city = dict(경기='수원', 전북='전주')
for value in city.values() :
    print(value, end=' ')
```

city = dict(경기='수원', 전북='전주') for item in city.items() : print(f'{item}', end=' ')

3. 딕셔너리 아이템(키, 값) 삭제 메서드 / LAB-3

번호	사용 예	출력(결과)	설 명
1	cities = {'대한민국':['서울', '부산'], '캐나다':['몬트리올', '오타와']}		dict() 생성
	print(cities)	{'대한민국': ['서울', '부산'], '캐나다': ['몬트리올', '오타와']}	
	print(cities.pop('캐나다'))	['몬트리올', '오타와']	키 '캐나다' 항목을 삭제하고, 값을 반환
	print(cities)	{'대한민국': ['서울', '부산']}	키 '캐나다' 항목이 삭죄된 것 확인
2	cities = {'대한민국':['서울', '부산'], '캐나다':['몬트리올', '오타와']}		dict() 생성
	print(cities)	{'대한민국': ['서울', '부산'], '캐나다': ['몬트리올', '오타와']}	
	print(cities.popitem())	['몬트리올', '오타와']	마지막 항목을 삭제하고, 값을 반환
	print(cities)	{'대한민국': ['서울', '부산']}	키 '캐나다' 항목이 삭제된 것 확인
3	cities = {'대한민국':['서울', '부산'], '캐나다':['몬트리올', '오타와']}		dict() 생성
	print(cities)	{'대한민국': ['서울', '부산'], '캐나다': ['몬트리올', '오타와']}	
	cities.clear()		모든 항목을 삭제
	print(cities)	{}	모든 항목이 삭제된 것 확인

LAB-3 딕셔너리 아이템(키, 값) 삭제 메서드

In [ ]:
```
cities = {'대한민국':['서울', '부산'], '캐나다':['몬트리올', '오타와']}
print(cities)
print(cities.pop('캐나다'))
print(cities)
```

In [ ]:
```
cities = {'대한민국':['서울', '부산'], '캐나다':['몬트리올', '오타와']}
print(cities)
print(cities.popitem())
print(cities)
```

In [ ]:
```
cities = {'대한민국':['서울', '부산'], '캐나다':['몬트리올', '오타와']}
print(cities)
```

```
cities.clear()
print(cities)
```

4. 딕셔너리의 내장 함수 활용 등 / LAB-4

번호	사용 예	출력(결과)	설 명
1	stock = {'GOOGLE':849, 'AAPL':136, 'AMZN':848}		
	print(len(stock))	3	항목수
2	score = {'국어':84, '영어':76, '수학':91}		
	print(max(score.values(), max.keys())	3	값 중 최댓값, 키중 최댓값 출력
3	score = {'국어':84, '영어':76, '수학':91}		
	'영어' in score	True	키 중 '영어'가 있는지?
4	score = {'국어':84, '영어':76, '수학':91}		
	'사회' not in score	True	키 중 '사회'가 없는지?

LAB-4 딕셔너리의 내장 함수 활용 등

In [ ]:
```
stock = {'GOOGLE':849, 'AAPL':136, 'AMZN':848}
print(len(stock))
```

In [ ]:
```
score = {'국어':84, '영어':76, '수학':91}
print(max(score.values()), max(score.keys()))
```

In [ ]:
```
score = {'국어':84, '영어':76, '수학':91}
'영어' in score
```

In [ ]:
```
score = {'국어':84, '영어':76, '수학':91}
'사회' not in score
```

LAB-5 수행 과제

1. 문제 : 로그 데이터 분석

- 회사는 특정 기간 동안의 웹 요청 로그를 처리해야 합니다. 로그는 사용자 ID, 요청 URL, 요청 상태 코드, 요청 시간을 포함합니다. 아래 조건에 따라 로그 데이터를 분석하세요.

2. 요구사항:

- 로그 데이터는 딕셔너리로 제공됩니다. 각 항목은 사용자 ID를 키로, 값은 해당 사용자의 요청 기록 목록입니다.
- 각 요청 기록은 URL, 상태 코드, 요청 시간을 포함한 튜플 형식입니다.

3. 목표:

- 사용자별 총 요청 수를 계산합니다.
- 500번대 상태 코드(서버 오류)가 발생한 사용자와 해당 요청 수를 출력합니다.
- 특정 URL(/admin)에 접근한 사용자와 그 요청 수를 출력합니다.

4. 제공된 데이터:

```
logs = {
    "user1": [("/home", 200, "2025-01-20T10:00:00"),
              ("/admin", 500, "2025-01-20T10:10:00"),
              ("/dashboard", 200, "2025-01-20T10:20:00")],
    "user2": [("/home", 200, "2025-01-20T11:00:00"),
              ("/admin", 403, "2025-01-20T11:15:00"),
              ("/settings", 500, "2025-01-20T11:30:00")],
    "user3": [("/home", 200, "2025-01-20T12:00:00"),
              ("/dashboard", 200, "2025-01-20T12:20:00")]
}
```

In [ ]:

```
# 사용자별 로그 데이터
logs = {
    "user1": [("/home", 200, "2025-01-20T10:00:00"),
              ("/admin", 500, "2025-01-20T10:10:00"),
              ("/dashboard", 200, "2025-01-20T10:20:00")],
    "user2": [("/home", 200, "2025-01-20T11:00:00"),
              ("/admin", 403, "2025-01-20T11:15:00"),
              ("/settings", 500, "2025-01-20T11:30:00")],
    "user3": [("/home", 200, "2025-01-20T12:00:00"),
              ("/dashboard", 200, "2025-01-20T12:20:00")]
}

# 1. 사용자별 총 요청 수 계산
total_requests = {}
for user, requests in logs.items() :
    total_requests[user] = len(requests)

# 2. 500번대 상태 코드 발생 사용자와 횟수 계산
error_requests = {}
for user, requests in logs.items():
    error_count = 0
    for url, status, time in requests:
        if 500 <= status < 600:
            error_count += 1

    # 500번대 오류가 있는 사용자만 딕셔너리에 추가
    if error_count > 0:
        error_requests[user] = error_count

# 3. 특정 URL('/admin')에 접근한 사용자와 횟수
admin_access = {}
for user, requests in logs.items():
    # '/admin' URL 접근 횟수 계산
    # 코딩하시오
    #
    #
    #

# 결과 출력
```

```
print("1. 사용자별 총 요청 수:", total_requests)
print("2. 500번대 오류 발생 사용자와 횟수:", error_requests)
print("3. '/admin' URL 접근 사용자와 횟수:", admin_access)
```

In [ ]: