
PYTHON

Python
Programming

10

- 담당교수 : 최희식
- eMail :choihs3054@seoultech.ac.kr

강의 내용

■ 학습목표

- 튜플에 이어 다양한 자료형에 대해 학습한다
- 파이썬 튜플, 다중 리스트, 딕셔너리 자료형에 대해 학습하고 이해한다.

■ 학습목차

- 튜플 살펴보기
- 파이썬 다중 리스트
- 딕셔너리

1

이번
차시에서는

- 튜플 구조

튜플 살펴보기

■튜플(tuple)

- 튜플은 리스트와 유사하게 여러 데이터를 담을 수 있다.
- 튜플은 한 번 생성된 항목 자료를 변경할 수 없다는 특징을 갖고 있다.
- 튜플은 간단하게 () 소괄호를 사용하여 튜플 생성을 초기화 할 수도 있다.
- 또한, 리스트 형태로 만들어진 자료형을 tuple() 함수를 사용하여 튜플로 전환하여 사용할 수 있다.

튜플 변수=(항목1, 항목2, 항목3...)

튜플 살펴보기

■튜플 변수

- 튜플 변수에는 숫자형 데이터, 문자열 데이터, 숫자와 문자열 혼합형 데이터 항목과 같은 여러 형태의 다양한 자료 항목도 저장할 수 있다.

```
① numList =()                                  #빈 튜플
② numList = (10, 20, 30, 40, 50)               #숫자 튜플 변수
③ strList=(“SBS”, “KBS”, “MBC”, “TVN”, “ICON”)  #문자열 튜플 변수
④ milkiest = (10, "KBS", 120.15, (1,2,3))      #혼합형 튜플 변수
```

튜플 살펴보기

■빈 튜플 생성

- 빈 튜플 생성 이후, 데이터 추가는 어렵지만 튜플 이용은 가능하다.

```
>>> num=()
>>> num
()
```

- 튜플은 리스트와는 다르게 빈 튜플을 생성 후 append(), insert() 함수를 이용하여 데이터 추가가 불가능하며, sort(), clear(), extend() 함수 사용도 할 수 없다.

튜플 살펴보기

■튜플 생성

- 튜플을 생성할 때 () 소괄호를 이용한다.

```
>>> num=(1,2,3)
>>> type(num)
<class 'tuple'>
```

- 튜플을 생성할 때 () 소괄호를 생략할 수 있다

```
>>> num=1,2,3
>>> type(num)
<class 'tuple'>
```

튜플 살펴보기

■리스트에서 튜플로 전환하기

- 리스트 변수를 만든 후, 파이썬 내장 함수 tuple() 함수를 사용하여 튜플을 생성하는 경우이다.

```
>>> str=["a","b","c"]
>>> str1=tuple(str)
>>> type(str1)
<class 'tuple'>
```

튜플 살펴보기

■튜플 항목에 또 다른 튜플 포함하기

- 튜플 항목에 또 다른 튜플을 포함하여 튜플 생성이 가능하다.

```
>>> mixStr=(1,2,3,("a","b","c"))
>>> type(mixStr)
<class 'tuple>
```

튜플 살펴보기

■튜플 1개 생성하기

- 튜플을 작성할 때 항목이 1 또는 (1)과 같이 작성하게 되면, 파이썬에서는 1이라는 숫자 정수로 인식된다.
- 튜플 1개를 생성하기 위해서는 반드시 (1,)과 같이 마지막에 , (콤마)를 넣어 주어야 튜플로 생성될 수 있다.

```
>>> num=(1)                #튜플 1개 항목 만들 때 ,(콤마) 생략 시 숫자 정수로 인식 됨
>>> type(num)
<class 'int'>
```

```
>>> num1=(1,)              #튜플로 1개의 항목을 만들 때 반드시 ,(콤마) 까지 사용해야 함
>>> type(num1)
<class 'tuple'>
```

튜플 살펴보기

■튜플 항목 추가하기

- flower 튜플에 새로운 항목 "민들레" 꽃을 추가해보자.
- 튜플에서 바로 항목을 추가할 수 없으므로 항목 추가도 마찬가지로 list()함수를 이용하여 먼저 리스트로 변환 후, append()함수를 이용하여 항목을 추가한다.

```
>>> flower=("들국화","채송와","봉숭화","맨드라미","해바라기")
>>> listFlower=list(flower)
>>> listFlower.append("민들레")
>>> listFlower
['들국화',  ' 채송와', '봉숭화', '맨드라미', '해바라기', '민들레']
```

튜플 살펴보기

■튜플 항목 변경하기

- "채송와"를 "채송화 " 로 수정해 보자.
- 현재 리스트 상태이므로 인덱스를 이용하여 수정이 가능하다.
- 추가와 수정이 이루어진 리스트는 tuple()함수를 이용하여 튜플로 전환한다.

```
>>> listFlower[1]="채송화"
>>> listFlower
['들국화', '채송화', '봉숭화', '맨드라미', '해바라기', '민들레']
>>> flower=tuple(listFlower)
>>> flower
('들국화', '채송화', '봉숭화', '맨드라미', '해바라기', '민들레')
```

튜플 살펴보기

■튜플 슬라이싱

▪튜플도 리스트나 문자열과 같이 자료형이므로 [] 대괄호를 사용하여 특정 항목을 인덱스하거나 특정 구간 범위를 지정하여 자료를 추출하기 위해서 슬라이싱(slicing)이 가능하다.

```
>>> num=(1,2,3,4,5,6,7,8,9,10)
>>> num[:]                  #전체 항목 슬라이싱
(1, 2, 3, 4, 5, 6, 7, 8, 9, 10)
>>> num[2]                  #인덱스 2번째
3
>>> num[0:3]                #인덱스 0부터 3번째
(1, 2, 3)
>>> num[-1]                 #인덱스 –1로 뒤에서 첫 번째
10
>>> num[-6:8]               #인덱스 뒤에서 6번째, 앞에서 숫자 1부터 8번째 까지
(5, 6, 7, 8)
```

[실습]

■튜플 변수 num=(11,12,13,14,15)에 숫자 항목을 기억시킨 후, 반복문을 이용하여 튜플에 기억된 데이터 값과 합계를 구하여 출력하는 프로그램을 작성하시오.

```
11
12
13
14
15
합계=65
```

감사합니다.

2

이번
차시에서는

- 다중 리스트

다중 리스트

■sorted() 함수

- 리스트에서 다루었던 sort() 정렬 함수로는 튜플에서는 사용이 불가능하지만 sorted() 함수 사용은 튜플에서도 가능하다.
- sorted() 함수의 가장 큰 특징은 리스트나 튜플에서 항목을 오름차순과 내림차순으로 선택하여 정렬할 수 있다.

새로운 리스트 변수=sorted(리스트변수, reverse=True)

다중 리스트

■sorted() 함수

▪ 리스트 변수 num=[12,15,8,4,9]에 숫자 정수 항목이 기억되어 있다. 오름차순 정렬된 결과를 newNum에 저장 후, 저장된 결과를 출력해 보자.

```
>>> num=[12,15,8,4,9]
>>> newNum=sorted(num)
>>> newNum
[4, 8, 9, 12, 15]
```

▪ 옵션 reverse=True를 추가하여 내림차순으로 정렬해 보자.

```
>>> newNum2=sorted(num, reverse=True)
>>> newNum2
[15, 12, 9, 8, 4]
```

다중 리스트

■**lambda() 함수**

- 다중 리스트나, 다중 튜플에서 여러 개의 필드 값을 가지고 있다면 key(키) 값을 인자 값으로 지정하여 lambda() 함수 key(키) 값 옵션을 지정할 수 있다.
- key=lambda x:x[0]의 의미는 다중 리스트에서 key(키) 값 0번째 인자에 대한 값을 지정하여 정렬을 한다.
- lambda 함수는 1차 리스트에서는 많이 사용하지 않지만, 만약, lambda() 함수를 1차원에서 사용하고자 한다면 인자에 대한 key(키) 값 기준은 무조건 key=lambda x:x[0]로 지정한다.

```
새로운변수=sorted(리스트변수, key=lambda x: x[0], reversed=True)
```

다중 리스트

■sorted() 함수

▪리스트변수 language=["Java","Python","R","C","Pascal","SQL"]에 문자열 항목을 저장한 후, lambda 함수를 사용하여 키 값을 지정한 후, 오름차순으로 정렬한 후, 새로운 리스트 변수 newLanguage에 저장해 보자.

```
>>> language=["Java","Python","R","C","Pascal","SQL"]
>>> newLanguage=sorted(language, key=lambda x:x[0], reverse=False)
>>> newLanguage
['C', 'Java', 'Python', 'Pascal', 'R', 'SQL']
```

▪옵션 reverse=True로 지정 후, 새로운 리스트 변수 newLanguage1에 저장한 후, 내림차순으로 정렬해 보자.

```
>>> newLanguage1=sorted(language, key=lambda x:x[0], reverse=True)
>>> newLanguage1
['SQL', 'R', 'Python', 'Pascal', 'Java', 'C']
```

[실습]

■튜플 변수 num=(20,30,15,31,23) 에 숫자 항목이 저장되어 있다. 오름차순으로 정렬하여 항목 값을 출력한 후, 합계도 출력하도록 프로그램을 작성하시오.

```
15
20
23
30
31
합계:119
```

다중 리스트와 튜플

■다중 리스트 구조

- 1차원 리스트 구조는 행으로만 구성된 것 구조
- 2차원 리스트 구조는 행과 열로 구성된 리스트 구조

0행 0열	0행 1열	0행 2열
1행 0열	1행 1열	1행 2열
2행 0열	2행 1열	2행 2열

■2차원 리스트, 2차원 튜플 구조

- 다중 리스트 = [[값, 값], [값, 값], [값, 값]]
- 다중 튜플 = ((값, 값),(값, 값),(값, 값))
- 다중 리스트 튜플 = ([값, 값], [값, 값], [값, 값])

다중 리스트와 튜플

■2차원 리스트에 접근하기

- 리스트[행][열] = 값 #데이터 값을 특정한 위치로 저장

다중 리스트와 튜플

■for 반복문을 이용하여 다중 리스트를 출력해보자.

```
>>> numList = [[10, 20], [30, 40], [10, 100]]
>>> for i, j in numList:
        print(i, j)
10 20
30 40
10 100
```

다중 리스트와 튜플

■외부 for, 내부 for 중첩 반복문을 사용하여 다중 리스트를 출력해 보자.

```
>>> numList = [[10, 20], [30, 40], [10, 100]]
>>> for i in numList:                    #외부 리스트에서 항목을 출력 : 행에 해당
        for j in i:                       #내부 리스트에서 항목을 출력 : 열에 해당
            print(j, end=" ")
        print()
10 20
30 40
10 100
```

다중 리스트와 튜플

■다중 리스트 튜플 정렬하기

▪다중 리스트 튜플에서 여러 개의 필드 값을 가지고 있다면 key(키) 값의 위치를 lambda() 함수 키 인자 값(도서대금)을 내림차순으로 지정하여 출력해보자.

```
book = [("Java", 15000),("Python", 25000), ("C", 18000), ("Java",19000)]
newBook=sorted(book, key=lambda x:x[1], reverse=True)
for i,j in newBook:
    print("{}₩t{}".format(i,j))
```

```
Python  25000
Java    19000
C       18000
Java    15000
```

[실습]

■다중 리스트 튜플에 데이터를 점수 기준 내림차순으로 정렬하여 출력되도록 프로그램을 작성하시오.

```
student=[
            ("인터넷","이정재","010-123-1234", 90),
            ("미디어","전지현","010-123-4560", 80),
            ("정통과","송혜교","010-123-2244", 100),
            ("컴퓨터","김소현","010-123-3355", 85),
            ("국문과","강하늘","010-123-7890", 95)
]
```

```
학   과   이   름   연락처          점수
정통과   송혜교   010-123-2244   100
국문과   강하늘   010-123-7890   95
인터넷   이정재   010-123-1234   90
컴퓨터   김소현   010-123-3355   85
미디어   전지현   010-123-4560   80
```

감사합니다.

3

이번
차시에서는

- 딕셔너리

딕셔너리

■딕셔너리(dictionary)

- 딕셔너리(dictionary) 개념은 key와 value가 한 쌍으로 이루어진 자료 형으로 리스트나 튜플과 같이 순차적으로 해당 값을 요구하는 것이 아니라 임의적으로 입력된 key 값을 참조하여 매칭되는 해당 value 값을 탐색하여 가져오는 자료 형이다.
- 딕셔너리 변수={key1:value1, key2:value2 … } 와 같이 {} 중괄호를 사용하며, key 값을 설정할 때는 변하지 않는 고유한 값을 사용하고, value 값 설정은 변하는 값과 변하지 않는 값 모두를 사용할 수 있다.

딕셔너리 변수={key1:value1, key2:value2 … }

딕셔너리

■빈 딕셔너리 만들기

- 빈 딕셔너리를 {} 중괄호를 이용하여 grade란 이름으로 생성해 보자.
- 생성된 빈 디렉터리에 key, value 하나의 쌍으로 데이터를 추가할 수 있다

```
>>> grade={}
>>> type(grade)
<class 'dict'>
```

▪빈 딕셔너리애 데이터 추가하기

딕셔너리[key] = value

딕셔너리

■빈 딕셔너리 항목 추가하기

▪생성된 grade에 key, value를 한 쌍으로 딕셔너리 grade에 추가해 보자.

key(키)	value(값)
아이키	90
박보검	95
유아인	80
전지현	100
송중기	90

```
>>> grade={}
>>> grade["아이키"]=90
>>> grade["박보검"]=95
>>> grade["유아인"]=80
>>> grade["전지현"]=100
>>> grade["송중기"]=90
>>> grade
{'아이키': 90, '박보검': 95, '유아인': 80, '전지현': 100, '송중기': 90}
```

딕셔너리

■key 값을 검색하여 value 값 찾기

- 입력된 딕셔너리는 딕셔너리[key]를 입력하여 할당된 값을 찾을 수 있다.
- 딕셔너리 grade에서 key 값에 "전지현"을 입력하여 할당된 value(값)를 찾아보자.

```
>>> grade["전지현"]
100
```

딕셔너리

■key 값을 검색하여 value 값 변경하기

- 할당된 value(값)는 언제든지 '딕셔너리[key] = 수정할 value'를 통해서 value(값)를 변경할 수 있다.
- grade 딕셔너리 key(키) "유아인"에 할당된 value(값)를 85점으로 변경해보자.

```
>>> grade["유아인"]=85
>>> grade
{'아이키': 90, '박보검': 95, '유아인': 85, '전지현': 100, '송중기': 90}
```

딕셔너리

■딕셔너리 출력하기

▪딕셔너리 key 값만 출력해보자.

```
>>> grade.keys()
dict_keys(['아이키', '박보검', '유아인', '전지현', '송중기'])
```

▪딕셔너리 value 값만 출력해보자.

```
>>> grade.values()
dict_values([90, 95, 85, 100, 90])
```

딕셔너리

■딕셔너리 출력하기

- **딕셔너리 key 값, value 값을 동시에 출력해보자.**

```
>>> grade.items()
dict_items([('아이키', 90), ('박보검', 95), ('유아인', 85), ('전지현', 100), ('송중기', 90)])
```

딕셔너리

■딕셔너리 출력하기

- **반복문을 활용하여 key 값만 출력해 보자.**

```
>>> for i in grade.keys() :
        print(i)
아이키
박보검
유아인
전지현
송중기
```

딕셔너리

■딕셔너리 출력하기

- **반복문을 활용하여 key 값, value 값을 동시에 출력해 보자.**

```
>>> for i in grade.keys() :
        print(i, grade[i])
아이키 90
박보검 95
유아인 85
전지현 100
송중기 90
```

딕셔너리

■리스트를 딕셔너리로 변경하기

▪리스트 변수를 딕셔너리로 변경하기 위해서는 dict() 함수와 zip() 함수를 허용하여 key 값과 value 값을 갖추는 딕셔너리 형식으로 매칭시켜 딕셔너리 형태의 자료 형으로 변경할 수 있다.

① zip() 함수 : 두 개의 리스트를 묶어서 새로운 리스트로 변환하는 역할을 하며 묶어진 두 리스트는 key, value로 사용되어질 기초 자료를 제공한다.

② dict() 함수 : 묶어진 리스트를 딕셔너리 자료 형으로 변환시켜주는데, 이 때 key, value는 완벽하게 딕셔너리에 사용될 수 있도록 변환된다.

새로운 딕셔너리 변수 = dict(zip(key, value))

딕셔너리

■리스트를 딕셔너리로 변경하기

▪리스트 변수 name=[”이순신","강감찬","홍길동"]과 score=[80,90,100]에 각각 기억시킨 후, dict 함수와 zip 명령으로 딕셔너리 형태의 자료형으로 변환하여 보자.

```
>>> name=[“이순신","강감찬","홍길동"]
>>> score=[80,90,100]
>>> grade=dict(zip(name,score))
>>> grade
{'이순신': 80, '강감찬': 90, '홍길동': 100}
```

[실습]

■딕셔너리변수 grade={'아이키': 90, '박보검': 95, '유아인': 85, '전지현': 100, '송중기': 90}에 서 key 값에 할당된 value 값 항목에 대한 합계를 [실행 결과]와 같이 출력하시오.

```
아이키 90
박보검 95
유아인 85
전지현 100
송중기 90
=========
합 계:460
```

[문제]

■파이썬 다중 리스트 movie_list=[("오징어게임",50.45), ("이터널스",35.46),("그레비티",9.8), ("노타임투다이",52.5), ("스파이더맨",15.47)]에 저장된 자료를 딕셔너리로 변환하여, [실행 결과]와 같이 영화제목, 예매율, 순위순으로 출력하시오. 단, 예매율이 높은 순으로 출력하시오.

영화제목	예매율	순위
=========	======	====
노타임투다이	52.5	1
오징어게임	50.45	2
이터널스	35.46	3
스파이더맨	15.47	4
그레비티	9.8	5

감사합니다.