Lecture 1 : 리스트와 튜플

1. 리스트

- 항목의 모임(collections of items)으로, 항목은 순서가 있는 시퀀스(sequence)
- 항목은 원소(element) 또는 요소(item)이라고도 불림
- 항목은 모든 유형의 객체(object) 가능
- 리스트는 가변(mutable) 객체로 항목은 수정, 추가 또는 삭제 가능

> [item_1, item_2, ... , item_n]
> data_list = [1, 2.4, 'py', [1, 2, 3]]

리스트 관련 내장 함수

내장함수	설명	사용 예	출력(결과)
list(iterable)	리스트 생성자, iterable로 리스트를 생성하는 함수	list()	[]
		list("python")	['p', 'y', 't', 'h', 'o', 'n']
		list((0, 1, 2, 3, 4))	[0, 1, 2, 3, 4]
		list({'banana', 'apple', 'graph'})	['graph', 'banana', 'apple']
		list({'JAVA':10, 'C#': 20})	['JAVA', 'C#']
		list({'JAVA':10, 'C#': 20}.items())	[('JAVA', 10), ('C#', 20)]
		list(range(10))	[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
sum(lst)	리스트의 항목의 합을 반환	sum([1, 3, 5, 6])	15
len(lst)	리스트의 항목의 갯수 반환	len([1, 3, 5, 6])	4
max(lst)	리스트의 항목 중에서 최댓값 반환	max([1, 3, 5, 6])	6
min(lst)	리스트의 항목 중에서 최솟값 반환	min([1, 3, 5, 6])	1
sorted(lst)	리스트의 항목을 정렬	sorted(['banana', 'apple', 'kiwi'])	['apple', 'banana', 'kiwi']
		sorted(['banana', 'apple', 'kiwi'],reverse=1)	['kiwi', 'banana', 'apple']

LAB-1 : 리스트 관련 내장 함수

In [ ]:
```
data_list = [1, 2.4, 'py', [1, 2, 3]]
data_list
```

In [ ]:
```
list()
```

In [ ]:
```
list("python")
```

In [ ]:
```
list((0, 1, 2, 3, 4))
```

In [ ]:
```
list({'banana', 'apple', 'graph'})
```

In [ ]:
```
list({'JAVA':10, 'C#': 20})
```

In [ ]:
```
list({'JAVA':10, 'C#': 20}.items())
```

In [ ]:
```
list(range(10))
```

In [ ]:
```
sum([1, 3, 5, 6])
```

In [ ]:
```
len([1, 3, 5, 6])
```

In [ ]:
```
 max([1, 3, 5, 6])
```

In [ ]:
```
min([1, 3, 5, 6])
```

In [ ]:
```
sorted(['banana', 'apple', 'kiwi'])
```

In [ ]:
```
sorted(['banana', 'apple', 'kiwi'],reverse=1)
```

2. 리스트 관련 메서드

리스트 메서드	설명	사용 예	출력(결과)
lst.insert(index, item)	리스트 lst의 첨자 index 앞에 item 추가	fruits = [] fruits.insert(0, "apple") fruits.insert(1, "banana") fruits	['banana', 'apple']
		language = ['Python', 'Java', 'C++', 'GO'] language.insert(2, "JavaScript") language	['Python', 'Java', 'JavaScript', 'C++', 'GO']
lst.append(item)	리스트 lst의 맨 뒤에 item 추가	language = ['Python', 'Java', 'C++', 'GO'] language.append("Kotlin") language	['Python', 'Java', 'C++', 'GO', 'Kotlin']
lst.remove(item)	리스트 lst에서 item 제거	fruits = ['apple', 'banana', 'kiwi'] fruits.remove("banana") fruits	['apple', 'kiwi']

리스트 메서드	설명	사용 예	출력(결과)
lst.pop(index)	리스트 lst에서 index번째 item 삭제후, item 값 반환	language = ['Python', 'Java', 'C++', 'GO'] print(language.pop()) print(langusge)	GO ['Python', 'Java', 'C++']
		language = ['Python', 'Java', 'C++', 'GO'] print(language.pop(1)) print(language)	Java ['Python', 'C++', 'GO']
lst.sort()	리스트 lst 항목을 정렬	language = ['Python', 'Java', 'C++', 'GO'] language.sort() language	['C++', 'GO', 'Java', 'Python']

LAB-2 : 리스트 메서드

In [ ]:
```
fruits = []
fruits.insert(0, "apple")
fruits.insert(1, "banana")
fruits
```

In [ ]:
```
language = ['Python', 'Java', 'C++', 'GO']
language.insert(2, "JavaScript")
language
```

In [ ]:
```
language = ['Python', 'Java', 'C++', 'GO']
language.append("Kotlin")
language
```

In [ ]:
```
fruits = ['apple', 'banana', 'kiwi']
fruits.remove("banana")
fruits
```

In [ ]:
```
language = ['Python', 'Java', 'C++', 'GO']
print(language.pop())
print(language)
```

In [ ]:
```
language = ['Python', 'Java', 'C++', 'GO']
print(language.pop(1))
print(language)
```

In [ ]:
```
language = ['Python', 'Java', 'C++', 'GO']
language.sort()
language
```

LAB 3 : 리스트에서 중복 제거

1. 문제 : 다음 숫자 리스트에서 중복을 제거하고, 중복이 제거된 리스트를 오름차순으로 정렬하여 출력하세요.
2. 조건 : 딕셔너리, 집합(set) 등 다른 자료형을 사용하지 말고 리스트만 사용
3. 입출력

- 입력 : numbers = [1, 3, 2, 3, 4, 1, 5]
- 출력 : [1, 2, 3, 4, 5]

> list.append(), list.sort()

In [ ]:
```
# 코드 예제
numbers = [1, 3, 2, 3, 4, 1, 5]

unique_numbers = []
for num in numbers:
    if num not in unique_numbers:  # 리스트에서 중복 확인
        unique_numbers.append(num)

unique_numbers.sort()  # 오름차순 정렬
print("중복 제거 및 정렬된 리스트:", unique_numbers)
```

LAB 4 : 문자열 정렬 프로그램 (리스트 활용)

1. 문제 : 사용자로부터 여러 문장을 입력받아, 각 문장의 단어를 알파벳 순서로 정렬
   - 사용자가 여러 문장을 입력합니다.
   - 각 문장의 단어를 알파벳순으로 정렬합니다.
   - 정렬된 문장을 출력합니다.
2. 조건 : 문자열 메서드와 리스트 메서드를 함께 사용하세요
3. 입출력

- 입력 :
  - 문장 입력: Python is great
  - 문장 입력: I love programming
  - 문장 입력: (빈 줄 입력 시 종료)
- 출력 :
  - 정렬된 문장:
  - is great Python
  - I love programming

> str.split(), str.join(), list.sort()

In [ ]:
```
# 코드
sentences = []

print("여러 문장을 입력하세요. (빈 줄 입력 시 종료)")
while True:
    sentence = input("문장 입력: ")
    if not sentence.strip():
        break
    sentences.append(sentence)

print("\n정렬된 문장:")
for sentence in sentences:
    #
```

```
# 코딩하오오
#
#
```

3. 튜플 (tuple)

- 항목의 모임(collections of items)으로, 항목은 순서가 있는 시퀀스(sequence)
- 튜플은 항목의 순서나 항목 추가, 삭제가 불가능
- 각 항목은 정수, 실수, 문자열, 리스트, 튜플 등 제한이 없음
- list보다 빠르게 실행 가능

> (item_1, item_2, ... , item_n) # 항목이 하나라도 콤마(,)가 필요
> item_1, item_2, ... , item_n # 괄호는 생략 가능하면, 항목이 하나라도 콤마(,)가 필요

튜플의 생성(선언) / LAB-5

사용 예	출력(결과)	설명
a = 1 print(a, type(a))	1 <class 'int'>	정수 변수 선언, 값과 자료형 출력
b = (1) print(b, type(b))	1 <class 'int'>	정수 변수 선언, 값과 자료형 출력
t1 = 1, print(t1, type(t1))	(1,) <class 'tuple'>	튜플 변수 선언, 값과 자료형 출력
t2 = (2,) print(t2, type(t2))	(2,) <class 'tuple'>	튜플 변수 선언, 값과 자료형 출력
t3 = ('월', '화', '수') print(t3)	('월', '화', '수')	튜플 변수 선언, 값과 자료형 출력
t4 = ('월', '화', '수',) print(t4)	('월', '화', '수')	튜플 변수 선언, 값과 자료형 출력

튜플 관련 내장 함수 / LAB-6

내장함수	설명	사용 예	출력(결과)
tuple(iterable)	튜플 생성자, iterable로 튜플을 생성하는 함수	tuple()	()
		tuple((1, 2))	(1, 2)
		tuple("python")	('p', 'y', 't', 'h', 'o', 'n')
		tuple(range(5))	(0, 1, 2, 3, 4)
		tuple({'banana', 'apple', 'graph'})	('banana', 'graph', 'apple')

튜플 패킹과 언패킹(Packing, Unpackang) / LAB-7

사용 예	출력(결과)	설명
day = ('월', '화', '수') a, b, c = day print(a, b, c)	월 화 수	언패킹(unpacking, 압축해제)

LAB-5 튜플의 생성(선언)

In [ ]:
```
a = 1
print(a, type(a))
```

In [ ]:
```
b = (1)
print(b, type(b))
```

In [ ]:
```
t1 = 1,
print(t1, type(t1))
```

In [ ]:
```
t2 = (2,)
print(t2, type(t2))
```

In [ ]:
```
t3 = ('월', '화', '수')
print(t3)
```

In [ ]:
```
t4 = ('월', '화', '수', )
print(t4)
```

LAB-6 튜플 내장 함수

In [ ]:
```
tuple()
```

In [ ]:
```
tuple((1, 2))
```

In [ ]:
```
tuple("python")
```

In [ ]:
```
tuple(range(5))
```

In [ ]:
```
tuple({'banana', 'apple', 'graph'})
```

LAB-7 패킹과 언패킹

In [ ]:
```
day = ('월', '화', '수')
a, b, c = day
print(a, b, c)
```

In [ ]:
```
def calc(a, b) :
    s = a + b
    m = a * b

    return (a, b)

x = calc(10, 20)
print(x, type(x))
```

```
x1, x2 = calc(10, 20)
print(x1, x2)
```

In [ ]: