CHAPTER 05

리스트, 튜플, 딕셔너리에 대해 알아봅시다.

문제해결을 위한 파이썬 첫걸음

이미향 교수

smilequeen@gmail.com

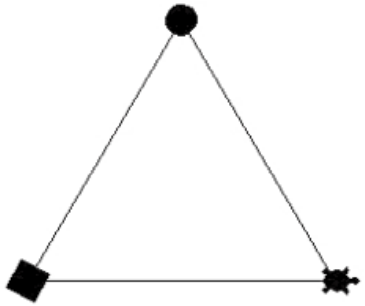
학습 목표

- 리스트를 이해하고 리스트의 요소를 생성, 추가, 삭제하는 방법을 이해합니다.
- 튜플을 이해하고, 튜플 요소를 생성, 접근하는 방법을 이해합니다.
- 딕셔너리를 이해하고 딕셔너리 요소를 생성, 삭제, 접근하는 방법을 이해합니다.

이번 장에서 만들 프로그램

리스트 이용, 터틀 커서 모양 표시 프로그램

- 리스트, 터틀 그래픽 모듈 이용



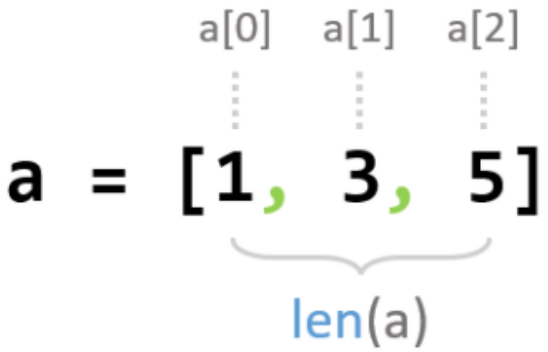
딕셔너리 활용, 동물 한영사전 프로그램

- 딕셔너리 이용

```
---------- 동물 한영 사전입니다. ----------
호랑이, 사자, 코끼리, 토끼, 거북이 중 하나를 입력하세요:사자
사자 영어 단어는
==> Lion 입니다
```

5.1 리스트(list)

- **리스트**
  - 여러 개의 데이터를 하나로 묶어서 처리할 수 있도록 해주는 자료형
  - 리스트 만들기
    - 비어 있는 리스트 만들 경우 **[]**만 지정
    - 항목(item)들을 **쉼표(,)**로 분리하여 **대괄호([ ])**안에 넣기



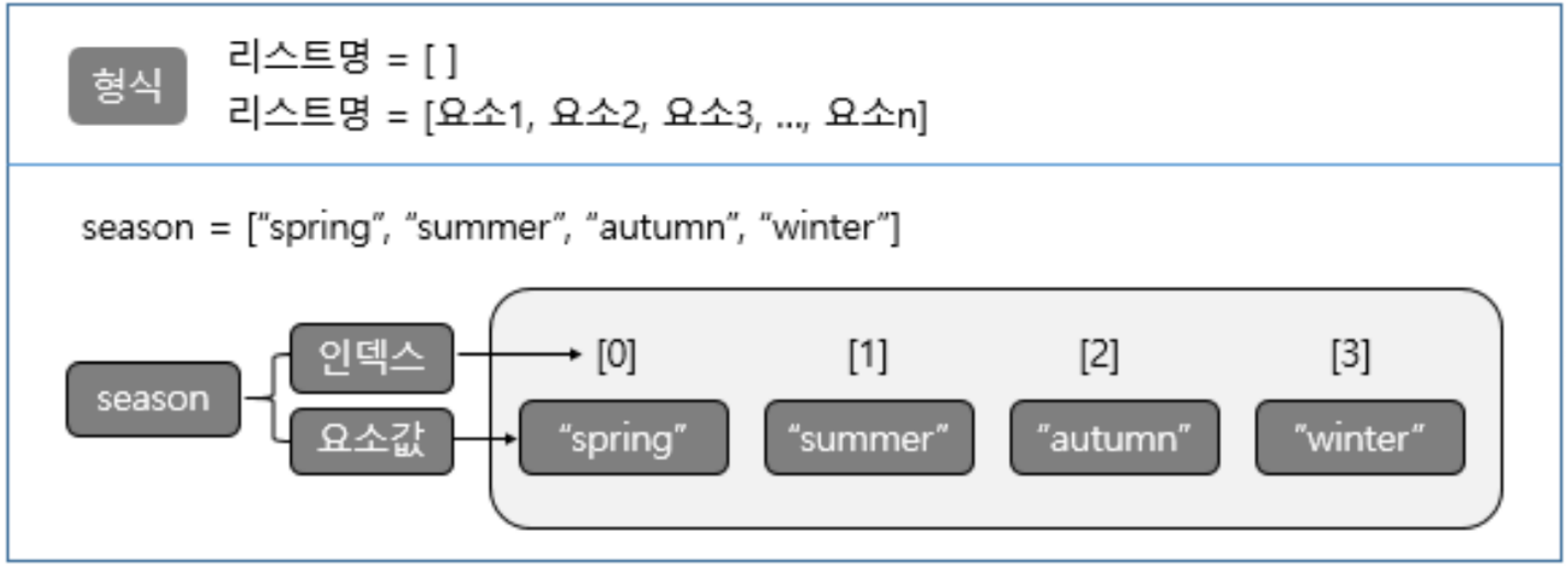
형식

```
리스트명 = [ ]
리스트명 = [요소1, 요소2, 요소3, … ]
```

```
a = []                                          # 빈 리스트
b = [1, 2, 3, 4, 5, 6, 7, 8, 9]                 # 숫자 가능
c = ["red", "orange", "yellow", "green", "blue"]  # 문자열 가능
d = [1, "고양이", 2, "강아지", 3, "병아리"]          # 숫자, 문자열 혼용 가능
e = [1,3,5,7,9,[2,4,6,8,10]]                    # 리스트 안에 리스트 포함 가능
```

리스트(list)

- **여러 개의 데이터를 하나의 이름으로 묶어서 처리**할 수 있도록 해주는 자료형
- 리스트를 생성할 때는 각 요소의 값을 **쉼표**로 구분하여 **대괄호 [ ]**안에 넣기
- 리스트는 **순서**를 가짐
- 리스트의 순서는 인덱스(index) 번호로 표현, 리스트 **첫 항목의 인덱스는 0**



리스트 인덱싱

- 리스트 인덱싱과 슬라이싱
  - 리스트 인덱싱
    - 인덱스(index) : 리스트 항목에 붙어있는 번호, 0부터 번호 시작
  - “red”,”green”,”blue” 문자열 데이터를 갖는 colors_list 생성하기

[소스코드] 5-1.py

```
colors_list = ["red", "green", "blue"]
print(colors_list)
print("----- 가장 좋아하는 색상은? -----")
print(colors_list[0])
```

[실행결과]

```
['red', 'green', 'blue']
### 가장 좋아하는 색상은? ###
red
```

실습_코딩하기1

터틀 커서 모양 도장 찍기

명령어 종류	명령어 설명
forward(픽셀수)	현재 진행하는 방향으로 지정된 픽셀 수 만큼 이동
left(각도)	현재 커서의 방향으로부터 지정된 각도만큼 왼쪽 방향으로 회전
stamp()	현재 커서의 모양을 화면에 찍기
hideturtle()	현재 커서의 모양을 화면에 표시되지 않게 숨기기

※ 리스트 변수 선언

- 리스트 변수명: cursor
- 저장 데이터 : "blue","red","gray"
- 리스트 선언 형태

cursor=["blue","red","gray"]

※ 리스트 변수 데이터 접근

- "blue" → cursor[0]
- "red" → cursor[1]
- "gray" → cursor[2]

프로그램1: 리스트 활용

- s리스트에 저장된 데이터를 이용하여 커서 모양 찍기

[소스코드] 5-2.py

```
import turtle
t=turtle.Turtle()
s=["turtle","circle","square"]
t.fd(200)
t.shape(s[0])
t.stamp()
t.left(120)
t.fd(200)
t.shape(s[1])
t.stamp()
t.left(120)
t.fd(200)
t.shape(s[2])
t.stamp()
t.ht()
```

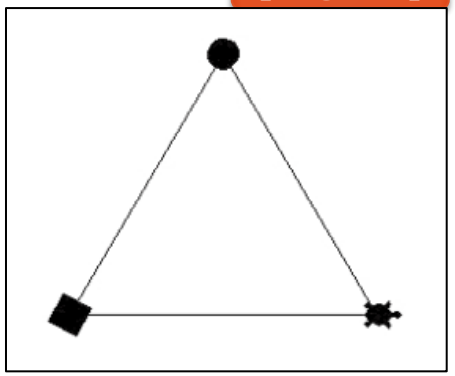
s리스트

s리스트의 0번 데이터 “turtle”

s리스트의 1번 데이터 “circle”

s리스트의 2번 데이터 “square”

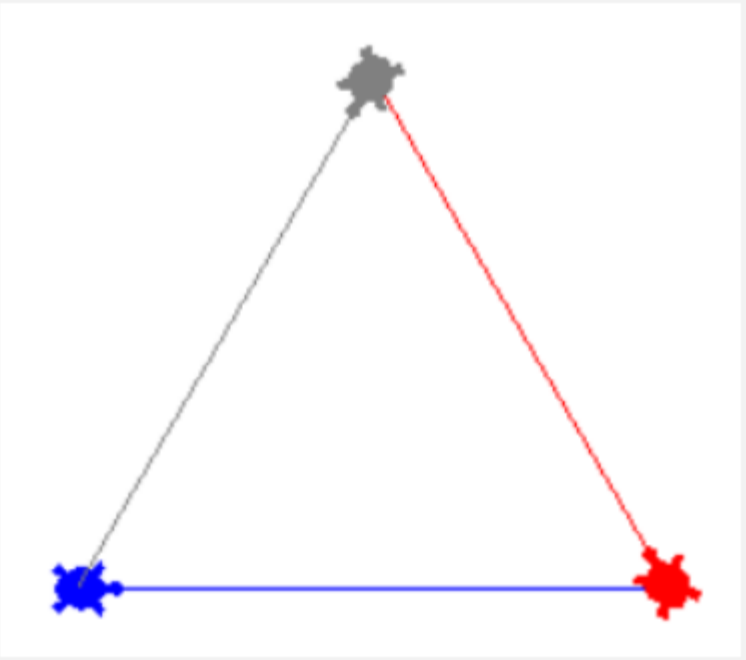
[실행결과]



터틀 커서 모양 도장 찍기

- turtle 모듈이 제공하는 “turtle” 커서의 모양을 색상을 달리해서 삼각형을 그리면서 도장을 찍는 프로그램

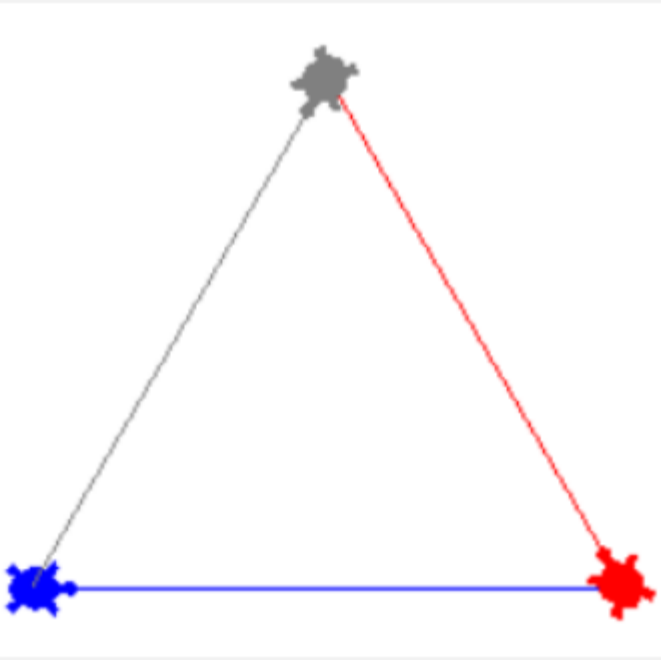
↘ 실행결과



터틀 커서 모양 도장 찍기

```
1  import turtle
2  t = turtle.Turtle()
3  t.shape("turtle")
4  cursor = ["blue", "red", "gray"]
5
6  t.color(cursor[0])
7  t.stamp()
8  t.fd(200)
9  t.left(120)
10
11 t.color(cursor[1])
12 t.stamp()
13 t.fd(200)
14 t.left(120)
15
16 t.color(cursor[2])
17 t.stamp()
18 t.fd(200)
19 t.ht()
```

↘ 실행결과

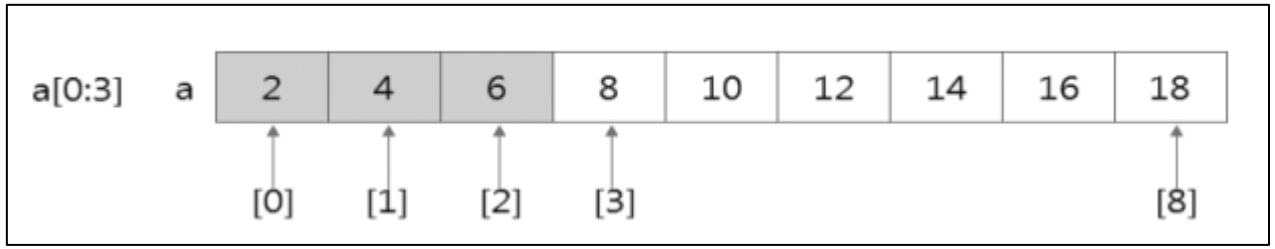


리스트 슬라이싱

- 리스트 슬라이싱
  - []안에 인덱스로 범위를 지정하여 리스트의 일부를 잘라줌
  - 범위 지정 [m:n]인 경우 **m부터 n-1까지** 선택됨

형식 리스트[시작 인덱스:끝인덱스]

끝인덱스는 가져오려는 내용보다 1 크게 지정



[소스코드] 5-3.py

```
a = [2,4,6,8,10,12,14,16,18]
print(a)
print(a[0:3])
```

가져오려는 끝 인덱스보다 1 크게 지정

[실행결과]

```
[2, 4, 6, 8, 10, 12, 14, 16, 18]
[2, 4, 6]
```

따라 해보기 : 리스트 슬라이싱

- 리스트 슬라이싱
  - []안에 인덱스로 범위를 지정하여 리스트의 일부를 잘라줌

- mylist = [0,1,2,3,4,5]
- mylist[1:3] -> mylist[1], mylist[2]
- mylist[:3] -> mylist[0], mylist[1], mylist[2]
- mylist[3:] -> mylist[3], mylist[4], mylist[5]
- mylist[:] -> mylist[0], mylist[1], mylist[2], mylist[3], mylist[4], mylist[5]

따라 해보기 : 리스트 슬라이싱

- 리스트 슬라이싱
  - []안에 인덱스로 범위를 지정하여 리스트의 일부를 잘라줌

```
>>> a = [3,6,9,12,15,18,21,27]
>>> a[:3]
[3, 6, 9]
>>> a[3:]
[12, 15, 18, 27]
>>> a[:]
[3, 6, 9, 12, 15, 18, 27]
```

리스트(list) 연산자

- 리스트 연산자
  - + : 리스트 결합
  - * : 리스트 반복

[소스코드] 5-4.py

```
a = [1,3,5,7,9]
b = [2,4,6,8,10]
print(a + b)
print(a * 2)
```

리스트 결합

리스트 반복

[실행결과]

```
[1, 3, 5, 7, 9, 2, 4, 6, 8, 10]
[1, 3, 5, 7, 9, 1, 3, 5, 7, 9]
```

- 리스트 요소의 개수 구하기 : len(list)

```
1 a = [1,3,5,7,9]
2
3 print(len(a))
4
5 print(len([1,3,5,7,9]))
```

a리스트 요소의 개수

[ ] 안의 요소의 개수

리스트 요소 값 수정

- 리스트 요소 값 수정하기

```
7 b = [3,6,9,13,15,18,21,27]
8 b[3] = 12
9 print(b)
```

b리스트 3번 인덱스 값을 수정

```
[3, 6, 9, 12, 15, 18, 21, 27]
```

12로 수정됨

리스트 관련 메서드 함수

- 리스트에 요소 **삽입** : list이름.insert(index, value)

```
a = [3,5,7,9]
a.insert(0,1)
print(a)
```

a리스트 0번 인덱스에 값 삽입

[실행결과]

[1, 3, 5, 7, 9]

0번 인덱스에 1 삽입

- 리스트 **확장** : list이름.extend(list)

```
1 a = ["카카오톡", "페이스북", "인스타그램"]
2 a.extend(["트위터","라인"])
3 print(a)
```

['카카오톡', '페이스북', '인스타그램', '트위터', '라인']

a리스트 뒤에 연결, 확장됨

리스트 관련 메서드 함수

- 리스트에 요소 추가 : list이름.append(value)

[소스코드] 5-6.py

```
future=[]
future.append("AI")
future.append("빅데이터")
future.append("자율주행")
future.append("사물인터넷")

print("미래 기술 핵심분야:")
print(future, sep=", ")
```

future 리스트에 값 삽입

출력할 때 ,와 공백 삽입

```
미래 기술 핵심분야:
['AI', '빅데이터', '자율주행', '사물인터넷']
```

따라 해보기

- title과 singer 리스트에 사용자로부터 입력 받은 데이터를 추가하여 출력하기

[소스코드] 5-7.py

```
title=["작은 것들을 위한 시", "하기나 해"]
singer=["방탄소년단", "GRAY"]
print("곡명과 가수를 저장해봅시다.")
title.append(input("곡명: "))
singer.append(input("가수: "))
print("\n곡 목록:", title )
print("가수 목록:", singer )
print()
print(f"곡명:{title[0]}, 가수:{singer[0]}")
```

title 리스트에 값 입력 받기

singer 리스트에 값 입력 받기

[실행결과]

```
곡명과 가수를 저장해봅시다.
곡명: 니 소식
가수: 송하예

곡 목록: ['작은 것들을 위한 시', '하기나 해', '니 소식']
가수 목록: ['방탄소년단', 'GRAY', '송하예']

곡명:작은 것들을 위한 시, 가수:방탄소년단
```

맨 뒤에 추가됨

리스트 정렬하기

- 리스트 정렬하기 : **list이름.sort()**
  - 리스트의 항목을 순서대로 정렬 후 재저장
  - 기본 정렬 : 오름차순 정렬

[소스코드] 5-8.py

```
future=[]
future.append("AI")
future.append("빅데이터")
future.append("자율주행")
future.append("사물인터넷")

print("미래 기술 핵심분야:")
print(future, sep=", ")

print("정렬후==>:")
future.sort()
print(future,sep=", ")
```

future 리스트 내용을 오름차순 정렬

[실행결과]

```
미래 기술 핵심분야:
['AI', '빅데이터', '자율주행', '사물인터넷']
정렬후==>:
['AI', '빅데이터', '사물인터넷', '자율주행']
```

리스트 정렬하기

- 리스트 정렬하기 : list이름.reverse() / list이름.sort(reverse=True)
  - 리스트의 항목을 순서대로 정렬 후 재저장
  - 저장된 순서 역순 정렬 : list이름.reverse()
  - 저장된 값 내림차순 정렬 : list이름.sort(reverse=True)

```
1  future=[]
2  future.append("AI")
3  future.append("빅데이터")
4  future.append("자율주행")
5  future.append("사물인터넷")
6
7  print("미래 기술 핵심분야:")
8  print(future, sep=", ")
9
10 print("정렬후==>:")
11 future.reverse()
12 print(future,sep=",")
```

future 리스트 순서를 역순으로 정렬

```
미래 기술 핵심분야:
['AI', '빅데이터', '자율주행', '사물인터넷']
정렬후==>:
['사물인터넷', '자율주행', '빅데이터', 'AI']
```

```
1  future=[]
2  future.append("AI")
3  future.append("빅데이터")
4  future.append("자율주행")
5  future.append("사물인터넷")
6
7  print("미래 기술 핵심분야:")
8  print(future, sep=", ")
9
10 print("정렬후==>:")
11 future.sort(reverse=True)
12 print(future,sep=",")
```

future 리스트 값을 내림차순으로 정렬

```
미래 기술 핵심분야:
['AI', '빅데이터', '자율주행', '사물인터넷']
정렬후==>:
['자율주행', '사물인터넷', '빅데이터', 'AI']
```

리스트 요소 제거

- 리스트 요소 제거
  - del list이름[x] : 리스트의 x번째 요소 값 삭제
  - list이름.remove(값) : 리스트에서 값을 찾아서 삭제
  - list이름.pop() : 리스트의 마지막 값을 삭제, 삭제한 값 반환
  - list이름.clear() : 리스트의 모든 값 삭제

따라 해보기 : 리스트 요소 제거

- 리스트 요소 제거

[소스코드] 5-9.py

```
rainbow = ["빨","주","노","초","파","남","보"]
print("원본==>",rainbow,"\n")
#del을 이용하여 삭제하기
del rainbow[0]                    0번 요소 값 삭제
print("del을 이용한 삭제 ==>",rainbow,"\n")
#remove()을 이용하여 삭제하기
rainbow.remove("주")              "주" 내용 값 검색하여 삭제
print("remove을 이용한 삭제 ==>",rainbow,"\n")
#pop()을 이용하여 삭제하기
rainbow.pop()                     맨 뒤의 요소 값 삭제
print("pop을 이용한 삭제 ==>",rainbow,"\n")
#clear()을 이용하여 삭제하기
rainbow.clear()                   전체 요소  값 삭제
print("clear를 이용한 삭제 ==>",rainbow,"\n")
```

[실행결과]

```
원본==> ['빨', '주', '노', '초', '파', '남', '보']

del을 이용한 삭제 ==> ['주', '노', '초', '파', '남', '보']

remove을 이용한 삭제 ==> ['노', '초', '파', '남', '보']

pop을 이용한 삭제 ==> ['노', '초', '파', '남']

clear를 이용한 삭제 ==> []
```