AIoT AutoCar Prime 으로 배우는

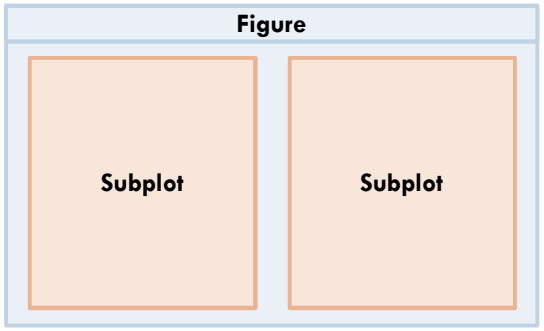
온디바이스 AI 프로그래밍

6.3 Matplotlib

6.3.1 그래프

Matplotlib

- Matplotlib : 데이터를 그래프로 시각화 할 수 있는 패키지
  - 그래프를 표현하기 위해 GUI 환경 사용
    - Subplot : 그래프를 표현하기 위한 공간에 대한 객체
    - Figure : 하나 이상의 Subplot을 Window에 표현하는 객체
    - Subplot에서는 대표적으로 선형 그래프, 막대그래프, 히스토그램, 산점도를 표현 가능



선형 그래프

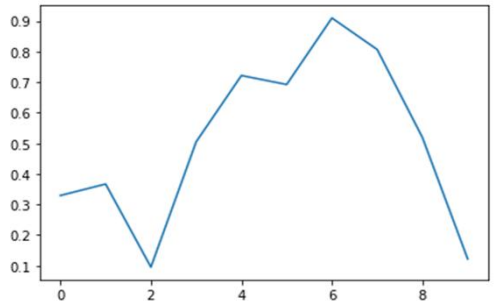
- 선형 그래프 : 1차원 배열 값을 순서대로 나열해 표현
  - Pyplot의 plot메소드를 사용하며 plot(list) 형식으로 사용

```
01:     import matplotlib.pyplot as plt
02:
03:     data=np.random.random(10) #10개의 랜덤 값을 가진 1차원 ndarray 생성
04:     plt.plot(data) #선형 그래프로 출력
05:     plt.show()
```

선형 그래프

- 선형 그래프 : 1차원 배열 값을 순서대로 나열해 표현
  - Pyplot의 plot메소드를 사용하며 plot(list) 형식으로 사용

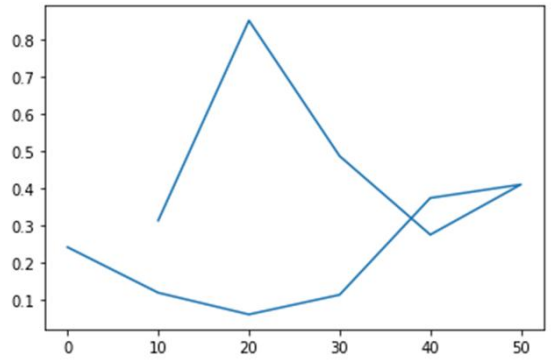
```
01:     import matplotlib.pyplot as plt
02:
03:     data=np.random.random(10) #10개의 랜덤 값을 가진 1차원 ndarray 생성
04:     plt.plot(data) #선형 그래프로 출력
05:     plt.show()
```



선형 그래프

- plot메소드에 2번째 파라미터를 입력할 경우
  - X축과 Y축 모두 직접 표현 가능
  - 단, 축을 순서 없이 지정할 수 있기에 선이 되돌아가는 형태로 표현 가능

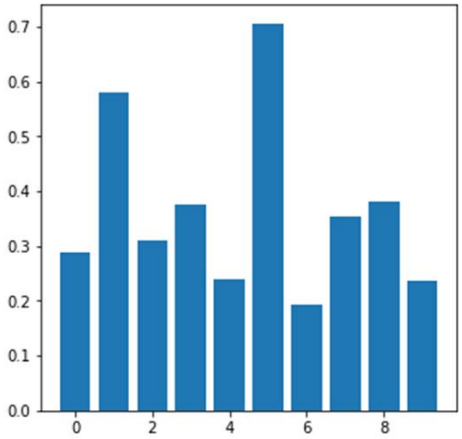
```
01:     import matplotlib.pyplot as plt
02:
03:     data=np.random.random(10)
        #10개의 랜덤 값을 가진 1차원 ndarray 생성
04:     plt.plot([0,10,20,30,40,50,40,30,20,10], data)
        #선형 그래프로 출력
05:     plt.show()
```



세로형 막대그래프

- 막대그래프 : 1차원 배열 값을 나열해 표연
  - 데이터를 비교하기 쉬운 형태
  - Pyplot의 bar메소드를 사용하며 bar(x, y) 형식으로 사용

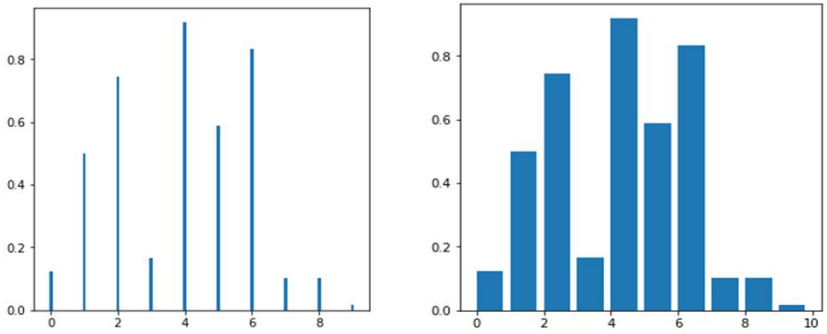
```
01:     import matplotlib.pyplot as plt
02:
03:     data=np.random.random(10) #10개의 랜덤 값을 가진 1차원 ndarray 생성
04:     plt.bar(np.arange(10), data) #막대 그래프로 출력
05:     plt.show()
```



세로형 막대그래프

- bar메소드에 추가 파라미터로 width 또는 align 입력할 경우
  - 막대의 두께를 조절하거나 값의 모서리, 값의 중앙에 정렬해 표현 가능
  - align 파라미터에는 'center' 또는 'edge' 만 사용 가능

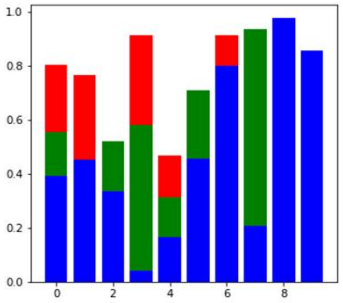
```
06:  plt.bar(np.arange(10), data, width=0.1) #얇은 막대 그래프로 출력
07:  plt.show()
08:  plt.bar(np.arange(10), data, align='edge') #모서리 정렬 막대 그래프로 출력
09:  plt.show()
```



세로형 막대그래프

- color 파라미터
  - 여러 개의 데이터를 하나의 Subplot에 표현하고자 할 때, 색상 구분 가능

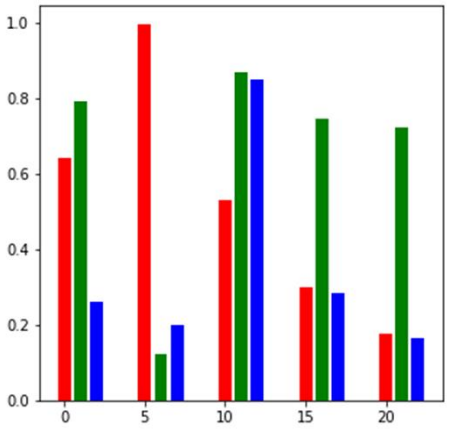
```
01:  import matplotlib.pyplot as plt
02:
03:  data=np.random.random(30).reshape(3,10) #30개의 랜덤 값을 가진 1차원 ndarray 생성하고 3,10 형태로 변환
04:  plt.bar(np.arange(10), data[0], color='r') #빨간색 막대 그래프로 출력
05:  plt.bar(np.arange(10), data[1], color='g') #초록색 막대 그래프로 출력
06:  plt.bar(np.arange(10), data[2], color='b') #파란색 막대 그래프로 출력
07:  plt.show()
```



세로형 막대그래프

- 막대그래프가 겹치지 않게 나열하고 싶다면 x값을 조절해 표현 가능

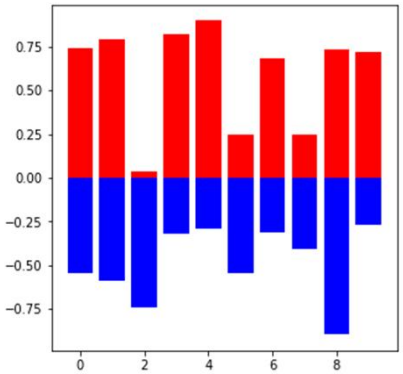
```
08:     data=np.random.random(15).reshape(3,5)
09:     plt.bar(np.arange(0,25,5), data[0], color='r') #빨간색 막대 그래프로 출력
10:     plt.bar(np.arange(0,25,5)+1, data[1], color='g') #초록색 막대 그래프를 1칸 띄워 출력
11:     plt.bar(np.arange(0,25,5)+2, data[2], color='b') #파란색 막대 그래프를 2칸 띄워 출력
12:     plt.show()
```



세로형 막대그래프

- color 파라미터를 응용해 Back to Back 그래프 생성 가능
  - x값을 반전시키고 color 파라미터를 입력

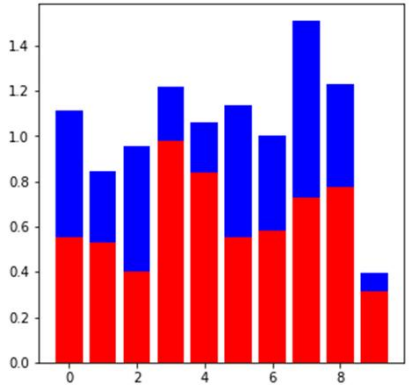
```
01:     import matplotlib.pyplot as plt
02:
03:     data=np.random.random(20).reshape(2,10) #20개의 랜덤 값을 가진 1차원 ndarray 생성하고 2,10 형태로 변환
04:     plt.bar(np.arange(10), data[0], color='r') #빨간색 막대 그래프로 출력
05:     plt.bar(np.arange(10), -data[1], color='b') #파란색 막대 그래프로 출력
06:     plt.show()
```



세로형 막대그래프

- 스택 바 그래프는 bottom 파라미터를 이용해 구현 가능

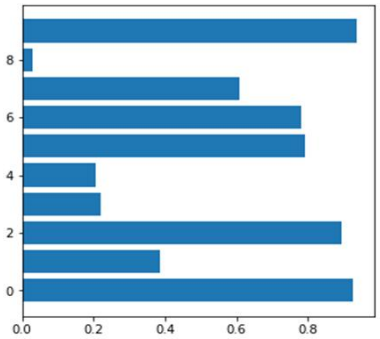
```
01:  import matplotlib.pyplot as plt
02:
03:  data=np.random.random(20).reshape(2,10) #20개의 랜덤 값을 가진 1차원 ndarray 생성하고 2,10 형태로 변환
04:  plt.bar(np.arange(10), data[0], color='r') #빨간색 막대 그래프로 출력
05:  plt.bar(np.arange(10), data[1], color='b', bottom=data[0]) #파란색 막대 그래프로 출력
06:  plt.show()
```



가로형 막대그래프

- 가로형 막대그래프 : Pyplot의 barh메소드 사용.
  - barh(x, y) 형식 사용

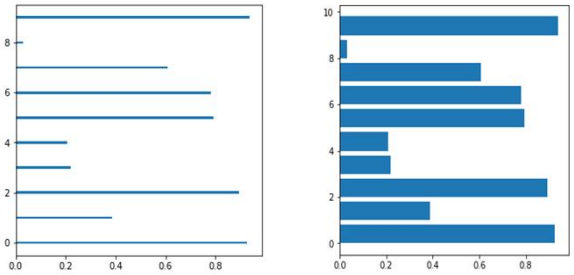
```
01:     import matplotlib.pyplot as plt
02:
03:     data=np.random.random(10) #10개의 랜덤 값을 가진 1차원 ndarray 생성
04:     plt.barh(np.arange(10), data) #막대 그래프로 출력
05:     plt.show()
```



가로형 막대그래프

- 추가 파라미터로 height 또는 align 입력할 경우
  - 막대의 두께를 조절하거나 값의 모서리, 값의 중앙에 정렬해 표현 가능
  - align 파라미터에는 'center' 또는 'edge'만 사용 가능

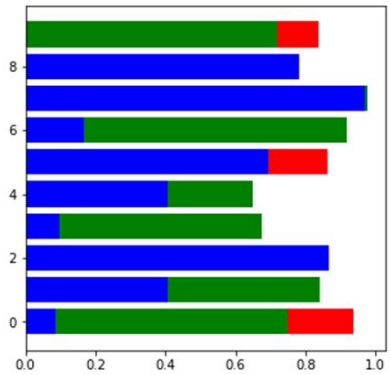
```
01:  plt.barh(np.arange(10), data, height=0.1) #얇은 막대 그래프로 출력
02:  plt.show()
03:  plt.barh(np.arange(10), data, align='edge') #모서리 정렬 막대 그래프로 출력
04:  plt.show()
```



가로형 막대그래프

- color 파라미터로 색상 변경 가능

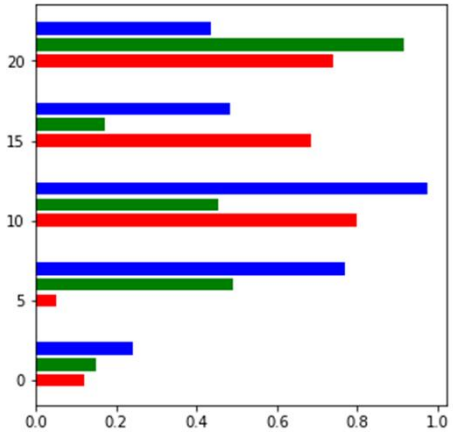
```
01:     import matplotlib.pyplot as plt
02:
03:     data=np.random.random(30).reshape(3,10) #30개의 랜덤 값을 가진 1차원 ndarray 생성하고 3,10 형태로 변환
04:     plt.barh(np.arange(10), data[0], color='r') #빨간색 막대 그래프로 출력
05:     plt.barh(np.arange(10), data[1], color='g') #초록색 막대 그래프로 출력
06:     plt.barh(np.arange(10), data[2], color='b') #파란색 막대 그래프로 출력
07:     plt.show()
```



가로형 막대그래프

- 막대그래프가 겹치지 않게 나열하고 싶다면 x값을 조절해 표현 가능

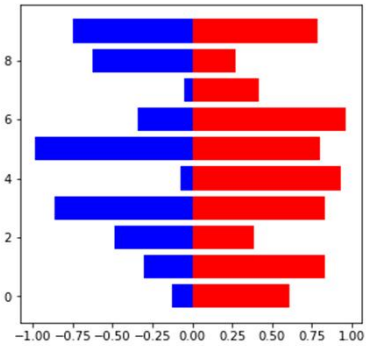
```
01:  data=np.random.random(15).reshape(3,5)
02:  plt.barh(np.arange(0,25,5), data[0], color='r') #빨간색 막대 그래프로 출력
03:  plt.barh(np.arange(0,25,5)+1, data[1], color='g') #초록색 막대 그래프를 1칸 띄워 출력
04:  plt.barh(np.arange(0,25,5)+2, data[2], color='b') #파란색 막대 그래프를 2칸 띄워 출력
05:  plt.show()
```



가로형 막대그래프

- color 파라미터를 응용해 Back to Back 그래프를 생성 가능
  - x값을 반전시키고 color 파라미터 입력

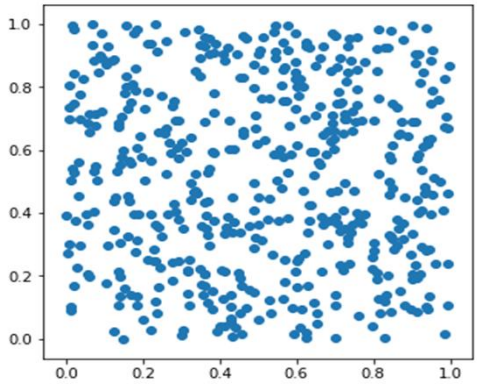
```
01:  import matplotlib.pyplot as plt
02:
03:  data=np.random.random(20).reshape(2,10) #20개의 랜덤 값을 가진 1차원 ndarray 생성하고 2,10 형태로 변환
04:  plt.barh(np.arange(10), data[0], color='r') #빨간색 막대 그래프로 출력
05:  plt.barh(np.arange(10), -data[1], color='b') #파란색 막대 그래프로 출력
06:  plt.show()
```



산점도 그래프

- 산점도 그래프 : 값을 점으로 표현한 그래프
  - 데이터 분포를 파악하기 쉬운 그래프
  - Pyplot의 scatter메소드를 사용하며 scatter(x, y) 형식으로 사용

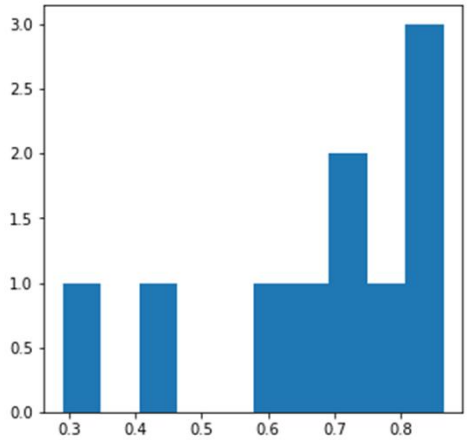
```
01: import matplotlib.pyplot as plt
02:
03: data=np.random.random(1000).reshape(2,500)
      #1000개의 랜덤 값을 가진 1차원 ndarray 생성하고 2,500 형태로 변환
04: plt.scatter(data[0], data[1]) #산점도 그래프로 출력
05: plt.show()
```



히스토그램

- 히스토그램 : 값의 분포를 막대 그래프로 표연
  - 구간별 확률분포나 밀도를 비교하기 좋은 그래프
  - Matplotlib에서 지원하는 hist(list) 메소드를 이용해 구연 가능

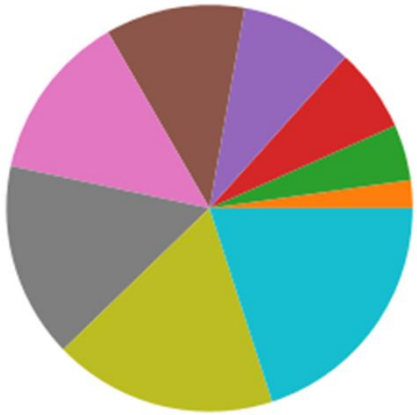
```
01:  import matplotlib.pyplot as plt
02:
03:  data=np.random.random(10)
     #10개의 랜덤 값을 가진 1차원 ndarray 생성
04:  plt.hist(data) #히스토그램 출력
05:  plt.show()
```



원판 그래프

- 원판 그래프
  - 각 값의 비율을 한눈에 비교하기 좋은 pie(list)메소드를 이용해 구현 가능
  - 입력되는 list의 값들은 100개까지 표현 가능

```
01:     import matplotlib.pyplot as plt
02:
03:     data=np.random.random(10)
        #10개의 랜덤 값을 가진 1차원 ndarray 생성
04:     plt.pie(np.arange(10)) #원판 그래프로 출력
05:     plt.show()
```



내용 정리

- Matplotlib : 데이터를 그래프로 시각화할 수 있는 패키지
- plot(list): list의 값들을 선으로 이은 그래프를 생성
- bar(index, list): 각 index별 값을 세로 막대로 표현한 그래프를 생성
- barh(index, list): 각 index별 값을 가로 막대로 표현한 그래프를 생성
- pie(list): 각 값을 원판으로 표현한 그래프를 생성
- hist(list): 값들의 분포를 표현한 그래프를 생성
- scatter(list_x, list_y): x, y 좌표에 점을 찍어 표현한 그래프 생성

연습문제

- 문제 31. 다음 데이터를 선형 그래프로 출력하는 코드를 작성해보세요.

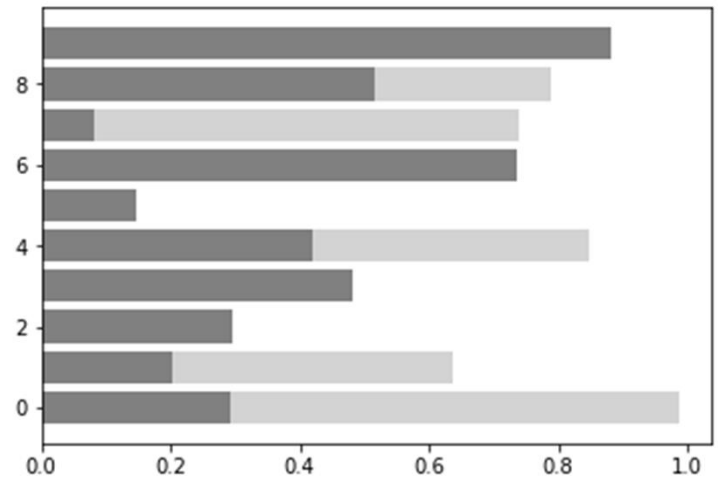
번호	1	2	3	4	5	6	7
값	10	15	13	17	18	15	21

연습문제

- 문제 32. 다음 코드와 출력을 읽고 틀린 부분을 고쳐보세요.

```
01: import matplotlib.pyplot as plt
02:
03: data=np.random.random(20).reshape(2,10)
04:
05: plt.bar(np.arange(10), data[0], color='lightgray')
06: plt.bar(np.arange(10), data[1], color='gray')
07:
08: plt.show()
```

[출력]



연습문제

- 문제 33. 아래의 코드는 Cds센서의 밝기 값을 그래프로 출력하는 코드입니다.

```
01:     from pop import Cds,delay
02:     import numpy as np
03:     import matplotlib.pyplot as plt
04:
05:     cds = Cds(7)
06:
07:     arr = [ ]
08:
09:     for i in range(100):
10:         arr.append(cds.readAverage())
11:         delay(50)
12:
13:     arr = np.array(arr)
14:
15:     plt.plot(arr)
16:     plt.show()
```

연습문제

- A. 코드를 실행시켜 결과를 확인해 보세요.
- B. 세로형 막대그래프로 출력하는 코드를 작성해 보세요.
- C. 산점도그래프로 출력하는 코드를 작성해 보세요.
- D. 히스토그램으로 출력하는 코드를 작성해 보세요.