1. 다음 코드의 결과로 올바른 것은?

```
print(np.arange(0, 10, 2))
```

① [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]

② [1, 3, 5, 7, 9] ③ [0, 2, 4, 6, 8, 10] ④ [0, 2, 4, 6, 8]

In [ ]:
```
import numpy as np
a = np.array([2,2,-2,2])
b = np.array([1, 3, 5])
c = np.array([4, 2, 1])

print(np.dot(b,c))  # 1차원 일때 내적
print(b*c)
print(np.sum(b*c))

print(np.linalg.norm(b)) # L2 노름 (유크리드 노름)
print(np.linalg.norm(b,1)) #L1 노름 (맨해튼 노름)
```

```
15
[4 6 5]
15
5.916079783099616
9.0
```

In [ ]:
```
import numpy as np
a = np.array([10,11,12,13,14])
b = np.array([1,2,3,4,5])
#벡터 덧셈뺄셈
print(a+b)
print(a-b)
c= np.array([1,2,3])
d= np.array([-1,-2,4])
#c d 벡터의 내적 외적
print(np.dot(c,d))
print(np.cross(c,d))

#d의 유클리드 노름을 구하시오
#d의 맨해튼 노름을 구하시오
print(np.linalg.norm(d))
print(np.linalg.norm(d,1))


e = np.array([[4,-2],[3,-2]])
#벡터의 행력실 과 역행렬 구하시오
print(np.linalg.det(e), np.linalg.inv(e))
```

```
[11 13 15 17 19]
[9 9 9 9 9]
7
[14 -7  0]
4.58257569495584
7.0
-2.0 [[ 1.  -1. ]
 [ 1.5 -2. ]]
```

In [ ]:
```
import numpy as np
a = np.arange(0,10,2)
print(a)
```

```
[0 2 4 6 8]
```

2. 다음 중 넘파이(numpy)로 생성한 다차원 배열의 속성과 연결이 올바르지 않은 것은?

① ndim : 배열의 축 혹은 차원의 개수를 나타냄
② shape : (m, n) 형식으로 배열의 형상을 나타냄
③ size : 배열 원소의 크기를 바이트 단위로 기술함
④ dtype : 배열 원소의 자료형을 기술함

In [ ]:
```
import numpy as np
a = np.array([1,2,3])
print(a.ndim)
print(a.shape)
print(a.size) #배열 원소의 개수 서술
print(a.dtype)
```

```
1
(3,)
3
int32
```

3. 다음 코드의 결과로 올바른 것은?

```
import numpy as np
a = np.array([1, 2, 3])
print(a.shape)
```

① (3,)　② (3, 1)
③ (1, 3)　④ (2, 3)

In [ ]:
```
import numpy as np
a = np.array([1,2,3])
print(a.shape)
#답 1
```

```
(3,)
```

Loading [MathJax]/extensions/Safe.js

4. 넘파이는 a * 10 을 수행할 때 np.array([10, 20, 30]) * np.array([10, 10, 10])와 같이 a의 차원에 맞게 스칼라 10을 벡터로 확장시켜주는 작업을 하는데 이를 무엇이라 하는가?

① 형상(shape) ② 브로드캐스팅(broadcasting)
③ 시각화(visualization) ④ 벡터화 연산(vectorized operation)

In [ ]:
```
#shape(행,열)
import numpy as np
ar = np.array([[1,2,1],
               [3,3,3]])
ar1 = np.array([1,2,3])
print("Broadcasting")
print(ar+ar1)
```

```
Broadcasting
[[2 4 4]
 [4 5 6]]
```

5. 다음 코드의 결과로 올바른 것은?

```
arr = np.array([[1, 2, 3],
                [4, 5, 6],
                [7, 8, 9],
                [0, 1, 2]])
print(arr[1, 1:])
```

① [2, 3] ② [1, 2, 3]
③ [5, 6] ④ [4, 5, 6]

In [ ]:
```
print("arr(행,열) 3번")
import numpy as np
arr = np.array([[1,2,3],[4,5,6],[7,8,9],[0,1,2]])
print(arr[1,1:])
```

```
arr(행,열) 3번
[5 6]
```

6. 다음 코드의 결과를 참고하여 빈공 올바른 것은?

```
import numpy as np
a = np.array([1, 3, 4])
np.insert(a, 1, 2) # 변수, 인덱스, 값
```

① array([1, 2, 3, 4]) ② array([1, 3, 4, 1])
③ array([1, 3, 4, 2]) ④ array([1, 2, 4, 1, 2])

In [ ]:
```
import numpy as np
a = np.array([1,3,4])
```

```
np.insert(a,1,2)
print(a)
```

```
[1 3 4]
```

7. 다음 코드의 결과로 올바른 것은?

```
for x in range(2, 5):
    print(x, end=' ')
```

① 1 3 5 ② 1 2 3 4
③ 2 3 4 ④ 2 3 4 5

In [ ]:
```
for x in range(2,5):
    print(x,end=" ")
```

```
2 3 4
```

8. 다음 코드의 결과를 참고하여 빈 공간에 들어갈 함수는 무엇인가?

```
import numpy as np
c = np.array([[1, 1], [3, 3], [4, 4]])
np.[      ](c, 1, 0, axis = 1)

array([[1, 0, 1],
       [3, 0, 3],
       [4, 0, 4]])
```

① append
② insert
③ flip
④ flatten

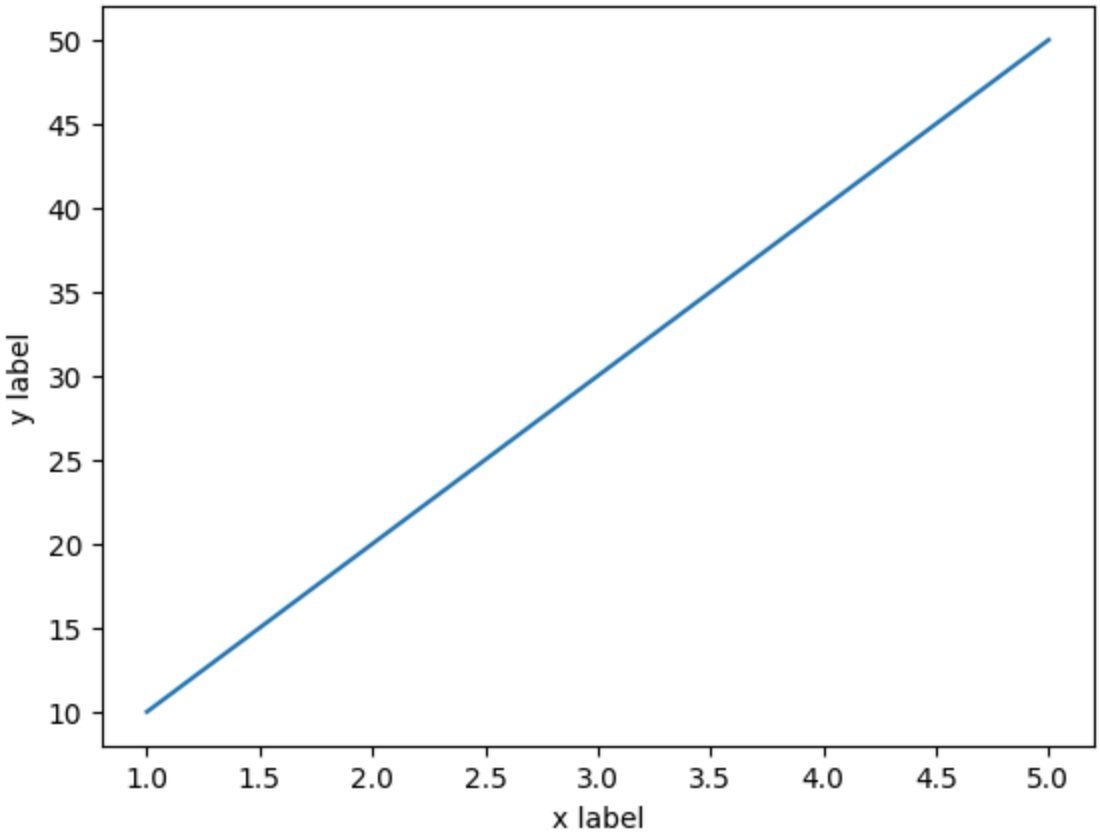
In [ ]:
```
import numpy as np
c = np.array([[1,2],[4,5],[5,6]])
c = np.insert(c,1,0, axis = 1)

print(c)
print("1열에 0을 추가하는건데... 추가가왜안되지...")

c = np.array([[1, 1], [3, 3], [4, 4]])
c = np.insert(c, 1, 2, axis=0)
np.insert(c, 1, 0, axis = 1)
```

```
[[1 0 2]
 [4 0 5]
 [5 0 6]]
1열에 0을 추가하는건데... 추가가왜안되지...
```

Out[ ]:
```
array([[1, 0, 1],
       [2, 0, 2],
       [3, 0, 3],
       [4, 0, 4]])
```

In [ ]:
```
import time
import matplotlib.pyplot as plt
import numpy as np


plt.plot([1, 2, 3, 4,5], [10, 20, 30, 40, 50])
```

```
plt.xlabel('x label')
plt.ylabel('y label')
plt.show()
```



9. 다음 코드의 결과로 올바른 것은?

```
import numpy as np
c = np.array([5, 3, 6, 2, 8, 4])
c.sort()
c[::-1]
```

① array([2, 3, 4, 5, 6, 8])
② array([2, 3, 4, 5, 6])
③ array([8, 6, 5, 4, 3])
④ array([8, 6, 5, 4, 3, 2])

In [ ]:

10. 다음 코드의 결과를 참고하여 빈 공간에 들어갈 함수는 무엇인가?

```
import numpy as np
a = np.arange(12)
print(a.        (3, 4))
```

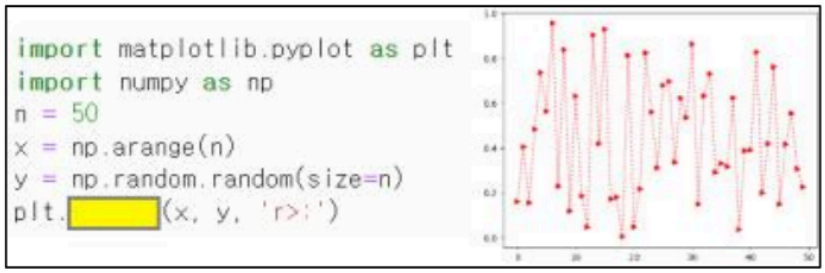
```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
```

① append
② reshape
③ flip
④ flatten

In [ ]:
```
import numpy as np
a = np.arange(12)
print(a.reshape(3,4))
print(np.flip(a))
print(a.flatten())
print(a.flatten()[::-1])
```

```
[[ 0  1  2  3]
 [ 4  5  6  7]
 [ 8  9 10 11]]
[11 10  9  8  7  6  5  4  3  2  1  0]
[ 0  1  2  3  4  5  6  7  8  9 10 11]
[11 10  9  8  7  6  5  4  3  2  1  0]
```

11. 다음과 같이 그래프가 출력되도록 하는 빈 공간에 들어갈 함수는 무엇인가?

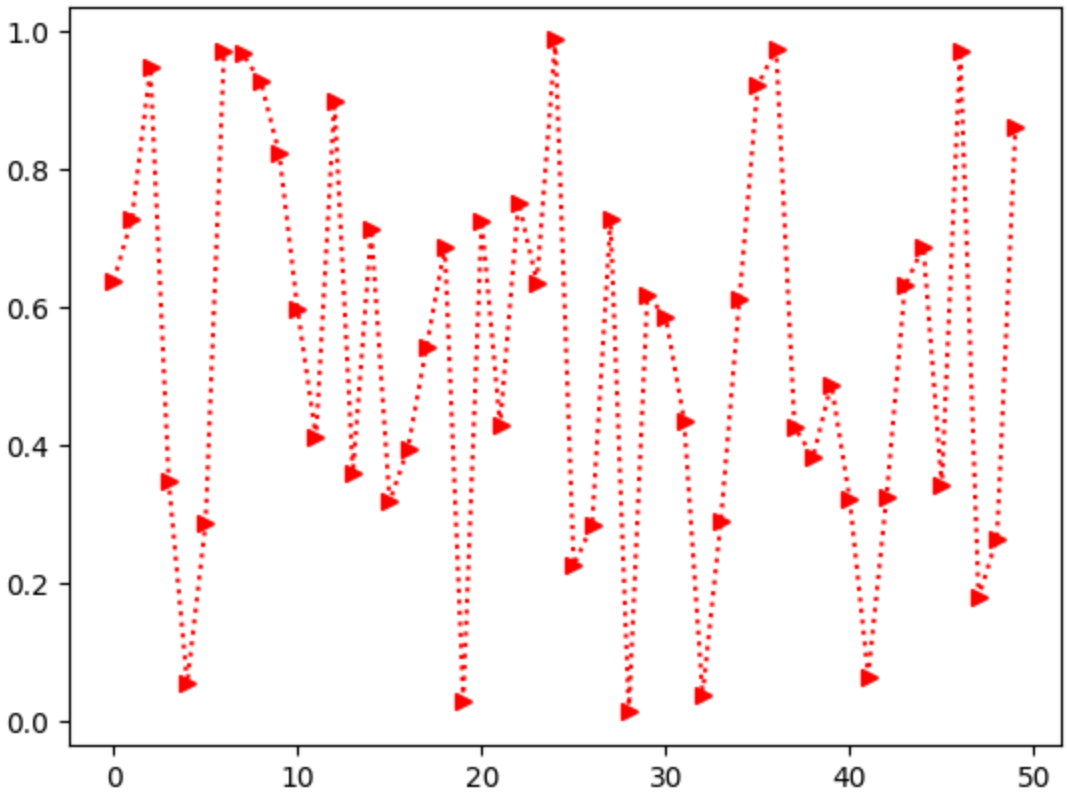


① scatter ② plot ③ bar ④ pie

In [ ]:
```
import matplotlib.pyplot as plt
import numpy as np
n = 50
x = np.arange(n)
y = np.random.random(size =n)
plt.plot(x,y,'r>:') # x,y array를 기준으로 빨간색 세모(>) 선
```

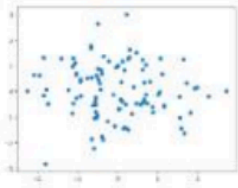
Out[ ]:
```
[<matplotlib.lines.Line2D at 0x24c731f2a88>]
```

Loading [MathJax]/extensions/Safe.js



12. 다음 코드와 그래프는 두 변수의 상관관계를 표현하는 산점도에 관한 것이다. 빈 공간에 들어갈 함수는 무엇인가?

```
import matplotlib.pyplot as plt
import numpy as np
x = np.random.randn(100)
y = np.random.randn(100)
plt.      (x, y)
```

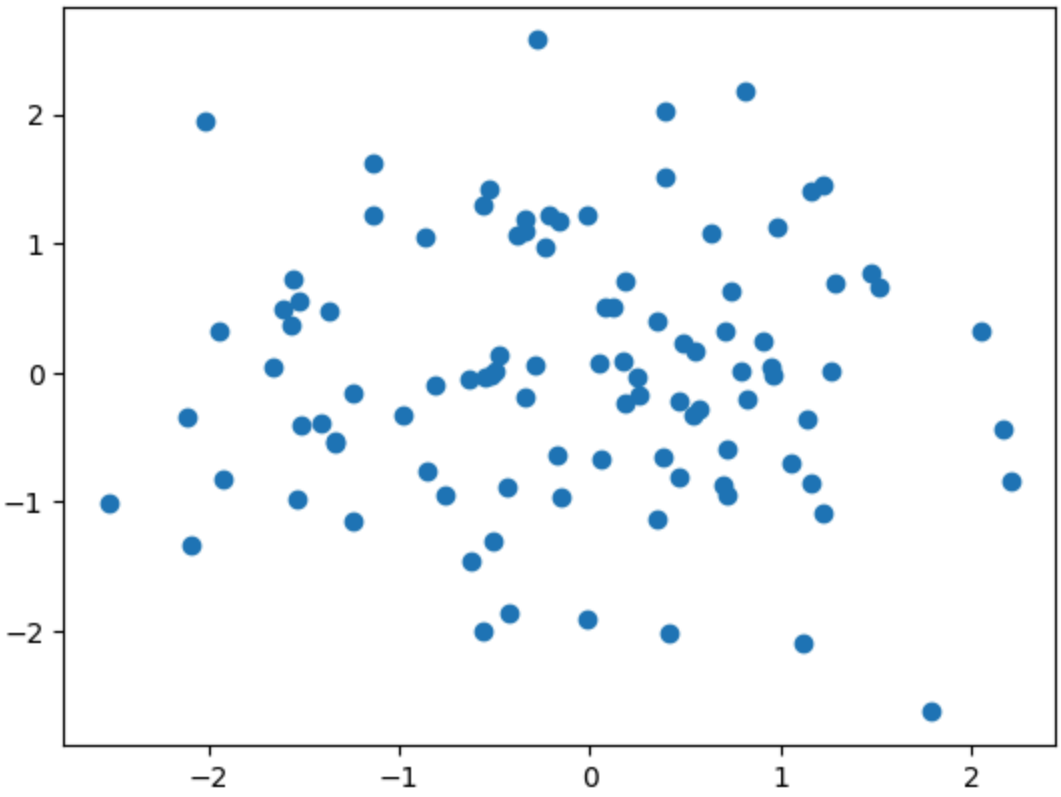


① scatter ② plot ③ bar ④ pie

In [ ]:
```
#점은 scatter!
import matplotlib.pyplot as plt
import numpy as np
x = np.random.randn(100) # 정규분포 -1~1
y = np.random.randn(100)
plt.scatter(x,y)
```

Out[ ]:
```
<matplotlib.collections.PathCollection at 0x24c7325b848>
```

Loading [MathJax]/extensions/Safe.js



13. 다음 코드의 결과로 올바른 것은?

```
a =[1, 2, 3]
for i in a:
    print(i * 2, end=' ')
```

① 1 2 3 ② 2 4 6
③ 1 2 3 1 2 3
④ 2 4 6 2 4 6

In [ ]:
```
a=[1,2,3]
print(*[i * 2 for i in a ])
```

```
2 4 6
```

In [ ]:
```

```

14. 다음 중 리스트 슬라이싱의 결과로 올바르지 않은 것은?

```
price = [100, 200, 300, 400, 500]
```

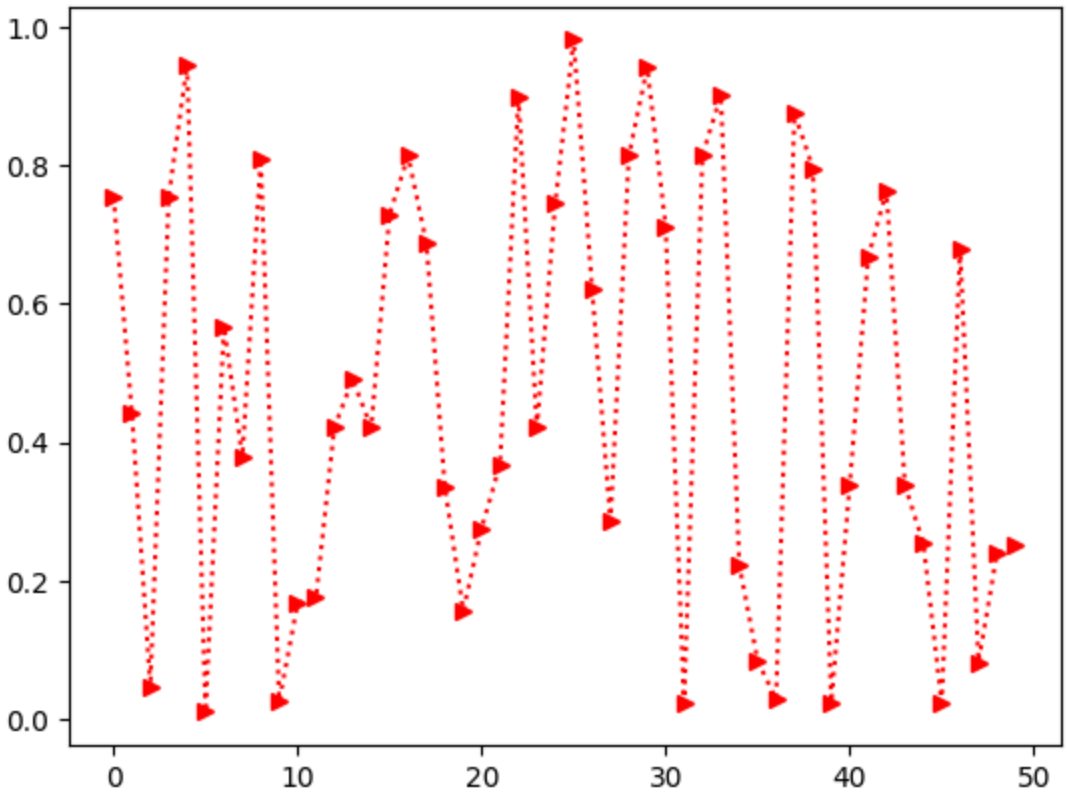
① print(price[0:3]) ☞ [100, 200, 300]
② print(price[0:4:2]) ☞ [100, 300]
③ print(price[1:2]) ☞ [200, 400]
④ print(price[1:-1]) ☞ [200, 300, 400]

In [ ]:
```
price = [100,200,300,400,500]
print(price[0:3])
print(price[0:4:2])
print(price[1:2])
print(price[1:-1])
```

```
[100, 200, 300]
[100, 300]
[200]
[200, 300, 400]
```

In [ ]:
```
import matplotlib.pyplot as plt
import numpy as np
n = 50
x = np.arange(n)
y = np.random.random(size =n)
plt.plot(x,y,'r>:') # x,y array를 기준으로 빨간색 세모(>) 선
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x24c732be3c8>]
```



15. 다음 코드의 결과는 무엇인가?

```
DB = [['사과', 100], ['딸기', 200], ['포도', 300]]
print(DB[1][0])
```

① 딸기 ② ['딸기', 200] ③ 사과 ④ ['사과', 100]

In [ ]:
```
DB =[['사과',100],['딸기',200],['포도'],300]
print(DB[1][0])
```

딸기

16. 다음 코드의 결과는 문자열의 길이를 반환한 것이다. 빈 공간에 들어갈 함수는 무엇인가?

```
str = '빅데이터분석'
print( 빈 공간 (str))
```

```
6
```

① len
② length
③ count
④ index

In [ ]:
```
str = 'ㅇㅇㅇㅇㅇㅇㅇㅇㅇㅇㅇ'
print(len(str))
```

11

Loading [MathJax]/extensions/Safe.js

17. 다음 리스트에서 출력 결과가 다른 것은 무엇인가?

```
fruits = ['사과', '오렌지', '포도', '복숭아']
```

① print(fruits[3])
② print(fruits[-4])
③ print(fruits[-1])
④ print(fruits[len(fruits)-1])

In [ ]:
```
fruits = ['사과', '오렌지', '포도', '복숭아']

print( fruits[3] )
print( fruits[-4] )
print( fruits[-1] )
print( fruits[len(fruits)-1] )
```

```
복숭아
사과
복숭아
복숭아
```

18. 다음 중 빈 공간에 들어갈 함수로 올바른 것은?

```
prime = [2, 3, 5]
prime.빈 공간(7)
print(prime)
```

```
[2, 3, 5, 7]
```

① delete
② append
③ insert
④ push

In [ ]:
```
prime =[2,3,5]
prime.append(7)
print(prime)
```

```
[2, 3, 5, 7]
```

07_2 선형 변환 및 고유값

In [ ]:
```
import numpy as np
a =[[1,2,3],[2,-3,2],[3,1,-1]]
b = np.array([6,14,-2])
print(np.linalg.solve(a,b))

a = np.array([[1,0,3],
              [0,2,6],
              [6, 14,-2]])
b = np.array([6,14,-2])
print(np.linalg.solve(a,b))
```

```
w, v = np.linalg.eig(a)
print("고유값",w)
print("고유백터",v)
```

```
[ 1. -2.  3.]
[-0.58064516  0.41935484  2.19354839]
고유값 [-10.37221314   1.17204084  10.2001723 ]
고유백터 [[ 0.23094495 -0.92214132 -0.25449436]
 [ 0.42455706  0.38322172 -0.5710592 ]
 [-0.87545173 -0.05288199 -0.78046397]]
```

In [ ]:
```
import numpy as np
a = np.array([[1,3],
              [2,6]])

w, v = np.linalg.eig(a)
print(w)
print(v)
```

```
[0. 7.]
[[-0.9486833  -0.4472136 ]
 [ 0.31622777 -0.89442719]]
```

1. 다음 중 데이터프레임(DataFrame)을 조작할 수 있는 다양한 함수를 지원하는 라이브러리는 무엇인가?

① numpy ② random ③ matplotlib ④ pandas

In [ ]:
```
#4번
import numpy as np
import pandas as pd
a = np.array([[1,2],[1,2],[1,2]])
print(pd.DataFrame(a))
```

```
   0  1
0  1  2
1  1  2
2  1  2
```

2. 다음 코드의 결과를 참고하여 빈 공간에 들어갈 코드는?

```
import numpy as np
import pandas as pd
file = 'vehicle_prod.csv'
df = pd.read_csv(file, index_col=0)
print(df)
print(          )
```

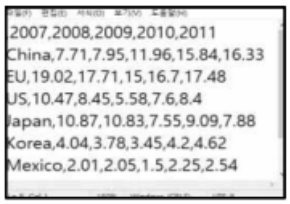
```
         2007   2008   2009   2010   2011
China    7.71   7.95  11.96  15.84  16.33
EU      19.02  17.71  15.00  16.70  17.48
US      10.47   8.45   5.58   7.60   8.40
Japan   10.87  10.83   7.55   9.09   7.88
Korea    4.04   3.78   3.45   4.20   4.62
Mexico   2.01   2.05   1.50   2.25   2.54
Index(['2007', '2008', '2009', '2010', '2011'], dtype='object')
```

① df.index
② df.columns
③ df.head(5)
④ df.tail(5)

In [ ]:
```
import numpy as np
import pandas as pd
file = 'vehicle_prod.csv'
df = pd.read_csv(file,index_col=0)
print(df)
print(df.index)
```

```
         2007   2008   2009   2010   2011
China    7.71   7.95  11.96  15.84  16.33
EU      19.02  17.71  15.00  16.70  17.48
US      10.47   8.45   5.58   7.60   8.40
Japan   10.87  10.83   7.55   9.09   7.88
Korea    4.04   3.78   3.45   4.20   4.62
Mexico   2.01   2.05   1.50   2.25   2.54
Index(['China', 'EU', 'US', 'Japan', 'Korea', 'Mexico'], dtype='object')
```

3. 다음 그림과 같이 쉼표로 구분한 변수 구성된 파일을 무엇이라 하는가?



① CSV (comma separated variables)
② TSV (tab separated variables)
③ JSON (savascript object notation)
④ py (python)

In [ ]:
```
#1번 csv파일
import numpy as np
import pandas as pd
file = 'vehicle_prod.csv'
df = pd.read_csv(file,index_col=0)
print(df)
```

```
         2007   2008   2009   2010   2011
China    7.71   7.95  11.96  15.84  16.33
EU      19.02  17.71  15.00  16.70  17.48
US      10.47   8.45   5.58   7.60   8.40
Japan   10.87  10.83   7.55   9.09   7.88
Korea    4.04   3.78   3.45   4.20   4.62
Mexico   2.01   2.05   1.50   2.25   2.54
```

4. 다음 코드와 결과를 참고하여 노란색 화살표가 가리키는 결과처럼 출력되지 않는 것은 무엇인가??

```
df = pd.read_csv(file, index_col=0)
print(df)
```

	2007	2008	2009	2010	2011
China	7.71	7.95	11.96	15.84	16.33
EU	19.02	17.71	15.00	16.70	17.48
US	10.47	8.45	5.58	7.60	8.40
Japan	10.87	10.83	7.55	9.09	7.88
Korea	4.04	3.78	3.45	4.20	4.62
Mexico	2.01	2.05	1.50	2.25	2.54

	2007	2008	2009	2010	2011
US	10.47	8.45	5.58	7.60	8.40
Japan	10.87	10.83	7.55	9.09	7.88

① df[3, 5]
② df.iloc[2:4]
③ df.iloc[[2, 3]])
④ df[2:4]

In [ ]:
```
#1번 csv파일
import numpy as np
import pandas as pd
file = 'vehicle_prod.csv'
df = pd.read_csv(file,index_col=0)
print(df)
```

```
         2007   2008   2009   2010   2011
China    7.71   7.95  11.96  15.84  16.33
EU      19.02  17.71  15.00  16.70  17.48
US      10.47   8.45   5.58   7.60   8.40
Japan   10.87  10.83   7.55   9.09   7.88
Korea    4.04   3.78   3.45   4.20   4.62
Mexico   2.01   2.05   1.50   2.25   2.54
```

5. 다음 중 데이터프레임의 데이터를 갱신하는 속성은 무엇인가?

① inplace=false ② inplace=true ③ update=false ④ update=true

In [ ]:
```
import numpy as np
import pandas as pd
df = pd.DataFrame(data=[[10, 20, 30, 40], [50, 60, 70, 80]],
columns=['A', 'B', 'C', 'D'])
new_df = df.drop('B', axis=1, inplace=True)
print(df)
```

```
    A   C   D
0  10  30  40
1  50  70  80
```

6. 다음 중 데이터프레임의 처음 3행만 가져오는 함수는?

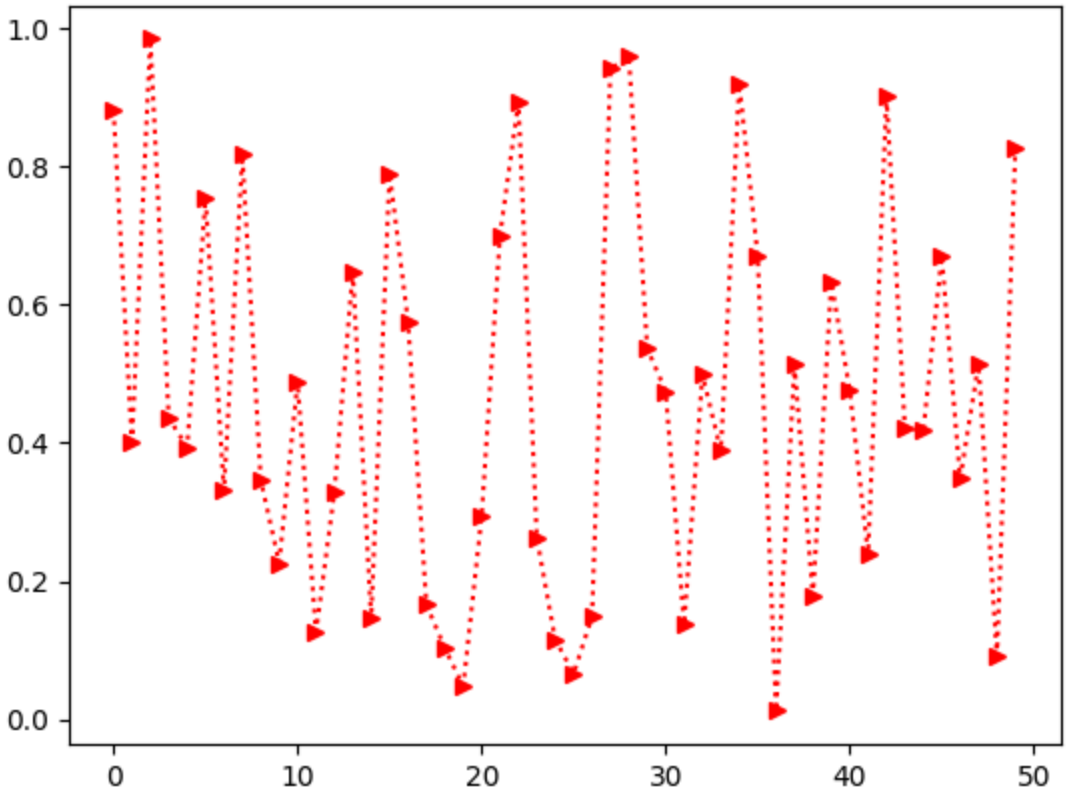
① first(3) ② start(3) ③ head(3) ④ tail(3)

In [ ]:
```
import numpy as np
import pandas as pd
file = 'weather.csv'
df = pd.read_csv(file, index_col=0, encoding='CP949')
print(df.head(3))
```

```
            평균기온  최대풍속  평균풍속
일시
2010-08-01  28.7   8.3   3.4
2010-08-02  25.2   8.7   3.8
2010-08-03  22.1   6.3   2.9
```

In [ ]:
```
import matplotlib.pyplot as plt
import numpy as np
n = 50
x = np.arange(n)
y = np.random.random(size =n)
plt.plot(x,y,'r>:') # x,y array를 기준으로 빨간색 세모(>) 선
```

Out[ ]: [<matplotlib.lines.Line2D at 0x24c705b4a48>]




```
import numpy as np
from sklearn import linear_model
regr = linear_model.LinearRegression()
x = [[163],[179],[166],[169],[171]]
y = [54,63,57,56,58]
regr.  A  (x,y) # 학습수행 : 독립변수x,종속변수y
coef=regr.coef_                 # 직선의 기울기
intercept = regr.intercept_     # 직선의 절편
score = regr.  B  (x,y)         # 학습된 직선이 데이터를 잘 수렴하는지
print("y = {}* x +{:.2f} ".format(coef.round(2),intercept))
print("데이터와 선형회귀 직선의 관계 점수 :{:.1%}".format(score))
```

```
y = [0.53]* x +-32.50
데이터와 선형회귀 직선의 관계 점수 :91.9%
```

7. 상기 코드에서 빈 공간 A 에 들어갈 함수는?

① fit ② train ③ predict ④ score

8. 상기 코드에서 빈 공간 B 에 들어갈 함수는?

① fit ② train ③ predict ④ score

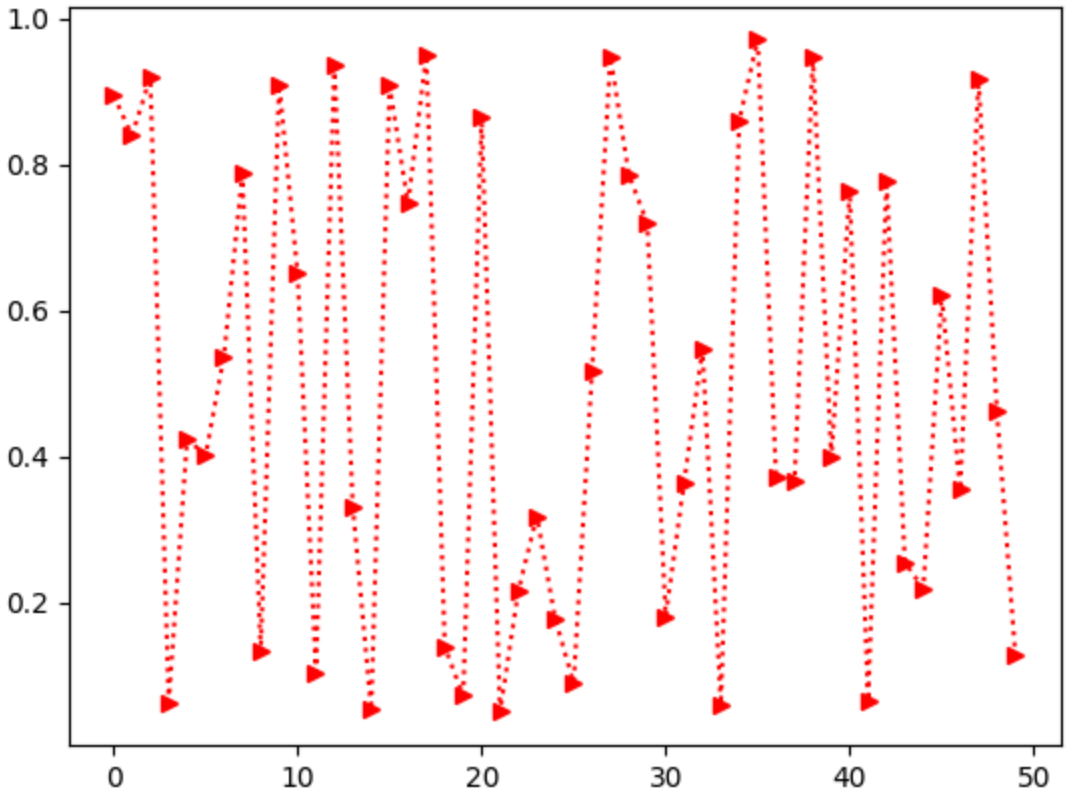
In [ ]:
```
import numpy as np
from sklearn import linear_model
```

```
regr = linear_model.LinearRegression()
X = [[163], [179], [166], [169], [171]]
y = [54, 63, 57, 56, 58]
regr.fit(X, y)
coef = regr.coef_            # 직선의 기울기
intercept = regr.intercept_  # 직선의 절편
score = regr.score(X, y)  # 모델 점수
print("y = {}* X + {:.2f}".format(coef.round(2), intercept))
print("데이터와 선형회귀 직선의 관계점수: {:.1%}".format(score))
```

```
y = [0.53]* X + -32.50
데이터와 선형회귀 직선의 관계점수: 91.9%
```

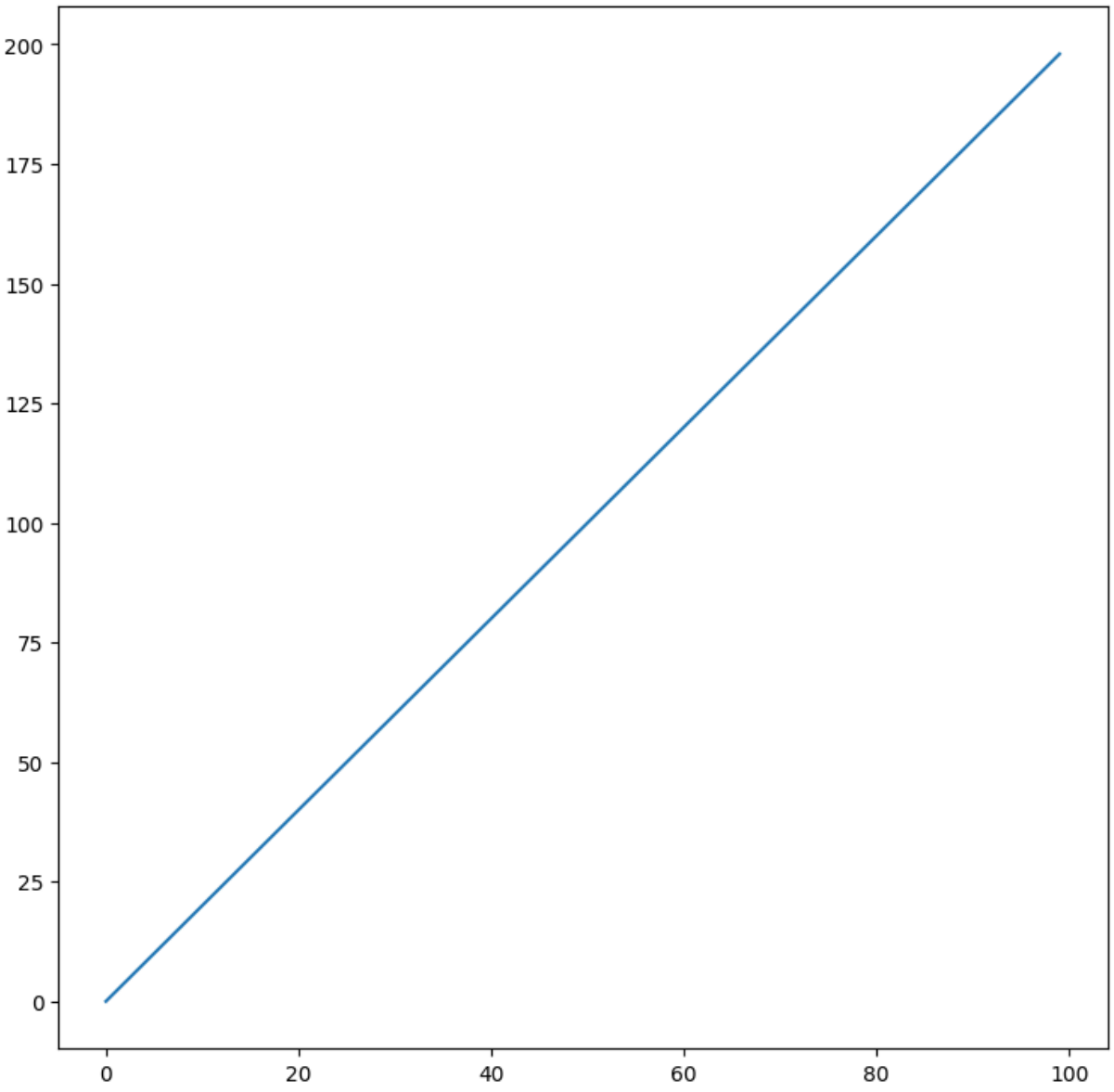
In [ ]:
```
import matplotlib.pyplot as plt
import numpy as np
n = 50
x = np.arange(n)
y = np.random.random(size =n)
plt.plot(x,y,'r>:') # x,y array를 기준으로 빨간색 세모(>) 선
```

Out[ ]: [<matplotlib.lines.Line2D at 0x24c737c0e88>]



In [ ]:
```
import numpy as np
import matplotlib.pyplot as plt
x = np.arange(0,100)
y = 2 * np.arange(0,100)
plt.figure(figsize=(9,9))  # 가로 세로 9인치
plt.plot(x,y)
```

Out[ ]: [<matplotlib.lines.Line2D at 0x24c736b0988>]



In [ ]:
```
import numpy as np
a = np.array([[1,2,3,4,5],[7,8,9,10,11]])
print(np.flip(a))
print(np.flip(a,axis=0))
print(np.flip(a,axis=1))
```

```
[[11 10  9  8  7]
 [ 5  4  3  2  1]]
[[ 7  8  9 10 11]
 [ 1  2  3  4  5]]
[[ 5  4  3  2  1]
 [11 10  9  8  7]]
```

In [ ]:
```
a = np.array([[1,2,3,4,5],[7,8,9,10,11]])
print(a)
a.sort(axis=0)
print(a)
```

```
[[ 1  2  3  4  5]
 [ 7  8  9 10 11]]
[[ 1  2  3  4  5]
 [ 7  8  9 10 11]]
```

In [ ]:
```python
# 리스트로 행렬 표현
A = [ [1, 0], [0, 1] ]
B = [ [1, 1], [1, 1] ]

A + B    # 행렬 연산이 아닌 리스트 연산
[[1, 0], [0, 1], [1, 1], [1, 1]]

# numpy matrix
A = np.array([ [1, 0], [0, 1] ])
B = np.array([ [1, 1], [1, 1] ])

print("행렬연산+",A+B)

# vector A
A = np.arange(0,10)
print("A ==", A)

# vector A형상 출력 => shape(행개수,열개수)
print("A.shape ==", A.shape)
```

```
행렬연산+ [[2 1]
 [1 2]]
A == [0 1 2 3 4 5 6 7 8 9]
A.shape == (10,)
```

In [ ]:
```python
#형변환(reshape) →말그대로 행열을 변환
A = np.array([1, 2, 3])
print(A)
A = A.reshape(1,3)
print(A)
print(A.shape)

# vector A차원 출력 => ndim
print("A.ndim ==", A.ndim)

#1열 n행
A = A.reshape(-1,1)
print(A)
print(A.size)
print(A.ndim)
```

```
[1 2 3]
[[1 2 3]]
(1, 3)
A.ndim == 2
[[1]
 [2]
 [3]]
3
2
```

In [ ]:
```
a = np.array([1, 2, 3])
b = np.array([[4, 5, 6], [7, 8, 9]])
print(a.shape, b.shape, a.ndim, b.ndim)
np.append(a, b)
#1차원 배열에 추가시 1차원배열내에 추가
```

```
(3,) (2, 3) 1 2
```

Out[ ]:
```
array([1, 2, 3, 4, 5, 6, 7, 8, 9])
```

In [ ]:
```
# 행렬곱(셈)
x = np.array([[1, 2], [3, 4]])
y = np.array([[5, 6], [7, 8]])
print(x)
print(y)
# x * y
print(np.matmul(x, y),"=",x@y)
print(x*y)

#1차원 배열일 때
x = np.array([[1, 2]])
y = np.array([[5, 6]])

print(x*y)
print(np.matmul(x, y),"=",x@y)
```

```
[[1 2]
 [3 4]]
[[5 6]
 [7 8]]
[[19 22]
 [43 50]] = [[19 22]
 [43 50]]
[[ 5 12]
 [21 32]]
[[ 5 12]]
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_14972\1167494180.py in <module>
     13
     14 print(x*y)
---> 15 print(np.matmul(x, y),"=",x@y)
     16

ValueError: matmul: Input operand 1 has a mismatch in its core dimension 0, with guf
unc signature (n?,k),(k,m?)->(n?,m?) (size 1 is different from 2)
```

In [ ]:
```
# 4차원 tensor
C = np.array([[[[1, 2], [3, 4]],
               [[5, 6], [7, 8]]],

              [[[9, 10], [11, 12]],
               [[13, 14], [15, 16]]]])

# 배열 출력
```

```
[[[[ 1  2]
   [ 3  4]]

  [[ 5  6]
   [ 7  8]]]


 [[[ 9 10]
   [11 12]]

  [[13 14]
   [15 16]]]]
```

In [ ]:
```
a = np.random.randint(10, 51, size=7)
print(a)
# 정수형 rand 0~1 을 size = 7 <= 7개수

#심화 버전 randn = -1~1사이로 float 자리수 난수생성후 평균 표준 편차 정의
c = np.random.randn(7) * 5 + 20
#평균
c,c.mean()
#소수점 2자리수까지 반올림
print(c.round(2))
#자료형 변경
c.astype(int)
```

```
[12 50 30 41 25 46 15]
[22.4  19.23 28.7  11.87 19.35 28.92 23.02]
```

Out[ ]:
```
array([22, 19, 28, 11, 19, 28, 23])
```

In [ ]:
```
#더 심화버전
# 난수발생 조건: 평균(loc) 20, 표준편차(scale) 3, 정규분포(size), 3행 * 4열, 소수점 2자
z = np.random.normal(loc=20, scale=3, size=[3, 4]).round(2)
z, z.mean().round(2)
```

Out[ ]:
```
(array([[21.48, 29.49, 20.14, 21.69],
        [20.18, 19.69, 21.05, 22.48],
        [17.  , 27.35, 22.39, 18.98]]),
 21.83)
```

In [ ]:
```
x = np.arange(10)
print(x)
#영어 말 그대로 random적으로 순서 바꾸기
np.random.shuffle(x)
print(x)
```

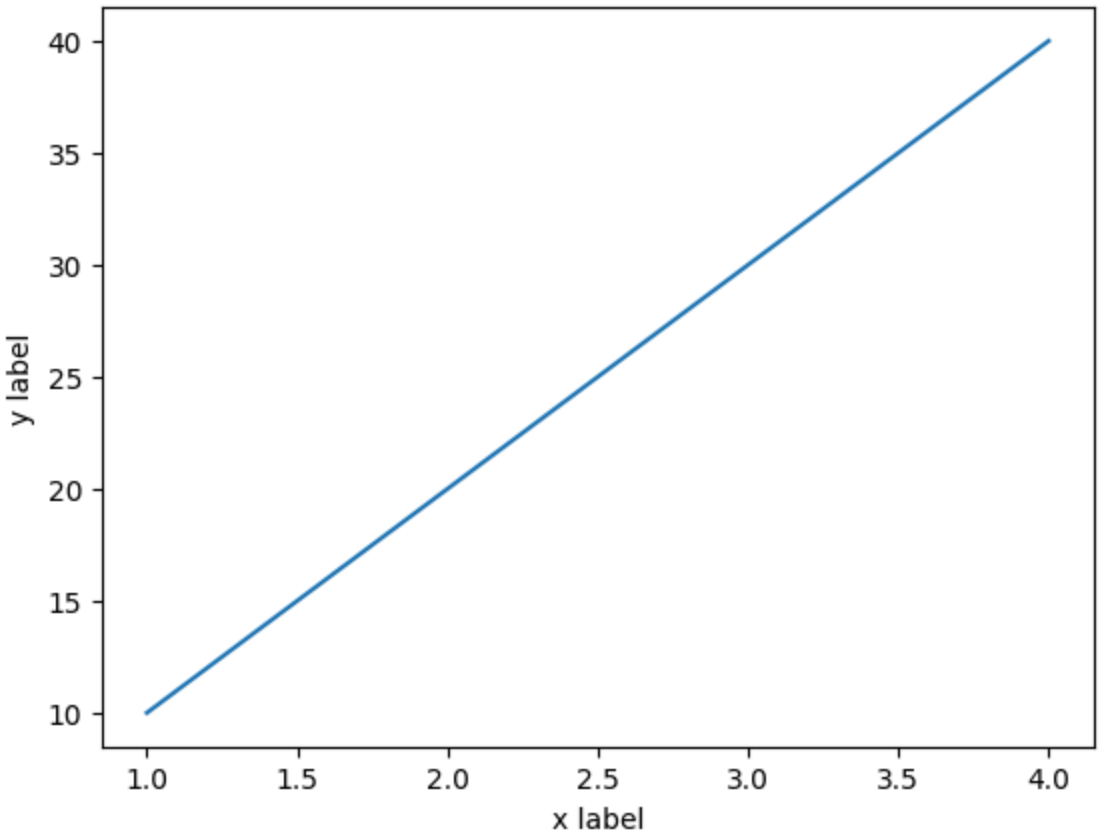
```
[0 1 2 3 4 5 6 7 8 9]
[2 4 0 8 5 3 7 1 9 6]
```

In [ ]:
```
X = np.array([[1,2,3],[4,5,6]])
result = np.sum(X, axis=0)
print(result)
result1 = np.sum(X,  axis=1)
print(result1)
print(np.sum(X,axis=(0,1)), np.sum(X))
```

```
#각각 행만 열만 또는 행,열 합을 구할 수 있다
```

```
[5 7 9]
[ 6 15]
21 21
```

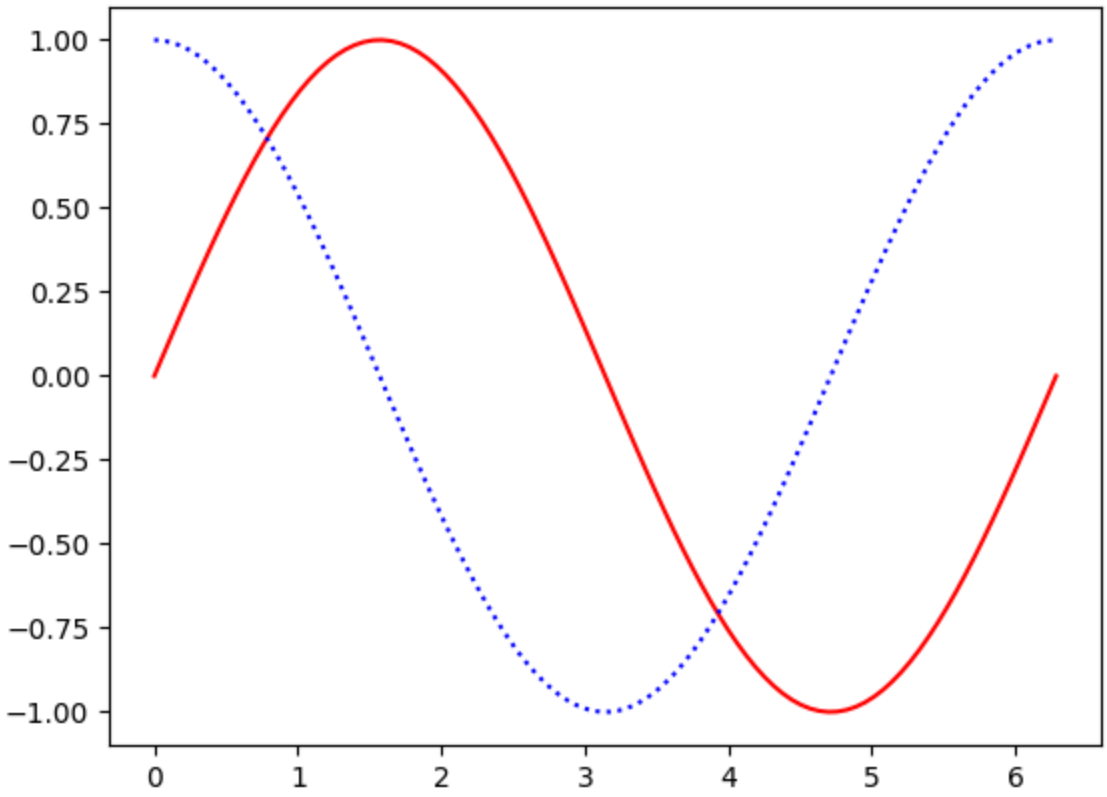
In [ ]:
```
#matplot 기본 생성
plt.plot([1, 2, 3, 4], [10, 20, 30, 40])
plt.xlabel('x label')
plt.ylabel('y label')
plt.show() # 생략 가능
```



In [ ]:
```
x = np.arange(10)
y = x**2
plt.plot(x, y)
plt.xlabel('x label')
plt.ylabel('y label')
plt.axis([0, 9.5, 0, 85])  # 행 0~ 9.5까지 열 0~85까지 범위를 나타냄
```

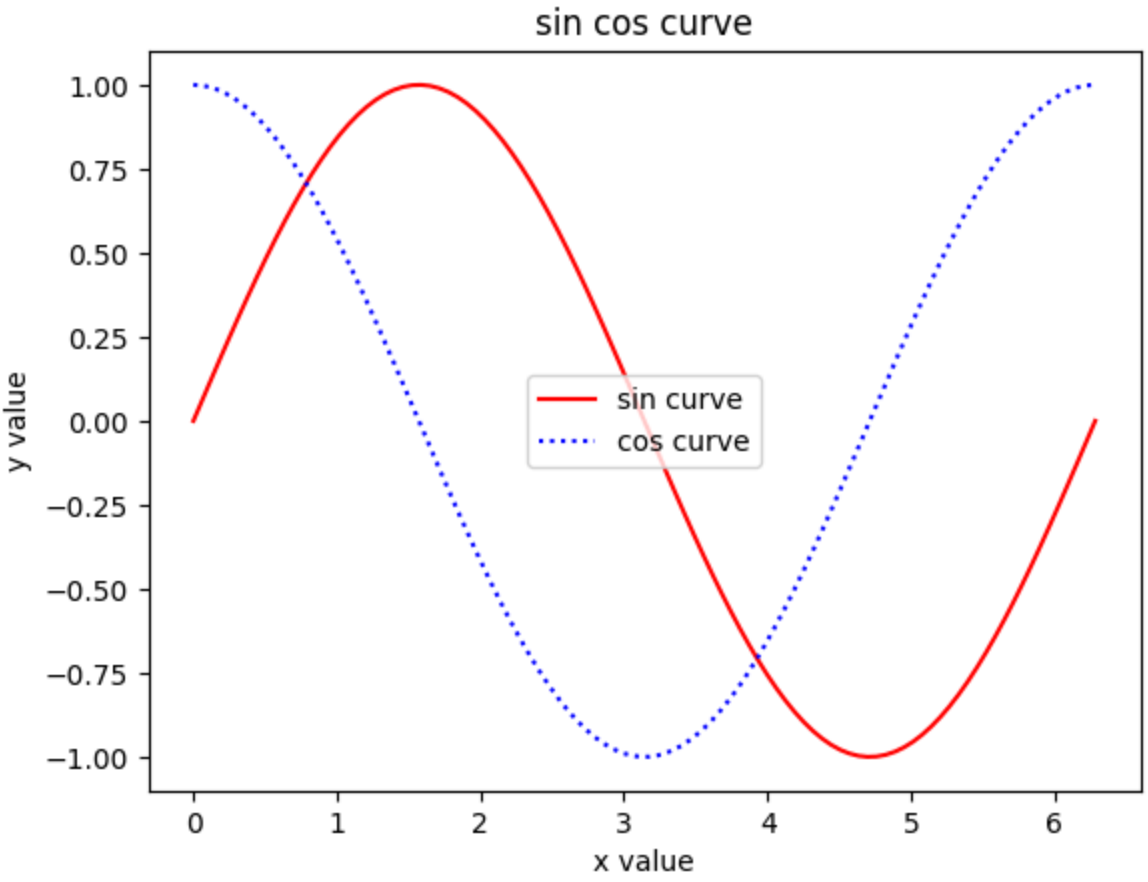
In [ ]:
```
fig = plt.figure()  # 이럴때는 선언 안해도됨

x = np.linspace(0, np.pi * 2, 100) #0~ 2pi 까지 100개 생성
y1 = np.sin(x)
y2 = np.cos(x)
plt.plot(x, y1, 'r-') # 빨간색 선
plt.plot(x, y2, 'b:') # 블루 점선
```

Out[ ]:
```
[<matplotlib.lines.Line2D at 0x1c869708708>]
```



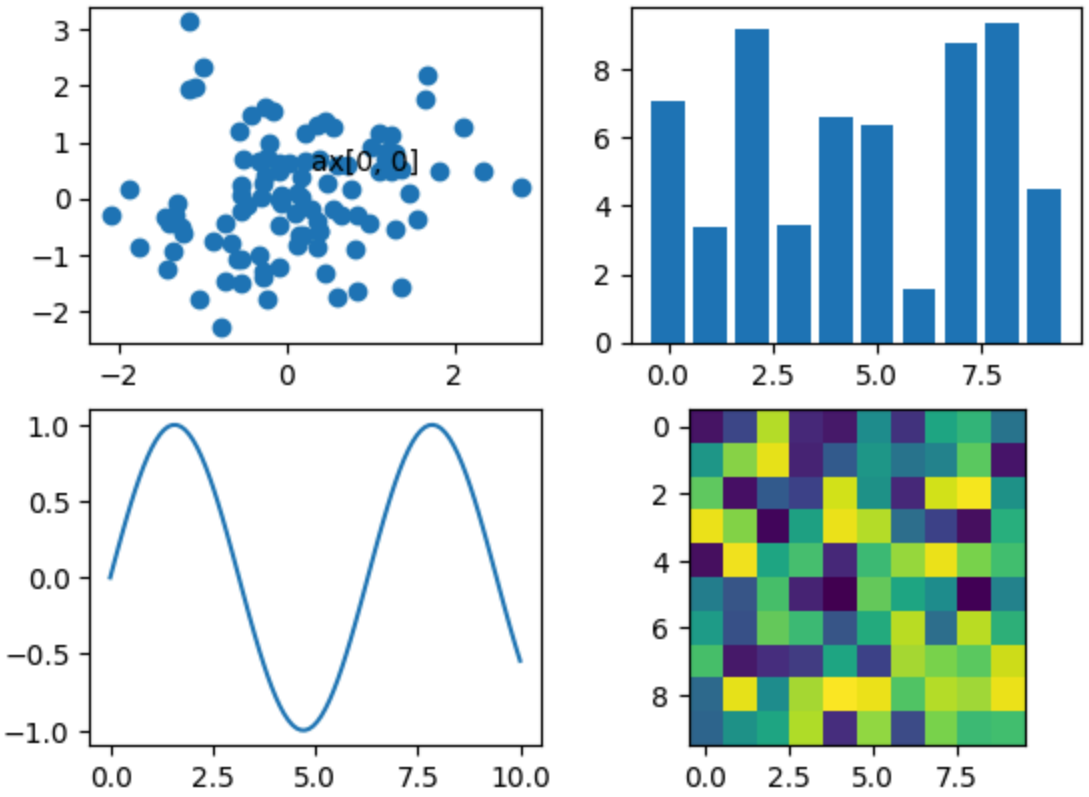
In [ ]:
```
plt.plot(x, y1, 'r-', label='sin curve')
plt.plot(x, y2, 'b:', label='cos curve')

plt.title('sin cos curve') #제목
plt.xlabel('x value') #x축 text
plt.ylabel('y value') #y축  text
plt.legend(loc='center') # 범례 (디폴트) upper lower center left right 그래프 범례 및
print(plt.style.available) #스타일 목록들!
```

```
['Solarize_Light2', '_classic_test_patch', '_mpl-gallery', '_mpl-gallery-nogrid', 'b
mh', 'classic', 'dark_background', 'fast', 'fivethirtyeight', 'ggplot', 'grayscale',
'seaborn', 'seaborn-bright', 'seaborn-colorblind', 'seaborn-dark', 'seaborn-dark-pal
ette', 'seaborn-darkgrid', 'seaborn-deep', 'seaborn-muted', 'seaborn-notebook', 'sea
born-paper', 'seaborn-pastel', 'seaborn-poster', 'seaborn-talk', 'seaborn-ticks', 's
eaborn-white', 'seaborn-whitegrid', 'tableau-colorblind10']
```



In [ ]:
```
#fig 여러개만들기
# scatter, bar, plot, imshow
fig, ax = plt.subplots(nrows=2, ncols=2) #2행2열
# fig, ax = plt.subplots(2,2) 동일
x = np.random.randn(100) # 정규분포를 가지는 데이터
y = np.random.randn(100)
ax[0, 0].scatter(x, y) #점
x = np.arange(10)
y = np.random.uniform(1, 10, 10) # 균일한 분포값 생성 #균등분포  최소 1 최대 10 개수 1
ax[0, 1].bar(x, y) #막대그래프
x = np.linspace(0, 10, 100) #0~10까지 100개
ax[0,0].text(0.3, 0.5, 'ax[0, 0]', fontsize=10) #해당 figure안에 text를 넣을수있다 항
#print(x)
y = np.sin(x)
ax[1, 0].plot(x, y) #선
z = np.random.uniform(0, 1, (10, 10))
ax[1, 1].imshow(z) # 이미지맵
```

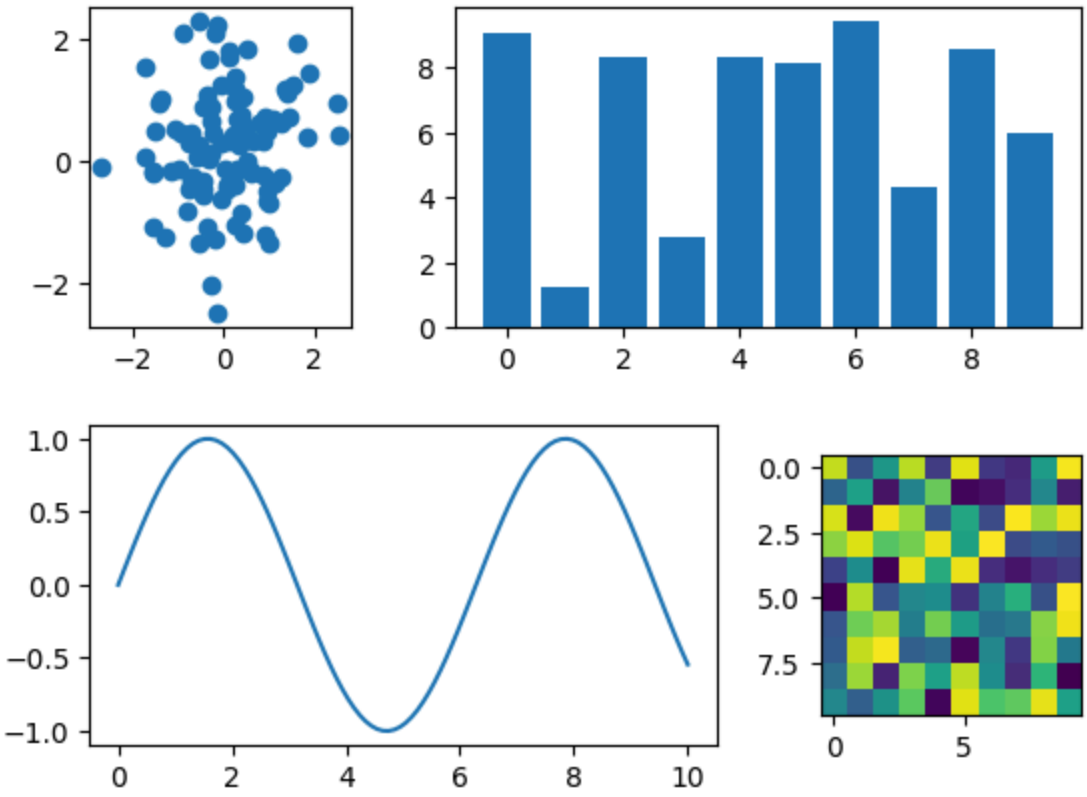
Out[ ]:
```
<matplotlib.image.AxesImage at 0x12d78bb5648>
```



In [ ]:
```
import matplotlib.pyplot as plt
import numpy as np

#그림 figure 두개 크기로
fig, ax = plt.subplots(2, 3)
grid = plt.GridSpec(2, 3, wspace=0.4, hspace=0.3) ##2,3 wspace,hsapace 셀공간 여백 =
x = np.random.randn(100) # 정규분포를 가지는 데이터
y = np.random.randn(100)
plt.subplot(grid[0,0]).scatter(x, y)
x = np.arange(10)
y = np.random.uniform(1, 10, 10)
plt.subplot(grid[0,1:]).bar(x, y) # 0행 1부터 2까지 자리를 먹음
x = np.linspace(0, 10, 100)
y = np.sin(x)
plt.subplot(grid[1,:2]).plot(x, y) # 0~1 자리를 먹음
z = np.random.uniform(0, 1, (10, 10))
plt.subplot(grid[1,2]).imshow(z)
```

Out[ ]:
```
<matplotlib.image.AxesImage at 0x12d78f86b48>
```

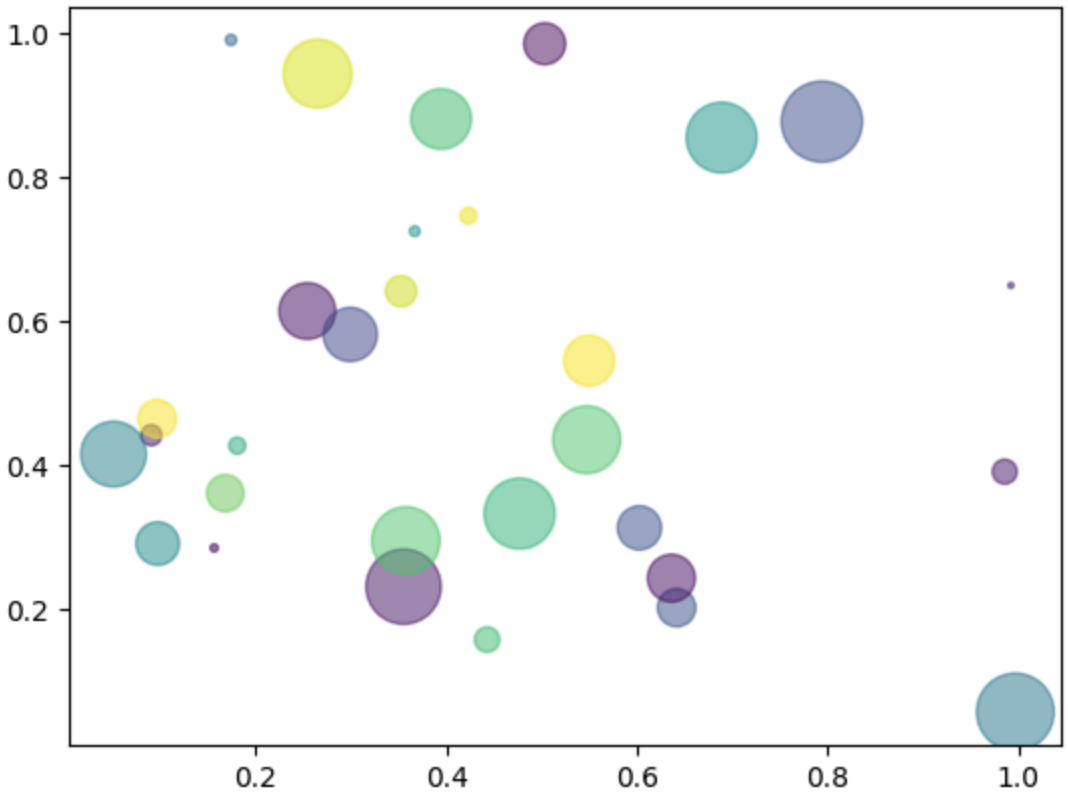
Loading [MathJax]/extensions/Safe.js



In [ ]:
```
n = 30
x = np.random.rand(n)
y = np.random.rand(n)
color = np.random.rand(n)
area = (30*np.random.rand(n))**2
plt.scatter(x, y, s=area, c=color, alpha=0.5) #색상 램덤 영역도 랜덤  alpha:투명도 설
#상관관계가 낮음
```

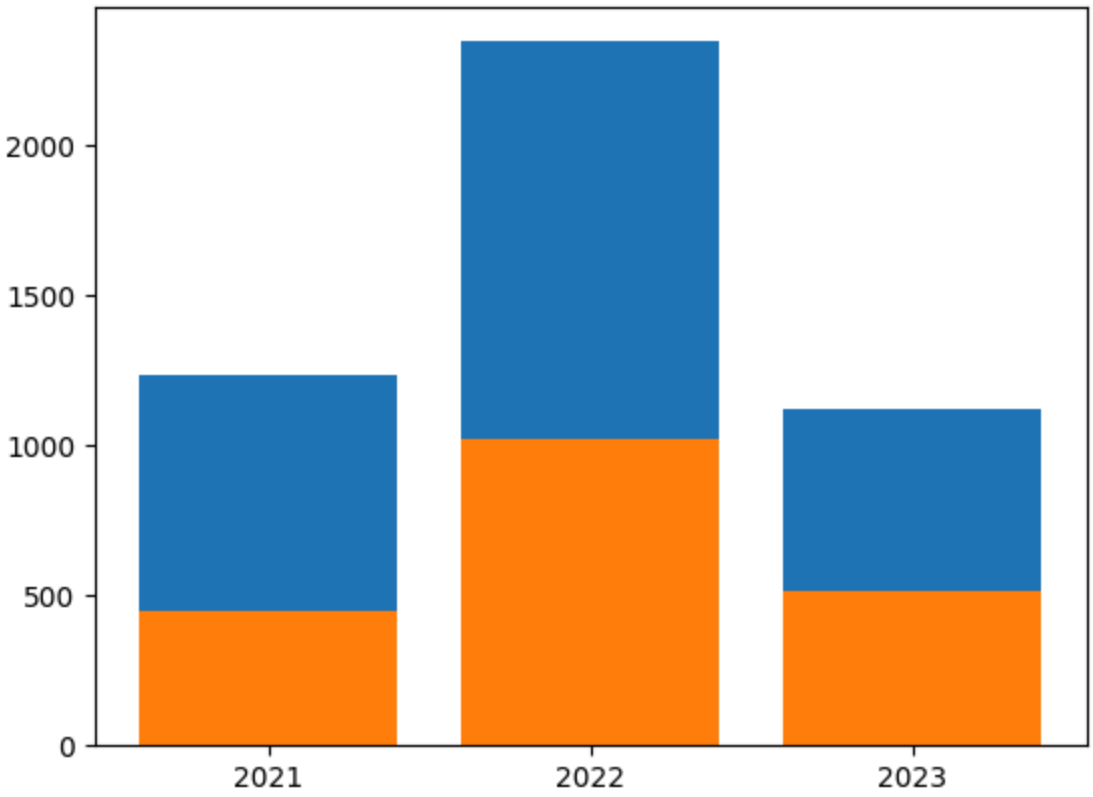
Out[ ]:
```
<matplotlib.collections.PathCollection at 0x12d787d6d08>
```

Loading [MathJax]/extensions/Safe.js



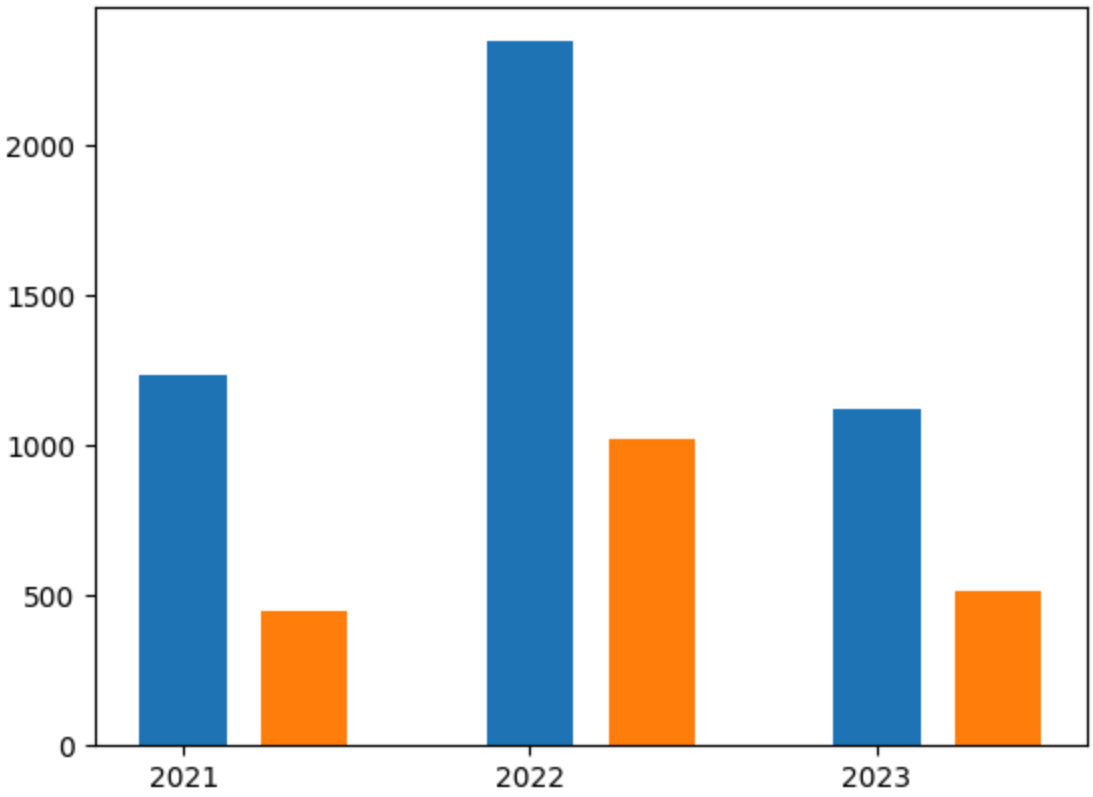
In [ ]:
```
x = np.arange(3)
year = ['2021', '2022', '2023']
y1 = [1234, 2345, 1124]
y2 = [451, 1024, 512]
plt.bar(x, y1) #y1 바 bar: 세로 barh:가로
#plt.barh(x, y1)
plt.bar(x, y2) # y2 바
#plt.barh(x, y2)
plt.xticks(x, year) #라벨 붙이기
#plt.yticks(x, year)
```

Out[ ]:
```
([<matplotlib.axis.XTick at 0x12d78584b08>,
  <matplotlib.axis.XTick at 0x12d78587a48>,
  <matplotlib.axis.XTick at 0x12d78584808>],
 [Text(0, 0, '2021'), Text(1, 0, '2022'), Text(2, 0, '2023')])
```



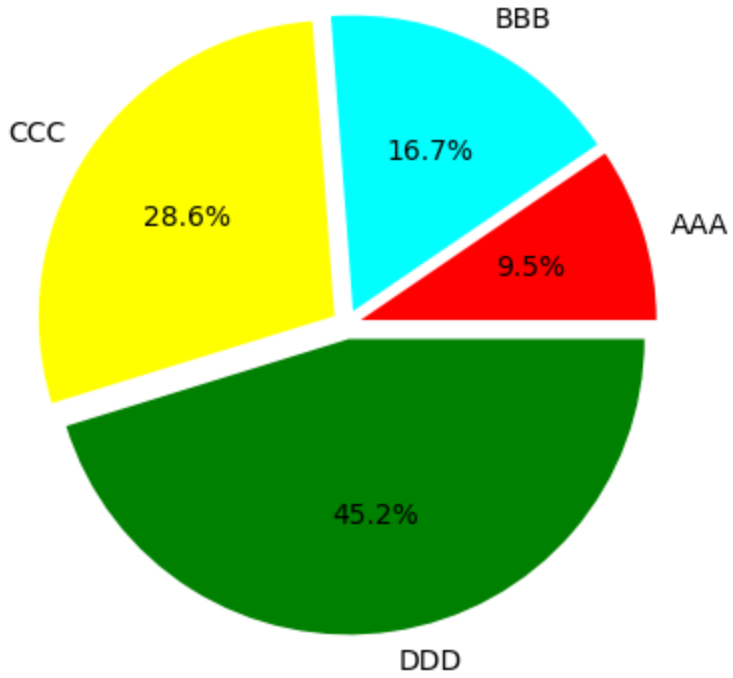
In [ ]:
```
x = np.arange(3)
year = ['2021', '2022', '2023']
y1 = [1234, 2345, 1124]
y2 = [451, 1024, 512]
plt.bar(x, y1, width=0.3) #30프로로 줄이기
plt.bar(x + 0.35, y2, width=0.35) # y1 y2 0.35간격
plt.xticks(x, year)
```

Out[ ]:
```
([<matplotlib.axis.XTick at 0x12d78fc0748>,
  <matplotlib.axis.XTick at 0x12d77533ec8>,
  <matplotlib.axis.XTick at 0x12d78fc0448>],
 [Text(0, 0, '2021'), Text(1, 0, '2022'), Text(2, 0, '2023')])
```



In [ ]:
```
data = [4, 7, 12, 19]
label = ['AAA', 'BBB', 'CCC', 'DDD']
color = ['red', 'cyan', 'yellow', 'green']
exp = [0.05, 0.05, 0.05, 0.05]
plt.pie(data, colors=color, autopct='%.1f%%', labels=label, explode=exp)
#color 색상, autopct: 퍼센트소수점 1까지 배열순서에 맞는 배열  explode 각 배열 간격
```

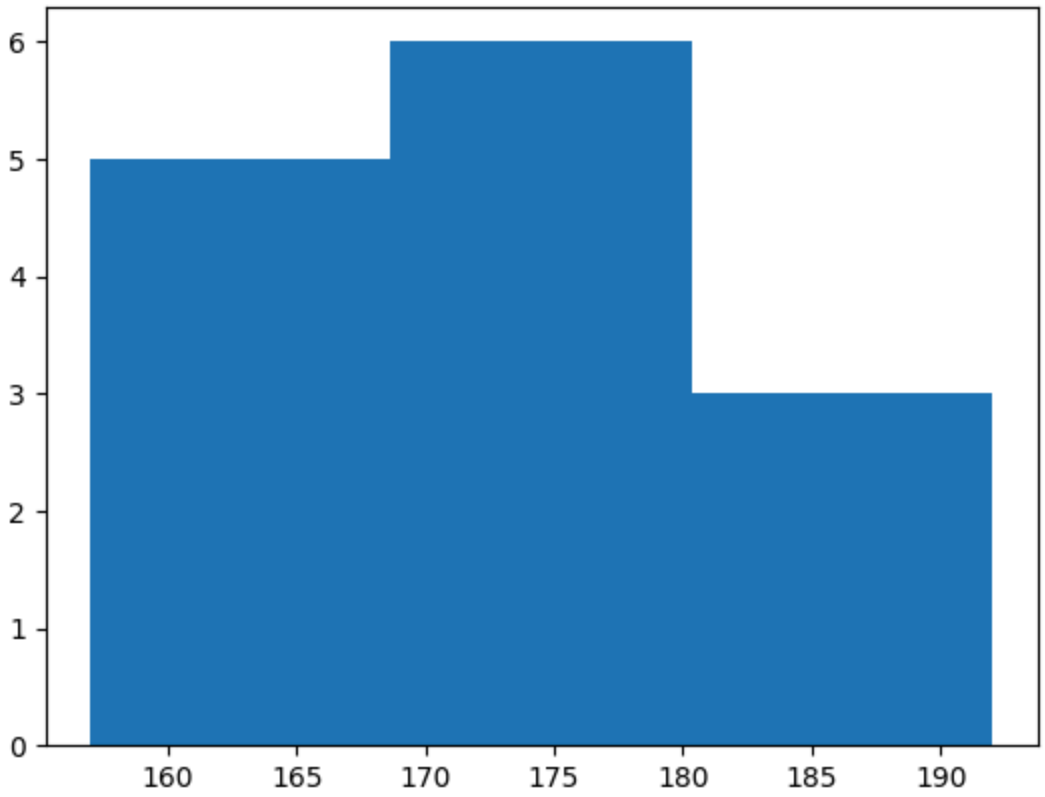
Out[ ]:
```
([<matplotlib.patches.Wedge at 0x12d79055888>,
  <matplotlib.patches.Wedge at 0x12d7905e5c8>,
  <matplotlib.patches.Wedge at 0x12d79067088>,
  <matplotlib.patches.Wedge at 0x12d7905ec88>],
 [Text(1.0989087247649836, 0.3389684566967815, 'AAA'),
  Text(0.49896627226861684, 1.0361142114353807, 'BBB'),
  Text(-0.9501746438958879, 0.6478179884020842, 'CCC'),
  Text(0.1713987556650378, -1.1371554276159777, 'DDD')],
 [Text(0.6211223226932515, 0.19159086682861565, '9.5%'),
  Text(0.28202441476052253, 0.5856297716808673, '16.7%'),
  Text(-0.5370552335063714, 0.36615799344465616, '28.6%'),
  Text(0.09687755754980397, -0.6427400243046829, '45.2%')])
```



In [ ]:
```
import matplotlib.pyplot as plt
import numpy as np
h = np.array([174, 157, 178, 178, 192, 163, 161, 163, 163, 178, 178, 179, 192, 191]
# 히스토그램(histogram)
plt.hist(h, bins=3) #구간을 3개로 나뉨
```

Out[ ]:
```
(array([5., 6., 3.]),
 array([157.        , 168.66666667, 180.33333333, 192.        ]),
 <BarContainer object of 3 artists>)
```

Loading [MathJax]/extensions/Safe.js



In [ ]:
```
import numpy as np
import pandas as pd
data = [1, 2, np.nan, 4]
# data = = {'1월':9500, '2월':6200, '3월':6050, '4월':7000} dictionary로 사용도 가능
se = pd.Series(data, index=['a', 'b', 'c', 'd']) # 1차원 list columns x
print(se, se[1], se[2])
#각 index 행 번호 data = 열
print(se.index, se.columns)
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_7044\3232786687.py in <module>
      3 data = [1, 2, np.nan, 4]
      4 # data = = {'1월':9500, '2월':6200, '3월':6050, '4월':7000} dictionary로 사용도 가능
----> 5 se = pd.Series(data, index=['a', 'b', 'c', 'd'], columns=[1,2,3,4]) # 1차원
      6 print(se, se[1], se[2])
      7 #각 index 행 번호 data = 열

TypeError: __init__() got an unexpected keyword argument 'columns'
```

In [ ]:
```
import numpy as np
import pandas as pd
month_se = pd.Series(['1월', '2월', '3월', '4월'])
income_se = pd.Series([9500, 6200, 6050, 7000])
expense_se = pd.Series([5400, 2350, 7800, 4800])
# 데이터프레임(df)
df = pd.DataFrame({'월':month_se, '수입':income_se, '지출':expense_se})
print(df)
print(df.index, df.columns)
```

```
   월    수입    지출
0  1월  9500  5400
1  2월  6200  2350
2  3월  6050  7800
3  4월  7000  4800
RangeIndex(start=0, stop=4, step=1) Index(['월', '수입', '지출'], dtype='object')
```

In [ ]:
```
m_ind = np.argmax(income_se) # 해당 리스트, 자료구조에서 가장 큰 값의 인덱스
print(month_se[m_ind])
print(income_se.max())
print(expense_se[np.argmax(expense_se)])
```

```
1월
9500
7800
```

In [ ]:
```
import numpy as np
import pandas as pd
file = 'vehicle_prod.csv'
df = pd.read_csv(file, index_col=0) # index_col: 첫번째열을 index(행 ex: 1 2 3 4 )
```

In [ ]:
```
import numpy as np
import pandas as pd
file = 'vehicle_prod.csv'
df = pd.read_csv(file, index_col=0) # 첫번째 열이 인덱스로 사용함.
print(df)
print(df['2007'].tolist()) #list형으로 만들어줌
print(df['2007'])
```

```
         2007   2008   2009   2010   2011
China    7.71   7.95  11.96  15.84  16.33
EU      19.02  17.71  15.00  16.70  17.48
US      10.47   8.45   5.58   7.60   8.40
Japan   10.87  10.83   7.55   9.09   7.88
Korea    4.04   3.78   3.45   4.20   4.62
Mexico   2.01   2.05   1.50   2.25   2.54
[7.71, 19.02, 10.47, 10.87, 4.04, 2.01]
China      7.71
EU        19.02
US        10.47
Japan     10.87
Korea      4.04
Mexico     2.01
Name: 2007, dtype: float64
```

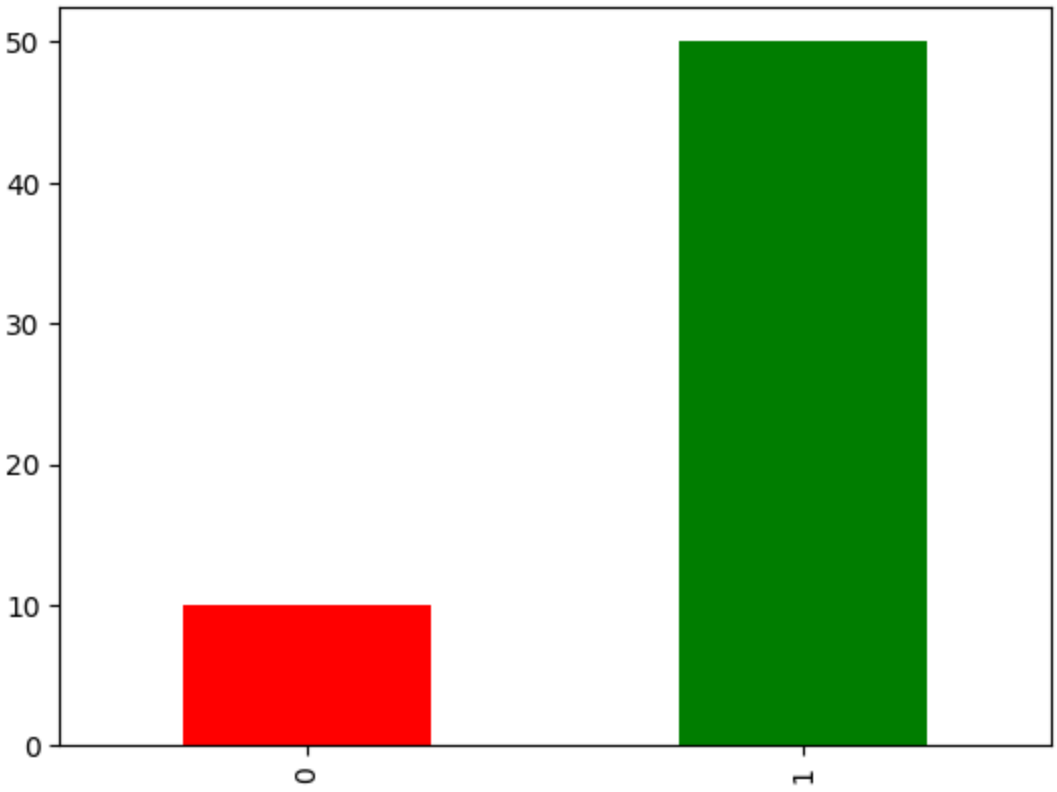
In [ ]:
```
import numpy as np
import pandas as pd
file = 'vehicle_prod.csv'
df = pd.read_csv(file, index_col=0) # 첫번째 열이 인덱스로 사용함.
df['total'] = df.sum(axis=1)  # ex:1행에있는 값들을 모두 합침
df['mean'] = df.mean(axis=1)
# 삭제
df.drop('2007', inplace=True, axis=1)
df
```

Out[ ]:

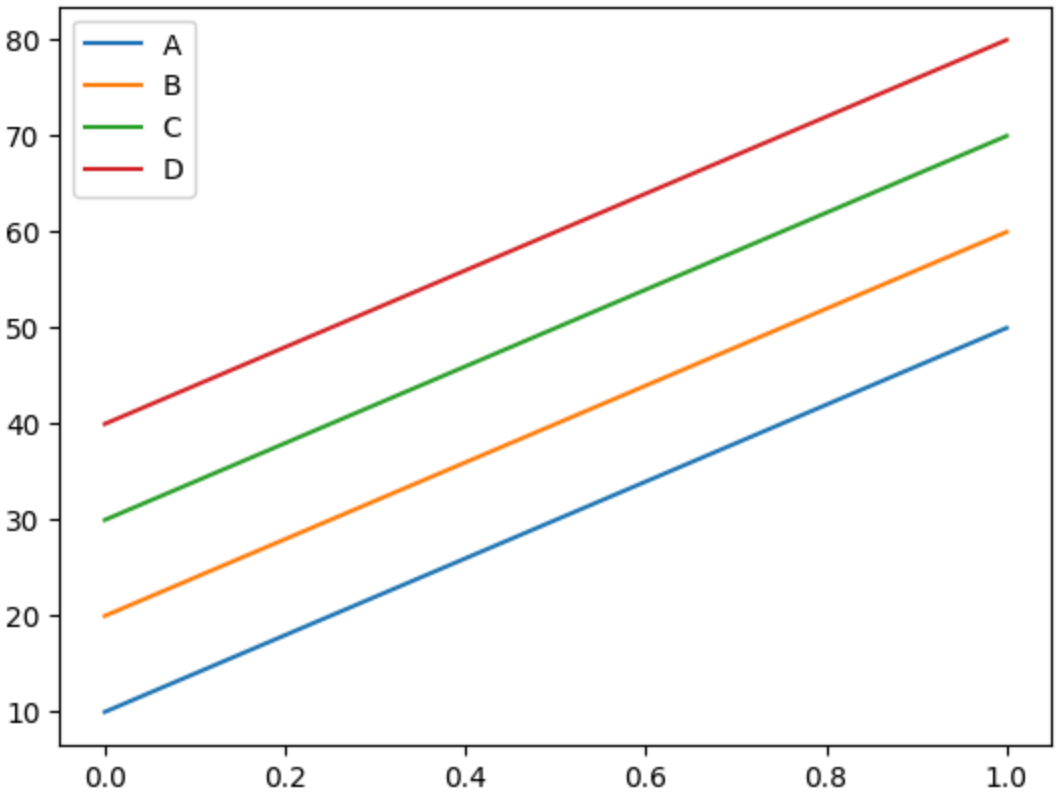
	2008	2009	2010	2011	total	mean
China	7.95	11.96	15.84	16.33	59.79	19.930000
EU	17.71	15.00	16.70	17.48	85.91	28.636667
US	8.45	5.58	7.60	8.40	40.50	13.500000
Japan	10.83	7.55	9.09	7.88	46.22	15.406667
Korea	3.78	3.45	4.20	4.62	20.09	6.696667
Mexico	2.05	1.50	2.25	2.54	10.35	3.450000

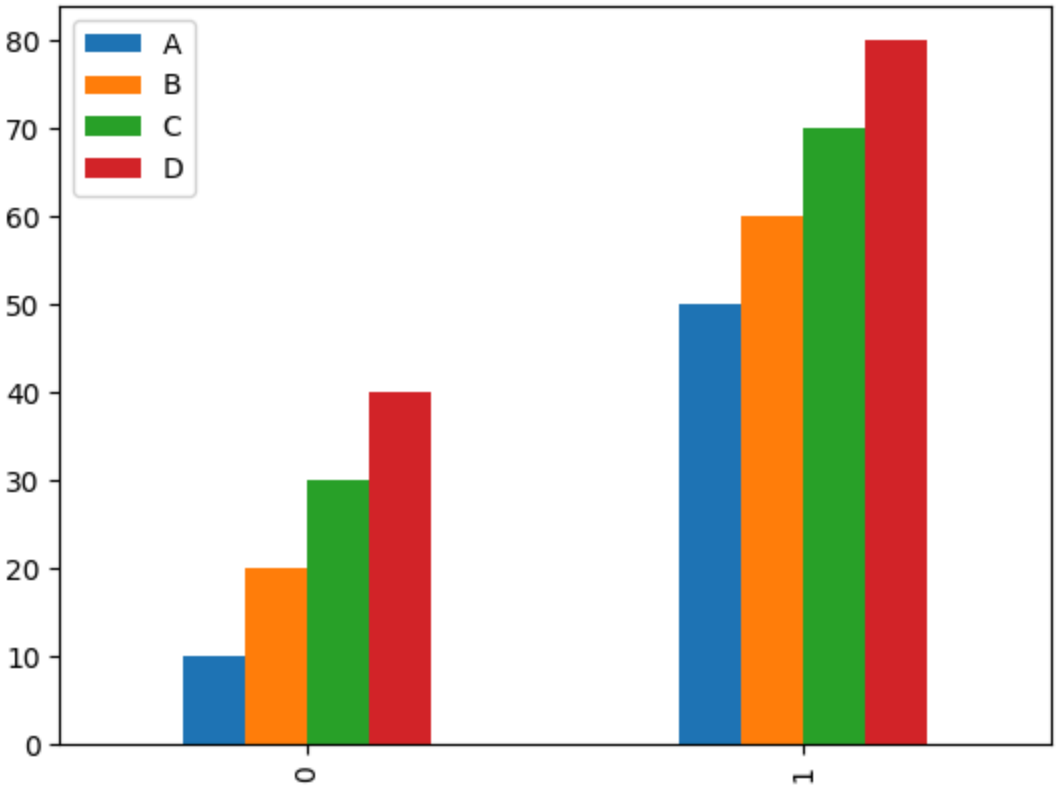
In [ ]:
```
# inplace로 데이터프레임 갱신하기
import numpy as np
import pandas as pd
df = pd.DataFrame(data=[[10, 20, 30, 40], [50, 60, 70, 80]],
columns=['A', 'B', 'C', 'D'])
new_df = df.drop('B', axis=1, inplace=False) # inplace시 초기생성 데이터 프레임도 변경
df

df['A'].plot(kind='bar', color=('r', 'g', 'b', 'c', 'm', 'y'))
#df['A'].plot(kind='pie') #벤다이어그램
df.plot.line() #선
df.plot.bar()
```

Out[ ]: <AxesSubplot:>



Loading [MathJax]/extensions/Safe.js





In [ ]:
```
print(df)
print(df.head(2)) # 0부터 2개
print(df.tail(2)) # 마지막부터 2개
```

```
   A   B   C   D
0  10  20  30  40
1  50  60  70  80
   A   B   C   D
0  10  20  30  40
1  50  60  70  80
   A   B   C   D
0  10  20  30  40
1  50  60  70  80
```

In [ ]:
```python
# 인덱싱과 슬라이싱
import numpy as np
import pandas as pd
file = 'vehicle_prod.csv'
df = pd.read_csv(file, index_col=0)
print(df)
# loc[] : 인덱스에서 특정 레이블이 있는 행을 가져옴
# iloc[] : 인덱스에서 정수형 인덱스값을 사용하여 특정 행을 가져옴
print(df.loc[['Korea', 'US']]) #특정행 추출 주의사항:: []꼭쓰기!
print(df.iloc[4])
print()
print(df.head(3)['2008']) # 인덱서(indexer)
df.iloc[[2, 3, 4]]
print(df[1])
```

```
         2007   2008   2009   2010   2011
China    7.71   7.95  11.96  15.84  16.33
EU      19.02  17.71  15.00  16.70  17.48
US      10.47   8.45   5.58   7.60   8.40
Japan   10.87  10.83   7.55   9.09   7.88
Korea    4.04   3.78   3.45   4.20   4.62
Mexico   2.01   2.05   1.50   2.25   2.54
        2007  2008  2009  2010  2011
Korea   4.04  3.78  3.45   4.2  4.62
US     10.47  8.45  5.58   7.6  8.40
2007    4.04
2008    3.78
2009    3.45
2010    4.20
2011    4.62
Name: Korea, dtype: float64

China     7.95
EU       17.71
US        8.45
Name: 2008, dtype: float64
```

```
---------------------------------------------------------------------------
KeyError                                  Traceback (most recent call last)
c:\Users\user\anaconda3\envs\tensorflow_env\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   3360             try:
-> 3361                 return self._engine.get_loc(casted_key)
   3362             except KeyError as err:

c:\Users\user\anaconda3\envs\tensorflow_env\lib\site-packages\pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

c:\Users\user\anaconda3\envs\tensorflow_env\lib\site-packages\pandas\_libs\index.pyx in pandas._libs.index.IndexEngine.get_loc()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

pandas\_libs\hashtable_class_helper.pxi in pandas._libs.hashtable.PyObjectHashTable.get_item()

KeyError: 1

The above exception was the direct cause of the following exception:

KeyError                                  Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_2452\2687039450.py in <module>
     12 print(df.head(3)['2008']) # 인덱서(indexer)
     13 df.iloc[[2, 3, 4]]
---> 14 print(df[1])

c:\Users\user\anaconda3\envs\tensorflow_env\lib\site-packages\pandas\core\frame.py in __getitem__(self, key)
   3456             if self.columns.nlevels > 1:
   3457                 return self._getitem_multilevel(key)
-> 3458             indexer = self.columns.get_loc(key)
   3459             if is_integer(indexer):
   3460                 indexer = [indexer]

c:\Users\user\anaconda3\envs\tensorflow_env\lib\site-packages\pandas\core\indexes\base.py in get_loc(self, key, method, tolerance)
   3361                 return self._engine.get_loc(casted_key)
   3362             except KeyError as err:
-> 3363                 raise KeyError(key) from err
   3364
   3365         if is_scalar(key) and isna(key) and not self.hasnans:

KeyError: 1
```

In [ ]:
```
print(df.iloc[2:4])
print(df.iloc[[2,3]])
```

```
        2007   2008  2009  2010  2011
US     10.47   8.45  5.58  7.60  8.40
Japan  10.87  10.83  7.55  9.09  7.88
        2007   2008  2009  2010  2011
US     10.47   8.45  5.58  7.60  8.40
Japan  10.87  10.83  7.55  9.09  7.88
```

In [ ]:

```
---------------------------------------------------------------------------
NameError                                 Traceback (most recent call last)
~\AppData\Local\Temp\ipykernel_2452\1274877492.py in <module>
----> 1 print(df.describe())
      2 print()
      3 print(df.std()) #표준 편차
      4 print()
      5 print(df.max()) #최대

NameError: name 'df' is not defined
```