[Random 함수]

- np.random.seed # seed를 통한 난수 생성
- np.random.randint # 균일분포의 정수 난수 1개 생성
- np.random.rand # 0부터 1사이의 균일분포에서 난수 매트릭스 array 생성
- np.random.randn # 가우시안 표준 정규 분포에서 난수 매트릭스 array 생성
- np.random.shuffle # 기존의 데이터의 순서 바꾸기
- np.random.choice # 기존의 데이터에서 sampling

In [6]:

```
# 시드가 같으면 생성되는 난수도 같다. 시드를 설정하지 않으면
import numpy as np

np.random.seed(42) # 42는 시드 값
```

In [14]:

```
np.random.seed(42)
np.random.rand()
```

```
0.3745401188473625
```

In [23]:

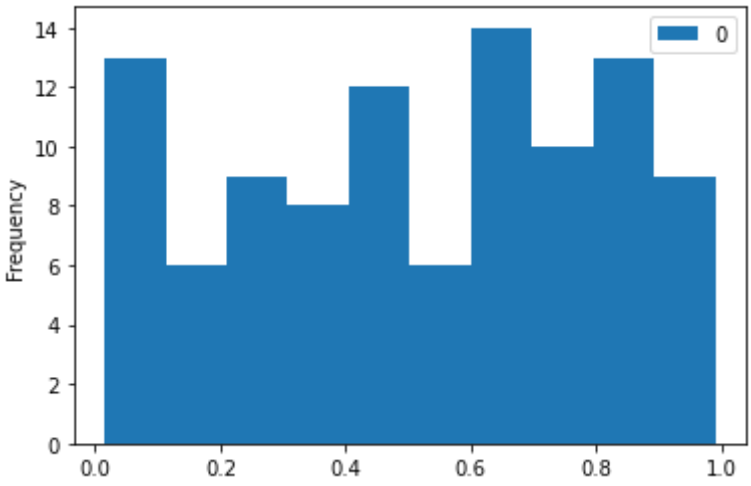
```
# rand() : 0 ~ 1 사이의 균일 분포
np.random.rand(3,2)
```

```
array([[0.70807258, 0.02058449],
       [0.96990985, 0.83244264],
       [0.21233911, 0.18182497]])
```

In [65]:

```
import pandas as pd
a = np.random.rand(100)
df = pd.DataFrame(a)
df.plot(kind='hist',bins=10)
```

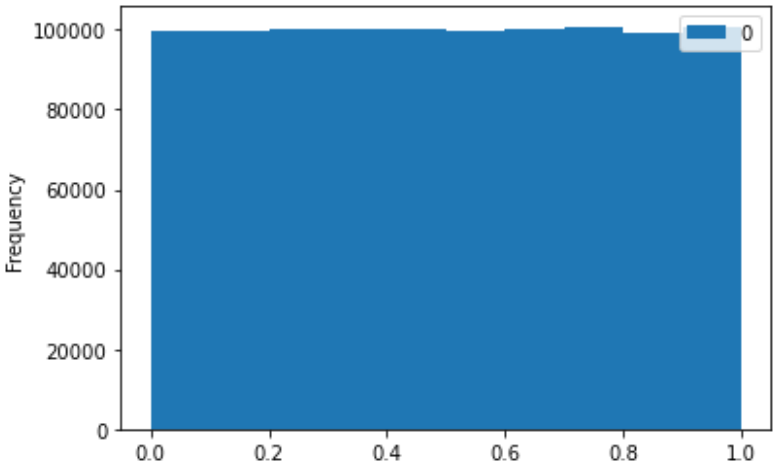
```
<matplotlib.axes._subplots.AxesSubplot at 0x272332
67700>
```



In [66]:

```
a = np.random.rand(1000000)
df = pd.DataFrame(a)
df.plot(kind='hist',bins=10)
```

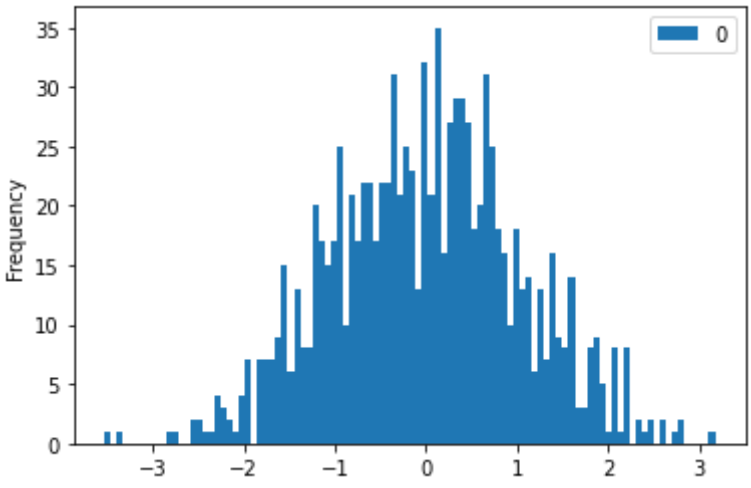
```
<matplotlib.axes._subplots.AxesSubplot at 0x272332
e85e0>
```



In [88]:

```
# randn은 표준정규분포에서 난수를 생성
data = np.random.randn(1000)
df = pd.DataFrame(data)
df.plot(kind='hist',bins=100)
```

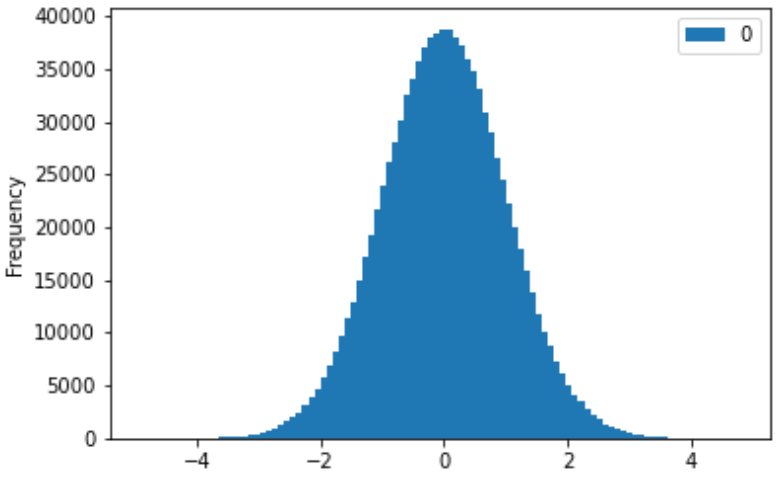
```
<matplotlib.axes._subplots.AxesSubplot at 0x272337
8a190>
```



In [86]:

```
data = np.random.randn(1000000)
df = pd.DataFrame(data)
df.plot(kind='hist',bins=100)
```

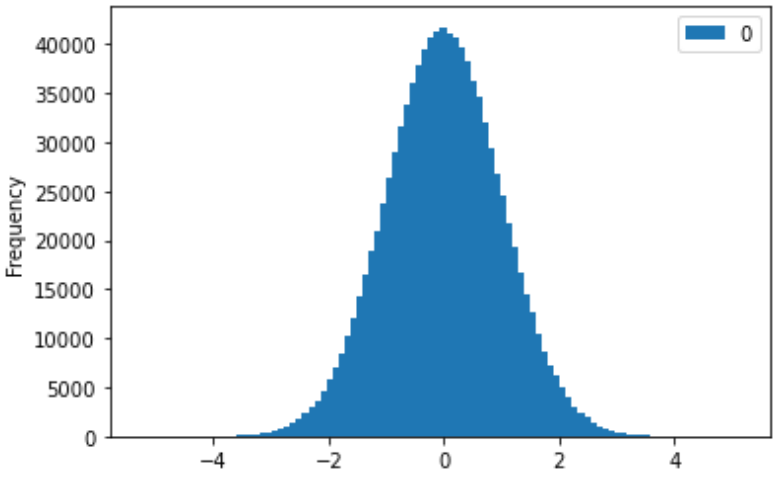
```
<matplotlib.axes._subplots.AxesSubplot at 0x27232a
68f70>
```



In [111]:

```
# 정규분포 : 평균 5, 표준편차 10
data = np.random.normal(0,1,1000000) # 평균, 표준편차, 난수의
df = pd.DataFrame(data)
df.plot(kind='hist',bins=100)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x272338
ba760>
```



In [83]:

```
# randint
print(np.random.randint(6),'₩n') # 0 ~ 5까지 정수인 난수 1개
print(np.random.randint(1,20),'₩n') # 1 ~ 19까지 정수인 난수
# 0 ~ 9까지 정수인 난수 10개
# 10 ~ 19까지 정수인 난수 10개
# 10 ~ 19까지 정수인 난수로 3행 5열 배열 생성
```

```
2

17
```

In [91]:

```
#randint
print(np.random.randint(6),'₩n') #0~5까지 정수인 난수 1개
print(np.random.randint(1,20),'₩n') #1~19까지 정수인 난수 1개
print(np.random.randint(0,10,10),'₩n') #0~9까지 정수인 난수
print(np.random.randint(10,20,10),'₩n')#10~19까지 정수인 난수
print(np.random.randint(10, 20, size = (3,5)))#10~19까지 정수
```

```
2


18


[4 8 9 2 1 8 0 3 1 3]


[10 19 18 10 14 14 12 10 11 19]


[[19 13 18 13 14]
 [10 16 17 16 17]
 [13 18 16 15 12]]
```

In [92]:

```
# choice
np.random.choice([1,2,3,4,5],10)
```

```
array([3, 3, 3, 4, 5, 1, 2, 2, 5, 2])
```

In [93]:

```
fruits = ['apple', 'banana', 'cherries', 'durian', 'grapes',
np.random.choice(fruits, 3)
```

```
array(['durian', 'lemon', 'cherries'], dtype='<U
8')
```

In [95]:

```
# 숫자별 갯수를 index, count 두개의 배열로 출력
a = np.array([11,11,3,3,2,2,2,7])
index, count = np.unique(a,return_counts=True)
print(index)
print(count)
```

```
[ 2  3  7 11]
[3 2 1 2]
```

In [96]:

```
# uniform()은 최솟값과 최댓값을 인자로 받아서 해당 범위에서
np.random.uniform(low=0.0,high=1.0,size=5)
```

```
array([0.81902179, 0.15152388, 0.25142762, 0.93426
263, 0.55890057])
```

In [107]:

```
print(np.random.rand()) # 0 ~ 1 사이의 균일분포
print(np.random.uniform(0,1),'₩n') # 0 ~ 1 사이의 균일분포
print(np.random.uniform(0,5),'₩n') # 0 ~ 5 사이의 균일분포
print(np.random.uniform(0,5,5),'₩n')
print(np.random.uniform(0,5,(2,5,5)))
```

```
0.7635829537205072
0.21290175900146358

3.4952411310484583

[0.38369837 0.85899664 4.44034055 1.03225969 0.456
58739]

[[[3.81139036 4.68748184 2.83523573 1.78844077 0.3
6598772]
  [3.98612584 3.34550596 0.99861639 0.48030023 2.7
8069091]
  [4.67528536 1.92397173 0.74501017 4.68352333 4.6
9480853]
  [3.43142555 2.67519992 3.43271903 0.49313706 4.0
3313763]
  [3.14220412 0.15374708 4.49051836 0.85329503 0.0
6259283]]

 [[0.41611336 4.8691921  3.34512711 3.39527601 2.2
0380727]
  [1.98613235 4.88260674 1.3680756  3.96851879 4.5
1224441]
  [0.66257076 2.80495397 2.03426317 3.88800685 3.7
0172965]
  [3.57739387 0.48379229 3.10258329 2.7183792  1.9
3784989]
  [2.21142145 3.64470136 1.81832298 0.4865411  4.0
5307903]]]
```

In [130]:

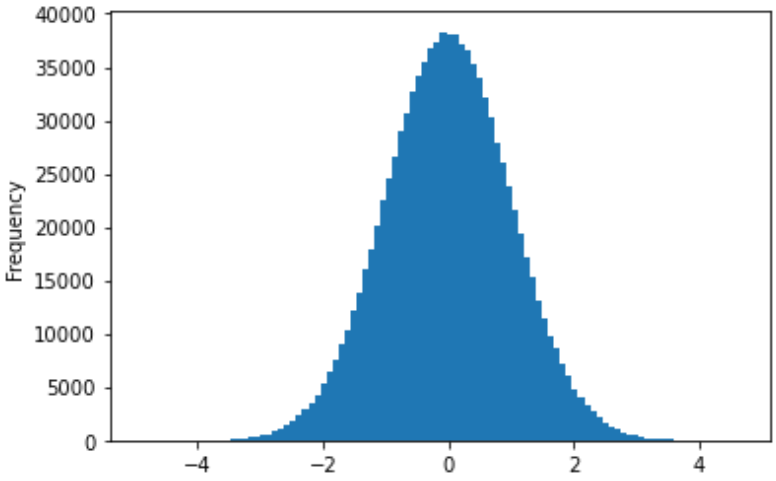
```
a = np.random.randn(1000000)
sr = pd.Series(a)
sr.describe()
```

```
count    1000000.000000
mean          -0.000708
std            0.999455
min           -4.913586
25%           -0.676239
50%           -0.000161
75%            0.673328
max            4.656776
dtype: float64
```

In [131]:

```
sr.plot(kind='hist',bins=100)
```

```
<matplotlib.axes._subplots.AxesSubplot at 0x27234e
25070>
```



In [117]:

```
a=np.random.randn(10000).reshape(100,100)     #1~100 사이 10x
a=a*100
df=pd.DataFrame(a)
df.head()
```

	0	1	2	3	4
0	97.546139	-37.594424	-6.622902	5.434741	1
1	-22.324473	-139.420125	18.288413	-1.140592	-
2	37.100686	80.204234	-45.266001	-91.423518	-
3	-63.474718	83.009561	27.270947	-185.257905	-
4	-112.582050	39.049712	-53.349742	178.771500	3

5 rows × 100 columns

In [116]:

```
# Q. 표준 정규분포를 따르는 모집단에서 난수 샘플 10000개를 추
# 데이터프레임을 생성하세요.
a= np.random.normal(0,1,10000).reshape(100,100)
a = a*100
df=pd.DataFrame(a)
df.head()
```

	0	1	2	3	4
0	-157.837277	45.431063	-59.011623	191.758286	-8
1	33.178490	-112.048156	152.563521	48.912897	18
2	-100.096455	-35.720518	-51.092207	50.127501	2
3	-89.128542	-132.078834	117.066135	27.268033	26
4	49.526606	-137.344030	53.006288	38.741647	-8

5 rows × 100 columns

In [133]:

```
# Q. 0 ~ 1 사이의 균일분포로 구성된 4행 8열 배열을 생성하세요
np.random.rand(4,8)
```

```
array([[0.65273284, 0.25907747, 0.41384799, 0.6757
6529, 0.39413556,
        0.96197282, 0.53219626, 0.93650379],
       [0.8231296 , 0.50026173, 0.37566533, 0.6643
2579, 0.69565547,
        0.05283045, 0.54841574, 0.85900734],
       [0.57538617, 0.57954693, 0.24904032, 0.8303
6691, 0.67844189,
        0.24338039, 0.76868594, 0.57801715],
       [0.18057704, 0.21453749, 0.24674968, 0.3511
9288, 0.97668987,
        0.54639745, 0.65672347, 0.58793852]])
```

In [134]:

```
data = np.random.uniform(0, 1, (4, 8))
data
```

```
array([[0.01020753, 0.86693399, 0.41630466, 0.3994
5665, 0.97475958,
        0.07341453, 0.1844995 , 0.04060677],
       [0.32715893, 0.47954531, 0.26583135, 0.3886
3099, 0.62937491,
        0.98663247, 0.03301224, 0.76012544],
       [0.54843051, 0.60736649, 0.31706064, 0.2092
225 , 0.70883711,
        0.21208445, 0.40739645, 0.38012498],
       [0.23977511, 0.40945316, 0.32457904, 0.5083
9723, 0.6896522 ,
        0.52302148, 0.52308607, 0.39032056]])
```

In [ ]:

```
# 평균이 3이고 표준편차가 3인 정규분포를 따르는 난수 100개를

np.random.normal(3,3,100).reshape(20,5)
```

[과제] a에서 중복을 허용하지 않고 값을 3개 출력하세요

a = [1, 2, 3, 4, 5]

In [141]:

```
# shuffle : 배열의 요소를 랜덤하게 섞는 함수
arr = np.array([1, 2, 3, 4, 5])
np.random.shuffle(arr)
arr
```

```
array([3, 5, 2, 4, 1])
```

In [143]:

```
arr = np.array([[1, 2, 3], [4, 5, 6], [7, 8, 9]])
print(arr,'₩n')
np.random.shuffle(arr)
arr
```

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

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

In [145]:

```
# 0 ~ 1사이의 균일분포로 구성된 4행 8열 배열을 생성
np.random.seed(0)
ar = np.random.rand(4,8)
# 배열을 df로 변환
columns = ['id','gender','age','region','product','price','q
df = pd.DataFrame(ar,columns=columns)
df
```

	id	gender	age	region	product	pr
0	0.548814	0.715189	0.602763	0.544883	0.423655	0.(
1	0.963663	0.383442	0.791725	0.528895	0.568045	0.9
2	0.020218	0.832620	0.778157	0.870012	0.978618	0.1
3	0.118274	0.639921	0.143353	0.944669	0.521848	0.4

In [ ]:

```
# Q. id, gender, age 컬럼에 대하여 의미있는 값으로 변환하여
```

In [146]:

```
df.id = np.array([1,2,3,4])
df.gender = np.array([0,1,0,1])
df.age = np.array([20,30,40,50])
df[['id','gender','age']]
```

	id	gender	age
0	1	0	20
1	2	1	30
2	3	0	40
3	4	1	50

Q. 아래 가이드에 따라서 고객 구매데이터를 생성하고 데이터 프레임으로 변환하여 출력하세요.(관측치 1000개)

- id : 1번 ~ 1000번 일련번호
- gender : 0,1 정수 난수 생성
- age : 10 ~ 80사이 정수 난수를 생성
- region : 1 ~ 10 사이 정수 난수 생성
- prod_22 : 제품코드 11111 ~ 99999
- prod_23 : 제품코드 11111 ~ 99999
- price_22 : 1000 ~ 50000 사이 실수 난수 생성
- price_23 : 1000 ~ 50000 사이 실수 난수 생성
- qty_22 : 1 ~ 100 사이 정수 난수 생성하여('19년 구매 수량)
- qty_23 : 1 ~ 100 사이 정수 난수 생성하여('20년 구매 수량)
- time_22 : 01 ~ 24 사이 정수 난수 생성('19년에 가장 자주 구매한 시간대)
- time_23 : 01 ~ 24 사이 정수 난수 생성('20년에 가장 자주 구매한 시간대)
- amount_22: price_22 * qty_22
- amount_23: price_23 * qty_23
- sales : 22년 대비 23년 구매금액이 증가면 1, 감소면 0