Python 기본

1. 쥬피터 노트북 사용법

- 노트북은 셀로 구성되어 있다.
  - 셀은 In[]과 Out[]으로 구성되어 있다. In[]에서는 입력 Out[]에는 출력
- 각 셀은 code 또는 markdown(주석)으로 지정할 수 있다
- 각 셀 단위로 수행을 할 수 있다.
- 셀을 실행 (셀에 커서를 위치함)
  - 코드 셀은 연산이 실행되고, 마크다운셀은 표시형태로 바뀐다.
  - 메뉴: cell -> run cell
  - 키보드 단축기: Ctrl+Enter
  - Run cell icon 클릭
- 셀에 변수나 숫자로 끝나면 해당 값의 출력에 표시된다.
  - 한 셀에 여러개의 라인이 있고 중간에 여러 변수가 있더라도 마지막 변수만 출력
  - 한 셀에서 여러 개의 출력을 원하면 print()함수를 사용한다

2. 파이썬 코딩 스타일

- 변수이름은 영문자, 숫자, 밑줄(_)로 구성. 숫자는 처음에 쓸 수 없다
  - 변수이름이 긴 경우는 중간에 밑줄 또는 대문자로 표시한다.
    - student_name, studentName
  - 대소문자는 구분됨
  - 한글도 사용가능
- 주석(comment) 사용은 #를 이용한다.
  - #는 아무 위치에서나 사용
  - # 오른쪽은 모두 주석으로 처리됨
  - 다중 주석줄은 3개 따옴표를 사용함
- 블록은 들여쓰기를 사용한다
  - 들여쓰기가 틀린 경우는 에러가 발생
  - 한개의 들여쓰기는 4개의 space를 사용한다.
  - 탭 사용은 권장되지 않음 (에디터에 따라 이동 정도가 달라짐)
  - 단 쥬피터 노트북에서는 탭이 빈칸 4개와 동일함
- 한 줄은 79칸 이하로 한다.
  - 가독성을 위한 것으로 79칸 이상이어도 작동에는 상관 없음

3. 기본 입출력 함수

Python에서는 기본 입력함수는 input(), 기본 출력함수는 print()을 사용한다.

- input()함수는 사용자에게 화면으로 메시지를 출력할 수 있는데 이것을 프롬프트(prompt)라고 한다.
- Python의 input() 함수는 모든 입력을 문자열로 받아들인다. 따라서 입력받은 문자열을 정수, 실수로 사용하려는 형변환을 하여야 한다.

출력함수 print()

In [1]:
```
name = '이세종'
age = 35
print(name)
print(age)
print("당신의 이름은",name,"그리고 나이는",age)
```

```
이세종
35
당신의 이름은 이세종 그리고 나이는 35
```

입력함수 input()

In [2]:
```
print("Enter your name: " )
name = input()   # John
print("Your name is",name)
```

```
Enter your name:
John
Your name is John
```

In [3]:
```
# input() 내부에 prompt 사용
name = input("Enter your name: ")   # Alice
print("Your name is",name)
```

```
Enter your name: Alice
Your name is Alice
```

In [4]:
```
# 입력받은 문자열을 정수로 형변환
age = int(input("Enter your age: "))    # 30
print("Your age next year is",age+1)
```

```
Enter your age: 30
Your age next year is 31
```

In [5]:
```
# 입력받은 문자열을 실수로 형변환
gpa = float(input("Enter your GPA: "))    # 3.4
print("Your GPA is",gpa)
```

```
Enter your GPA: 3.4
Your GPA is 3.4
```

4. 파이썬 모듈 및 패키지

- 패키지는 설치가 필요하다 (대부분 pip install을 사용)
- 설치된 패키지와 모듈을 사용하기 위해서는 해당 패키지와 모듈은 import해야 함
  - 예) math 모듈 사용 시 'import math'를 실행해야 함
- import된 패키지 또는 모듈에 있는 함수나 클래스를 사용하고자 할 때는 모듈또는 패키지 이름을 앞에 붙여 사용한다.
  - 예) sqrt()함수 사용시 math.sqrt()로 사용

In [6]:
```
import math
math.sqrt(2)
```

Out[6]:
```
1.4142135623730951
```

- from import를 시용하면 앞에 패키지 이름을 생략할 수 있다.

In [7]:
```
from math import sqrt
sqrt(2)
```

Out[7]: 1.4142135623730951

- from import에서 패키지에서 전체 함수나 클래스를 불러오려면 *를 사용

In [8]:
```
from math import *
sin(pi/2)   # math.sin(math.pi/2)
```

Out[8]: 1.0

5. 간단한 연산

In [9]:
```
2+3
```

Out[9]: 5

In [10]:
```
2**3   #  ** is used to raise power
```

Out[10]: 8

In [11]:
```
pow(2,3)    #  pow() is also avaiable
```

Out[11]: 8.0

In [12]:
```
1/2    # / exxecuted float division
```

Out[12]: 0.5

In [13]:
```
17 // 3  # floor division (integer division)
```

Out[13]: 5

In [14]:
```
17 % 3   # remainder
```

Out[14]: 2

In [15]:
```
sqrt(3)
```

Out[15]: 1.7320508075688772

In [16]:
```
exp(1)    # exponential function exp()
```

Out[16]: 2.718281828459045

In [17]:
```
log(exp(1))  # log(): natural log
```

Out[17]: 1.0

In [18]:
```
log10(10)   # log10() common log
```

Out[18]: 1.0

In [19]:
```
abs(-1)   # absolute
```

Out[19]: 1

In [20]:
```
factorial(5)   # factorial
```

Out[20]: 120

In [21]:
```
comb(5, 2)    # combination in math module
```

Out[21]: 10

In [22]:
```
perm(5, 2)  # permutation in math module
```

Out[22]: 20

In [23]:
```
1+ 2j    #  complex number
```

Out[23]: (1+2j)

6. 변수 사용

- 변수의 종류는 미리 선언하지 않는다.
- 변수의 종류는 중간에 변할 수 있다.

In [24]:
```
x = 3
print(x+4)
x = 'Python'
print(x)
```

```
7
Python
```

In [25]:
```
width = 25.3
height = 12.4
area = width*height
print('area = ', area)
```

```
area =  313.72
```

순환가능한 변수 (iterables)

- list: []로 표시
- tuple: ()로 표시
- set: {}로 표시
- string: '', "", ''' '''로 표시

string

In [26]:
```
s1 = "I can't stop."
print(s1)
```

```
I can't stop.
```

In [27]:
```
s2 = '"Double quote" is included.'
print(s2)
```

```
"Double quote" is included.
```

In [28]:
```
s3 = '''Mix of '' and "".'''
print(s3)
```

```
Mix of '' and "".
```

In [29]:
```
s = 'Py'+'thon'   # + operation for string
print(s)
print(3*s)    # * operation for string
```

```
Python
PythonPythonPython
```

In [30]:
```
len(s)    # length of string
```

Out[30]: 6

In [31]:
```
s[0]    # index starts from 0
```

Out[31]: 'P'

In [32]:
```
s[-1]   #  negative index means backward count from the end
```

Out[32]: 'n'

In [33]:
```
s[2:5]   # slicing index 2~5 (5 is not included)
```

Out[33]: 'tho'

In [34]:
```
s[2:]   # index 2~last
```

Out[34]: 'thon'

list

- 복합 데이터 저장 가능
- 가장 유연하고 사용성이 높은 자료형

In [35]:
```
a = [1, 4, 9, 16]
b = ['a', 'b', 'c']
c = [a, b]   # Nested list
print(c)
print(type(c))
```

```
[[1, 4, 9, 16], ['a', 'b', 'c']]
<class 'list'>
```

In [36]:
```
c[0]
```

Out[36]: [1, 4, 9, 16]

In [37]:
```
# indexing of nested list
c[0][3]
```

Out[37]: 16

tuple

- 생성 및 삭제는 가능하지만 원소의 변경은 불가능
- ()는 생략 가능함

In [38]:
```
a = (1,2)
print(a)
print(type(a))
```

```
(1, 2)
<class 'tuple'>
```

In [39]:
```
# tuple unpack
a = (1,2)
x, y = a
print('x = ',x)
print('y = ',y)
```

```
x =  1
y =  2
```

In [40]:
```
# Chang valuess
x = 2
y = 3
x, y = y, x  # swap x, y values
print('x = ',x)
print('y = ',y)
```

```
x =  3
y =  2
```

7. Control flow

In [41]:
```
# input()은 모든 입력을 string으로 받는다.
x = int(input('정수를 입력하세요'))  # convert to int
if x > 0:
    print('양의 정수')
elif x == 0:
    print('영')
else:
    print('음의 정수')
```

```
정수를 입력하세요-5
음의 정수
```

In [42]:
```
for i in range(5):   # range(5): 0, 1, 2, 3, 4
    print(i)
```

```
0
1
2
3
4
```

In [43]:
```
a = 'Python'
for i in a:
    print(i)
```

```
P
y
t
h
o
n
```

In [44]:
```
#range(5, 8): 5, 6, 7 # 마지막 인덱스 8은 포함 안함
list(range(5, 8))
```

Out[44]:
```
[5, 6, 7]
```

In [45]:
```
#range(0, 10, 3): 0, 3, 6, 9
list(range(0, 10, 3))
```

Out[45]:
```
[0, 3, 6, 9]
```

In [46]:
```
#range(-10, -100, -40): -10, -40, -70
list(range(-10, -100, -40))
```

Out[46]: [-10, -50, -90]

In [47]:
```
list_a = [85, 95, 72, 83, 92]
for number in list_a:
    print(number, end = ' ')   # place space instead of new line
```

```
85 95 72 83 92
```

In [48]:
```
member = ['Tom', 'Alice', 'John', 'Mark']
name = 'Peter'
name in member
```

Out[48]: False

In [49]:
```
days = ['Sun', 'Mon', 'Tue', 'Wed', 'Thr', 'Fri', 'Sat']
for day in days:
    if day in ['Sat', 'Sun']:
        print(day,'is weekend.')
    else:
        print(day,'is weekday.')
```

```
Sun is weekend.
Mon is weekday.
Tue is weekday.
Wed is weekday.
Thr is weekday.
Fri is weekday.
Sat is weekend.
```

In [50]:
```
list_a = [85, 95, 72, 83, 92]
for i, score in enumerate(list_a):   # return index and value
    print(i, score)
```

```
0 85
1 95
2 72
3 83
4 92
```

In [51]:
```
# list method
```

In [52]:
```
fruits = ['apple', 'pear', 'banana', 'kiwi', 'melon', 'pear', 'apple']
fruits.count('apple')
```

Out[52]: 2

In [53]:
```
fruits.index('kiwi')
```

Out[53]: 3

In [54]:
```
fruits.index('pear',2)
```

Out[54]: 5

In [55]:
```
fruits.append('grape')
fruits
```

Out[55]: ['apple', 'pear', 'banana', 'kiwi', 'melon', 'pear', 'apple', 'grape']

In [56]:
```
fruits.sort()
fruits
```

Out[56]: ['apple', 'apple', 'banana', 'grape', 'kiwi', 'melon', 'pear', 'pear']

In [57]:
```
fruits.sort(reverse=True)
fruits
```

Out[57]: ['pear', 'pear', 'melon', 'kiwi', 'grape', 'banana', 'apple', 'apple']

In [58]:
```
# List comprehension (리스트 내포)
squares = [x**2 for x in range(5)]
squares
```

Out[58]: [0, 1, 4, 9, 16]

8. 함수

- 사용자 정의 함수

In [59]:
```
# Function
def squared(x):
    return x**2

squared(55)
```

Out[59]: 3025

- default argument

In [60]:
```
# Function
def power(x, n=2):
    return x**n

print(power(5))      # n = 2 is used, since n is not given
print(power(5,3))    # n = 3 is given. then use n = 3
```

```
25
125
```

- 인수가 미리 정해지지 않은 함수

입력 인수의 갯수가 정해져 있지 않고 여러 개가 있을 수 있는 경우
argument에 *를 붙여주면 입력된 값을 튜플로 만들어 준다

In [61]:
```
def add_many(*args):
    result = 0
    for i in args:
        result = result + i
    return result

print(add_many(1))
print(add_many(1,2))
print(add_many(1,2,3))
```

```
1
3
6
```

In [62]:
```
def add_mul(choice, *args):
    if choice == 'add':
        result = 0
        for i in args:
            result = result + 1
    elif choice == 'mul':
        result = 1
        for i in args:
            result = result*i
    return result
```

In [63]:
```
result = add_mul('add',1,2,3,4,5)
print(result)
```

```
5
```

In [64]:
```
result = add_mul('mul',1,2,3,4,5)
print(result)
```

```
120
```

- 여러 개의 값은 반환할 때는 튜플로 반환을 한다.

In [65]:
```
def add_and_mul(a, b):
    return a + b, a * b    # return 2 values as a tuple

add_and_mul(3,4)
```

Out[65]: (7, 12)

9. 출력 format

파이썬에서는 여러가지 포맷 형식이 있다. 여기서는 format문과 % 문의 예를 소개한다.

In [66]:
```
# %포맷
g = 9.8
t = 7.3
v = g*t
# 출력변수 2개 ()안에 사용
print("%0.1f 초 후 속도는 %0.2f m/s 입니다."%(t, v))
```

```
7.3 초 후 속도는 71.54 m/s 입니다.
```

In [67]:
```
# format statement
print ('{:0.1f}초 후 속도는 {:0.2f} m/s 입니다.'.format(t, v))
```

```
7.3초 후 속도는 71.54 m/s 입니다.
```

In [68]:
```
# format statement
age = 12

name = 'Alice'
print('{} is {} years old.'.format(name, age))
```

```
Alice is 12 years old.
```

In [69]:
```
print('{1} is {0} years old.'.format(age, name))
```

```
Alice is 12 years old.
```

In [70]:
```
print('{name} is {age} years old.'.format(age=age, name=name))
```

```
Alice is 12 years old.
```

In [71]:
```
age2 = 15
name2 = 'Elsa'
print('{name} is {age} years old.'.format(age=age2, name=name2))
```

```
Elsa is 15 years old.
```

In [72]:
```
b = 13
c = 3
print('{}/{} = {}.'.format(b, c, b/c))
```

```
13/3 = 4.333333333333333.
```

In [73]:
```
print('{}/{} = {:.2f}.'.format(b, c, b/c))
```

```
13/3 = 4.33.
```

In [74]:
```
print('{1}/{0} = {2:.4f}.'.format(c, b, b/c))
```

```
13/3 = 4.3333.
```

10. Numpy

참고: https://numpy.org/devdocs/user/quickstart.html

In [75]:
```
import numpy as np
a = np.arange(15).reshape(3, 5)
a
```

Out[75]:
```
array([[ 0,  1,  2,  3,  4],
       [ 5,  6,  7,  8,  9],
       [10, 11, 12, 13, 14]])
```

In [76]:
```
a.shape
```

Out[76]:
```
(3, 5)
```

In [77]:
```
a.ndim
```

Out[77]:
```
2
```

In [78]:
```
a.dtype.name
```

Out[78]:
```
'int32'
```

In [79]:
```
a.size
```

Out[79]:
```
15
```

In [80]:
```
type(a)
```

Out[80]:
```
numpy.ndarray
```

In [81]:
```
a = np.array([2, 3, 4])
a.dtype
```

Out[81]:
```
dtype('int32')
```

In [82]:
```
b = np.array([1.2, 3.5, 5.1])
b.dtype
```

Out[82]:
```
dtype('float64')
```

기본 연산

In [83]:
```
A = np.array([[1, 1],
              [0, 1]])
B = np.array([[2, 0],
              [3, 4]])
```

In [84]:
```
A * B    # elementwise product (not inner product)
```

Out[84]:
```
array([[2, 0],
       [0, 4]])
```

In [85]:
```
A @ B     # matrix product
```

Out[85]:
```
array([[5, 4],
       [3, 4]])
```

In [86]:
```
A.dot(B)  # another matrix product
```

Out[86]:
```
array([[5, 4],
       [3, 4]])
```

In [87]:
```
# linespace 함수
b = np.linspace(0, 5, 11)
b
```

Out[87]:
```
array([0. , 0.5, 1. , 1.5, 2. , 2.5, 3. , 3.5, 4. , 4.5, 5. ])
```

In [88]:
```
a = np.arange(12).reshape(3, 4)
a
```

Out[88]:
```
array([[ 0,  1,  2,  3],
       [ 4,  5,  6,  7],
       [ 8,  9, 10, 11]])
```

In [89]:
```
a.sum()
```

Out[89]:
```
66
```

In [90]:
```
a.min()
```

Out[90]:
```
0
```

In [91]:
```
a.max()
```

Out[91]:
```
11
```

In [92]:
```
a.sum(axis=0)   # sum of each column
```

Out[92]:
```
array([12, 15, 18, 21])
```

In [93]:
```
a.min(axis=1)     # min of each row
```

Out[93]:
```
array([0, 4, 8])
```

- indexing, slicing

In [94]:
```
a = np.arange(10)**2
a
```

Out[94]:
```
array([ 0,  1,  4,  9, 16, 25, 36, 49, 64, 81], dtype=int32)
```

In [95]:
```
a[2]    # 3rd element (index starts from 0)
```

Out[95]:
```
4
```

In [96]:
```
a[2:5]
```

Out[96]:
```
array([ 4,  9, 16], dtype=int32)
```

주의 (array data type)

array data type을 주의하지 않는 경우 틀린 계산이 될 수 있다.

아래는 정수형 array에 첫번째 요소 값을 1.2로 변경하려는 것을 보인다.

In [97]:
```
a[0] = 1.2    # replace value
a
```

Out[97]:
```
array([ 1,  1,  4,  9, 16, 25, 36, 49, 64, 81], dtype=int32)
```

- 첫번째 요소 값에 1.2를 대입했지만 정수로 형변환 되어 1로 저장되어 에러가 발생할 수 있음.
- 따라서 정수형 배열을 먼저 float64로 형변환을 한 후 값을 입력함

In [98]:
```
a = a.astype('float64')
a
```

Out[98]:
```
array([ 1.,  1.,  4.,  9., 16., 25., 36., 49., 64., 81.])
```

In [99]:
```
a[0] = 1.2
a
```

Out[99]:
```
array([ 1.2,  1. ,  4. ,  9. , 16. , 25. , 36. , 49. , 64. , 81. ])
```

- 요소가 정수형인 리스트는 바로 numpy array로 변환할 수 있다.

In [100]:
```
a_list = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
a_array = np.array(a_list)
a_array
```

Out[100]:
```
array([[1, 2, 3],
       [4, 5, 6],
       [7, 8, 9]])
```

In [101]:
```
a_array.dtype
```

Out[101]:
```
dtype('int32')
```

- numpy array는 요소 형과 관계없이 리스트로 항상 변환할 수 있다.

In [102]:
```
b_array = np.array([[1.2, 2, 3], [4, 5, 6], [7, 8, 9]])
b_array
```

Out[102]:
```
array([[1.2, 2. , 3. ],
       [4. , 5. , 6. ],
       [7. , 8. , 9. ]])
```

In [103]:
```
b_array.dtype
```

Out[103]:
```
dtype('float64')
```

In [104]:
```
b_list = b_array.tolist()
b_list
```

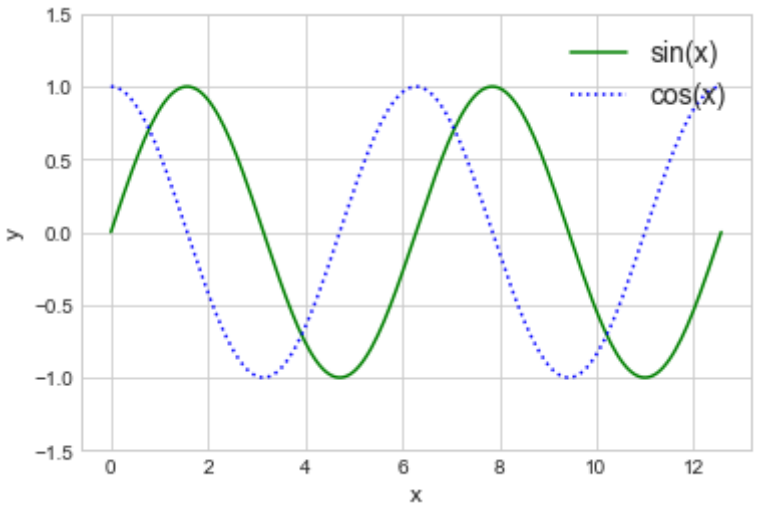
Out[104]:
```
[[1.2, 2.0, 3.0], [4.0, 5.0, 6.0], [7.0, 8.0, 9.0]]
```

11. Plot

- 참고: matplotlib tutorial: https://matplotlib.org/stable/tutorials/introductory/pyplot.html

Line plot

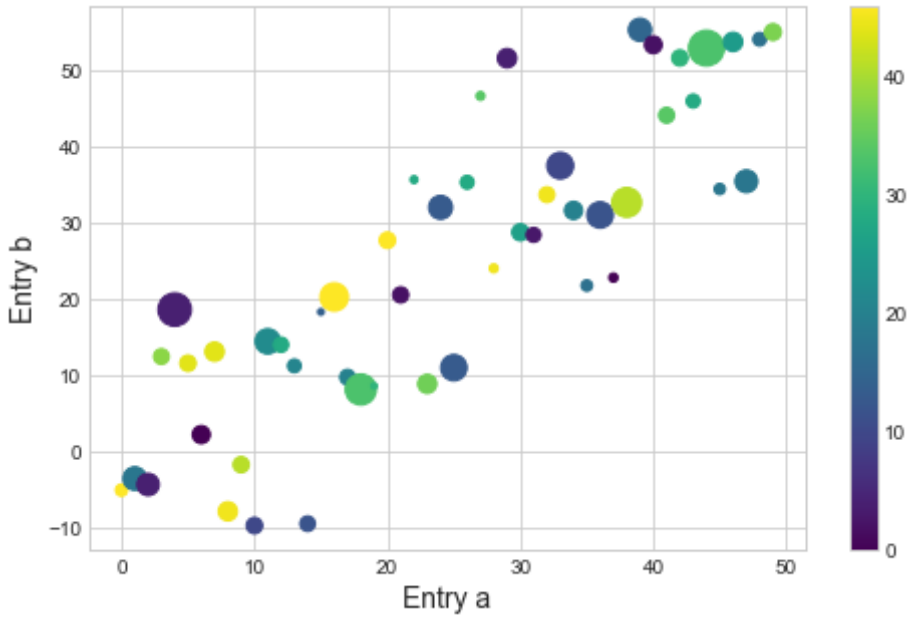
In [105]:
```
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
import numpy as np

fig = plt.figure()
x = np.linspace(0, 4*pi, 200)
plt.plot(x, np.sin(x), '-g', label='sin(x)')
plt.plot(x, np.cos(x), ':b', label='cos(x)')
plt.xlabel('x',fontsize=12)
plt.ylabel('y',fontsize=12)
plt.ylim([-1.5, 1.5])
#plt.axis('equal')

plt.legend(fontsize=14);
plt.show()
```



Scatter plot

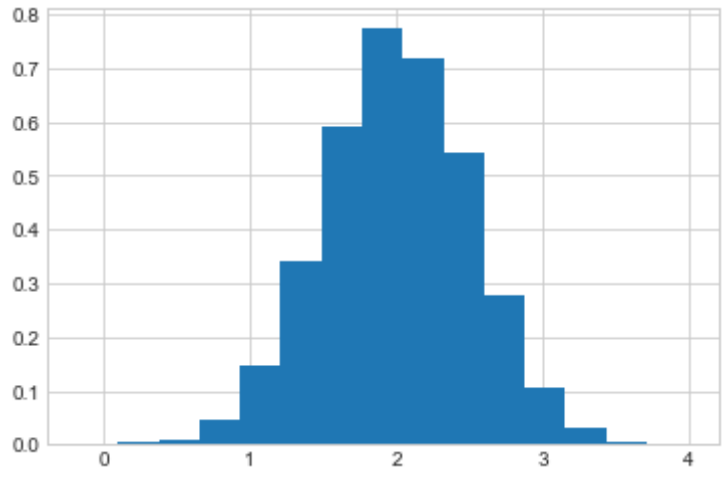
In [106]:
```
%matplotlib inline
import matplotlib.pyplot as plt
import numpy as np
plt.style.use('seaborn-whitegrid')

data = {'a': np.arange(50),
        'c': np.random.randint(0, 50, 50),
        'd': np.random.randn(50)}
data['b'] = data['a'] + 10 * np.random.randn(50)
data['d'] = np.abs(data['d']) * 100
```

```
fig = plt.figure(figsize=(8,5))
plt.scatter('a', 'b', c='c', s='d', data=data, cmap='viridis')
plt.xlabel('Entry a',fontsize=14)
plt.ylabel('Entry b', fontsize=14)
plt.colorbar();  # show color scale
plt.show()
```



Histogram

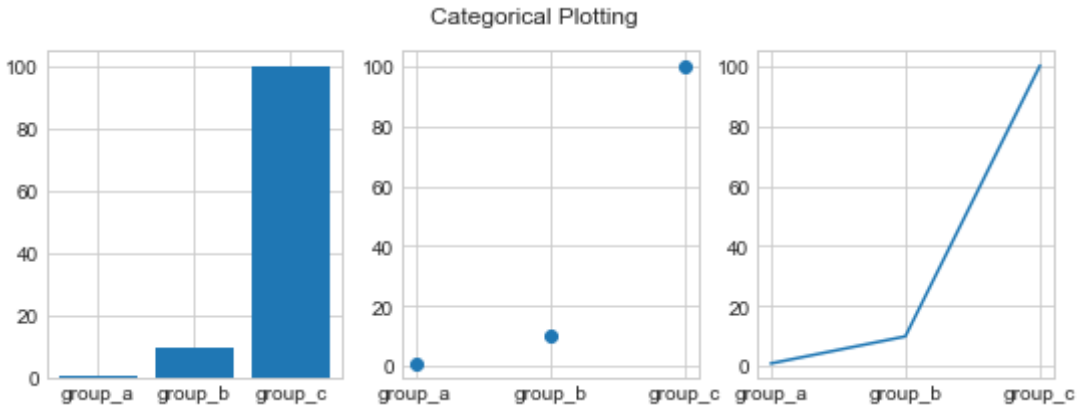
In [107]:
```
%matplotlib inline
import matplotlib.pyplot as plt
plt.style.use('seaborn-whitegrid')
mu, sigma = 2, 0.5
v = np.random.normal(mu, sigma, 10000)
plt.hist(v, bins=15, density=1)
plt.show()
```



Plotting with categorical variables

In [108]:
```
names = ['group_a', 'group_b', 'group_c']
values = [1, 10, 100]

plt.figure(figsize=(9, 3))

plt.subplot(131)
plt.bar(names, values)
plt.subplot(132)
```

```
plt.scatter(names, values)
plt.subplot(133)
plt.plot(names, values)
plt.suptitle('Categorical Plotting')
plt.show()
```



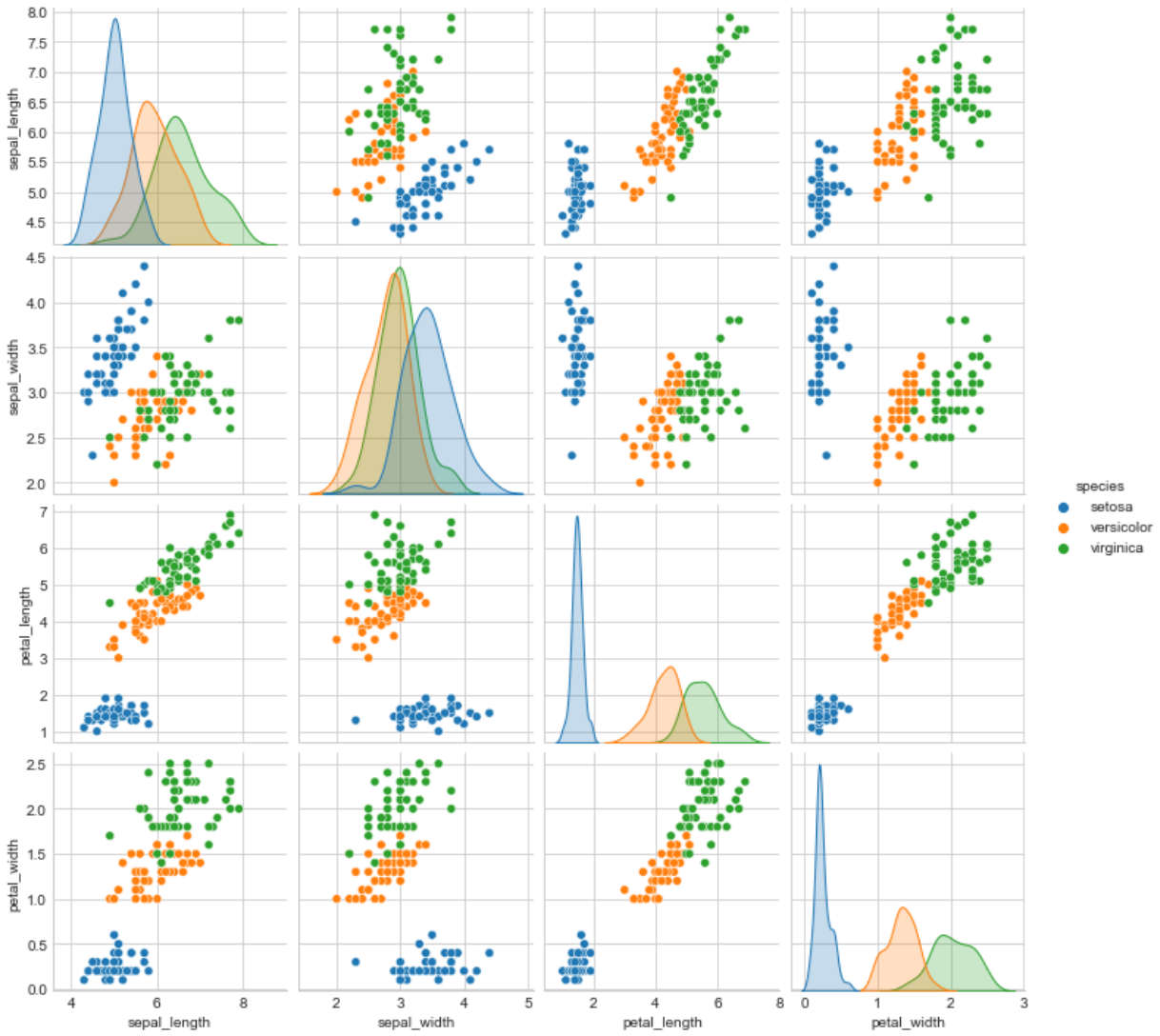
Pair plot

In [109]:
```
import seaborn as sns
iris = sns.load_dataset("iris")
iris.head()
```

Out[109]:

	sepal_length	sepal_width	petal_length	petal_width	species
0	5.1	3.5	1.4	0.2	setosa
1	4.9	3.0	1.4	0.2	setosa
2	4.7	3.2	1.3	0.2	setosa
3	4.6	3.1	1.5	0.2	setosa
4	5.0	3.6	1.4	0.2	setosa

In [110]:
```
sns.pairplot(iris, hue='species', height=2.5);
```


sepal_length
sepal_width
petal_length
petal_width
species
setosa
versicolor
virginica