

데이터 분석을 위한 파이썬 기초

[3차시]

목 차

- 01. 함수의 이해
- 02. 함수의 정의와 변수
- 03. 함수 활용과 랜덤 함수
- 04. 모듈의 이해
- 05. 다양한 모듈 활용
- 06. 재귀함수의 이해
- 07. 튜플의 이해
- 08. 딕셔너리의 이해
- 09. 리스트/튜플/딕셔너리의 활용
- 10. 시각화의 이해

01. 함수의 이해

[3차시]

학습목표

- 함수를 왜 사용하는지 이해하기
- 내장함수가 무엇인지 이해하기
- 사용자 정의 함수를 만드는 과정을 따라해 보기

함수사용 이유

- 코드가 길어질 때 모듈화하여 간결성 높인다
- 반복되는 구분을 모듈화 하여
 - 고치기 쉽고(easy to modify)
 - 운영과 관리를 용이하게 하며(flexible to maintain)
 - 프로그램 가독성을 높게 한다(readability)

함수

- 함수의 종류
 - 내장 함수(Built-in function)
 - 사용자 정의 함수(User defined function)
- 함수를 정의하려면
 - 함수 이름과 명령문들을 순서대로 쓴다
 - 함수 이름을 불러서 함수를 "호출" 한다 (function call)

사용해 본 내장함수

- print()
- input()
- int()
- float()
- range()
- len()
- deepcopy()

내장 함수(Built-in Function)

- 시스템에서 제공해 주는 함수들

- 사용한 내장 함수

```
>>> int('55')
```

```
>>> print( 'kmkim' )
```

```
>>> range(10)
```

- 어떤 함수들은 모듈을 import한 후 사용 가능

```
>>> import math
```

```
>>> math.sqrt(4) / 2.0 1.0
```


내장함수 활용 1

```
# type conversion function
>>> int('32')
32
>>> int(3.99999)
3
>>> float(-2)
-2.0
# Unicode -> character
>>> chr(65)
'A'
>>> chr(97)
'a'

# character -> unicode
>>> ord('a')
97
>>> ord("F")
70
```

내장함수 활용 2

```
>>> print(5)
5
>>> i = input("enter your age : ")
enter your age : 12
>>> print(i * 2)
1212

# sum()
>>> num_list=[1,2,3,4,5]
>>> sum(num_list)
15

#sorted()
>>> num_l=[5,6,2,9,1]
>>> sorted(num_l)
[1, 2, 5, 6, 9]
>>> num_l
[5, 6, 2, 9, 1]
```

내장함수 활용3

<http://docs.python.org/3.10/library/functions.html>

abs() all() any() ascii() b in() bool() bytearray() bytes() callable() chr() classmethod() compile() complex() delattr()	dict() dir() divmod() enumerate() eval() exec() filter() float() format() frozenset() getattr() globals() hasattr() hash()	help() hex() id() input() int() isinstance() i ssubclass() it er() len() list() locals() map() max() memoryview()	min() next() object() oct() open() ord() pow() print() property() range() repr() reversed() round() set()	setattr() slice() sorted() staticmethod() str() sum() super() tuple() type() vars() zip() __import__()
---	---	---	--	---

내장함수 활용 4

파이썬에는 친숙한 수학적 함수들을 제공하는 `math` module이 있다

- 모듈은 관련된 함수들의 모음이다
- 모듈을 사용하려면 `import` 사용해야 한다

```
# math module
>>> import math
>>> print(math)
<module 'math' (built-in)>

>>> degrees = 45
>>> radians = degrees / 360.0 * 2 * math.pi
>>> math.sin(radians)
0.707106781187
>>> math.sqrt(2) / 2.0
0.707106781187
```

내장함수, Argument and return value

```
>> int_num = int(23.5)
```

argument

```
>> int_num
```

23

return value

```
>> r = round(2.3456, 2)
```

2 arguments

Function call

- Argument는 함수를 호출했을 때 값을 받게 된다
- 리턴 값은 함수가 끝날 때 그 결과를 기억한다

내장함수, Sorting

- 정렬

- 특정한 순차(sequence) 혹은 다른 집합(set)들에 따라 항목(item)을 배열하는 과정을 말한다
- 순서화(Ordering)
 - 같은 종류, 클래스의 항목(items)들을 순서 있는 나열로 배열하는 것
- 카테고리화(Categorizing)
 - 비슷한 특징을 지닌 항목끼리 그룹화(grouping)하고 표시하는 것

내장함수, Sorting

<https://wiki.python.org/moin/HowTo/Sorting/>

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

```
>>> a.sort()
```

```
>>> a
```

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

```
>>> import operator
```

```
>>> data = [('red', 1), ('blue', 1), ('red', 2), ('blue', 2)]
```

```
>>> sorted(data, key=operator.itemgetter(1))
```

```
[('blue', 1), ('red', 1), ('blue', 2), ('red', 2)]
```

```
>>> data
```

```
[('red', 1), ('blue', 1), ('red', 2), ('blue', 2)]
```

sort by second item

사용자 정의 함수(User defined function)

- 사용자 정의 함수란?
 - 사용자가 필요하다고 판단하는 루틴을 구상하여 함수로 정의하고 사용하는 것
- 함수 정의하기(define function)
- 선언된 함수 사용하기(function call)

함수 정의하기

```
def plus(num1, num2) :  
    result = num1 + num2  
    return result
```

2개의 parameters
num1, num2

Statement

한 개의 값을 돌려주는 리턴 문

사용자 정의 함수 정의 과정

- 1단계. 함수의 기능을 정리하고, 이름을 정한다
- 2단계. 함수에서 입력과 결과가 무엇인지 정한다
- 3단계. 입력으로 사용하는 파라미터와 리턴값의 데이터형을 정한다
- 4단계. 원하는 함수의 기능을 Pseudo code로 쓴다
- 5단계. 함수를 만든다
- 6단계. 제대로 실행되는지 테스트 한다

사용자 정의함수 만들기 예제

- 3개 숫자를 입력 받아서 평균을 계산하여 돌려준다
 - 평균을 구하는 함수 이름은 `calculate_avg`
- Parameter는 3개, 데이터형 : int
return value 데이터형 : float
- Pseudo code 써보기
 - `sum = parameter1 + parameter2 + parameter3`
 - `return sum/3`

사용자 정의함수 만들기 예제

- 함수 만들기

```
def calculate_avg(n1, n2, n3):  
    sum = n1 + n2 + n3  
    return sum/3
```

- 제대로 실행되는지 확인해 본다

사용자 정의함수 만들기 예제

정의한 함수 사용해 보기

```
# Define a function
def calculate_avg(n1, n2, n3):
    sum = n1 + n2 + n3
    return sum/3

# Call the function
avg=calculate_avg(1,5,9)
print("result of function call = ", avg)

avg=calculate_avg(11, 9, 16)
print("result of function call = ", avg)
```

```
result of function call = 5.0
result of function call = 12.0
```

강의 요약

- 함수를 왜 사용하는지 이해하기
 - 반복되는 구문을 모듈화
 - 수정이 간편
 - 운영과 관리가 용이
 - 가독성이 높아짐
- 내장함수가 무엇인지 이해하기
 - 시스템에서 제공하는 함수들
- 사용자 정의 함수를 만드는 과정을 따라해 보기

퀴즈 1

- 함수의 특징을 설명한 것이 아닌 문장은?
 - ① 반복되는 구문을 모듈화 한다
 - ② 수정과 운영과 관리가 간편해진다
 - ③ 자동으로 처리하는 것이 쉬워진다
 - ④ 코드의 가독성이 높아진다

퀴즈 1-답안

- 함수의 특징을 설명한 것이 아닌 문장은?
 - ① 반복되는 구문을 모듈화 한다
 - ② 수정과 운영과 관리가 간편해진다
 - ③ 자동으로 처리하는 것이 쉬워진다
 - ④ 코드의 가독성이 높아진다

퀴즈 2

• 다음 함수 중 내장함수를 모두 선택하시오

- ① input()
- ② input_num()
- ③ range()
- ④ len()
- ⑤ calculate()

퀴즈 2-답안

- 다음 함수 중 내장함수를 모두 선택하시오

- ① `input()`
- ② `input_num()`
- ③ `range()`
- ④ `len()`
- ⑤ `calculate()`

02. 함수의 정의와 변수

[3차시]

학습목표

- 함수 정의 연습하기
- 파라미터와 리턴값 이해하기
- 원하는 함수를 기술하는 과정 익히기
- 지역 변수와 전역 변수의 차이점 알아보기

사용자 정의 함수

no parameter, no return value

```
def print_script() :  
    print("Blessed are the poor in spirit,")  
    print("for theirs is the kingdom of heaven.")  
    print("  ")
```

```
print("result of first function call, ")  
print("=" * 25)  
print_script()      # first function call
```

```
def repeat_script() :  
    print_script()  
    print_script()
```

```
print("result of second function call, ")  
print("=" * 25)  
repeat_script()     # second function call
```

```
➞ result of first function call,  
=====  
Blessed are the poor in spirit,  
for theirs is the kingdom of heaven.  
  
result of second function call,  
=====  
Blessed are the poor in spirit,  
for theirs is the kingdom of heaven.  
  
Blessed are the poor in spirit,  
for theirs is the kingdom of heaven.
```

사용자 정의 함수

parameters, no return value

```
def print_chr(times) :  
    for i in range(times) :  
        print("*")
```

```
print("result of first function call, ")  
print("=" * 25)  
print_chr(5)
```

```
def print_line(t) :  
    print("*" * t)
```

```
print("result of second function call, ")  
print("=" * 25)  
print_line(10)
```

```
→ result of first function call,  
=====
```

*
*
*
*
*

```
result of second function call,  
=====
```

사용자 정의 함수

parameters, return value

```
def plus(num1, num2) :  
    return num1 + num2
```

```
result_plus = plus(2, 12)  
print("result of plus(2, 12) ")  
print(result_plus)
```

```
def mul(num1, num2) :  
    return num1 * num2
```

```
result_mul = mul(2, 12)  
print("result of mul(2, 12) ")  
print(result_mul)
```

```
➡ result of plus(2, 12)  
14  
result of mul(2, 12)  
24
```

Parameters and Arguments 1

- 함수 내에서, 인수(argument)는 매개변수(parameter)라는 변수에 일대일로 매치된다

```
def print_twice( something ) :  
    print(something)  
    print(something)
```

Parameter

```
print_twice( 'Love' )
```

Argument

Parameters and Arguments 2

```
def print_twice( something ) :  
    print(something)  
    print(something)
```

```
print_twice('Love is real')
```

```
print('-'*50)
```

```
sentence = 'YOU SHALL LOVE YOUR NEIGHBOR.'  
print_twice(sentence)
```



```
Love is real  
Love is real
```

```
-----  
YOU SHALL LOVE YOUR NEIGHBOR .  
YOU SHALL LOVE YOUR NEIGHBOR .
```

Parameters and Arguments 3

```
>>> def User_pow(number1, number2):  
    result = 1  
    if number2 == 0:  
        result = 1  
    else:  
        for i in range(number2):  
            result = result*number1  
    print(result)
```

```
>>> User_pow(2,0)
```

```
1
```

```
>>> User_pow(2,3)
```

```
8
```

Parameters and Arguments 4

```
>>> def User_Max(number1, number2, number3):  
    Max = number1  
    if number2 > number1 and number2 > number3:  
        Max = number2  
    if number3 > number2 and number3 > number1:  
        Max = number3  
  
    print(Max)
```

```
>>> User_Max(1,2,3)  
3  
>>> User_Max(3,2,1)  
3  
>>> User_Max(1,3,2)  
3
```

return statement 1

- 함수 바디(body)들은 한 개 이상의 return을 포함할 수 있다
 - 함수 바디의 어느 곳에든 위치할 수 있다
 - return문은 함수 호출의 실행을 종료하고
 - 결과 즉, 반환(return) 키워드 다음에 오는 표현의 값을 호출자(caller)에게 "반환" 한다
 - 만약 반환문에 return 문이 없다면
 - 제어 흐름이 함수 바디의 끝에 도착했을 때 함수가 종료된다
 - return되는 값이 여러 개인 경우, 그 값은 tuple 형식으로 전달된다

return statement 2

```
# define function plus( a, b )
```

```
def plus( a, b ) :
```

```
    return( a+b )
```

```
i = plus(3, 5)
```

```
print("return value of function plus(3,5) = ", i)
```

```
# define function plus_list( listname )
```

```
num_list=[1,3,5,7,9]
```

```
def plus_list( listname ) :
```

```
    sum = 0
```

```
    for i in listname :
```

```
        sum = sum+i
```

```
    print (i, sum)
```

```
    return sum, listname
```

```
pl=plus_list(num_list)      #function call
```

```
print("return value of funtion plus_list([1,3,5,7,9]) = ", pl)
```

```
➡ return value of function plus(3,5) = 8  
9 25  
return value of funtion plus_list([1,3,5,7,9]) = (25, [1, 3, 5, 7, 9])
```

return statement 3

```
def fahrenheit(celsius) :  
    # returns the temperature in degrees Fahrenheit  
    return (celsius * 9 / 5) + 32  
  
for t in (22.6, 25.8, -10, 0.0) :  
    print("celcius =",t, "farenheit =", fahrenheit(t))
```

```
↳ celcius = 22.6 farenheit = 72.68  
celcius = 25.8 farenheit = 78.44  
celcius = -10 farenheit = 14.0  
celcius = 0.0 farenheit = 32.0
```

연습문제 1

- 2개의 파라미터를 지정한다
- 지정한 파라미터 두개의 곱을 리턴하는 함수를 작성
- 이 함수를 호출하기 전에,
 - 사용자에게 2개의 숫자를 입력 받는다
 - 위에 만든 함수를 호출한다
 - 결과값을 받아서 출력한다

연습문제1-코드

```
def multi(num1, num2):  
    result = num1 * num2  
    return result
```

```
num1 = 12 #@param {type:"integer"}  
num2 = 10 #@param {type:"integer"}
```

```
print(f"{num1:d} x {num2:d} = {multi(num1, num2):d}")
```

 12 × 10 = 120

예제 1

```
def sum(i, j, k) :
    return i + j + k

for i in range(10) :
    print(f"{i:2d} + {i+1:2d} + {i+2:2d} = {sum(i, i+1, i+2):2d}")

print("=" * 30)

num1 = 12 #@param {type:"integer"}
num2 = 10 #@param {type:"integer"}
num3 = 28 #@param {type:"integer"}

print(f"{num1:2d} + {num2:2d} + {num3:2d} = {sum(num1, num2, num3):2d}")
```

```
0 + 1 + 3 = 4
1 + 2 + 4 = 7
2 + 3 + 5 = 10
3 + 4 + 6 = 13
4 + 5 + 7 = 16
5 + 6 + 8 = 19
6 + 7 + 9 = 22
7 + 8 + 10 = 25
8 + 9 + 11 = 28
9 + 10 + 12 = 31
=====
12 + 10 + 28 = 50
```

예제 2

```
# define ex function : exchange list element
def exchange(listN,i,j) :
    temp = listN[i]
    listN[i] = listN[j]
    listN[j] = temp

nums=[1,3,5,7,9,11]
fr=['apple', 'banana', 'bluberry', 'lemon', 'melon']

print("*"*50)
print(nums)
print(fr)

exchange(nums,0,1)
exchange(fr,0,1)
print("exchange index 0,1 = ", nums)
print(fr)
# continue to..
```

```
# ~~~
exchange(nums,1,4)
exchange(fr,1,4)

print("exchange index 1,4 = ", nums)
print(fr)

print("*"*50)
```

```
↳ *****
[1, 3, 5, 7, 9, 11]
['apple', 'banana', 'bluberry', 'lemon', 'melon']
exchange index 0,1 = [3, 1, 5, 7, 9, 11]
['banana', 'apple', 'bluberry', 'lemon', 'melon']
exchange index 1,4 = [3, 9, 5, 7, 1, 11]
['banana', 'melon', 'bluberry', 'lemon', 'apple']
*****
```

지역 변수와 전역 변수

- 전역 변수(Global Variable)
 - 지금까지 사용한 형태의 변수
 - 프로그램 시작 시에 값을 지정하여, 계속 활용 가능하다
- 지역 변수(Local Variable)
 - Variable with local scope
 - 변수가 선언된 함수나 블록 내에서만 사용하는 변수
 - Variable with global scope
 - 변수를 함수나 블록 내에서 값을 지정하지만, 어디서나 사용 가능하게 정의
 - `global count`

전역 변수(Global Variables)

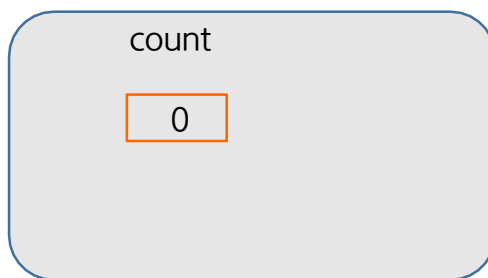
- 파이썬 인터프리터 셸에서 값을 지정하는 변수는 글로벌 변수이다
- 어디에서나 사용 가능한 변수이다
 - >>> count = 10
 - >>> count = count + 1
 - >>>
 - count 11

지역변수(Variables with Local Scope)

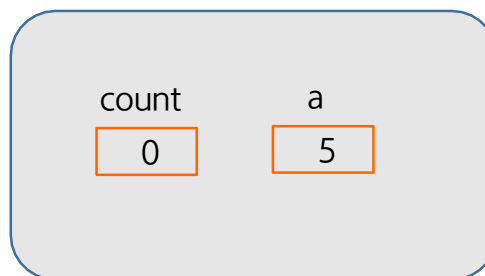
```
def fun(count):           # count: local scope
    a = 5                 # a: local scope
    count = count + a
    return count

count = 0                 # count: global scope

print('result = ', fun(count))
print(count)              # count: global scope
```



Module scope



Function fun()

```
>>> def fun(count):
        a=5
        count = count + a
        return count

>>> count = 0
>>> print('result = ', fun(count))
result = 5
>>> print(count)
0
>>>
```

전역 변수(Global Variables)

```
def fun(count):           #count: local scope
    global a              #global a is changed
    a = 5
    count = count + a
    return count
```

```
count = 0                 #count: global scope
print('result = ', fun(count))
```

```
print(a)                  #count: global scope
```

```
>>> def fun(count):           #count: local scope
    global a                  #global a is changed
    a = 5
    count = count + a
    return count

>>> a = 0
>>> print('result = ', fun(count))
result = 5
>>> print(a)
5
>>> |
```

연습문제 2

- 잔액, 예금할 금액, 출금할 금액을 입력하면 계산해 주는 프로그램을 작성한다
- 함수 2개를 작성한다
- 잔액을 global 변수로 지정하여 계산한다

연습문제 2-코드

```
def withdraw(금액):
    global 잔액
    if 잔액 - 금액 < 0 :
        return -1
    else:
        잔액 = 잔액 - 금액
        return
```

```
def deposit(금액):
    global 잔액
    잔액 = 잔액 + 금액
```

```
잔액 = 10000 #@param {type:"integer"}
입금 = 2000 #@param {type:"integer"}
deposit(입금)
print(f"입금후 현재 잔액 : {잔액:,}")
```

```
출금 = 13000 #@param {type:"integer"}
if withdraw(출금)== -1 :
    print("잔액 부족으로 출금 불가")
else:
    print("출금후 현재 잔액 : {잔액:,}")
```

```
# 연습문제 2-코드

def withdraw(금액):
    global 잔액
    if 잔액 - 금액 < 0 :
        return -1
    else:
        잔액 = 잔액 - 금액
        return

def deposit(금액):
    global 잔액
    잔액 = 잔액 + 금액

잔액 = 10000 #@param {type:"integer"}
입금 = 2000 #@param {type:"integer"}
deposit(입금)
print(f"입금후 현재 잔액 : {잔액:,}")

출금 = 13000 #@param {type:"integer"}
if withdraw(출금)== -1 :
    print("잔액 부족으로 출금 불가")
else:
    print("출금후 현재 잔액 : {잔액:,}")
```

입금후 현재 잔액 : 12,000
잔액 부족으로 출금 불가

잔액 : 10000

입금 : 2000

출금 : 13000

강의 요약

- 함수 정의 연습하기
 - parameter과 return 값의 유무에 따라
- 원하는 함수를 기술하는 과정 익히기
- 지역 변수와 전역 변수의 차이점 알아보기
 - 지역 변수: 특정 범위 내에서 사용 가능
 - 전역 변수: 프로그램 전체에서 사용 가능

03. 함수 활용과 랜덤 함수

[3차시]

학습목표

- 함수 예제를 통해 함수 이해하기
- 함수를 다양한 방식으로 활용해 보기
- 랜덤 함수 사용법을 이해하고 활용하기

함수 이해도 확인하기

- 다음에 나타나는 함수에서
 - 함수 이름은 무엇인가?
 - 파라미터는 몇 개이며, 데이터 형은?
 - 해당 함수의 기능은 무엇인가?
 - 함수 호출은 몇 번 이루어졌는가?

다각형 그리기

```
def drawPolygon(t, sideLength, numSides, color):
    t.color(color)
    turnAngle= 360 / numSides
    for i in range(numSides):
        t.pendown()
        t.forward(sideLength)
        t.right(turnAngle)

t.clearscreen()
t.speed(13)
t.penup()
t.setposition(-50, 0)
drawPolygon(t, 50, 5, 'blue')

t.penup()
t.setposition(100, 0)
drawPolygon(t, 50, 8, 'hotpink')
```

함수명	
파라미터 수	
파라미터 데이터 형	
함수의 기능	
함수 호출 횟수	

다각형 그리기-답안

함수명	drawPolygon()
파라미터 수	4개
파라미터 데이터 형	Object, 정수, 정수, 문자
함수의 기능	다각형을 그린다 t turtle을 받는 object sideLength 다각형의 한면의 크기 numSides 몇 각형인지 지정 color 다각형 선의 색상
함수 호출 횟수	2회

사각형 그리기

```
def square(t, size, color):
    t.color(color)
    for i in range(4):
        t.forward(size)
        t.right(90)

t.clearscreen()
t.speed(13)
t.pensize(3)
colors = ['red', 'orange', 'yellow', 'green', 'blue',
          'violet']

i = 30
for color in colors:
    square(t, i, color)
    i = i + 30
```

함수명	
파라미터 수	
파라미터 데이터 형	
함수의 기능	
함수 호출 횟수	

사각형 그리기-답안

함수명	square()
파라미터 수	3개
파라미터 데이터 형	Object, 정수, 문자
함수의 기능	사각형을 그린다 t turtle을 받는 object size 사각형 면의 길이 color 사각형 선의 색상
함수 호출 횟수	6회

꽃 모양 그리기

```
def flower(t, n, r):
    angle = 360/n
    for i in range(n):
        for i in range(2):
            t.circle(r,angle)
            t.left(180-angle)
            t.left(angle)

def move(t, length):
    t.penup()
    t.forward(length)
    t.pendown()

t.clearscreen()
t.speed(13)
t.color("violet")
move(t, -100)

for i in range(3):
    flower(t, 6, 30+(50*i))
```

함수명	
파라미터 수	
파라미터 데이터 형	
함수의 기능	
함수 호출 횟수	

꽃 모양 그리기-답안

함수명	flower()
파라미터 수	3개
파라미터 데이터 형	Object, 정수, 정수
함수의 기능	꽃모양을 그린다 t turtle을 받는 object n 꽃잎의 수 r 꽃잎의 반지름
함수 호출 횟수	3회

벌집 그리기

```
def hexagon(t):
    for i in range(6):
        t.forward(80)
        t.left(60)
```

```
t.speed(13)
t.color('red')
hexagon(t)
```

```
t.clearscreen()
for i in range(6):
    hexagon(t)
    t.forward(80)
    t.right(60)
```

함수명	
파라미터 수	
파라미터 데이터 형	
함수의 기능	
함수 호출 횟수	

벌집 그리기-답안

함수명	hexagon()
파라미터 수	1개
파라미터 데이터 형	Object
함수의 기능	벌집 모양을 그린다 t turtle을 받는 object
함수 호출 횟수	7회

난수를 발생시켜 활용하는 경우

- 규칙적으로 변하지 않는 값이 필요한 경우
- 게임을 만드는데, 적이 움직이는 방향을 예상치 못하도록 움직이도록 해야 할 때
- 방향을 16가지 경우로 수로 발생시키고, 각각 나타나는 결과가 예측하지 못해야 하는 경우
- 컴퓨터가 생각한 숫자를 맞추는 게임을 할 때, 컴퓨터가 만들어 내는 숫자가 필요 할 때

랜덤 함수

- 랜덤 모듈을 먼저 import 하여야 사용 가능하다

```
>> import random
```

- 랜덤한 값을 만드는 함수는 다양하다

- 정수형 랜덤값 생성하기

```
>> random.randrange(1,9,1)
```

- 실수형 랜덤값 생성하기

```
>> random.uniform(1,2)
```

- 리스트에서 선택하기

```
>> random.choice([1,2,3,4,5,6,7])
```

다양한 랜덤 함수

<https://docs.python.org/3/library/random.html?highlight=random#module-random>

함수	기능
<code>random.seed()</code>	랜덤 숫자 발생기를 초기화 한다
<code>random.randrange(start, stop, step)</code>	정수형 난수를 발생시킨다
<code>random.uniform(start, stop)</code>	실수형 난수를 발생시킨다
<code>random.choice(list)</code>	리스트에서 한 개를 선정한다
<code>random.shuffle(list)</code>	리스트가 기억하는 아이템들의 위치를 임의로 바꾼다
<code>random.sample(list, n)</code>	리스트에서 중복없이 원하는 개수만큼 선정한다

정수형 난수

```
#generate random integer

import random
originNumber=[]

for i in range(3):
    originNumber.append( random.randrange(1,9) )

print(originNumber)

for i in range(5):
    originNumber.append( random.randrange(10,100,5) )

print(originNumber)
```

```
➤ [7, 7, 6]
   [7, 7, 6, 50, 35, 10, 15, 65]
```


실수형 난수

```
#generate random float

import random

for i in range(3):
    print("random float from 1 to 2 = ", random.uniform(1,2) )

print("***50)

for i in range(3):
    print("random float from 11 to 13 = ", random.uniform(11,13) )
```

```
random float from 1 to 2 = 1.296858359081268
random float from 1 to 2 = 1.9432065322241545
random float from 1 to 2 = 1.5337214921173756
*****
random float from 11 to 13 = 11.357557367208155
random float from 11 to 13 = 12.4481086229418
random float from 11 to 13 = 12.881873449687294
```

리스트에서 아이템 한 개 선정

```
#generate random value from list

import random
numList=[1,3,5,2.2,1.35,5,9,9.5,11,15,5.7]
fruitList=["apple", "banana", "citrus", "blueberry", "blackberry", "lemon"]

for i in range(5):
    print("select from numList = ", random.choice(numList) )

print("*"*50)

for i in range(3):
    print("select from fruitList = ", random.choice(fruitList) )
```

```
select from numList = 2.2
select from numList = 2.2
select from numList = 11
select from numList = 5.7
select from numList = 9
*****
select from fruitList = blueberry
select from fruitList = blueberry
select from fruitList = blackberry
```

리스트에서 중복없이 여러 개 선정

```
# select random integer without repetition

import random

random_list = random.sample(range(1,10), 5)
print("random.sample(range(1,10), 5) = ", random_list)

random_list = random.sample(random_list, 2)
print("random.sample(random_list, 2) = ", random_list)

random_list = random.sample(range(5, 90, 4), 10)
print("random.sample(range(5, 90, 4), 10) = ", random_list)
```

```
➞ random.sample(range(1,10), 5) = [4, 5, 3, 1, 8]
   random.sample(random_list, 2) = [5, 4]
   random.sample(range(5, 90, 4), 10) = [37, 33, 81, 17, 45, 69, 25, 65, 9, 13]
```

연습문제 1

- List이름과 정수를 parameter로 입력하는 함수를 만든다
- 함수는 List 아이템 중 하나를 랜덤으로 선택해서 보여준다. 입력한 개수 만큼 랜덤 값을 보여준다.
- return 문은 사용하지 않는다

연습문제 1-코드

```
import random

def random_picker(lists, number) :
    for i in range(number) :
        print(random.choice(lists))

num_list=[3,1, 7, 11, 25, 5, 4, 9]
random_picker(num_list, 3)
```

```
3
25
9
```

연습문제 2

- 함수 정의
 - 두 개의 정수 매개변수(parameter)를 설정
 - 첫 번째 매개변수보다 크고 두 번째 매개변수보다 작은 구간에서 정수형 난수 생성
 - ("A"+생성된 정수)에 해당하는 대문자를 출력
- 함수를 10번 호출하여 결과를 출력한다

연습문제 2-코드

```
import random

def gen_random(a,b):
    r=random.randrange(a,b)
    c=chr(65+r)
    print(r, c)

for i in range(10) :
    gen_random(i+1, i+10)
```

```
1 B
7 H
6 G
11 L
6 G
14 O
11 L
8 I
15 P
10 K
```

연습문제 3

- 주사위 두 개를 던져 같은 숫자가 나올 때까지 반복하는 프로그램
- 함수 정의
 - 1부터 6 사이의 정수 난수 생성과 반환


연습문제 3-코드

```
import random

def dice():
    d = random.randrange(1, 6)
    return d

n1 = dice()
n2 = dice()
print(n1, n2)

while( n1 != n2 ):
    n1 = dice()
    n2 = dice()
    print(n1, n2)
```



```
1 5
5 4
3 2
1 3
4 5
5 1
5 5
```

강의 요약

- 함수 예제를 통해 함수 이해하기
 - 함수명, parameter, 기능, 함수 호출 이해하기
- 함수를 다양한 방식으로 활용해 보기
- 랜덤 함수 사용법을 이해하고 활용하기
 - `import random`
 - 정수형, 실수형, 리스트 내 랜덤값 생성 가능

퀴즈 1

return()의 특징을 설명한 것이 아닌 문장은?

- ① return()문은 함수 바디의 어느 곳에든 위치할 수 있다
- ② return()문은 함수 호출의 실행을 종료한다
- ③ return되는 값이 여러 개이며 list 형식으로 전달된다
- ④ return()문은 한 개의 함수에서 여러 번 기술 가능하다

퀴즈 1-답안

return()의 특징을 설명한 것이 아닌 문장은?

- ① return()문은 함수 바디의 어느 곳에든 위치할 수 있다
- ② return()문은 함수 호출의 실행을 종료한다
- ③ return되는 값이 여러 개이며 list 형식으로 전달된다
- ④ return()문은 한 개의 함수에서 여러 번 기술 가능하다

퀴즈 2

랜덤 함수를 이용하여 특정한 리스트에서 3개의 값을 임의로 선택하려고 한다. 어떤 함수를 사용해야 하는가?

- ① `random.uniform()`
- ② `random.randrange()`
- ③ `random.sample()`
- ④ `random.choice()`

퀴즈 2-답안

랜덤 함수를 이용하여 특정한 리스트에서 3개의 값을 임의로 선택하려고 한다. 어떤 함수를 사용해야 하는가?

- ① `random.uniform()`
- ② `random.randrange()`
- ③ `random.sample()`
- ④ `random.choice()`

04. 모듈의 이해

[3차시]

학습목표

- 모듈 이해하기
- string 모듈 활용해 보기
- math 모듈 활용해 보기
- datetime 모듈 활용해 보기

모듈(Modules)

- 표준라이브러리(standard library)의 일부분
 - 파이썬 프로그램에서 사용되기 위한 명령문을 포함하고 있는 파일
 - 프로그래밍에 앞서 구현하려는 기능이 파이썬 라이브러리 모듈에 있는지 여부 확인 필요
- turtle, random 모듈을 이미 사용해 봄
- random, datetime, math, string, turtle, tkinter, file 등 200여개 모듈 존재

내장형 모듈 1

```
def dump(expression) :  
    result = eval(expression)  
    print(expression, "=>", result, type(result))
```

```
dump("1")  
dump("1.0")  
dump("'string'")  
dump("1.0 + 2.0")  
dump("'*' * 10")  
dump("len('world')")
```

```
1 => 1 <class 'int'>  
1.0 => 1.0 <class 'float'>  
'string' => string <class 'str'>  
1.0 + 2.0 => 3.0 <class 'float'>  
'*' * 10 => ********** <class 'str'>  
len('world') => 5 <class 'int'>
```

내장형 모듈 2

```
def f1(a):  
    print(type(a), a)  
f1(2)  
f1(2.00)  
f1('two')
```

```
<class 'int'> 2  
<class 'float'> 2.0  
<class 'str'> two
```

모듈 String

```
import string
```

```
text = "All that I need you"  
print("capword", "=>", string.capwords(text))  
  
print("upper", "=>", text.upper())  
print("lower", "=>", text.lower())  
print("split", "=>", text.split(" "))  
print("replace", "=>", text.replace("you", "him"))  
print("find", "=>", text.find("All"))  
print("count", "=>", text.count("e"))
```

```
capword => All That I Need You  
upper => ALL THAT I NEED YOU  
lower => all that i need you  
split => ['All', 'that', 'I', 'need', 'you']  
replace => All that I need him  
find => 0  
count => 2
```

String을 숫자로 변환하기

```
# to convert strings to numbers
```

```
print(int("4711"))
```

```
print(float("4711"))
```

```
4711  
4711.0
```

연습문제 1

스트링 변환하기

- 사용자에게 30글자 이상의 영어문장을 입력 받는다.
- 알파벳이 몇 개로 구성되었는지 출력한다.
- 대문자, 소문자 각각 몇 개로 구성되었는지 출력한다.

연습문제 1-코드

```
import string

input_str = "Simple is Better than Complex."  #@param{type:"string"}
alpha = 0
upper = 0
lower = 0

for letter in input_str:
    alpha = alpha + letter.isalpha()
    upper = upper + letter.isupper()
    lower = lower + letter.islower()

print(f"알파벳 개수: {alpha}, 대문자 개수: {upper}, 소문자 개수: {lower}")
```

알파벳 개수: 25, 대문자 개수: 3, 소문자 개수: 22

math Module (1)

<http://docs.python.org/3/library/math.html>

Methods with Description	
math.ceil(x) ($N \geq x$)를 만족하는 가장 큰 정수 N을 반환	math.fsum(<i>iterable</i>) 합계
math.copysign(x, y) y의 부호만 x에 복사해서 반환	math.isfinite(x) 유한수인 경우 참
math.fabs(x) 절대값	math.isinf(x) 무한수인 경우 참
math.factorial(x)	math.fmod(x, y) 실수의 mod 함수
math.floor(x) ($N \leq x$)를 만족하는 가장 큰 정수 N을 반환	math.modf(x) 입력받은 x를 순수 소수 부분과 정수 부분으로 분리해 tuple 로 반환, 분리된 부분은 모두 부호가 할당

math Module (2)

- 승, 로그 함수

<http://docs.python.org/3/library/math.html>

Methods with Description	
math.exp(x)	math.log2(x) Return the base-2 logarithm of x
math.log(x[, base])	math.log10(x) Return the base-10 logarithm of x
math.pow(x, y)	math.sqrt(x)

math Module (3)

- 삼각함수, 각도 변환 <http://docs.python.org/3/library/math.html>

Methods with Description	
math.asin(x)	math.sin(x)
math.atan(x)	math.tan(x)
math.atan2(y, x)	math.cos(x)
math.degrees(x)	math.radians(x)

math Module (4)

- Hyperbolic functions(쌍곡선 함수) 과 constants

<http://docs.python.org/3/library/math.html>

Methods with Description	
math.acosh(x)	math.sinh(x)
math.asinh(x)	math.tanh(x)
math.atanh(x)	math.pi constant $\pi = 3.141592...$
math.cosh(x)	math.e constant $e = 2.718281...$

math Module 예제

```
>>> import math

>>> math.fmod(5.0, 2.0)      # 실수형의 나머지 연산
1.0

>>> math.factorial(5)        # 정수의 factorial
120

>>> math.log(100,10)         # log값
2.0

>>> math.pow(2,3)
8.0

>>> math.sqrt(16)
4.0

>>> math.e                   # 자연로그 값
2.718281828459045
```

cmath module

- Math module 동일한 기능을 복소수 대상으로 지원
- 참조 : <https://docs.python.org/3/library/cmath.html#module-cmath>

연습문제 2

- Math module을 활용하여
 - 이차방정식의 근의 공식을 만든다
 - 이차방정식의 a, b, c값을 입력 받아 실근을 출력한다
 - a, b, c값에 따라 함수를 선택하여 근을 계산한다
 - 근의 공식 = $\frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$

연습문제 2-코드, 실근만 처리


```
import math

def roots_formula(a, b, c):
    d = b**2 - 4*a*c
    if d > 0:
        root1 = (-b + math.sqrt(d)) / (2*a)
        root2 = (-b - math.sqrt(d)) / (2*a)
        return root1, root2
    elif d == 0:
        root1 = (-b / (2*a))
        return root1
    else:
        return False

r = roots_formula(1,5,4)
print( r )
```

판별식 계산

허근일 경우 false로 처리

 (-1.0, -4.0)

연습문제 2-코드, 허근 포함


```
import math, cmath

def deter(a, b, c) :
    return math.pow(b, 2) - 4*a*c

def roots_formula(a, b, c):
    if deter(a,b,c) >= 0:
        root01 = (-b + math.sqrt(deter(a,b,c)))/ (2*a)
        root02 = (-b - math.sqrt(deter(a,b,c)))/ (2*a)
    else:
        root01_real = -b/(2*a)
        root01_imag = (math.sqrt(math.fabs(deter(a,b,c))))/ (2*a)
        root02_real = -b/(2*a)
        root02_imag = (math.sqrt(math.fabs(deter(a,b,c))))/ (2*a)
        root01 = root01_real + root01_imag * 1j
        root02 = root02_real - root02_imag * 1j

    return [root01, root02]

print(roots_formula(1,3,5))
```

 `[(-1.5+1.6583123951777j), (-1.5-1.6583123951777j)]`

연습문제 3

Math module을 활용하여

- 사용자에게 각도(degree)를 입력 받는다
- 입력 값을 radian 값으로 바꾼 후
- 다음의 삼각함수 값을 출력 하시오
 $\sin()$, $\cos()$, $\tan()$

연습문제 3-코드

```
import math

angle = 90 #@param {type:"integer"}
rad = math.radians(angle)           # radian 값으로 바꾸기

result_sin = math.sin(rad)
result_cos = math.cos(rad)
result_tan = math.tan(rad)

print("degree:", angle, ", radian:", rad)
print("sin:", result_sin, ", cos:", result_cos, ", tan:", result_tan)
```



```
degree: 90 , radian: 1.5707963267948966
sin: 1.0 , cos: 6.123233995736766e-17 , tan: 1.633123935319537e+16
```

Datetime module (1)

<https://docs.python.org/3/library/datetime.html?highlight=datetime#module-datetime>

- 날짜와 시간 관련 기능을 제공
- class datetime.date
 - Gregorian calendar 기준
 - Year, month, day
- class datetime.time
 - Hour, minute, second, microsecond
- class datetime.datetime
 - Class date, time을 합친 것

예제, date object

- 날짜를 불러오고, 날짜 간의 마이너스가 가능하다

```
import time
from datetime import date

today = date.today()
print(today)

bday = date(today.year+1, 2, 24)
print(bday)

due = abs(bday - today)
print(due)
```

```
2022-04-18
2023-02-24
312 days, 0:00:00
```

예제, datetime object

.datetime

.utcnow, 협정 세계시(Coordinated Universal Time, UTC)

.astimezone, 타임존 기준 시각

```
from datetime import date, time, datetime
from pytz import timezone
```

```
print(datetime(1990, 7, 5, 12, 30))
```

```
now = datetime.utcnow()
print(now)
print(now.astimezone(timezone('Asia/Seoul')))
```

```
1990-07-05 12:30:00
2022-04-18 08:42:07.704803
2022-04-18 17:42:07.704803+09:00
```

연습문제 4

친구의 생일이 며칠 남았는지 계산하려고 한다

- 사용자에게 생일을 입력 받는다
- 오늘부터 생일까지 며칠 남았는지 계산하여 출력한다

연습문제 4-코드

```
from datetime import date

birthday = input("생일을 입력하세요 [ex)3월15일 = 3 15] : ")
birthday = birthday.split()
today = date.today()
birthday = date(today.year, int(birthday[0]), int(birthday[1]))

due = birthday - today
if due.days < 0:
    next_birthday = date(today.year + 1, birthday.month, birthday.day)
    due = next_birthday - today

print("생일까지 남은 날짜는: ", due.days)
```

→ 생일을 입력하세요 [ex)3월15일 = 3 15] : 3 15
생일까지 남은 날짜는: 331

연습문제 5

날짜를 입력하면 해당 날짜의 요일을 출력하려고 한다

- `datetime.date(년, 월, 일).weekday()` 함수 사용한다
- `weekday()` 는 return 값이 숫자이므로
- 요일 리스트를 만들어 변환하는 과정을 추가한다

연습문제 5-코드

```
import datetime

yy = 2022 #@param{type:"integer"}
mm = 4    #@param{type:"integer"}
dd = 12   #@param{type:"integer"}

weekString = ["월", "화", "수", "목", "금", "토", "일"]
day = datetime.date(yy,mm,dd).weekday()
day = weekString[day]

print(yyyy,"년", mm, "월", dd, "일은", day, "요일 입니다")
```

📄 2022 년 4 월 12 일은 화 요일 입니다

강의 요약

- **모듈(module)이란**
 - 표준 라이브러리의 일부분
 - 파이썬 프로그램에서 사용되기 위한 명령문을 포함하고 있는 파일
- **string 모듈 활용해 보기**
 - .upper(), .lower(), .split() 등
- **math 모듈 활용해 보기**
 - .fmod(), .factorial(), .log(), .pow() 등
- **datetime 모듈 활용해 보기**
 - .today(), .year(), .month(), .day() 등

05. 다양한 모듈 활용

[3차시]

학습목표

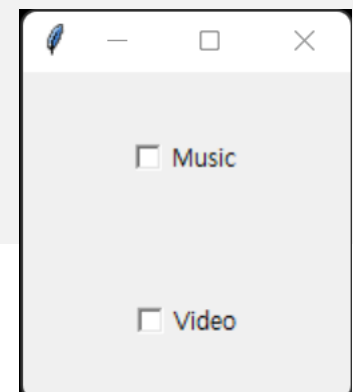
- tkinter 모듈 활용하기
- os 모듈 활용하기
- 사용자가 만드는 모듈 정의하기

그래픽 처리 Tkinter module

- 웹 프로그래밍을 할 수 있도록 GUI를 제공
- Tool Kit for interactive programming
- ttk library를 같이 활용
- Tkinter widgets
 - Button
 - Checkbutton
 - Entry
 - Frame
 - Label
 - Menubutton

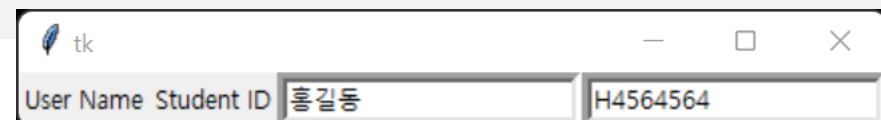
Checkbox

```
from tkinter import *  
  
top = Tk()  
  
CheckVar1 = IntVar()  
CheckVar2 = IntVar()  
  
C1 = Checkbutton(top, text = "Music", variable = CheckVar1, onvalue = 1,  
                  offvalue = 0, height=5, width = 20)  
C2 = Checkbutton(top, text = "Video", variable = CheckVar2, onvalue = 1,  
                  offvalue = 0, height=5, width = 20)  
  
C1.pack()  
C2.pack()  
  
top.mainloop()
```



Entry widget, 입력 받기

```
from tkinter import *  
  
top = Tk()  
L1 = Label(top, text="User Name")  
L1.pack(side = LEFT)  
  
E1 = Entry(top, bd =5)  
E1.pack(side = RIGHT)  
  
L2 = Label(top, text="Student ID")  
L2.pack(side = LEFT)  
  
E2 = Entry(top, bd =5)  
E2.pack(side = RIGHT)  
  
top.mainloop()
```



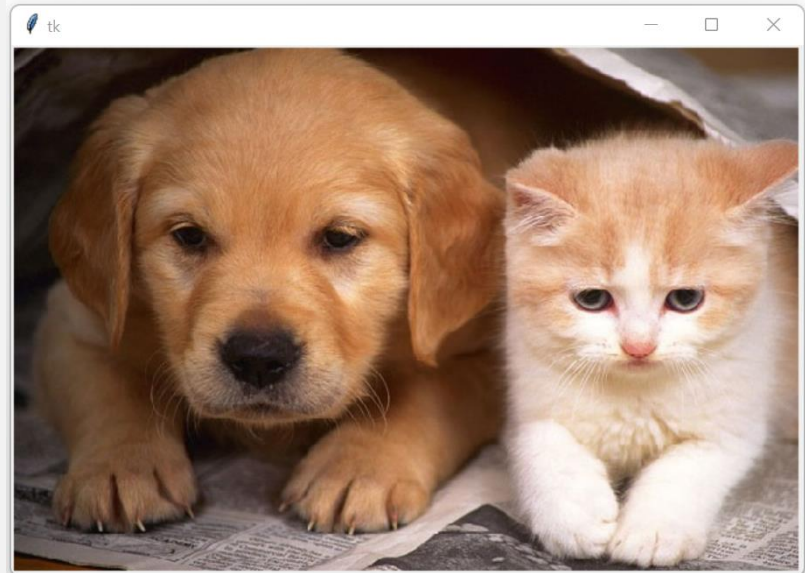
Displaying Images

```
# 이미지 표시
from tkinter import *
from tkinter import Tk, Canvas
from PIL import ImageTk, Image

root = Tk()
canvas = Canvas(root, width=600, height=400)
canvas.pack()

im = Image.open('./image/acatdog.jpg')
canvas.image = ImageTk.PhotoImage(im)
canvas.create_image(0, 0, image=canvas.image, anchor='nw')

root.mainloop()
```



다양한 모듈

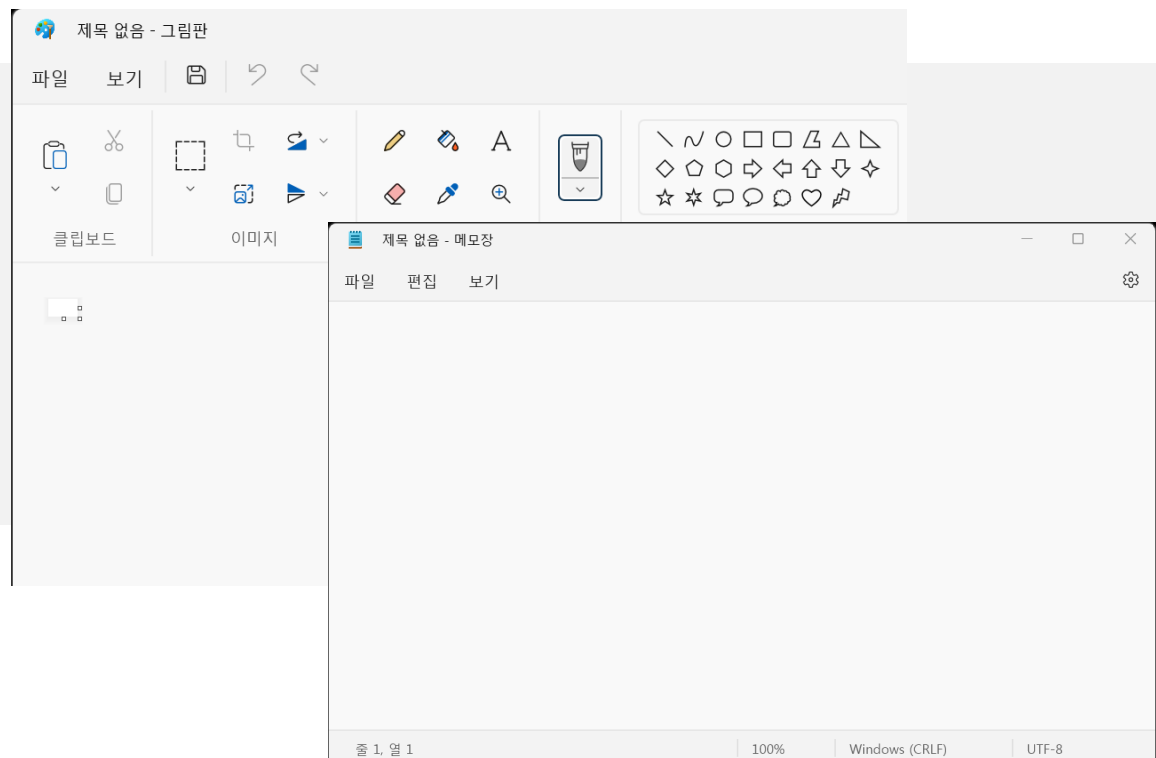
다양한 모듈	
sys	프로그램 실행 환경과 관련한 정보를 제공
os	파일, 프로세스, 디렉토리 등 다양한 운영체제 기능 제공
htmllib	HTML 분석 모듈
cgi	cgi-bin에서 파이썬으로 웹 응용 작성을 도움

Operating System Modules (1)

os module

- 다수의 운영체제 함수들에 통일된 인터페이스를 제공

```
>>> from os import *  
  
>>> system('mspaint')  
  
>>> system('notepad')
```



Operating System Modules (2)

Using the os.path module to handle filename

```
import os
```

```
filename = "my/little/pony"
```

```
print("using", os.name, "...")
```

```
print("split", "=>", os.path.split(filename))
```

```
print("splittext", "=>", os.path.splittext(filename))
```

```
print("dirname", "=>", os.path.dirname(filename))
```

```
print("basename", "=>", os.path.basename(filename))
```

```
print("join", "=>", os.path.join(os.path.dirname(filename), os.path.basename(filename)))
```

```
using posix ...
split => ('my/little', 'pony')
splittext => ('my/little/pony', '')
dirname => my/little
basename => pony
join => my/little/pony
```

사용자가 만드는 모듈 1

```
# filename : fibo.py  
# Fibonacci numbers module
```

```
def fib(n) :  
    if n == 0 :  
        return 0  
    elif n == 1 :  
        return 1  
    else :  
        return fib(n-1) + fib(n-2)
```

```
def ifib(n) :  
    a = 0  
    b = 1  
    for i in range(n) :  
        a = b  
        b = a + b  
    return a
```

fibo.py안에 함수 2개를 선언한다
이 함수들을 사용하고 싶은 곳에서,
`import fibo`
쓰면, 저장되어 있는 함수 2개 사용 가능하다

```
import fibo
```

```
f = fibo  
print(f.fib(11))  
print(f.ifib(11))
```

```
89  
1024
```

사용자가 만드는 모듈 2

- 자주 사용하는 함수들을 모아서 모듈로 사용 가능
- 관련 있는 함수들은 모아서, 몇 개의 사용자 정의 모듈을 만들면
 - 코딩 소요 시간 감소
 - 다른 사람과 같이 작업할 때, 공유해야 함

연습문제 1

- 자주 사용하는 함수 3개를 저장하여 'freq.py'에 저장한다
- `import freq` 사용하여 저장된 함수 3개를 사용한다

연습문제 1-코드

```
# freq.py
from datetime import date
Import math, cmath

def cal_birthday(month, day):
    today = date.today()
    birthday = date(today.year, month, day)
    due = birthday - today
    if due.days < 0 :
        next_birthday = date(today.year + 1, birthday.month, birthday.day)
        due = next_birthday - today

    print("생일까지 남은 날짜는: ", due.days)
def deter(a, b, c):
    return math.pow(b, 2) - 4*a*c

# continue to..
```

연습문제 1-코드

```
def roots_formula(a, b, c):  
    if deter(a,b,c) >= 0:  
        root01 = (-b + math.sqrt(deter(a,b,c)))/ (2*a)  
        root02 = (-b - math.sqrt(deter(a,b,c)))/ (2*a)  
    else:  
        root01_real = -b/(2*a)  
        root01_imag = (math.sqrt(math.fabs(deter(a,b,c))))/ (2*a)  
        root02_real = -b/(2*a)  
        root02_imag = (math.sqrt(math.fabs(deter(a,b,c))))/ (2*a)  
        root01 = root01_real + root01_imag * 1j  
        root02 = root02_real - root02_imag * 1j  
  
    return [root01, root02]
```

```
import freq
```

```
f = freq  
print(f.roots_formula(1, 2, 3))  
f.cal_birthday(11, 24)
```

```
➞ [(-1+1.4142135623730951j), (-1-1.4142135623730951j)]  
생일까지 남은 날짜는: 220
```


연습문제 2

- 'operators.py'에 덧셈, 뺄셈, 곱셈 과정과 결과를 함께 출력하는 함수 3개를 만든다.
- Import 하여 사용한다.

연습문제 2 코드

```
# operators.py

def add(a, b):
    print(a, '+', b, '=', a+b)

def mul(a, b):
    print(a, '*', b, '=', a*b)

def min(a, b):
    print(a, '-', b, '=', a-b)
```

```
import operators

equation = operators
equation.add(1, 5)
equation.min(90, 42)
equation.mul(9, 9)
```

```
➤ 1 + 5 = 6
   90 - 42 = 48
   9 * 9 = 81
```

다양한 모듈 1

String 관련 Module	
Re	String을 효과적으로 분석, 처리하는 정규식 지원
struct	C언어 API 지원, struct 다루고 binary file 처리 지원
Difflib	file비교 관련 module
text wrap	word-wrapping들 텍스트 처리 지원
codecs	텍스트 인코딩 관련 module

다양한 모듈 2

Data type 관련 Module	
calendar	달력, 윤달 확인, 주 단위 시작 종료 추출
bisect	stack, queue 관련 지원
array	이미지나 음성 파일 처리 시 사용 배열 지원
copy	복잡한 오브젝트의 복사본 관리
pprint	내포한 리스트나 딕셔너리 보기 쉽게 출력
sets	임의의 집합 관리

다양한 모듈 3

인터넷 데이터 처리 Module	
mimify	메일 메시지의 인코딩 및 디코딩
Binscii	binary data와 ASCII 간의 변환
Binhex	MAC용 binhex 의 압축 및 압축해제
quopri	복잡한 오브젝트의 복사본 관리

다양한 모듈 4

마크업 처리 Module	
htmllib	HTML 분석 module
sgmlib	SGML 분석 module
xml.sax, xml.dom	XML 분석 module
formatter	HTML, XML 및 기타 형식으로 출력 지원

다양한 모듈 5

인터넷 프로토콜 처리 Module	
cgi	cgi-bin에서 파이썬으로 웹응용 작성 지원
urllib, urlparse	URL 열고 결과를 파싱 지원
httplib, ftplib, gopherlib	HTTP, FTP 프로토콜 client 사용 지원
Poplib, imaplib	메일 읽는 POP3, IMAP 지원
SocketServer	TCP, UDP server 제작 지원
SimpleHTTPServer, CGIHTTPServer, BasedHTTPServer	간단한 웹 서버 구축 지원
smtplib	메일 전송을 위한 SMTP/ESMTP client 구현

강의 요약

- **tkinter module 활용하기**
 - tkinter module : 그래픽 처리 기능. GUI 제공
 - ttk library를 같이 활용
- **os 모듈 활용하기**
 - 다수의 운영체제 함수들에 통일된 인터페이스 제공
- **사용자가 만드는 모듈 정의하기**
 - 자주 사용하는 함수들을 모아서 모듈로 정의
 - 관련 함수들을 모아 공동 작업자와 공유 가능

퀴즈 1

다음 중 모듈의 이름이 아닌 것을 찾으시오

- ① date
- ② math
- ③ ossys
- ④ random
- ⑤ turtle
- ⑥ cmath
- ⑦ tkinter

퀴즈 1-답안

다음 중 모듈의 이름이 아닌 것을 찾으시오

- ① date
- ② math
- ③ **ossys**
- ④ random
- ⑤ turtle
- ⑥ cmath
- ⑦ tkinter

퀴즈 2

사용자 정의 모듈에 대한 설명 중 틀린 것은?

- ① 자주 사용하는 함수들을 모아서 모듈로 정의한다
- ② 사용자 정의 모듈과 내장형 모듈의 사용 방식은 다르 다
- ③ 사용자 정의 모듈은 코딩 소요 시간을 감소하게 한다
- ④ 다른 사람과 같이 작업하면서 공유할 때 효율적이다

퀴즈 2-답안

사용자 정의 모듈에 대한 설명 중 틀린 것은?

- ① 자주 사용하는 함수들을 모아서 모듈로 정의한다
- ② 사용자 정의 모듈과 내장형 모듈의 사용 방식은 다르다
- ③ 사용자 정의 모듈은 코딩 소요 시간을 감소하게 한다
- ④ 다른 사람과 같이 작업하면서 공유할 때 효율적이다

06. 재귀함수의 이해

[3차시]

학습목표

- 재귀 함수가 무엇인지 이해하기
- 재귀 함수를 반복문으로 변경해보기

재귀함수 (Recursive function)

- 재귀(Recursion)
 - 함수가 바디에서 자기 자신을 호출하는 프로그래밍의 메소드 혹은 함수
- 재귀함수
 - 재귀로 정의된 함수를 지칭(비공식적으로 종종 사용됨)
 - 재귀 함수에는 **종료 조건을 반드시 명시해야 함**
그렇지 않은 경우에 무한 루프로 빠질 수 있음

스스로 부르는 함수

```
def countdown(n) :  
    print(n)  
    if n > 1 :  
        countdown(n-1)  
  
countdown(5)
```



```
5  
4  
3  
2  
1
```


재귀 함수, pow

```
def pow(n1, n2) :  
    if n2 == 0 :  
        return 1  
    else:  
        return( n1 * pow(n1, n2-1) )  
  
print(pow(3, 8))
```

 6561

재귀 함수, $f(n) = 3 * n$

$f(n) = 3 * n$, 즉, 3의 배수의 재귀 함수 버전을 생각해보자

```
def mult3(n) :  
    mul=0  
    if n >=1 :  
        for i in range(n):  
            mul=mul+3  
    return mul  
  
for i in range(1,10) :  
    print(mult3(i))
```



```
3  
6  
9  
12  
15  
18  
21  
24  
27
```

연습문제 1

아래 코드를 재귀함수를 사용하여 실행 되도록 수정해 보시오

```
def mult3(n) :  
    if n == 1 :  
        return 3  
    else:  
        for i in range(1, n):  
            return 3*n  
  
for i in range(1,10) :  
    print(mult3(i))
```

연습문제 1-코드

```
def mult3(n) :  
    if n == 1 :  
        return 3  
    else:  
        return 3 + mult3(n-1)  
  
for i in range(1,10) :  
    print(mult3(i))
```



```
3  
6  
9  
12  
15  
18  
21  
24  
27
```

재귀 함수, pattern()

```
def pattern(n) :  
    if n == 0 :  
        print(0, end=' ')  
    else:  
        pattern(n-1)  
        print(n, end=' ')
```

```
pattern(5)
```

```
print()
```

```
pattern(11)
```

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

재귀 함수, factorial

```
def factorial(n) :  
    print(f"함수 호출 n = {n}")  
    if n == 1 : # 종료 조건  
        return 1  
    else :  
        res = n * factorial(n-1)  
        print(f"중간 계산값 {n} * factorial({n-1}) = {res}")  
    return res
```

```
i = 5 #@param{type:"integer"}  
print(f"최종 결과값 factorial(i) = {factorial(i)}")
```

```
↳ 함수 호출 n = 5  
함수 호출 n = 4  
함수 호출 n = 3  
함수 호출 n = 2  
함수 호출 n = 1  
중간 계산값 2 * factorial(1) = 2  
중간 계산값 3 * factorial(2) = 6  
중간 계산값 4 * factorial(3) = 24  
중간 계산값 5 * factorial(4) = 120  
최종 결과값 factorial(i) = 120
```

연습문제 2

- 피보나치 수열을 재귀 함수로 만드시오
- 다음과 같이 항이 생성되는 수열

$$f1 = 1$$

$$f2 = 1$$

$$f3 = f1 + f2$$

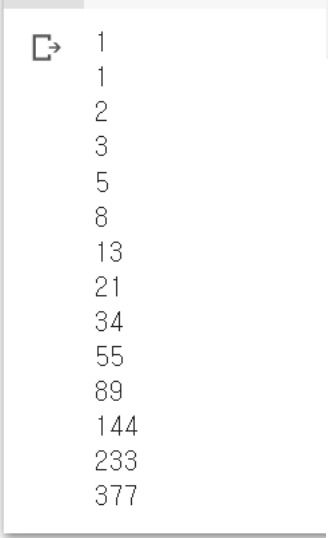
$$f4 = f2 + f3$$

...

$$fn = fn-1 + fn-2$$

연습문제 2-코드

```
def fibo(n) :  
    if n == 1 or n == 2 :  
        return 1  
    else :  
        return fibo(n-1) + fibo(n-2)  
  
for i in range(1,15) :  
    print(fibo(i))
```



```
1  
1  
2  
3  
5  
8  
13  
21  
34  
55  
89  
144  
233  
377
```


연습문제 3

- 정수를 입력 받아
- 1부터 입력 받은 정수까지의 합을 구하는 함수를 작성한다

연습문제 3-코드

```
def sum(n) :  
    if n<=0 :  
        return 0  
    else :  
        return n + sum(n-1)  
  
num = int(input("숫자를 입력하세요: "))  
sum = sum(num)  
  
print(f"1부터 {num}까지의 합은 {sum}입니다.")
```

☞ 숫자를 입력하세요 : 5
1부터 5까지의 합은 15입니다 .

☞ 숫자를 입력하세요 : 100
1부터 100까지의 합은 5050입니다 .

강의 요약

- 재귀 함수가 무엇인지 이해하기
 - 함수가 body에서 자기 자신을 호출하는 프로그래밍 의 메소드 혹은 함수
 - 반드시 종료조건을 명시해야 함
- 재귀 함수를 반복문으로 변경해보기

퀴즈 1

- 재귀함수에 대한 설명 중 가장 적절한 것은?
 - ① 종료 조건을 명시하는 것이 가장 중요하다
 - ② 파라미터를 지정하는 것이 가장 중요하다
 - ③ 반환문(return)을 지정하는 것이 가장 중요하다
 - ④ 위의 3가지 모두 맞다

퀴즈 1-답안

- 재귀함수에 대한 설명 중 가장 적절한 것은?
 - ① 종료 조건을 명시하는 것이 가장 중요하다
 - ② 파라미터를 지정하는 것이 가장 중요하다
 - ③ 반환문(return)을 지정하는 것이 가장 중요하다
 - ④ 위의 3가지 모두 맞다

퀴즈 2

- 다음 코드의 결과는?

```
def pattern(n) :  
    if n == 0 :  
        print(0, end=' ')  
    else:  
        pattern(n-1)  
        print(n, end=' ')  
  
pattern(5)
```

- ① 1 2 3 4 5
- ② 0 1 2 3 4 5
- ③ 5 4 3 2 1
- ④ 5 4 3 2 1 0

퀴즈 2-답안

- 다음 코드의 결과는?

```
def pattern(n) :  
    if n == 0 :  
        print(0, end=' ')  
    else:  
        pattern(n-1)  
        print(n, end=' ')
```

pattern(5)

- ① 1 2 3 4 5
- ② 0 1 2 3 4 5
- ③ 5 4 3 2 1
- ④ 5 4 3 2 1 0

0 1 2 3 4 5

07. 튜플의 이해

[3차시]

학습목표

- 튜플이 무엇인지 알기
- 리스트와 튜플의 차이점을 이해하기
- 여러 명령어 사용해 보기

데이터 구조

- List
 - num=[1,2,3,4,5]
 - index는 0부터 시작하여 숫자로 구성
- Tuple
 - 변경 불가능한 리스트
 - num=(1,2,3,4,5)
 - index는 0부터 시작하여 숫자로 구성
- Dictionary
 - 모든 종류의 데이터 형을 index로 사용 가능
 - days = { 'Sun':'Sunday', 'Mon':'Monday', 'Tue':'Tuesday', 'Wed':'Wednesday', 'Thu':'Thursday', 'Fri':'Friday', 'Sat':'Saturday' }

Tuples (튜플)

- 튜플은 값들의 나열(sequence)
 - 모든 데이터형 가능
 - 정수로 색인(index)
 - 튜플은 list와 유사
 - 서로 다른 데이터형 혼합하여 저장 가능
 - 차이점은
 - 튜플은 변경 불가능, 리스트는 변경 가능
- 함수에서 여러 개의 값을 리턴 하면 튜플로 생성됨
 - 리턴된 값이 변경되지 못하도록 튜플로 처리

Tuples (튜플)

- 튜플에는 서로 다른 데이터형 저장 가능
- 정의 할 때 괄호를 사용

```
# create tuples
>>> t = ('a', 'b', 'c', 'd', 'e')
>>> print(t)
('a', 'b', 'c', 'd', 'e')

>>> t1 = ( 1, 2, 3, 'a', 'b')
>>> print(t1)
( 1, 2, 3, 'a', 'b' )

# convert string to tuple
>>> t = tuple('lupins')
>>> print(t)
('l', 'u', 'p', 'i', 'n', 's')
```

Tuples are immutable

- 튜플 원소(element)의 값 변경 불가
 - 튜플 내 아이템을 추가, 변경, 삭제 불가
 - 튜플을 다른 것으로 대체하여 변경 가능

```
# modify the element
```

```
>>> t = ('a', 'b', 'c', 'd', 'e')
```

```
>>> print(t[0])
```

```
'a'
```

```
>>> t[0] = 'A'
```

```
TypeError: object doesn't support item assignment
```

```
# replace one tuple with another
```

```
>>> t = ('A',) + t[1:]
```

```
>>> print(t)
```

```
('A', 'b', 'c', 'd', 'e')
```

Lists and tuples

- 정렬 등을 해야 할 때 튜플은 리스트로 바꾸어 사용
- **list**(tuplename) → list로 변환

```
>>> t = ( 11, 3, 15, 7, 9)
>>> l = list(t)
>>> l
[ 11, 3, 15, 7, 9 ]

>>> l.sort()
>>> l
[ 3, 7, 9, 11, 15 ]
```

zips a string and a list

- **zip**은 두 개 이상의 스트링, 리스트 등을 받아서 리스트로 "묶는" 내장된 함수
 - index가 같은 것끼리 한 덩어리로 묶어줌
 - 리스트를 생성함

```
>>> s = 'abc'
>>> t = [0, 1, 2]
>>> list( zip(s, t) )
[('a', 0), ('b', 1), ('c', 2)]
```

2개 문자열의 개수가 다른 경우에는, 작은 개수 기준으로 생성

```
>>> list( zip('Anne', 'Elk') )
[('A', 'E'), ('n', 'l'), ('n', 'k')]
```

연습문제 1

- 다음 함수를 실행하여 돌려 받은 값은 리스트이다
- 돌려 받은 리스트를 튜플로 바꾼 후 문자열 "Python"과 쌍을 구성하는 리스트로 바꾸시오

```
import random

def select_item(list, n) :
    result=random.sample(list, n)
    return result

r=select_item([1,3,5,7,11,15,21], 6)
print(r)
```


연습문제 1-코드

```
import random

def select_item(list, n) :
    result=random.sample(list, n)
    return result

r=select_item([1,3,5,7,11,15,21], 6)
print(r)

t = tuple(r)
print(t)

t1 = list(zip(r, "Python"))
print(t1)
```

```
☞ [21, 15, 11, 7, 3, 1]
   (21, 15, 11, 7, 3, 1)
   [(21, 'P'), (15, 'y'), (11, 't'), (7, 'h'), (3, 'o'), (1, 'n')]
```

연습문제 2

- 랜덤 사다리 타기
- 인원과 전체 인원의 이름을 입력받는다
- 각 사람에게 'o' 또는 'x'가 매칭되도록 한다
- 결과 리스트는 튜플로 구성되며 다음과 같은 형식이다.
[('지혜', 'x'), ('민지', 'o'), ('광현', 'o')]

연습문제 2-코드 및 결과

```
import random

namelist= []
oxlist= []

count = int(input("인원을 입력하세요: "))

for i in range (count) :
    name = input("이름을 입력하세요: ")
    namelist.append(name)

for i in range (count) :
    oxlist.append(random.choice(['o', 'x']))

tname = tuple(namelist)
print(tname)

t = list(zip(tname, oxlist))
print(t)
```

```
➡ 인원 을 입력 하세요 : 4
이름 을 입력 하세요 : 재환
이름 을 입력 하세요 : 길동
이름 을 입력 하세요 : 민현
이름 을 입력 하세요 : 은지
('재환', '길동', '민현', '은지')
[('재환', 'o'), ('길동', 'o'), ('민현', 'o'), ('은지', 'x')]
```

연습문제 3

다음과 같은 문자열과 리스트가 존재한다

- `s = 'abcdefghijklmnopqrstuvwxyz'`
- `n = [0,1,2,3,4,5,6,7,8,9,10,11,12,13,14,15,16,17,18,19,20,21, 22,23,24,25]`
- 서로 다른 2개의 자료형을 일대일로 묶어서, 튜플로 구성된 리스트를 생성 하시오

연습문제 3-코드

```
#s = list(string.ascii_lowercase) # import string 필요
#s = list(map(chr, range(ord('a'), ord('z')+1)))
s = [chr(i) for i in range(ord('a'), ord('z')+1)]
n = range(0,25+1)

sn = list(zip(s, n))
print(sn)
```

```
[('a', 0), ('b', 1), ('c', 2), ('d', 3), ('e', 4), ('f', 5), ('g', 6), ('h', 7), ('i', 8), ('j', 9), ('k', 10),
```

연습문제 4

암호화

- `s = 'abcdefghijklmnopqrstuvwxyz'`
- `base_s = []`
- `random.sample(s, 26)` 사용하여 순서를 바꾸어 알파벳 26개를 `base_s`에 저장한다
- `zip()` 함수를 사용하여, `s`와 `base_s`로 튜플로 이루어진 리스트를 생성한다
- 사용자에게 문자열을 입력 받는다
- 각 문자를 튜플의 쌍으로 존재하는 다른 문자로 바꿔서 출력한다

연습문제 4-코드

```
import random

s = list(map(chr, range(ord('a'), ord('z')+1)))
base_s = random.sample(s, 26)
password = [ ]
password_hint = list(zip(s, base_s))
print(password_hint)

input_str = input("문자열 입력: ")
for letter in input_str:
    for i in range(26):
        if letter == password_hint[i][0]:
            password.append(password_hint[i][1])

print(password)
```

```
▶ [('a', 'w'), ('b', 'g'), ('c', 'b'), ('d', 'c'), ('e', 'e'), ('f', 'a'), ('g', 'm'), ('h', 'k'), ('i', 'p'), ('j', 'u'),
문자열 입력: hongkildong
['k', 'v', 'z', 'm', 'n', 'p', 'y', 'c', 'v', 'z', 'm']
```

강의 요약

- 튜플이 무엇인지 알기
 - 값들의 나열(sequence)
- 리스트와 튜플의 차이점을 이해하기
 - 리스트: 변경 가능
 - 튜플: 변경 불가능
- 여러 명령어 사용해 보기
 - list() : 튜플을 리스트로 바꾸기
 - zip() : 두개 이상의 스트링, 리스트 등을 받아서 리스트로 묶기

퀴즈 1

튜플과 리스트의 차이점은 무엇인가?

- ① 리스트는 정수로 색인(index)되어 있지만, 튜플은 문자열로 색인된다
- ② 리스트는 서로 다른 데이터형 혼합하여 저장 가능하지만 튜플은 그렇지 않다
- ③ 튜플은 아이템을 변경 불가능하지만, 리스트는 변경 가능하다
- ④ 함수에서 여러 개의 값을 리턴 하면 자료 변경이 불가능한 튜플로 생성된다

퀴즈 1-답안

튜플과 리스트의 차이점은 무엇인가?

- ① 리스트는 정수로 색인(index)되어 있지만, 튜플은 문자열로 색인된다
- ② 리스트는 서로 다른 데이터형 혼합하여 저장 가능하지만 튜플은 그렇지 않다
- ③ 튜플은 **아이템을 변경 불가능하지만, 리스트는 변경 가능하다**
- ④ 함수에서 여러 개의 값을 리턴 하면 자료 변경이 불가능한 튜플로 생성된다

퀴즈 2

다음 코드의 실행 결과는?

```
>>> list( zip('Anne', [1, 2, 3]) )
```

- ① [('A', 1), ('n', 2), ('n', 3), ('e', 0)]
- ② [('A', 1), ('n', 2), ('n', 3)]
- ③ [('A', 1], ['n', 2], ['n', 3], ['e', 0])
- ④ [('A', 1], ['n', 2], ['n', 3])

퀴즈 2-답안

다음 코드의 실행 결과는?

```
>>> list( zip('Anne', [1, 2, 3]) )
```

```
↳ [('A', 1), ('n', 2), ('n', 3)]
```

- ① [('A', 1), ('n', 2), ('n', 3), ('e', 0)]
- ② [('A', 1), ('n', 2), ('n', 3)]
- ③ [('A', 1], ['n', 2], ['n', 3], ['e', 0])
- ④ [('A', 1], ['n', 2], ['n', 3])

08. 딕셔너리의 이해

[3차시]

학습목표

- 딕셔너리 이해하기
- 딕셔너리 정의하기, 읽기, 자료 추가 및 삭제 연습
- 딕셔너리 메소드 `items()`, `get()`, `keys()` 기능 이해하기

사전형(Dictionary) 이란?

- 사전은 리스트와 비슷하지만
 - 리스트에서는 index 값이 정수여야 함
 - 사전에서는 모든 종류의 데이터 형을 index로 사용 가능 (user-defined indexes)
 - 정의할 때, '{'와 '}'를 사용

사전형 정의하기

- 정의하기

```
>>> days = {'Sun':'Sunday', 'Mon':'Monday', 'Tue':'Tuesday',  
'Wed':'Wednesday', 'Thu':'Thursday', 'Fri':'Friday', 'Sat':'Saturday'}
```

- 첫번째 기술한 아이템이 검색키(index)가 된다.

```
>>> days['Sun']  
'Sunday'  
>>> days['Fri']  
'Friday'
```


사전형 생성 후 자료 추가

- 비어있는 사전 생성

```
>>> d = { }
```

- 함수 dict는 항목(item)없는 새로운 사전을 생성한다
- 생성 후, 다음과 같이 아이템을 한 개씩 추가 한다

```
# create dictionary
>>> eng2sp = dict()
>>> print(eng2sp)
{}
>>> eng2sp['one'] = 'uno'
>>> print(eng2sp)
{'one': 'uno'}
```

사전형 연산

operator	Description
len()	사전형 변수에 저장된 아이템의 개수를 알려준다
k in dictionary_name	k 가 해당 사전형에 존재하면 True, 존재하지 않으면 False
dictionary_name[k]	k 가 해당 사전형에 존재하면, 해당 아이템 값을 출력한다
dictionary_name[k] = "item 값"	k 가 해당 사전형에 존재하지 않으면, 해당 아이템 index, 값이 추가되어 저장된다
dictionary_name.pop(k)	k 가 해당 사전형에 존재하면, 삭제한다

사전형 연산, 개수와 아이템 확인

- 국가명과 국제전화 코드를 사전형으로 정의

```
>>> country_code = {1:"미국", 20:"이집트", 30:"그리스", 39:"이태리", 81:"일본", 82:"한국 "}
>>> len(country_code)
6

>>> country_code[30]
"그리스"

>>> country_code[82]
"한국"
```

사전형 연산, 자료 추가와 삭제

```
>>> 82 in country_code  
True
```

```
>>> 60 in country_code  
False
```

```
>>> country_code[60] = "말레지아"  
>>> 60 in country_code  
True
```

append an item

```
>>> country_code.pop(81)  
'일본'  
>>> 81 in country_code  
False
```

delete an item

사전형 연산, 자료 읽기

```
# 사전형 city 값을 읽어내기
```

```
>>> city = {"New York City":8175133, "Los Angeles": 3792621, "Washington":632323,  
"Chicago": 2695598, "Toronto":2615060, "Montreal":11854442, "Ottawa":883391,  
"Boston":62600}
```

```
>>> city["Toronto"]
```

```
2615060
```

```
>>> city["Boston"]
```

```
62600
```

```
# 사전형 food 아이템 추가하기
```

```
>>> food = {"ham" : "yes", "egg" : "yes", "spam" : "no" }
```

```
>>> food
```

```
{'egg': 'yes', 'ham': 'yes', 'spam': 'no'}
```

```
>>> food["spam"] = "yes"
```

```
>>> food
```

```
{'egg': 'yes', 'ham': 'yes', 'spam': 'yes'}
```

연습문제 1

- 아이템이 없는 사전형 birthdate를 정의한다
- 사용자에게 이름과 생일을 입력 받아서 사전형 birthdate에 추가한다
- 5명의 이름과 생일을 입력 받아서 추가한 후, 추가된 내용을 화면에 출력한다
- 특정한 한 사람의 이름을 입력 받아서, 생일을 화면에 출력한다

연습문제 1-코드

```
birthdate = { }  
def add_birth(num):  
    for  
        i in range(num):  
            input_name = input("이름: ")  
            input_birth = int(input("생일: "))  
            birthdate[input_name] = input_birth  
  
add_birth(5)  
print(birthdate)  
  
name=input("생일을 찾고 싶은 사람의 이름을 입력하세요: ")  
print(birthdate[name])
```

```
이름: jslee  
생일: 0902  
이름: yjkim  
생일: 1022  
이름: dshong  
생일: 1209  
이름: ygpark  
생일: 0125  
이름: kdchoi  
생일: 0812  
{'jslee': 902, 'yjkim': 1022, 'dshong': 1209, 'ygpark': 125, 'kdchoi': 812}  
생일을 찾고 싶은 사람의 이름을 입력하세요 : yjkim  
1022
```

연습문제 2

- 연습문제 1에서 생성한 dictionary 사용한다
- 입력 받은 사람의 자료를 삭제한다
- 삭제 후 in을 사용하여 삭제 여부를 확인한다

연습문제 2-코드

```
birthdate = { }
def add_birth(num):
    for i in range(num):
        input_name = input("이름: ")
        input_birth = int(input("생일: "))
        birthdate[input_name] = input_birth
```

```
add_birth(5)
print(birthdate)
```

```
name=input("생일을 삭제하고 싶은 사람의 이름을 입력하세요: ")
birthdate.pop(name)
print(name in birthdate)
print(birthdate)
```

```
이름 : jslee
생일 : 0902
이름 : yjkim
생일 : 1022
이름 : dshong
생일 : 1209
이름 : ygpark
생일 : 0125
이름 : kdchoi
생일 : 0812
{'jslee': 902, 'yjkim': 1022, 'dshong': 1209, 'ygpark': 125, 'kdchoi': 812}
생일을 삭제하고 싶은 사람의 이름을 입력하세요: yjkim
False
{'jslee': 902, 'dshong': 1209, 'ygpark': 125, 'kdchoi': 812}
```

Dictionary Method

Method	Description
d.items	사전형 d을 index-값 형식의 튜플로 구성된 리스트로 생성
d.get(k)	d[k] 와 같은 결과로 index k에 해당하는 값을 보여준다
d.keys(k)	사전형 d에서 index값만 찾아서 보여준다
d.pop(k)	사전형 d에서 index k인 아이템을 삭제한다
d.values(k)	사전형 d에서 index값을 제외하고, 값들만 찾아서 보여준다
d.update(d2)	사전형 d의 모든 저장된 내용에, 사전형 d2의 내용이 추가된다
dict(listname)	index-값 형식의 튜플로 구성된 리스트를 dictionary로 변환한다, d.items와 반대의 기능을 수행

Dictionary Methods, items

- **items**라는 메소드
 - index-값 형식의 튜플로 구성된 리스트를 생성

```
>>> d = {'a':0, 'b':1, 'c':2}          # create dictionary
>>> t = d.items()                     # create a list contains tuples
>>> t
[('a', 0), ('c', 2), ('b', 1)]
```

Dictionary Methods, get(k), keys()

.get(k) .keys()

```
>>> sp={'one':'uno', 'two':'dos', 'three':'tres'}  
>>> sp.get('two')  
'dos'  
>>> sp['one']  
'uno'  
>>> sp.keys()  
dict_keys(['one', 'two', 'three'])
```

연습문제 3

- 저장된 인물의 정보가 어떤 것이 있는지 출력한다
- 알고 싶은 정보를 입력받아 해당 정보를 출력한다
- 다음과 같은 딕셔너리를 생성한다
 - `info = {'이름': '홍길동', '나이': 21, '주소': '서울시 동작구 상도동', '전공': '컴퓨터공학'}`

연습문제 3-코드

```
info = {'이름': '홍길동', '나이': 21, '주소': '서울시 동작구 상도동', '전공': '컴퓨터공학'}  
print(info['이름'], "님에 대한 다음의 정보가 저장되어 있습니다. ")  
print(info.keys())  
  
key = input("알고 싶은 정보를 입력하세요: ")  
print(info.get(key))
```

☞ 홍길동 님에 대한 다음의 정보가 저장되어 있습니다.
dict_keys(['이름', '나이', '주소', '전공'])
알고 싶은 정보를 입력하세요: 나이
21

강의 요약

딕셔너리 이해하기

- 중괄호 사용하여 정의
- 모든 종류의 데이터형을 index로 사용 가능
- 딕셔너리 정의하기
- 읽기, 자료 추가 및 삭제 연습

딕셔너리 메소드 기능 이해하기

- items(): index-값 형식의 튜플로 구성된 리스트 생성
- get(): 해당 index의 값 반환
- keys(): 전체 dictionary의 key값 반환

퀴즈 1

다음 코드의 실행 결과는?

```
>>> country_code = {1:"미국", 20:"이집트", 30:"그리스", 39:"이태리", 81:"일본",  
82:"한국"}  
>>> country_code.pop(81)  
>>> 81 in country_code
```

- ① "일본"
- ② 1
- ③ True
- ④ False

퀴즈 1-답안

다음 코드의 실행 결과는?

```
>>> country_code = {1:"미국", 20:"이집트", 30:"그리스", 39:"이태리", 81:"일본",  
82:"한국"}  
>>> country_code.pop(81)  
>>> 81 in country_code
```

- ① "일본"
- ② 1
- ③ True
- ④ False

False

퀴즈 2

- 다음 코드의 실행 결과는?

```
>>> sp={'one':'uno', 'two':'dos', 'three':'tres'}  
>>> sp.keys()
```

- ① dict_values(['uno', 'dos', 'tres'])
- ② dict_keys(['one', 'two', 'three'])
- ③ {'one':'uno', 'two':'dos', 'three':'tres'}
- ④ 답이 없음

퀴즈 2-답안

- 다음 코드의 실행 결과는?

```
>>> sp={'one':'uno', 'two':'dos', 'three':'tres'}  
>>> sp.keys()
```

- ① dict_values(['uno', 'dos', 'tres'])
- ② dict_keys(['one', 'two', 'three'])
- ③ {'one':'uno', 'two':'dos', 'three':'tres'}
- ④ 답이 없음

```
dict_keys(['one', 'two', 'three'])
```

09. 리스트/튜플/딕셔너리의 활용

[3차시]

학습목표

- 딕셔너리에서 사용하는 메소드 활용하기
- 딕셔너리에서 여러 개의 값을 지정하고 활용하기
- 리스트로 구성된 딕셔너리를 활용하기

Dictionary Method

Method	Description
d.items	사전형 d을 index-값 형식의 튜플로 구성된 리스트로 생성
d.get(k)	d[k] 와 같은 결과로 index k에 해당하는 값을 보여준다
d.keys(k)	사전형 d에서 index값만 찾아서 보여준다
d.pop(k)	사전형 d에서 index k인 아이템을 삭제한다
d.values(k)	사전형 d에서 index값을 제외하고, 값들만 찾아서 보여준다
d.update(d2)	사전형 d의 모든 저장된 내용에, 사전형 d2의 내용이 추가된다
dict(listname)	index-값 형식의 튜플로 구성된 리스트를 dictionary로 변환한다, d.items와 반대의 기능을 수행

Dictionary Methods, pop(k), values()

- .pop(k) .values()

```
>>> sp={'one':'uno', 'two':'dos', 'three':'tres'}
>>> sp.get('two')
'dos'
>>> sp['one']
'uno'
>>> sp.keys()
dict_keys(['one', 'two', 'three'])
>>> sp.pop('one')
'uno'
>>> sp
{'two': 'dos', 'three': 'tres'}
>>> sp.values()
dict_values(['dos', 'tres'])
```

Dictionary Methods, update(d2)

다른 딕셔너리의 내용을 추가한다

```
>>> sp={'one':'uno', 'two':'dos', 'three':'tres'}
>>> birthdate={'kmkim': 224, 'sjkang': 512, 'kdhong': 1212, 'ychoi': 409, 'hjkim': 1103}

>>> sp.update(birthdate)

>>> sp
{'one': 'uno', 'two': 'dos', 'three': 'tres', 'kmkim': 224, 'sjkang': 512, 'kdhong': 1212, 'ychoi': 409, 'hjkim': 1103}

>>> birthdate
{'kmkim': 224, 'sjkang': 512, 'kdhong': 1212, 'ychoi': 409, 'hjkim': 1103}
```


Dictionary Methods, dict

dict 메소드

- index-값 형식의 튜플로 구성된 리스트를 dictionary로 변환

```
# 튜플로 구성된 리스트를 딕셔너리로 바꾸는 dict
>>> t = [('a', 0), ('c', 2), ('b', 1)]
>>> d = dict(t)
>>> d
{'a': 0, 'c': 2, 'b': 1}

>>> d = dict(zip('abc', range(3)))
>>> d
{'a': 0, 'b': 1, 'c': 2}
```

Dictionary 다수 값 정의

Index는 하나이지만, 해당하는 값을 리스트로 구성하여 저장되는 사전형

```
>>> sp={'one':['uno', 1], 'two':['dos',2], 'three':['tres',3]}
>>> sp['one']
['uno', 1]

>>> sp.values()
dict_values(['uno', 1], ['dos', 2], ['tres', 3])

>>> sp.keys()
dict_keys(['one', 'two', 'three'])

>>> sp
{'one': ['uno', 1], 'two': ['dos', 2], 'three': ['tres', 3]}
```

연습문제 1

- 친구 이름과 전화번호로 구성된 사전형 phone을 구성한다
- 아이템 추가로 5명의 자료를 입력, 이 때 입력은 함수로 구성한다
- 입력된 자료의 index만 모두 출력한다
- 입력된 자료의 값들만 모두 출력한다

연습문제 1-코드

```
phone = {}

def add_phonenumber(n):
    for i in range(n):
        print(i+1)
        index = input('이름: ')
        phonenumber = int(input('전화번호: '))
        phone[index] = phonenumber
```

```
add_phonenumber(5)
```

```
print(phone.keys())
print(phone.values())
```

```
1
이름: jslee
전화번호: 01056457445
2
이름: yjkim
전화번호: 01034561575
3
이름: ygpark
전화번호: 01064548744
4
이름: dshong
전화번호: 01064443777
5
이름: kdchoi
전화번호: 01089743656
dict_keys(['jslee', 'yjkim', 'ygpark', 'dshong', 'kdchoi'])
dict_values([1056457445, 1034561575, 1064548744, 1064443777, 1089743656])
```

연습문제 2

- 교사명과 담당교과목명을 사전형으로 구성한다
 - '홍길동' : '수학', '과학'
 - '최영희' : '영어', '수학'
 - '강동원' : '영어'
 - '정필수' : '사회', '역사'
 - '박희수' : '국어'
 - '이승철' : '수학', '과학'
- 교사명을 입력하면 담당교과목명을 출력한다

연습문제 2-코드

```
Subjects = {'홍길동':['수학','과학'],'최영희':['영어','수학'],  
            '강동원':['영어','정필수':['사회','역사'],  
            '박희수':['국어','이승철':['수학','과학']]}
```

```
lec = input('담당교사명을 입력하시오: ')  
print(Subjects[lec])
```

☞ 담당교사명을 입력하시오: 최영희
['영어', '수학']

연습문제 3

- 연습문제 3과 같은 딕셔너리를 사용한다
- 교사명과 담당교과목명을 사전형으로 구성한다
 - '홍길동' : '수학', '과학'
 - '최영희' : '영어', '수학'
 - '강동원' : '영어'
 - '정필수' : '사회', '역사'
 - '박희수' : '국어'
 - '이승철' : '수학', '과학'
- 담당교과목명을 입력하면 교사명을 출력한다
- 담당교과목을 여러명의 교사가 강의하는 경우, 모든 교사명을 다 출력한다

연습문제 3 코드

```
def find_teacher(sub):
    Subjects = {'홍길동':['수학','과학'],'최영희':['영어','수학'],
                '강동원':['영어','정필수':['사회','역사'],
                '박희수':['국어','이승철':['수학','과학']]

    teacher = []
    for key in Subjects.keys() :
        if sub in Subjects[key] :
            teacher.append(key)

    return teacher

sub = input('과목을 입력하시오: ')
print(find_teacher(sub))
```

➡ 과목을 입력하시오: 수학
['홍길동', '최영희', '이승철']

강의 요약

- 딕셔너리에서 사용하는 메소드 활용하기
 - .pop(), .values(), .update(), dict()
- 딕셔너리에서 여러 개의 값을 지정하고 활용하기
 - index는 하나이지만, 해당 값을 리스트로 구성하여 저장하는 사전형
- 리스트로 구성된 딕셔너리를 활용하기

10. 시각화의 이해

[3차시]

학습목표

- Matplotlib 라이브러리 활용하기
- 한글폰트 설치 및 깨짐 방지
- 다양한 그래프 표현 활용하기

Matplotlib 라이브러리

- 파이썬 표준 시각화 도구
- 다양한 그래프와 옵션 지원
- Pandas와 연동이 용이

참고 : <https://matplotlib.org/stable/gallery/index.html>

환경 설정 (1/2)

- 버전확인

```
import matplotlib  
  
matplotlib.__version__
```

→ '3.2.2'

- 그림 선명하게 표시

```
%config InlineBackend.figure_format = 'retina'
```

- 인라인으로 결과 표시

```
%matplotlib inline
```

환경 설정 (2/2)

• 한글 폰트 설치(재시작 필요)

```
# 네이버 나눔 폰트
```

```
!sudo apt-get install -y fonts-nanum
```

```
!sudo fc-cache -fv
```

```
!rm ~/.cache/matplotlib -rf
```

```
Reading package lists... Done
Building dependency tree
Reading state information... Done
The following NEW packages will be installed:
  fonts-nanum
0 upgraded, 1 newly installed, 0 to remove and 39 not upgraded.
Need to get 9,604 kB of archives.
After this operation, 29.5 MB of additional disk space will be used.
Get:1 http://archive.ubuntu.com/ubuntu bionic/universe amd64 fonts-nanum all 20170925-1 [9,604 kB]
Fetched 9,604 kB in 0s (43.1 MB/s)
debconf: unable to initialize frontend: Dialog
```

• 한글 깨짐 방지(폰트 변경)

```
import matplotlib.pyplot as plt
```

```
# 폰트 나눔고딕으로 지정
```

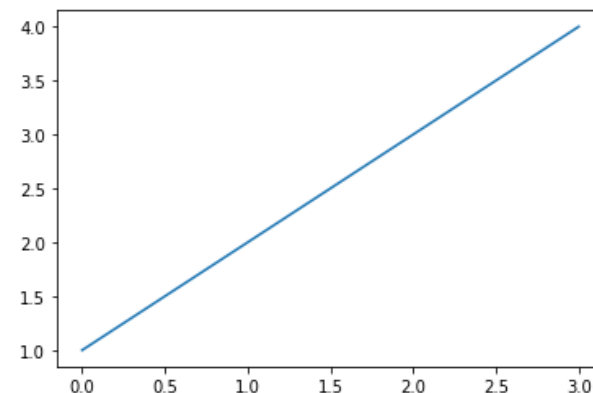
```
plt.rc('font', family='NanumBarunGothic')
```

기본 그래프 (1/5)

• 기본 그래프(y값만 지정)

```
import matplotlib.pyplot as plt
```

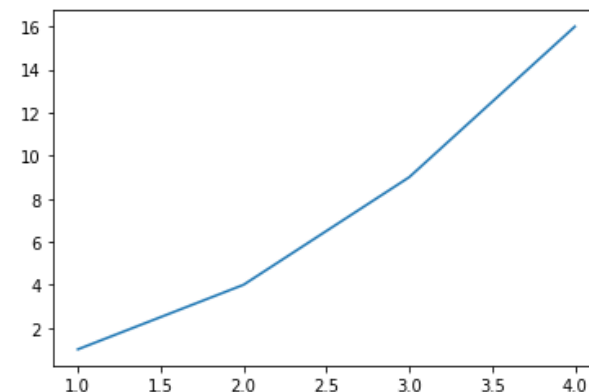
```
plt.plot([1, 2, 3, 4]) # y값  
plt.show()
```



• 기본 그래프(x-y값 지정)

```
import matplotlib.pyplot as plt
```

```
plt.plot([1, 2, 3, 4], [1, 4, 9, 16]) # x-y값  
plt.show()
```

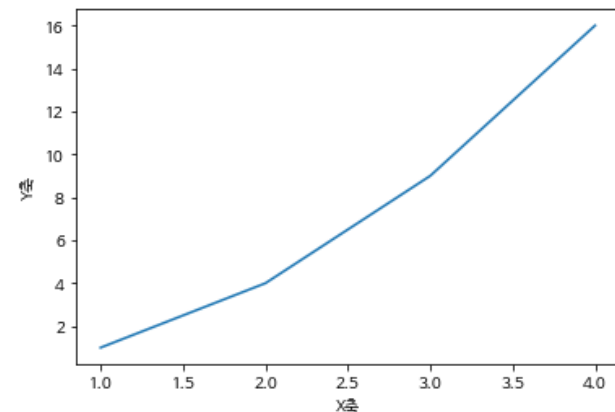


기본 그래프 (2/5)

• 레이블 지정

```
import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 4], [1, 4, 9, 16])
plt.xlabel('X축')
plt.ylabel('Y축')
plt.show()
```

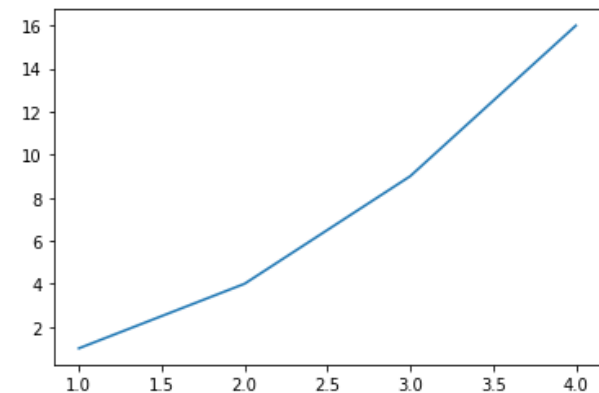


• 범례 표시

```
import matplotlib.pyplot as plt

plt.plot([1, 2, 3, 4], [2, 3, 5, 10], label='가격($)')
plt.xlabel('X축')
plt.ylabel('Y축')
plt.legend()

plt.show()
```

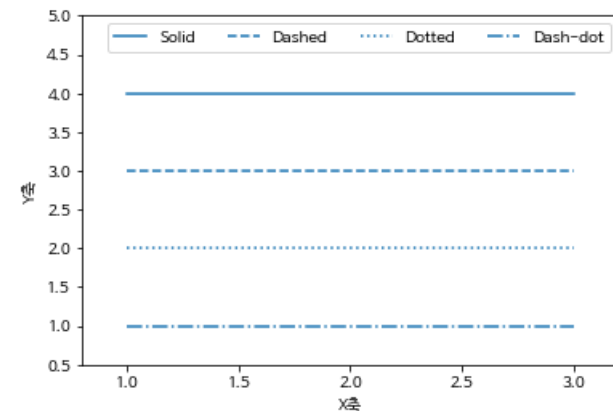


기본 그래프 (3/5)

• 선종류 지정

```
import matplotlib.pyplot as plt
```

```
plt.plot([1, 2, 3], [4, 4, 4], '-', color='C0', label='Solid')
plt.plot([1, 2, 3], [3, 3, 3], '--', color='C0', label='Dashed')
plt.plot([1, 2, 3], [2, 2, 2], ':', color='C0', label='Dotted')
plt.plot([1, 2, 3], [1, 1, 1], '-.', color='C0', label='Dash-dot')
plt.xlabel('X축')
plt.ylabel('Y축')
plt.axis([0.8, 3.2, 0.5, 5.0])
plt.legend(loc='upper right', ncol=4)
plt.show()
```



——— Solid
plt.plot(x, y, '-')

- - - - - Dashed
plt.plot(x, y, '--')

..... Dotted
plt.plot(x, y, ':')

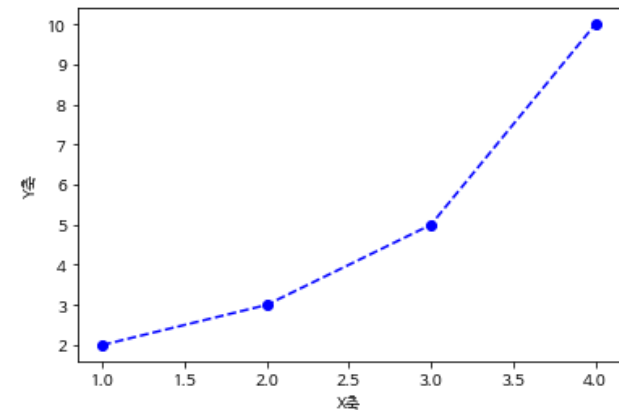
- . - . - . Dash-dot
plt.plot(x, y, '-.')

기본 그래프 (4/5)

• 마커 지정

```
import matplotlib.pyplot as plt
```

```
# plt.plot([1, 2, 3, 4], [2, 3, 5, 10], 'bo-') # 파란색 + 마커 + 실선  
plt.plot([1, 2, 3, 4], [2, 3, 5, 10], 'bo--') # 파란색 + 마커 + 점선  
plt.xlabel('X축')  
plt.ylabel('Y축')  
plt.show()
```



색상, 선 종류, 마커

Colors

character	color
'b'	blue
'g'	green
'r'	red
'c'	cyan
'm'	magenta
'y'	yellow
'k'	black
'w'	white

Line Styles

character	description
'-'	solid line style
'--'	dashed line style
'-.'	dash-dot line style
':'	dotted line style

Markers

character	description
'.'	point marker
','	pixel marker
'o'	circle marker
'v'	triangle_down marker
'^'	triangle_up marker
'<'	triangle_left marker
'>'	triangle_right marker
'1'	tri_down marker
'2'	tri_up marker
'3'	tri_left marker
'4'	tri_right marker
's'	square marker
'p'	pentagon marker
'*'	star marker
'h'	hexagon1 marker
'H'	hexagon2 marker
'+'	plus marker
'x'	x marker
'D'	diamond marker
'd'	thin_diamond marker
' '	vline marker
'_'	hline marker

기본 그래프 (5/5)

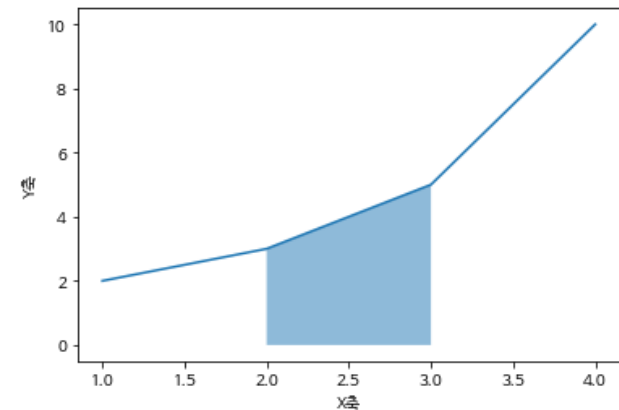
• 그래프 영역 채우기

```
import matplotlib.pyplot as plt

x = [1, 2, 3, 4]
y = [2, 3, 5, 10]

plt.plot(x, y)
plt.xlabel('X축')
plt.ylabel('Y축')
plt.fill_between(x[1:3], y[1:3], alpha=0.5)

plt.show()
```



array → transparency

```
plt.fill_between(x[1:3], y[1:3], alpha=0.5)
```

막대 그래프

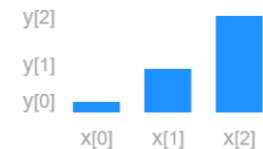
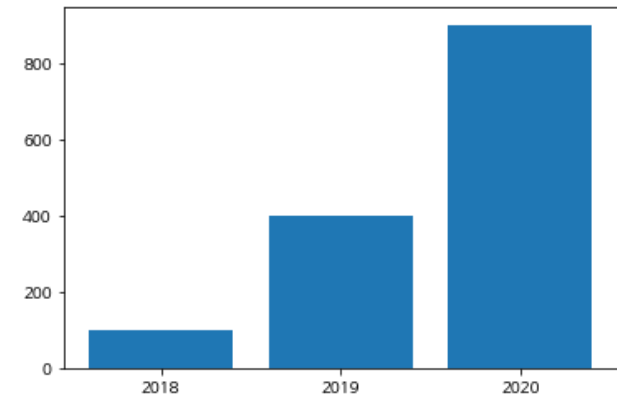
• 막대 그래프 / y축 눈금 레이블

```
import matplotlib.pyplot as plt  
import numpy as np
```

```
x = np.arange(3)  
years = ['2018', '2019', '2020']  
values = [100, 400, 900]
```

```
plt.bar(x, values)  
plt.xticks(x, years)
```

```
plt.show()
```



```
plt.bar(x, y)
```

산점도 그래프

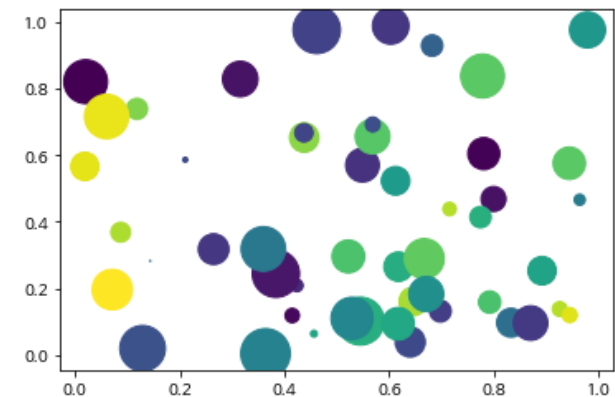
• 산점도 그래프 / 크기 / 색상

```
import matplotlib.pyplot as plt
import numpy as np

np.random.seed(0)

n = 50
x = np.random.rand(n)
y = np.random.rand(n)
area = (30 * np.random.rand(n))**2
colors = np.random.rand(n)

plt.scatter(x, y, s=area, c=colors)
plt.show()
```



size color
↓ ↓
plt.scatter(x, y, s=area, c=colors)

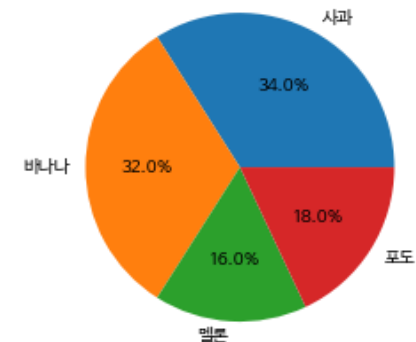
파이 차트

• 파이 차트 / 부채꼴 스타일

```
import matplotlib.pyplot as plt

ratio = [34, 32, 16, 18]
labels = ['사과', '바나나', '멜론', '포도']

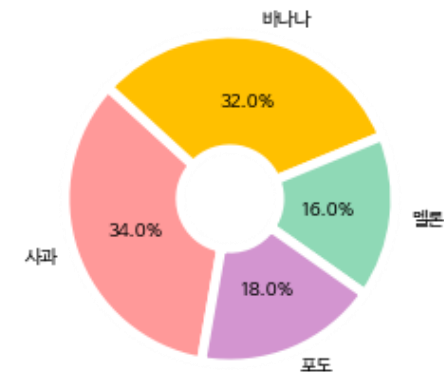
plt.pie(ratio, labels=labels, autopct='%1f%%')
plt.show()
```



```
import matplotlib.pyplot as plt

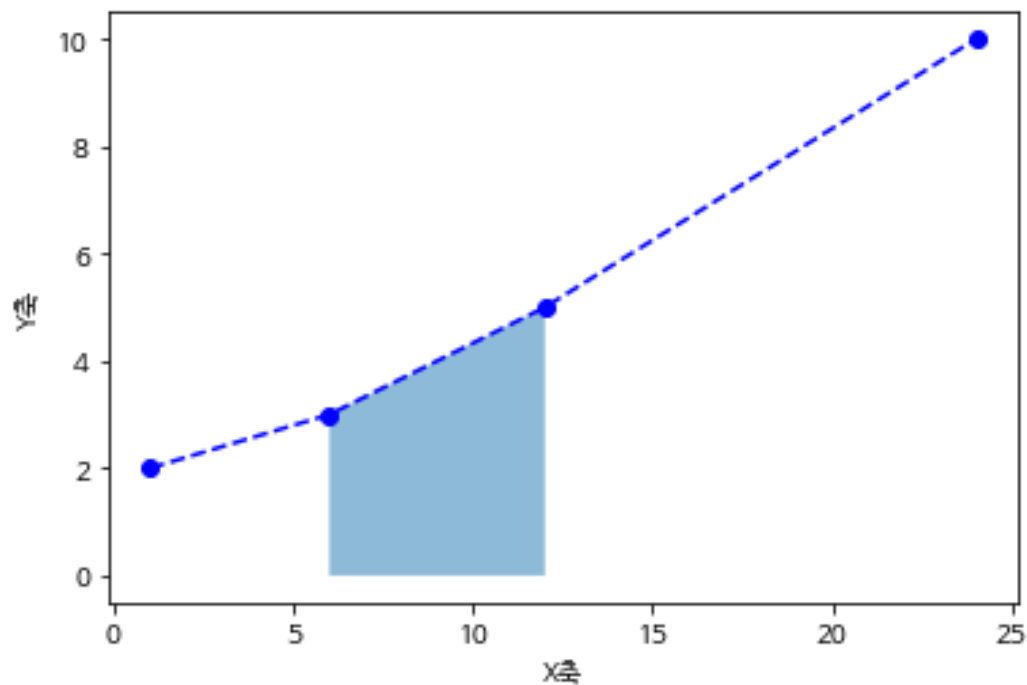
ratio = [34, 32, 16, 18]
labels = ['사과', '바나나', '멜론', '포도']
colors = ['#ff9999', '#ffc000', '#8fd9b6', '#d395d0']
wedgeprops = {'width': 0.7, 'edgecolor': 'w', 'linewidth': 5}

plt.pie(ratio, labels=labels, autopct='%1f%%', startangle=260,
        counterclock=False, colors=colors, wedgeprops=wedgeprops)
plt.show()
```



연습문제 1

- 아래와 같은 그래프를 그려본다.



연습문제 1-코드

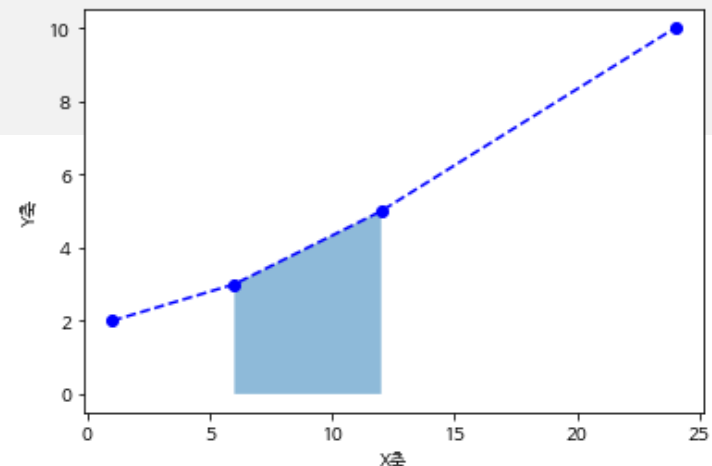
```
import matplotlib.pyplot as plt

x = [1, 6, 12, 24]
y = [2, 3, 5, 10]

plt.plot(x, y, 'bo--', label='가격($))

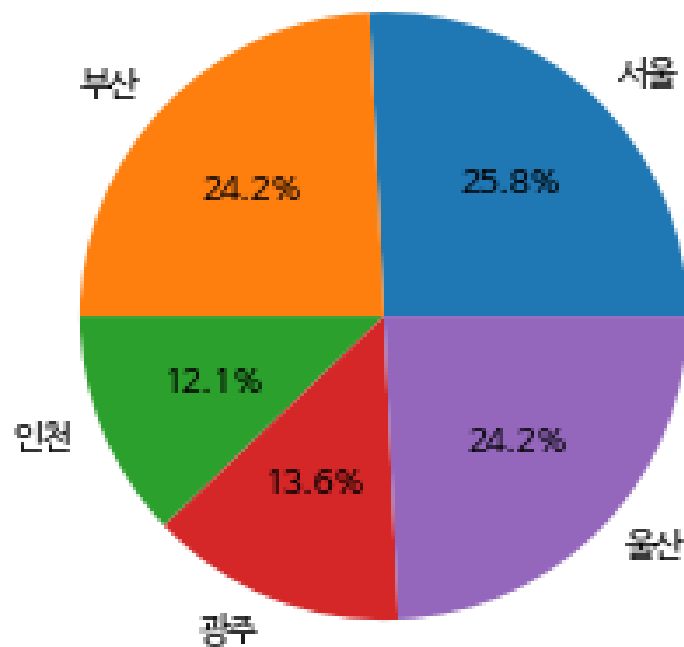
plt.xlabel('X축')
plt.ylabel('Y축')
plt.fill_between(x[1:3], y[1:3], alpha=0.5)

plt.show()
```



연습문제 2

- 아래와 같은 그래프를 그려본다.

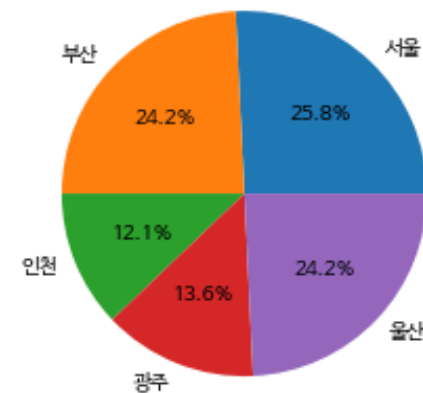


연습문제 2-코드

```
import matplotlib.pyplot as plt

ratio = [34, 32, 16, 18, 32]
labels = ['서울', '부산', '인천', '광주', '울산']

plt.pie(ratio, labels=labels, autopct='%0.1f%%')
plt.show()
```



강의 요약

- 시각화 라이브러리(matplotlib) 활용하기
- 환경설정하기
 - 버전 확인, 선명하게 표시, 한글 폰트 설치, 깨짐 방지
- 기본 그래프 그리기
 - x-y값, 레이블, 범례, 선종류, 마커, 영역 채우기
- 다양한 그래프 그리기
 - 막대 그래프, 산점도 그래프, 파이 차트

감사합니다
[3차시]