내장함수

■ 내장함수

- 파이썬 내부에 미리 정의되어 있는 함수
- 외부모듈과는 달리 import 과정없이 사용 가능

  ex) print(), type(), len(), list(), map(), set(), ...
- __builtins__ 속성을 통해서 내장함수 목록 확인 가능

```
dir(__builtins__)
```

```
>>> dir(__builtins__)
['ArithmeticError', 'AssertionError', 'AttributeError', 'BaseException', 'Blocki
ngIOError', 'BrokenPipeError', 'BufferError', 'BytesWarning', 'ChildProcessError
', 'ConnectionAbortedError', 'ConnectionError', 'ConnectionRefusedError', 'Conne
ctionResetError', 'DeprecationWarning', 'EOFError', 'Ellipsis', 'EnvironmentErro
r', 'Exception', 'False', 'FileExistsError', 'FileNotFoundError', 'FloatingPoint
Error', 'FutureWarning', 'GeneratorExit', 'IOError', 'ImportError', 'ImportWarni
ng', 'IndentationError', 'IndexError', 'InterruptedError', 'IsADirectoryError',
'KeyError', 'KeyboardInterrupt', 'LookupError', 'MemoryError', 'ModuleNotFoundEr
ror', 'NameError', 'None', 'NotADirectoryError', 'NotImplemented', 'NotImplement
edError', 'OSError', 'OverflowError', 'PendingDeprecationWarning', 'PermissionEr
ror', 'ProcessLookupError', 'RecursionError', 'ReferenceError', 'ResourceWarning
', 'RuntimeError', 'RuntimeWarning', 'StopAsyncIteration', 'StopIteration', 'Syn
taxError', 'SyntaxWarning', 'SystemError', 'SystemExit', 'TabError', 'TimeoutErr
or', 'True', 'TypeError', 'UnboundLocalError', 'UnicodeDecodeError', 'UnicodeEnc
odeError', 'UnicodeError', 'UnicodeTranslateError', 'UnicodeWarning', 'UserWarni
ng', 'ValueError', 'Warning', 'WindowsError', 'ZeroDivisionError', '__build_clas
s__', '__debug__', '__doc__', '__import__', '__loader__', '__name__', '__package
__', '__spec__', 'abs', 'all', 'any', 'ascii', 'bin', 'bool', 'bytearray', 'byte
s', 'callable', 'chr', 'classmethod', 'compile', 'complex', 'copyright', 'credit
s', 'delattr', 'dict', 'dir', 'divmod', 'enumerate', 'eval', 'exec', 'exit', 'fi
lter', 'float', 'format', 'frozenset', 'getattr', 'globals', 'hasattr', 'hash',
'help', 'hex', 'id', 'input', 'int', 'isinstance', 'issubclass', 'iter', 'len',
'license', 'list', 'locals', 'map', 'max', 'memoryview', 'min', 'next', 'object'
, 'oct', 'open', 'ord', 'pow', 'print', 'property', 'quit', 'range', 'repr', 're
versed', 'round', 'set', 'setattr', 'slice', 'sorted', 'staticmethod', 'str', 's
um', 'super', 'tuple', 'type', 'vars', 'zip']
```

■ 내장함수

● 자주 사용되는 내장함수 목록

abs()	all()	any()	chr()	dir()
divmod()	**enumerate()**	eval()	**filter()**	hex()
id()	input()	**int()**	isinstance()	**len()**
list()	**map()**	max()	min()	oct()
open()	ord()	pow()	**range()**	sorted()
str()	tuple()	**type()**	**zip()**	

■ 내장함수

● abs()

- 숫자의 절대값으로 돌려주는 함수

```
print(abs(3))
print(abs(-3.14))
print(abs(-99999))
```

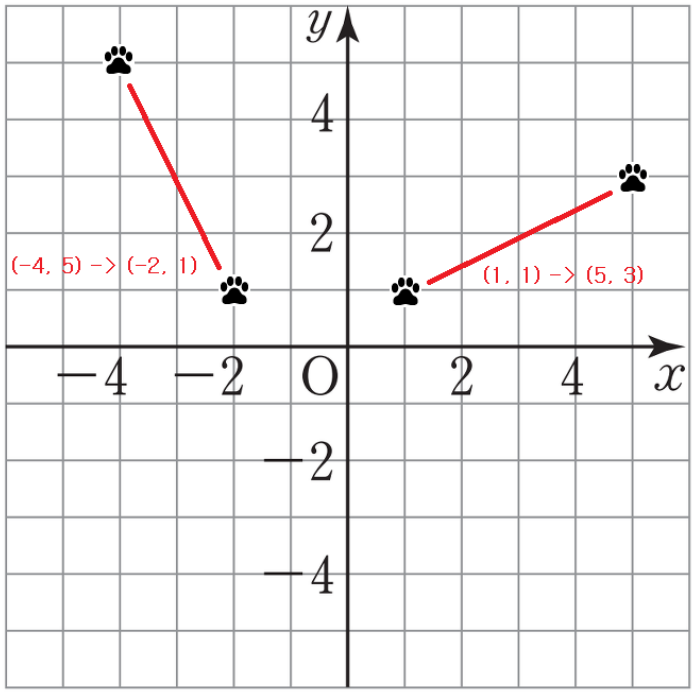
● all()

- 반복 가능한(iterable) 자료를 입력받아서 요소가 모두 참이면 True,
  거짓인 요소가 하나라도 있으면 False를 돌려주는 함수

```
print(all([1, 2, 3, 4]))
print(all([1, 2, 0]))
print(all(['a', 'b', ' ']))
print(all({'a', 'b', ''}))
```

■ 연습문제

● 그래프와 같이 이동한 거리 dx, dy 를 구할 수 있는 distance() 함수 작성



(-4, 5) → (-2, 1)
dx → 2, dy → 4

(1, 1) → (5, 3)
dx → 4, dy → 2

```
def distance(start, end):
    # 코드 작성

start = (1, 1)
end = (5, 3)
dx, dy = distance(start, end)
print('dx:', dx, 'dy:', dy)
```

```
dx: 4 dy: 2
```

■ 연습문제

● 모든 문제에 대해 답이 작성되었는지 확인하는 코드 작성

```
print('컴파일 언어가 아닌 것은?')
answer1 = input('1. C 2. C# 3. Python 4. swift => ')

print('컴퓨터 장치가 아닌 것은?')
answer2 = input('1. HDD 2. SDD 3. RAM 4. CPU => ')

print('파이썬의 자료형이 아닌 것은?')
answer3 = input('1. str 2. tuple 3. list 4. Map => ')

answer_list = list()
answer_list.append(answer1)
answer_list.append(answer2)
answer_list.append(answer3)

# 코드 작성
```

```
컴파일 언어가 아닌 것은?
1. C  2. C#  3. Python  4. swift => 3

컴퓨터 장치가 아닌 것은?
1. HDD  2. SDD  3. RAM  4. CPU => 2

파이썬의 자료형이 아닌 것은?
1. str  2. tuple  3. list  4. Map => 4

제출 완료
```

```
컴파일 언어가 아닌 것은?
1. C  2. C#  3. Python  4. swift => 3

컴퓨터 장치가 아닌 것은?
1. HDD  2. SDD  3. RAM  4. CPU => 2

파이썬의 자료형이 아닌 것은?
1. str  2. tuple  3. list  4. Map =>

재입력
```

■ 내장함수

● any()

- 반복 가능한(iterable) 자료를 입력받아서 요소가 참인 요소가 하나라도 있으면 True, 모두 거짓이면 False를 돌려주는 함수

```
print(any([1, 2, 3, 4]))
print(any([1, 2, 0]))
print(any(('a', 'b', '')))
print(any([0, '', False, [], {}, ()]))
```

● chr()

- 아스키(ASCII)코드를 입력받아 해당하는 문자를 돌려주는 함수

```
print(chr(48))
print(chr(65))
print(chr(97))
print(chr(122))
```

코드	문자
…	…
48	0
49	1
50	2
…	…
65	A
66	B
67	C
…	…
97	a
98	b
99	c
…	…

■ 내장함수

● dir()

- 객체가 가지고 있는 변수와 함수를 돌려주는 함수

```
print(dir([1, 2, 3]))
```

```
['__add__', '__class__', '__class_getitem__', '__contains__',
'__delattr__', '__delitem__', '__dir__', '__doc__', '__eq__',
'__format__', '__ge__', '__getattribute__', '__getitem__', '__gt__',
'__hash__', '__iadd__', '__imul__', '__init__', '__init_subclass__',
'__iter__', '__le__', '__len__', '__lt__', '__mul__', '__ne__',
'__new__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__',
'__rmul__', '__setattr__', '__setitem__', '__sizeof__', '__str__',
'__subclasshook__', 'append', 'clear', 'copy', 'count', 'extend',
'index', 'insert', 'pop', 'remove', 'reverse', 'sort']
```

- 별도의 클래스 생성 변수, 함수 내용 확인

```
class MyClass:
    age = 10
    name = 'ggoreb'
    def myFunc():
        return age

print(dir(MyClass))
```

```
['__class__', '__delattr__', '__dict__', '__dir__', '__doc__', '__eq__',
'__format__', '__ge__', '__getattribute__', '__gt__', '__hash__',
'__init__', '__init_subclass__', '__le__', '__lt__', '__module__',
'__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__',
'__setattr__', '__sizeof__', '__str__', '__subclasshook__',
'__weakref__', 'age', 'myFunc', 'name']
```

■ 내장함수

● divmod()

- 2개의 숫자를 입력받은 후 나누기 결과 몫과 나머지를 튜플로 돌려주는 함수

```
print(divmod(5, 3))
print(divmod(-10, 2))
print(divmod(5.4, 2))
```

● enumerate()

- 반복 가능한(iterable) 자료를 입력받아서 인덱스와 요소를 돌려주는 함수

```
for i, e in enumerate([1, 2, 3]):
    print(i, e)
for i, e in enumerate({1: 'a', 2: 'b', 3: 'c'}):
    print(i, e)
for i, e in enumerate('문자열'):
    print(i, e)
```

■ 연습문제

● a를 b로 나눈 몫과 나머지 구하기

```
a = 123
b = 16

# 코드 작성

print('몫', quotient)
print('나머지', remaind
```

```
몫 7
나머지 11
```

■ 연습문제

● 리스트를 딕셔너리로 변경하기

```
char_list = ['a', 'b', 'c', 'd', 'e', 'f', 'g']
char_dict = {}

# 코드 작성

print(char_dict)
```

```
{1: 'a', 2: 'b', 3: 'c', 4: 'd', 5: 'e', 6: 'f', 7: 'g'}
```

■ 내장함수

● eval()

- 실행 가능한 문자열을 입력받은 후 실행한 결과값을 돌려주는 함수

```
print(eval('1 + 2'))
eval('print("Hello", "Python")')
print(eval('any([1, 2, 3, 4])'))
```

● filter()

- 참, 거짓을 반환하는 함수와 반복 가능한(iterable) 자료를 입력받은 후 참인 값만 찾은 후 돌려주는 함수

```
# filter()
def even(n):
    return n % 2 == 0

print(list(filter(even, (1, 2, 3, 4, 5, 6))))
print(tuple(filter(even, (1, 2, 3, 4, 5, 6))))
print(set(filter(even, (1, 2, 3, 4, 5, 6))))
print(list(filter(lambda n: n % 2 == 0, (1, 2, 3, 4, 5, 6))))
```

■ 연습문제

● filter, lambda, 함수 등을 이용하여 리스트 요소 중 음수를 제거한 리스트 만들기

```
num_list = [1, -2, 3, -5, 8, -3]

# 코드 작성
```

```
[1, 3, 8]
```

● filter, lambda, 함수 등을 이용하여 리스트 요소 중 3의 배수를 제거한 리스트 만들기

```
num_list = range(1, 21)

# 코드 작성
```

```
[1, 2, 4, 5, 7, 8, 10, 11, 13, 14, 16, 17, 19, 20]
```

■ 내장함수

● hex()

- 10진수인 정수를 입력받은 후 16진수 정수로 돌려주는 함수

```
print(hex(5))
print(hex(10))
print(hex(15))
print(hex(16))
```

```
0x5
0xa
0xf
0x10
```

● id()

- 객체를 입력받은 후 객체의 고유 주소값을 돌려주는 함수

```
a = [1, 2, 3], b = [1, 2, 3]
print(id(a), id(b))

b = a
print(id(a), id(b))

c = ('a', 'b'), d = ('a', 'b')
print(id(c), id(d))

d = c
print(id(c), id(d))
```

```
1825340579776 1825340568384
1825340579776 1825340579776
1825340572416 1825340583296
1825340572416 1825340572416
```

■ 연습문제

● 색상표를 통해 확인된 숫자를 16진수로 표현

```
red = 190
green = 66
blue = 205

# 코드 작성

print(r, g, b)
```


```
0xbe 0x42 0xcd
```

■ 내장함수

● input()

- 사용자 입력을 받은 후 문자열로 돌려주는 함수

```
text1 = input()
print(text1)

text2 = input('value : ')
print(text2)
```

● int()

- 문자열 형태의 숫자, 소수점이 있는 숫자를 정수 형태로 돌려주는 함수

```
print(int('2324'))   # 문자열 → 정수
print(int(2.324))    # 실수 → 정수
print(int('11', 2))  # 2진수 → 10진수
print(int('11', 8))  # 8진수 → 10진수
print(int('11', 16)) # 16진수 → 10진수
print(int('ff', 16)) # 16진수 → 10진수
```

■ 연습문제

● 주어진 16진수 문자를 이용하여 실행결과와 같이 출력될 수 있도록 show_hex_to_ch() 함수 작성 (split(), chr(), int() 등 이용)

```
hexa_string1 = '48 45 4C 4C 4F'
hexa_string2 = '47 47 4F 52 45 42'

def show_hex_to_ch(hexa_string):

    # 코드 작성

    print()

show_hex_to_ch(hexa_string1)
show_hex_to_ch(hexa_string2)
```

```
HELLO
GGOREB
```

ASCII TABLE

Decimal	Hex	Char	Decimal	Hex	Char
32	20	[SPACE]	64	40	@
33	21	!	65	41	A
34	22	"	66	42	B
35	23	#	67	43	C
36	24	$	68	44	D
37	25	%	69	45	E
38	26	&	70	46	F
39	27	'	71	47	G
40	28	(	72	48	H
41	29	)	73	49	I
42	2A	*	74	4A	J
43	2B	+	75	4B	K
44	2C	,	76	4C	L
45	2D	-	77	4D	M
46	2E	.	78	4E	N
47	2F	/	79	4F	O
48	30	0	80	50	P
49	31	1	81	51	Q
50	32	2	82	52	R
51	33	3	83	53	S
52	34	4	84	54	T
53	35	5	85	55	U
54	36	6	86	56	V
55	37	7	87	57	W
56	38	8	88	58	X
57	39	9	89	59	Y
58	3A	:	90	5A	Z
59	3B	;	91	5B	[
60	3C	<	92	5C	\
61	3D	=	93	5D	]
62	3E	>	94	5E	^
63	3F	?	95	5F	_

■ 내장함수

● isinstance()

- 입력받은 변수와 클래스가 같은 인스턴스인지 확인해주는 함수

```
class Car: pass
a = Car()
print(isinstance(a, Car))

class Taxi(Car): pass
b = Taxi()
print(isinstance(b, Taxi))
print(isinstance(b, Car))

print(isinstance(10, int))
print(isinstance(10.0, float))
```

● len()

- 입력된 값의 길이(전체 요소의 개수)를 돌려주는 함수

```
print(len('python'))
print(len([1, 2, 3, 4, 5]))
print(len({1, 2, 3, 4, 5}))
```

■ 연습문제

● values에 저장되어 있는 요소 중 모든 정수(int)와 실수(float)의 합 구하기

(단, True 제외 - bool 자료형은 int 자료형을 상속받으므로 int에 포함)

```
values = [10, 'a', True, {}, [], -4, 23.24, (1,), -1.2]
result = 0

# 코드 작성

print(result) # 28.04
```

■ 내장함수

● list()

- 반복 가능한 자료형을 입력받아 리스트로 만들어 주는 함수

```
print(list('python'))
print(list({1, 2, 3, 4, 5}))
print(list({'a':1, 'b':2, 'c':3}))
print(list((1, 2, 3, 4, 5)))
```

● map()

- 입력된 값에 연산결과를 반영하여 반환하는 함수와 반복 가능한(iterable) 자료를 입력받은 후 함수 결과값들을 묶어서 돌려주는 함수

```
def two_times(x): return x * 2

print(list(map(two_times, [1, 2, 3, 4])))
print(list(map(lambda x: x * 2, [1, 2, 3, 4])))
```

■ 연습문제

● map, lambda, 함수 등을 이용하여 리스트 연산 후 결과와 같이 출력하기

- 요소 * 3

```
num_list = [1, 2, 3, 4]

# 코드 작성
```

```
[3, 6, 9, 12]
```

● map, lambda, 함수 등을 이용하여 리스트 연산 후 결과와 같이 출력하기

- 요소 ** 요소

```
num_list = [1, 2, 3, 4]

# 코드 작성
```

```
[1, 4, 27, 256]
```

■ 내장함수

● max()

- 반복 가능한 자료형을 입력받아 요소 중 최대값을 돌려주는 함수

```
print(max((1, 3, 1000, 9999, -1)))
print(max({'a':1, 'b':3, 'c':1}))
print(max('Python'))
print(max('pYthon'))
```

● min()

- 반복 가능한 자료형을 입력받아 요소 중 최소값을 돌려주는 함수

```
print(min((1, 3, 1000, 9999, -1)))
print(min({'a':1, 'b':3, 'c':1}))
print(min('Python'))
print(min('pYthon'))
```

■ 연습문제

● 리스트의 요소 중 최대값과 최소값 구하기

```
num_list = [-8, 2, 7, 5, -3, 5, 0, 1]

# 코드 작성
```

```
최대값 7
최소값 -8
```

■ 내장함수

● oct()

- 10진수인 정수를 입력받은 후 8진수 정수로 돌려주는 함수

```
print(oct(5))
print(oct(8))
print(oct(16))
print(oct(18))
```

● open()

- 입력된 mode에 따라 파일객체를 돌려주는 함수

```
file = open('test.txt', 'r', encoding='utf8')
print(file.read())
file.close()
```

■ 내장함수

● ord()

- 문자를 입력받아 해당하는 아스키(ASCII)코드를 돌려주는 함수 (↔ chr())

```
print(ord('0'))
print(ord('A'))
print(ord('a'))
print(ord('z'))
```

● pow()

- 2개의 x, y 숫자를 입력받은 후 x의 y제곱 결과를 돌려주는 함수

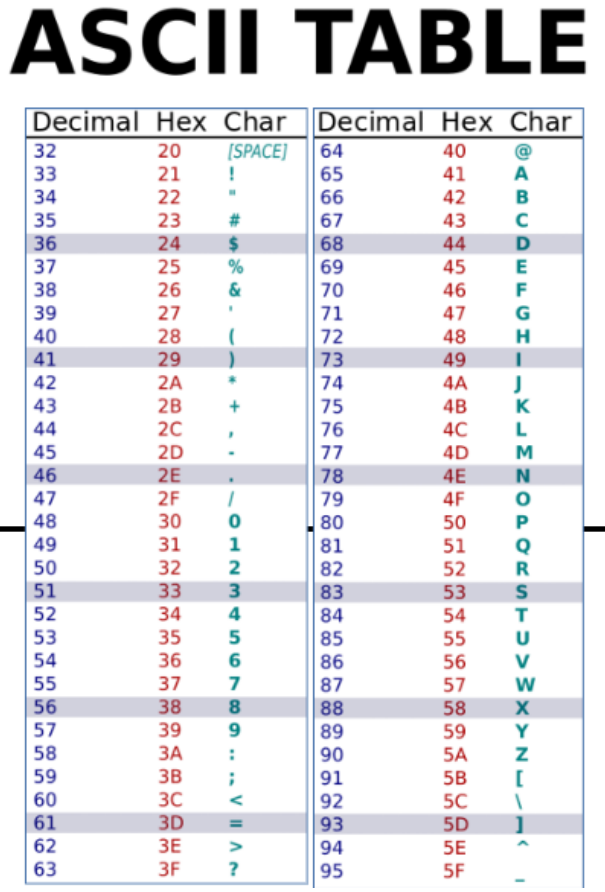
```
print(pow(2, 2))
print(pow(2, 10)) # 1KB (≒ 1,000)
print(pow(2, 20)) # 1MB (≒ 1,000,000)
print(pow(2, 30)) # 1GB (≒ 1,000,000,000)
```

■ 연습문제

● 주어진 문자열을 이용하여 실행결과와 같이 출력될 수 있도록 show_ch_to_ascii() 함수 작성

```
string1 = 'ASCII TABLE'
string2 = 'PYTHON'

def show_ch_to_ascii(string):

    # 코드 작성

    print()

show_ch_to_ascii(string1)
show_ch_to_ascii(string2)
```

```
65 83 67 73 73 32 84 65 66 76 69
80 89 84 72 79 78
```



■ 내장함수

● range()

- 주로 for문과 연계하여 사용되며, 입력받은 숫자에 해당하는 범위를 반복 가능한 객체로 만들어서 돌려주는 함수

```
print(list(range(5)))             # 마지막 숫자 제외
print(list(range(1, 6)))          # 마지막 숫자 제외
print(list(range(1, 11, 2)))      # 숫자 사이의 거리
print(list(range(0, -11, -2)))    # 숫자 사이의 거리
```

● sorted()

- 반복 가능한 자료형을 입력받아 오름차순 정렬된 리스트를 돌려주는 함수

```
print(sorted(('z', 'y', 'r', 'a', 'b')))
print(sorted((3, 5, 4, 9, -5)))
print(sorted({'1', 'a', '2', 'A', '가'}))
```

```
['a', 'b', 'r', 'y', 'z']
[-5, 3, 4, 5, 9]
['1', '2', 'A', 'a', '가']
```

■ 내장함수

● sorted()

- Dictionary 적용

```
d = { 'a': 10, 'c': 5, 'b': 12, 'd': 1 }
print(sorted(d))
print(sorted(d.items()))
```

```
['a', 'b', 'c', 'd']
[('a', 10), ('b', 12), ('c', 5), ('d', 1)]
```

- Dictionary value 정렬

```
d = { 'a': 10, 'c': 5, 'b': 12, 'd': 1 }

print(sorted(d.items(), key=lambda x: x[1]))
print(sorted(d.items(), key=lambda x: x[1], reverse=True))
```

```
[('d', 1), ('c', 5), ('a', 10), ('b', 12)]
[('b', 12), ('a', 10), ('c', 5), ('d', 1)]
```

■ 내장함수

● str()

- 입력받은 값을 문자열 형태로 돌려주는 함수

```
print(str(1))
print(str('123'))
print(str([1, 2, 3]))
print(str({1:1, 2:2, 3:3}))
```

● tuple()

- 반복 가능한 자료형을 입력받아 튜플로 만들어 주는 함수

```
print(tuple([1, 2, 3, 4, 5]))
print(tuple('12345'))
print(tuple({1, 2, 'a', 'b', 'c', '가', '나', '다'}))
```

■ 내장함수

● type()

- 입력받은 값의 자료형을 알려주는 함수

```
print(type('python'))
print(type(10.10))
print(type({}))
print(type(()))
print(type([]))
```

● zip()

- 반복 가능한 자료형 묶음별 요소들을 각 순번에 맞춰 튜플로 묶어진 리스트를 돌려주는 함수

```
print(list(zip([1, 2, 3], [4, 5, 6])))
print(list(zip([1, 2, 3], [4, 5, 6], [7, 8, 9])))
print(list(zip([1, 2, 3], [4, 5, 6], [7, 8])))
```

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

■ 연습문제

● 3개의 리스트 각 요소를 이용하여 튜플 리스트로 변환하기

```
list1 = [1, 2, 3, 4]
list2 = ['a', 'b', 'c', 'd']
list3 = ['가', '나', '다', '라']

# 코드 작성
```

```
[('가', 1, 'a'), ('나', 2, 'b'), ('다', 3, 'c'), ('라', 4, 'd')]
```

■ 연습문제

● 주어진 데이터를 결과와 같은 형식의 데이터로 저장하기

데이터

```
fish1_length = [
    25.4, 26.3, 26.5, 29.0, 29.0, 29.7, 29.7, 30.0, 30.0, 30.7,
    31.0, 31.0, 31.5, 32.0, 32.0, 32.0, 33.0, 33.0, 33.5, 33.5,
    34.0, 34.0, 34.5, 35.0, 35.0, 35.0, 35.0, 36.0, 36.0, 37.0,
    38.5, 38.5, 39.5, 41.0, 41.0
]
fish1_weight = [
    242.0, 290.0, 340.0, 363.0, 430.0, 450.0, 500.0, 390.0, 450.0,
    500.0, 475.0, 500.0, 500.0, 340.0, 600.0, 600.0, 700.0, 700.0,
    610.0, 650.0, 575.0, 685.0, 620.0, 680.0, 700.0, 725.0, 720.0,
    714.0, 850.0, 1000.0, 920.0, 955.0, 925.0, 975.0, 950.0
]
fish2_length = [
    9.8, 10.5, 10.6, 11.0, 11.2, 11.3, 11.8, 11.8, 12.0, 12.2,
    12.4, 13.0, 14.3, 15.0
]
fish2_weight = [
    6.7, 7.5, 7.0, 9.7, 9.8, 8.7, 10.0, 9.9, 9.8, 12.2,
    13.4, 12.2, 19.7, 19.9
]
```

결과