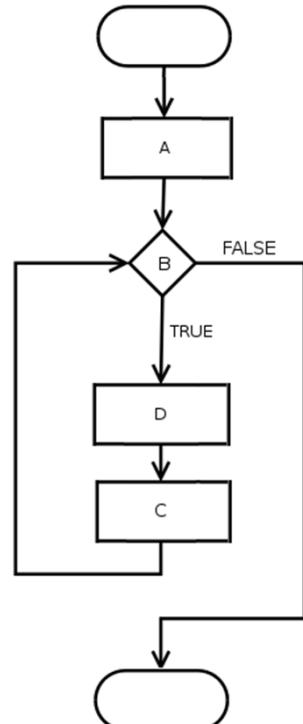


# 자료구조 및 알고리즘

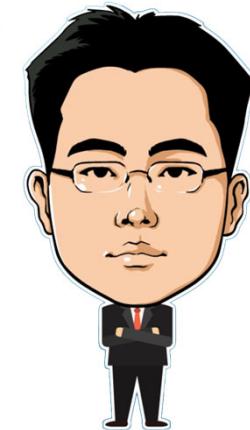
```
for(A;B;C)  
D;
```



## 파이썬 프로그래밍 (Python Programming)

Seo, Doo-Ok

Clickseo.com  
[clickseo@gmail.com](mailto:clickseo@gmail.com)



# 목 차



- 파이썬 프로그래밍 기초

백문이불여일타(百聞而不如一打)

- 자료형과 자료구조

- 객체지향 프로그래밍

- 파이썬 라이브러리



# 파이썬 언어 개요 (1/2)

## ● 파이썬 프로그래밍 언어: [python.org](http://python.org)

### ○ 파이썬 소프트웨어 재단(PSF, Python Software Foundation)

- 1991년 02월, **귀도 반 로섬(Guido van Rossum)**이 발표
  - 플랫폼 독립적이며 객체지향적 인터프리터, 동적 타이핑(dynamically typed) 대화형 언어
- **자유-오픈소스 소프트웨어(FOSS)**의 좋은 예

### ○ Python 릴리즈

- 2008년 12월, **Python 3.0** >> 2023년 08월, Python 3.11.5
- 2000년 10월, **Python 2.0** >> 2020년 04월, Python 2.7.18
  - 2020년 01월 01일부로 Python 2의 지원이 종료되었다.



### ○ 라이선스: PSFL(Python Software Foundation License)

- **GPL과 호환되는**  
**BSD 스타일의 자유 소프트웨어 라이선스**

### ○ 프로그래밍 언어: C, Python

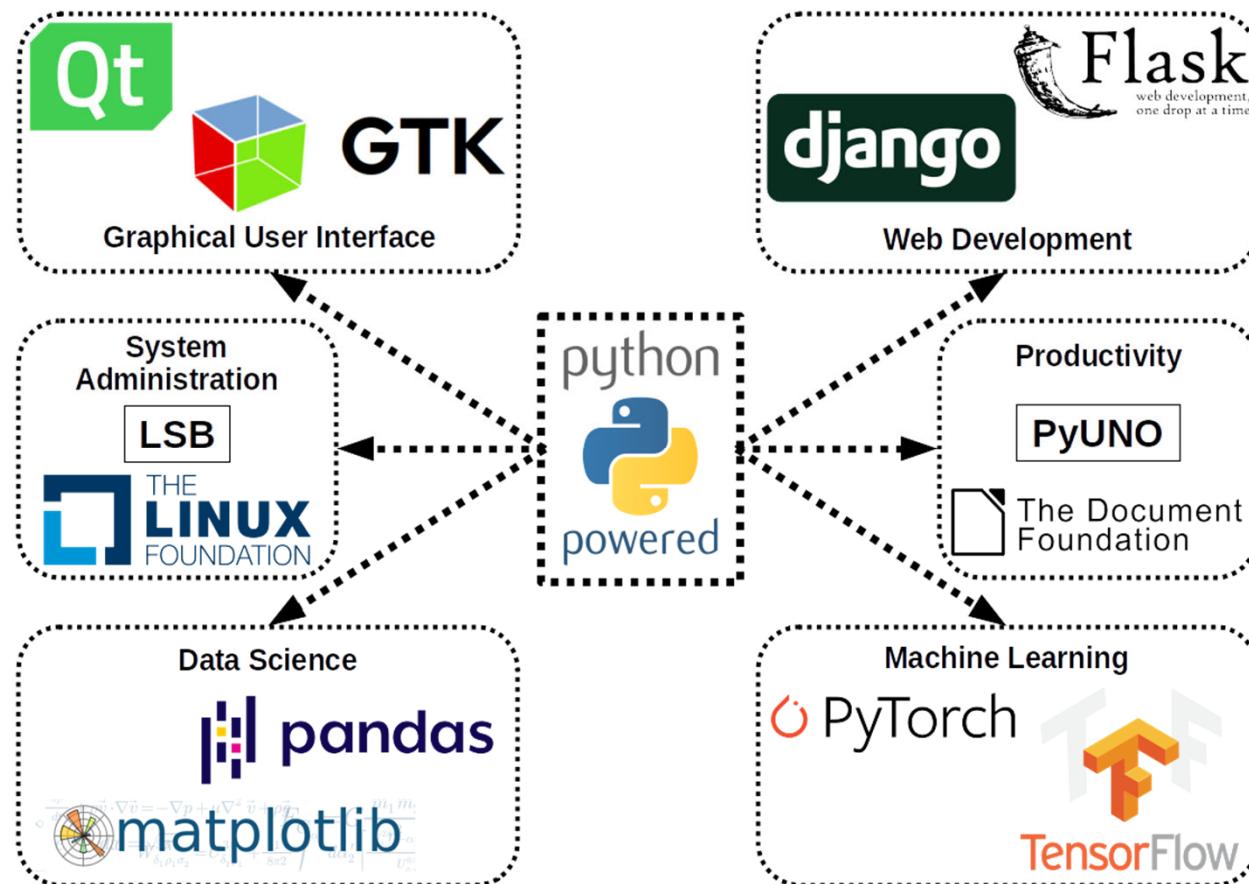
Python Repository(GitHub): [github.com](https://github.com)

<https://github.com/python/cpython>



# 파이썬 언어 개요 (2/2)

## ● 파이썬 언어의 다양한 활용



[ 출처: "Python(programming language)", Wikipedia. ]

# 파이썬 프로그래밍 기초



- 파이썬 프로그래밍 기초

- 파이썬 언어 개요
- 수식과 연산자
- 제어 흐름
- 함수

백문이불여일타(百聞而不如一打)

- 자료형과 자료구조

- 객체지향 프로그래밍

- 파이썬 라이브러리





# 파이썬 프로그래밍 기초

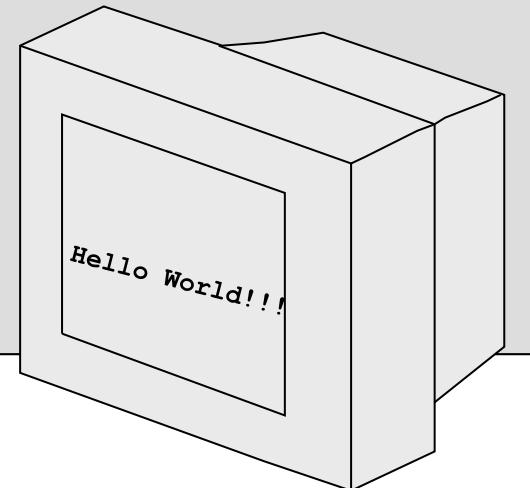
파이썬 언어 개요  
: 데이터 표현(변수와 상수, 자료형)



# 파이썬 언어 개요

## ● 파이썬 프로그램 구조

```
# 첫 번째 프로그램  
# 프로그램: Hello World  
# 작성자 : 서두옥(Clickseo) ^..^  
# 작성일 : 0000년 00월 00일  
# hello.py  
  
print('Hello World!!!!')  
print("Hello World!!!!")
```



```
# 파이썬 내장 변수: __name__  
if __name__ == '__main__' :  
    print('Hello World!!!!')  
    print("Hello World!!!!")
```



python

# 변수와 상수 (1/4)

## ● 변수(Variabes)

○ 파이썬 프로그래밍 언어는 동적(Dynamic) 타입 언어이다.

- 자료형을 선언하지 않아도 실행시간에 자동적으로 자료형이 정해진다.
  - C/C++와 Java와 같은 정적(Static) 타입 언어들과 비교하면 쓰기가 간결하다.

```
a = 10          # 정수형  
a = 10.5        # 실수형  
a = 'Hi~ Clickseo'    # 문자열
```

```
# 여러 개의 변수에 값을 대입  
a, b = 10, 20
```

```
# 두 변수의 값을 간단히 바꿀 수 있다.  
a = 10  
b = 20  
a, b = b, a
```

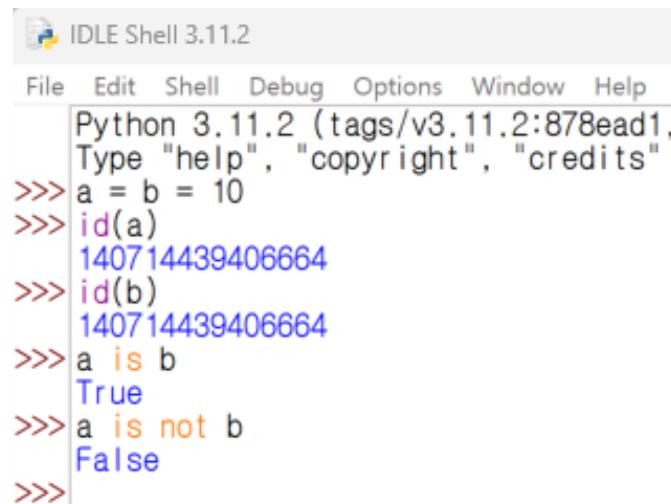
# 변수와 상수 (2/4)

## ● 변수(Variable)

○ 파이썬 언어에서 사용하는 변수는 객체를 가리키는 것.

```
# 여러 개의 변수에 값을 대입
```

```
a = b = 10
```



```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1,
Type "help", "copyright", "credits"
>>> a = b = 10
>>> id(a)
140714439406664
>>> id(b)
140714439406664
>>> a is b
True
>>> a is not b
False
>>>
```

- **is, is not 연산자:** 객체 비교 연산자
- **id 내장 함수:** 변수가 가리키고 있는 객체의 고유 주소 값을 반환하는 함수
  - `id(object)`

# 변수와 상수 (3/4)

## 예제 0-1: 변수 -- 동적 타입 언어

```
a = 10          # 정수형  
print(a)
```

```
a = 10.5       # 실수형  
print(a)
```

```
a = 'Python'   # 문자열  
print(a)
```

```
# 동일한 객체를 가리키는 변수: a, b  
a = b = 50  
print(a)  
print(b)
```

```
# id 내장 함수: 변수가 가리키고 있는 객체의 주소 값을 반환하는 함수  
print(id(a))  
print(id(b))
```

```
# 객체 비교 연산자: is, is not  
print(a is b)  
print(a is not b)
```

```
IDLE Shell 3.11.2  
File Edit Shell Debug Options Window Help  
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  
Type "help", "copyright", "credits" or "li  
==== RESTART: C:\Users\click\w  
10  
10.5  
Python  
50  
50  
140714439407944  
140714439407944  
True  
False  
>>>
```

# 변수와 상수 (4/4)

---

## ● 상수(Constants)

### ○ 내장 상수(Built-in Constants)

- **True** : 논리형(bool)의 참 값
- **False** : 논리형(bool)의 거짓 값
- **None** : 없음을 나타내는데 자주 사용되는 객체
  - **None**은 **NoneType** 유형의 유일한 인스턴스이다.
- **NotImplemented**
- **Ellipsis**
- **\_debug\_** : Python이 -O 옵션으로 시작되지 않은 경우에 참이다.

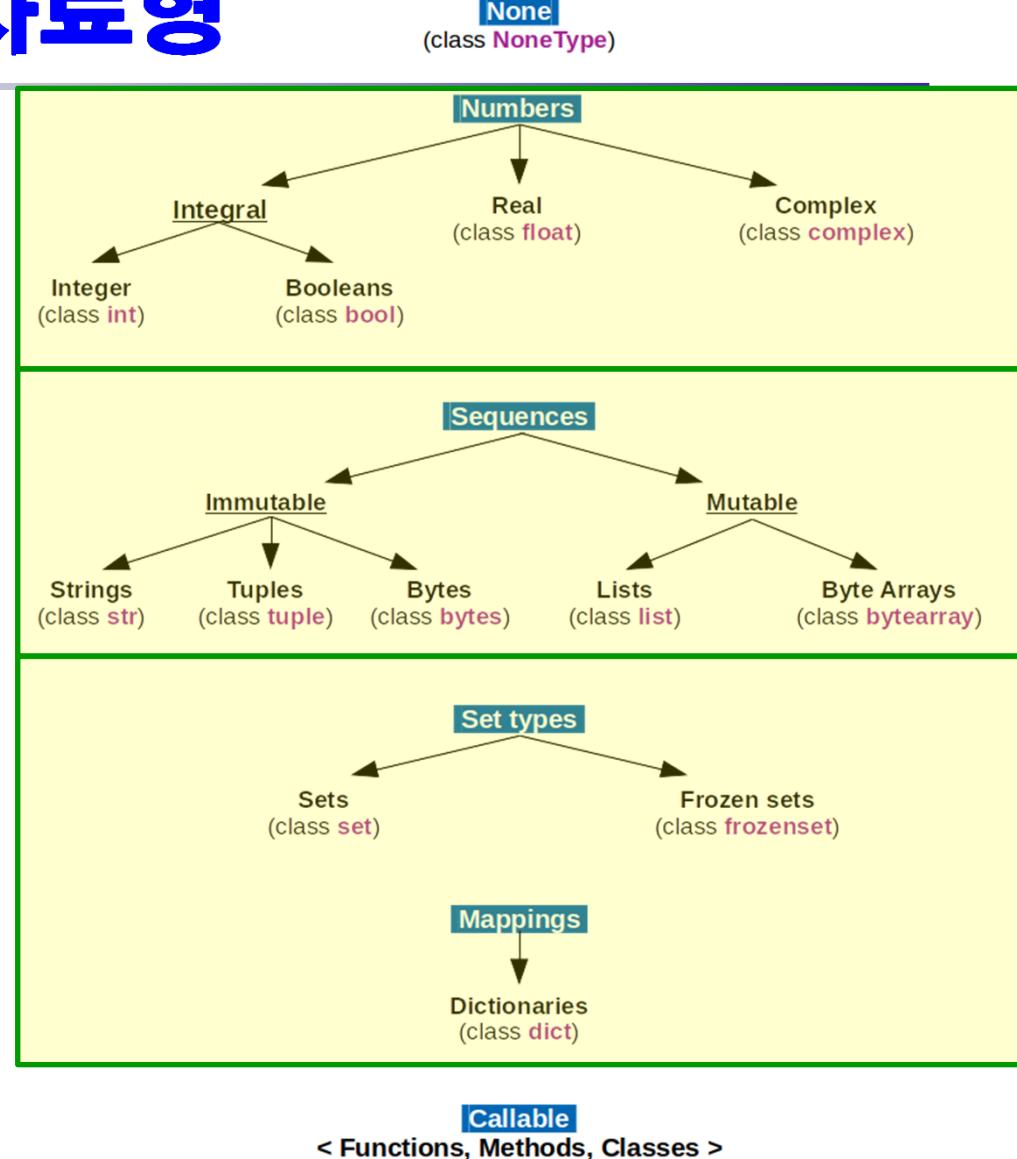
### ○ site 모듈에 의해 추가된 상수

- **quit(code=None)**
- **exit(code=None)**
- **copyright**
- **credits**
- **license**

# 자료형

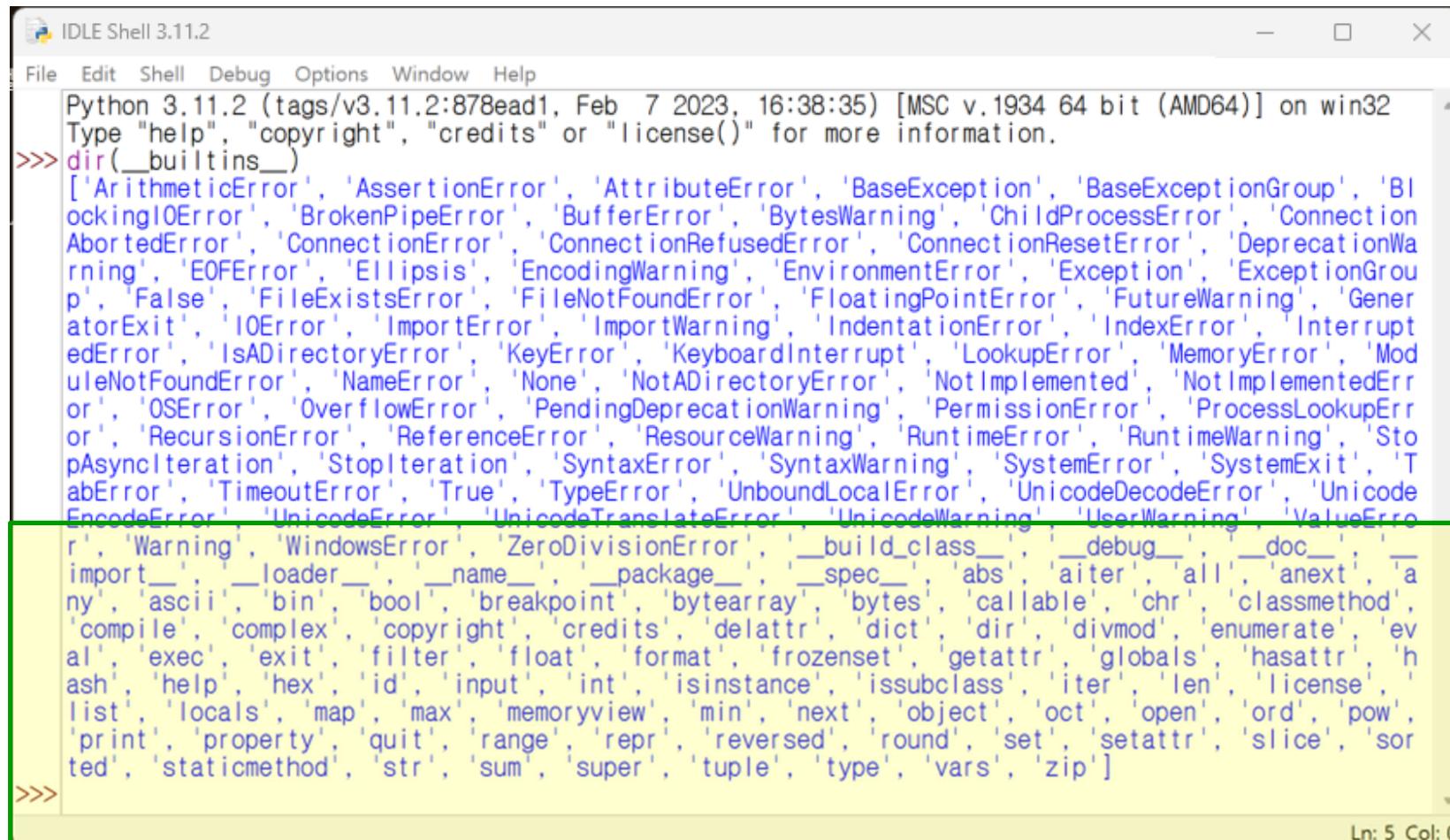
## ● 자료형(Data Types)

- **NoneType**: `None`
- **논리형**: `bool`
- **숫자형**
  - 정수: `int`
  - 실수: `float`
  - 복소수: `complex`
- **문자열**: `str`
- **리스트(List)**
- **튜플(Tuple)**
- **사전(Dictionary)**
- **집합(Set)**



# 자료형: 내장 클래스 (1/2)

## ● 파이썬 내장 객체: \_\_builtins\_\_



```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> dir(__builtins__)
['ArithError', 'AssertionError', 'AttributeError', 'BaseException', 'BaseExceptionGroup', 'BlockingIOError', 'BrokenPipeError', 'BufferError', 'BytesWarning', 'ChildProcessError', 'ConnectionAbortedError', 'ConnectionError', 'ConnectionRefusedError', 'ConnectionResetError', 'DeprecationWarning', 'EOFError', 'Ellipsis', 'EncodingWarning', 'EnvironmentError', 'Exception', 'ExceptionGroup', 'False', 'FileExistsError', 'FileNotFoundException', 'FloatingPointError', 'FutureWarning', 'GeneratorExit', 'IOError', 'ImportError', 'ImportWarning', 'IndentationError', 'IndexError', 'InterruptedError', 'IsADirectoryError', 'KeyError', 'KeyboardInterrupt', 'LookupError', 'MemoryError', 'ModuleNotFoundError', 'NameError', 'None', 'NotADirectoryError', 'NotImplemented', 'NotImplementedError', ' OSError', 'OverflowError', 'PendingDeprecationWarning', 'PermissionError', 'ProcessLookupError', 'RecursionError', 'ReferenceError', 'ResourceWarning', 'RuntimeError', 'RuntimeWarning', 'StopAsyncIteration', 'StopIteration', 'SyntaxError', 'SyntaxWarning', 'SystemError', 'SystemExit', 'TabError', 'TimeoutError', 'True', 'TypeError', 'UnboundLocalError', 'UnicodeDecodeError', 'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError', 'UnicodeWarning', 'UserWarning', 'ValueError', 'Warning', 'WindowsError', 'ZeroDivisionError', '__build_class__', '__debug__', '__doc__', '__import__', '__loader__', '__name__', '__package__', '__spec__', 'abs', 'aiter', 'all', 'anext', 'any', 'ascii', 'bin', 'bool', 'breakpoint', 'bytearray', 'bytes', 'callable', 'chr', 'classmethod', 'compile', 'complex', 'copyright', 'credits', 'delattr', 'dict', 'dir', 'divmod', 'enumerate', 'eval', 'exec', 'exit', 'filter', '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', 'reversed', 'round', 'set', 'setattr', 'slice', 'sorted', 'staticmethod', 'str', 'sum', 'super', 'tuple', 'type', 'vars', 'zip']
>>>
```

Ln: 5 Col: 0

# 자료형: 내장 클래스 (2/2)

## ● 내장 함수(Built-in Functions)

Built-in Functions			
A abs() aiter() all() any() anext() ascii()	E enumerate() eval() exec()	L len() list() locals()	R range() repr() reversed() round()
B bin() bool() breakpoint() bytearray() bytes()	F filter() float() format() frozenset()	M map() max() memoryview() min()	S set() setattr() slice() sorted() staticmethod() str() sum() super()
C callable() chr() classmethod() compile() complex()	G getattr() globals()	N next()	T tuple() type()
D delattr() dict() dir() divmod()	H hasattr() hash() help() hex()	O object() oct() open() ord()	V vars()
	I id() input() int() isinstance() issubclass() iter()	P pow() print() property()	Z zip()
			_import_()

[ 출처: "Built-in Functions", Python 3.13.2 documentation, <https://docs.python.org/3/library/functions.html>, 2025. ]

# 자료형: 기본 자료형 (1/5)

## ● 논리형(Boolean Type)

### ○ class bool

- 파이썬 예약어: True, False
- bool 내장 함수
  - class bool(x=False)

```
import keyword
keyword.kwlist[0]      # 'False'
keyword.kwlist[2]      # 'True'

type(True)             # <class 'bool'>
type(False)            # <class 'bool'>
```

class bool

- type 내장 함수: 입력된 값의 자료형이 무엇인지 알려주는 내장 함수
  - class type(object)
  - class type(name, bases, dict, \*\*kwds)

# 자료형: 기본 자료형 (2/5)

## ● 정수형(Integer Type)

### ○ class int

- 정수(소수점 이하 부분을 가지지 않는 숫자)를 표현하는 객체를 만드는 클래스
- 정수형(integer): 정수형의 범위는 무제한(Python3)
- 긴 정수형(long integer): 메모리가 하락하는 한 무제한의 자릿수로 정수를 계산
  - Python3에서 제거 됨.

```
a = 10

type(a)          # <class 'int'>
type(10)         # <class 'int'>
type(-10)        # <class 'int'>
type(1234567890) # <class 'int'>
type(123456789012345678901234567890) # <class 'int'>
```

# class int

- int 내장 함수: 문자열을 형태의 정수와 실수형 등을 정수 형태로 변환
  - class int(x=0)
  - class int(x, base=10)

# 자료형: 기본 자료형 (3/5)

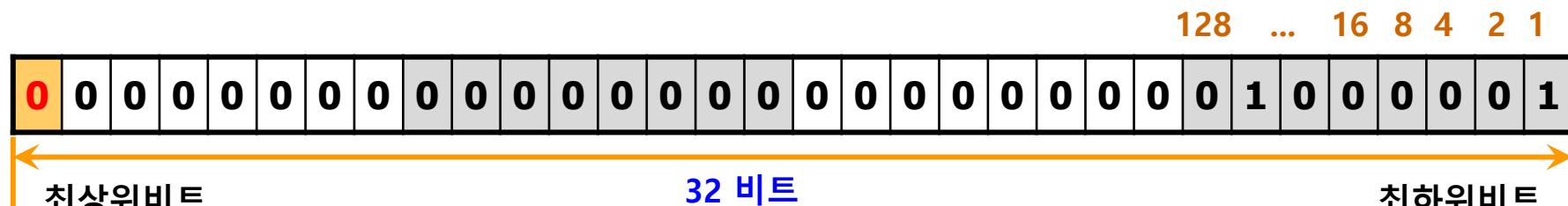
IDLE Shell 3.11.2

```
File Edit Shell De
Python 3.11.2
Type "help", '
>>> bin(65)
'0b1000001'
>>> bin(-65)
'-0b1000001'
>>>
```

- 정수형: 데이터 표현 방식

## ○ 양수 표현

i = 65

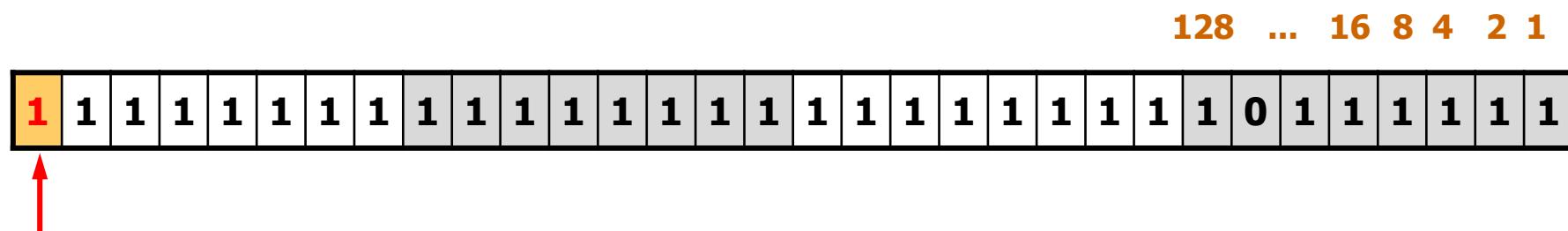


(**MSB** : Most significant Bit)

(**LSB** : Least Significant Bit)

### ○ 음수 표현: 2의 보수로 표현

$$i = -65$$



부호비트(MSB)가 1이면 음수(2의 보수로 표현)

# 자료형: 기본 자료형 (4/5)

## 예제 0-2: 정수형 -- 다양한 진법 표현

```
# 다양한 데이터 표현: 2진법, 8진법, 10진법, 16진법  
# class int(x, base=10): base 진수로 표현된 문자열 x 를 10진수로 반환  
  
print(int('00000010', 2))  
print(int('00000010', 8))  
print(int('00000010', 10))  
print(int('00000010', 16))
```

```
IDLE Shell 3.11.2  
File Edit Shell Debug Options Window  
Python 3.11.2 (tags/v3.11.2  
Type "help", "copyright", "credits" or "license" for more information  
RESTART:  
=>>> 2  
=>>> 8  
=>>> 10  
=>>> 16  
=>>> 0b1010  
=>>> 0o12  
=>>> 0xa
```

## # 다양한 진법 변환

```
print(bin(10))      # bin: 정수 형태의 숫자를 2진수(binary)로 변환(0b)  
print(oct(10))      # oct: 정수 형태의 숫자를 8진수(octal)로 변환(0o)  
print(hex(10))      # hex: 정수 형태의 숫자를 16진수(hexadecimal)로 변환(0x)
```

# 자료형: 기본 자료형 (5/5)

## ● 실수형(Real Type)

### ○ class float

- 부동소수점 방식으로 숫자를 표현하는 객체를 만드는 클래스

```
a = 10.0
```

```
type(a)          # <class 'float'>
type(10.0)        # <class 'float'>
type(10.123456789) # <class 'float'>
```

# class float

- float 내장 함수:** 문자열 형태의 정수와 실수형 등을 실수 형태로 변환
  - class float(x=0.0)

### ○ IEEE 754 표준:

- 실수 표현을 위해 4bytes 또는 8bytes 고정된 크기의 메모리 할당

```
a = 0.3 + 0.6
print(a)    # 0.8999999999999999

if a == 0.9 : print(True)
else : print(False)           # False
```

```
# round(a,4) : 소수점 다섯 번째 자리에서 반올림
a = 0.3 + 0.6
print(round(a,4))      # 0.9

if round(a,4) == 0.9 : print(True)    # True
else : print(False)
```

# 자료형: 문자열 (1/6)

## ● 문자열(String)

○ 문자열(string)은 문자들의 나열(sequence of characters)이다.

- 문자열을 만드는 방법
  - 큰 따옴표
  - 작은 따옴표

```
print('Hello World!!!!')
print("Hello World!!!!")
# 문자열 사용시 주의할 점
print("Hello World!!!!")      # SyntaxError: unterminated string literal
```

class str

- 큰따옴표(")로 시작했다가 작은따옴표(')로 끝내면 문법적인 오류이다.
- str 내장 함수: 문자열 형태로 객체를 반환한다.
  - class str(object='')
  - class str(object=b'', encoding='utf-8', errors='strict')

# b를 붙이면 바이트(bytes) 객체가 된다.

# 자료형: 문자열 (2/6)

## ● 문자열 보간: 형식 문자열

### ○ 다양한 포맷을 사용하여 데이터 값을 문자열에 포매팅 할 수 있다.

- 옛 스타일: % -- 파이썬 2.3에서 지원
  - `format_string % data` 형식
  - 포맷 문자열(format\_string) 안에 끼워 넣을 데이터(data)를 표시하는 형식
  - 변환 코드: %d, %o, %x, %c, %s, %f, %e, %g, %%
- 새 스타일: {}, format 메소드 -- 파이썬 2.6 이상에서만 지원
  - `format_string.format(data)` 형식
  - `format` 메소드의 인수는 포맷 문자열 내의 {} 순서대로 나타낸다.
  - 왼쪽 정렬(<, 기본값), 오른쪽 정렬(>), 가운데 정렬(^)
- 최신 스타일: f-문자열 -- 파이썬 3.6 이상에서만 지원
  - 첫 번째 인용 부호(문자열) 앞에 문자 `f` 또는 `F`를 붙이면 포매팅 기능을 사용할 수 있다.
  - 변수 이름이나 식을 중괄호 안에 포함해 값을 문자열로 가져온다.
  - `format` 메소드가 없고 포맷 문자열에 빈 괄호 {}와 위치 괄호{1}가 없다.
    - » `format` 메소드에서 수행할 수 있는 정의를 {} 안에 사용할 수 있다([코드 읽기 쉬움](#)).
  - f-문자열은 파이썬 3.8부터 이름과 값을 쉽게 출력할 수 있다([디버깅이 편리](#)).

# 자료형: 문자열 (3/6)

## ● 문자열 보간: 형식 문자열

### ○ input 내장 함수: 수치 데이터

```
# 1) 데이터 입력: 사용자로부터 두 개의 정수
x = int(input('첫 번째 정수 입력: '))
y = int(input('두 번째 정수 입력: '))

# 2) 수식 계산
sum = x + y
```

#### # 3-1) 결과 출력

```
print(x, '와', y, '의 합은', sum, '입니다.')
print(str(x) + '와' + str(y) + '의 합은' + str(sum) + '입니다.')
```

#### # 3-2) 결과 출력: 다양한 형식 문자열

```
print('%d 와 %d 의 합은 %d 입니다.' % (x, y, sum))
print('{} 와 {} 의 합은 {} 입니다.'.format(x, y, sum))
print(f'{x} 와 {y} 의 합은 {sum} 입니다.')
```

#### # map 내장 함수: 반복 가능한 객체(iterable)의 요소를 지정된 함수로 처리해주는 함수

```
print('두 개의 정수를 입력하세요: ', end=' ')
x, y = map(int, input().split())
# x, y = list(map(int, input().split()))
```

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 1 2023, 14:00:57)
Type "help", "copyright", "credits" or "license" for more information
>>> ===== RESTART: C:\Users\click\PycharmProjects\untitled\Untitled.py =====
첫 번째 정수 입력: 10
두 번째 정수 입력: 20
10 와 20 의 합은 30 입니다.
10와 20의 합은 30입니다.
10 와 20 의 합은 30 입니다.
10 와 20 의 합은 30 입니다.
10 와 20 의 합은 30 입니다.
```

# 자료형: 문자열 (4/6)

## ● 문자열 보간: 형식 문자열

- **f-문자열**: 형식 문자열 내의 {} 순서대로 나타낸다.

```
# f-문자열: 문자열에 데이터를 포함하여 문자열 출력
```

```
a = 12345  
b = 3.14159  
c = 'Python'
```

**f-문자열**

```
print( f'{a} {b} {c}' )      # 12345 3.14159 Python  
print( f'{b} {c} {a}' )      # 3.14159 Python 12345  
print( f'{c} {a} {b}' )      # Python 12345 3.14159
```



# 자료형: 문자열 (5/6)

## 예제 0-3: 문자열 보간(형식 문자열) -- f 문자열

(1/2)

# f-문자열: 문자열에 데이터를 포함하여 문자열 출력

a = 12345

```
print(f'[{a}]')           # print(f'[{a:d}]')
```

```
print(f'[{a:#o}]')
```

```
print(f'[{a:#x}]')
```

```
# print(f'[{a:8d}]')
```

```
print(f'[{a:8}]')
```

```
print(f'[{a:08}]')
```

```
print(f'[{a:+8}]')
```

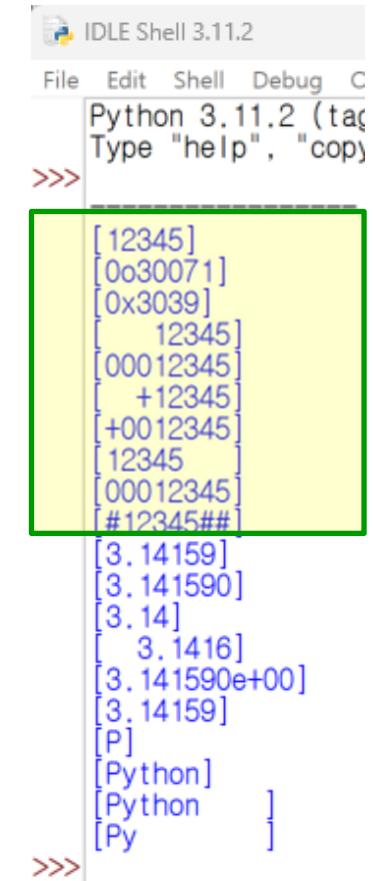
```
print(f'[{a:+08}]')
```

# 왼쪽 정렬(<), 오른쪽 정렬(>), 가운데 정렬(^)

```
print(f'[{a:<8}]')
```

```
print(f'[{a:0>8}]')
```

```
print(f'[{a:#^8}]')
```



```
IDLE Shell 3.11.2
File Edit Shell Debug C
Python 3.11.2 (tag)
Type "help", "cop"
>>>
[12345]
[0o30071]
[0x3039]
[12345]
[00012345]
[+12345]
[+0012345]
[12345]
[00012345]
[#12345##]
[3.14159]
[3.141590]
[3.14]
[3.1416]
[3.141590e+00]
[3.14159]
[P]
[Python]
[Python
[Py
]
>>>
```

# 자료형: 문자열 (6/6)

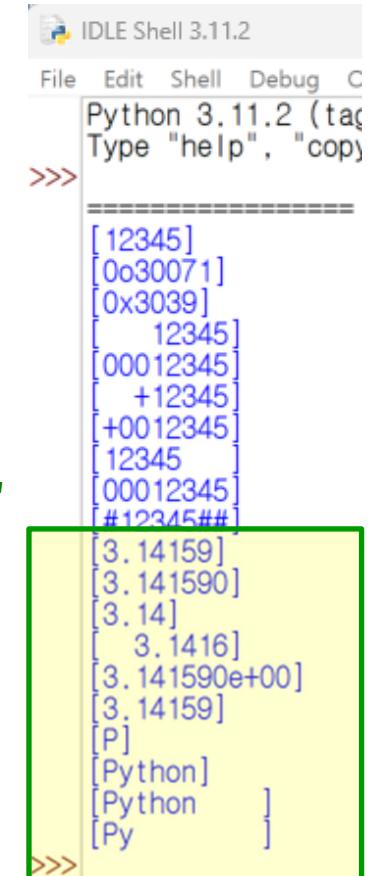
예제 0-3: 문자열 보간(형식 문자열) -- f 문자열

(2/2)

b = 3.14159

```
print(f'{b}')           # 3.14159
print(f'{b:f}')         # 3.141590
print(f'{b:.2f}')       # 소수점 2번째 자리 (.2) 까지
print(f'{b:8.4f}')      # 출력할 최소 자릿수(8)
print(f'{b:e}')         # 부동 소수점 표기법
print(f'{b:g}')         # %f 와 %e 중에서 더 짧은 표현 선택

# ValueError: Unknown format code 'c' for object of type 'str'
# print(f'{"P":c}')
# print(f'{"Python":c}')
print(f'{"P"}')          # print(f'{"P"}')
print(f'{"Python"}')      # print(f'{"Python":s}')
print(f'{"Python":10s}')
print(f'{"Python":10.2s}'')
```



```
IDLE Shell 3.11.2
File Edit Shell Debug C
Python 3.11.2 (tag
Type "help", "cop
>>> =====
[12345]
[0o30071]
[0x3039]
12345
[00012345]
+12345
+0012345
12345
[00012345]
[#12345##]
[3.14159]
[3.141590]
[3.14]
3.1416]
[3.141590e+00]
[3.14159]
[P]
[Python]
[Python
[Py]
```

# 자료형: 리스트 (1/5)

## ● **리스트(List): [ ]**

○ 여러 개의 자료들을 모아서 하나의 묶음으로 저장하는 것

- C/C++, Java 같은 프로그래밍 언어에서는 리스트가 없다.
  - 하지만, 리스트와 비슷한 개념인 배열(Array)을 사용한다.

○ 리스트 객체 생성

```
aList = [] # 빈 리스트 객체 생성
```

```
bList = [ 10, 20, 30, 40, 50 ] # 리스트 객체 생성
```

○ **list** 내장 함수

- 다른 데이터 유형(문자열, 튜플, 딕셔너리 등)을 리스트로 변환한다.

```
temp = list('1234567890') # 리스트 객체 생성
```

```
temp = ( 10, 20, 30, 40, 50 ) # 튜플 객체 생성
```

```
list(temp) # 튜플 객체를 리스트 객체로 변환
```

# class list

# 자료형: 리스트 (2/5)

- **리스트: 개별 항목 접근**
  - 리스트 객체의 개별 항목

```
# 리스트 생성
```

```
sList = [ 10, 20, 30, 40, 50 ]
```

```
# 리스트 객체의 개별 항목 접근: 순방향
```

```
sList[0] # 10
```

```
sList[1] # 20
```

```
sList[2] # 30
```

```
sList[3] # 40
```

```
sList[4] # 50
```

```
sList[5] # IndexError: list index out of range
```

```
# 리스트 객체의 개별 항목 접근: 역방향
```

```
sList[-1] # 50
```

```
sList[-2] # 40
```

```
sList[-3] # 30
```

```
sList[-4] # 20
```

```
sList[-4] # 10
```

```
# 리스트 객체 생성
```

```
sList = [ 10, 20, 30, 40, 50 ]
```

```
# 리스트 객체의 전체 항목 순회: 반복문(for)
```

```
for str in sList :  
    print(str)
```

```
for i in range(len(sList)) :  
    print(sList[i])
```

```
# List Comprehension
```

```
[ print(str) for str in sList ]
```

```
# 리스트 객체 생성
```

```
sList = [ 10, 20, 30, 40, 50 ]
```

```
# 리스트 객체의 전체 항목 순회: 반복문(while)
```

```
i = 0  
while i < len(sList) :  
    print(sList[i])  
    i += 1
```

# 자료형: 리스트 (3/5)

## ● **리스트**: 부분 항목 접근

### ○ 리스트 객체의 부분 항목 접근

```
# 리스트 객체 생성
```

```
sList = [ 10, 20, 30, 40, 50 ]
```

```
# 리스트 객체의 부분 원소 접근
```

```
sList[:] # [ 10, 20, 30, 40, 50 ]
```

```
sList[0:5] # [ 10, 20, 30, 40, 50 ]
```

```
sList[:4] # [ 10, 20, 30, 40 ]
```

```
sList[1:] # [ 20, 30, 40, 50 ]
```

```
# sList = [ 10, 20, 30, 40, 50 ]
```

```
sList[1] = 60 # [ 10, 60, 30, 40, 50 ]
```

```
sList[1:3] = [ 70, 80 ] # [ 10, 70, 80, 40, 50 ]
```

```
sList[1:2] = [ 90, 99 ] # [ 10, 90, 99, 80, 40, 50 ]
```

```
sList[1:3] = [ 88 ] # [ 10, 88, 80, 40, 50 ]
```

# 자료형: 리스트 (4/5)

## ● 리스트: 복사

### ○ 동일한 리스트 객체를 참조하는 객체 생성

```
# 기존 리스트 객체를 복사하여, 동일한 객체를 참조
```

```
a = [ 10, 20, 30, 40, 50 ]
```

```
b = a
```

```
a is b # True
```

### ○ 자신만의 값을 가지는 새로운 객체 생성

```
# 기존 리스트 객체를 복사하여, 자신만의 값을 가지는 새로운 객체 생성
```

```
a = [ 10, 20, 30, 40, 50 ]
```

```
b = a[:] # 1) 슬라이싱 [:]
```

```
c = a.copy() # 2) 리스트 조작 함수(copy)
```

```
d = list(a) # 3) list 내장 함수
```

```
a is b # False
```

```
a is c # False
```

```
a is d # False
```

# 자료형: 리스트 (5/5)

## 예제 0-4: 파이썬 내장 함수와 리스트 조작 함수

```
# 리스트 객체 생성  
sList = [10, 20, 30, 40, 50]
```

```
# 내장 함수: len  
print(f'전체 원소: {sList}')  
print(f'전체 원소 개수: {len(sList)}')
```

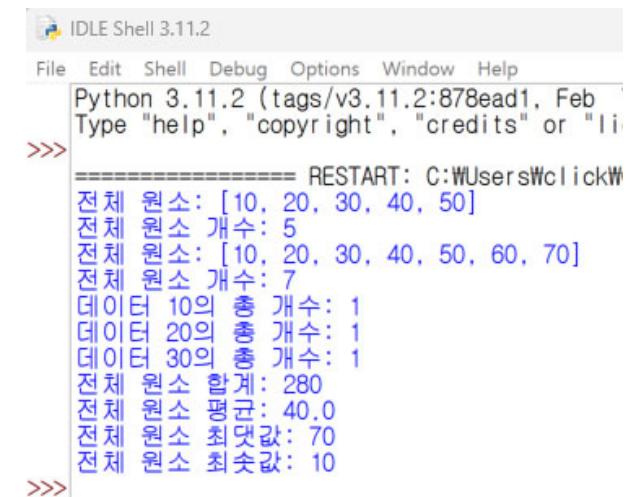
```
# 리스트 조작 함수	append: 리스트 맨 마지막 원소로 추가  
sList.append(60)  
sList.append(70)
```

```
# 내장 함수: len  
print(f'전체 원소: {sList}')  
print(f'전체 원소 개수: {len(sList)}')
```

```
# 리스트 조작 함수(count): 특정 원소의 총 개수  
print(f'데이터 10의 총 개수: {sList.count(10)}')  
print(f'데이터 20의 총 개수: {sList.count(20)}')  
print(f'데이터 30의 총 개수: {sList.count(30)}')
```

```
# 내장 함수: len, sum  
print(f'전체 원소 합계: {sum(sList)}')  
print(f'전체 원소 평균: {sum(sList)/len(sList)}')
```

```
# 내장 함수: max, min  
print(f'전체 원소 최댓값: {max(sList)}')  
print(f'전체 원소 최솟값: {min(sList)}')
```



```
IDLE Shell 3.11.2  
File Edit Shell Debug Options Window Help  
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  
Type "help", "copyright", "credits" or "li  
>>> ===== RESTART: C:\Users\clickw  
전체 원소: [10, 20, 30, 40, 50]  
전체 원소 개수: 5  
전체 원소: [10, 20, 30, 40, 50, 60, 70]  
전체 원소 개수: 7  
데이터 10의 총 개수: 1  
데이터 20의 총 개수: 1  
데이터 30의 총 개수: 1  
전체 원소 합계: 280  
전체 원소 평균: 40.0  
전체 원소 최댓값: 70  
전체 원소 최솟값: 10  
>>>
```

# 자료형: 튜플 (1/4)

## ● 튜플(Tuple): ()

### ○ 튜플은 순서가 있고 변경할 수 없다.

- 불변(immutable) 데이터 유형
  - 즉, 튜플이 생성된 후에는 항목을 변경, 추가 또는 제거할 수 없다.
  - 튜플은 리스트의 append, insert 등과 같은 튜플 조작 함수가 거의 없다.
- Tuple Comprehension은 없다.
  - 가변적인 데이터 유형(리스트, 딕셔너리와 셋)에는 Comprehension이 있다.
  - 문자열과 튜플은 불변 유형이기 때문에 다른 방법으로 만들어져야 한다.
- 튜플의 다양한 장점
  - 튜플은 리스트보다 더 적은 공간을 사용한다.
  - 프로그래머의 실수로 튜플의 항목이 손상될 염려가 없다.
  - 튜플을 딕셔너리의 키로 사용할 수 있다.
  - 네임드 튜플(named tuple)은 객체의 단순한 대안이 될 수 있다.

# 자료형: 튜플 (2/4)

## ● 튜플(Tuple): 튜플 생성 및 초기화

### ○ 튜플 생성 및 초기화

class tuple

```
aTuple = ()
```

# 빈 튜플 생성

```
bTuple = ( 10, 20, 30, 40, 50 )
```

# 튜플 생성 및 초기화

# 튜플을 생성할 때는 괄호를 생략 할 수 있다.

```
cTuple = 10, 20, 30, 40, 50
```

### ○ tuple 내장 함수

- 다른 데이터 유형(문자열, 리스트, 딕셔너리 등)을 튜플로 변환한다.

```
temp = tuple('1234567890')
```

# 튜플 객체 생성

```
temp = [ 10, 20, 30, 40, 50 ]
```

# 리스트 객체 생성

```
tuple(temp)
```

# 리스트를 튜플 객체로 변환

# 자료형: 튜플 (3/4)

## ● 튜플(Tuple): 튜플 생성 및 초기화

### ○ 하나의 항목을 가진 튜플 생성

- 항목이 하나만 있는 튜플을 만들려면 항목 뒤에 쉼표를 추가해야 합니다. 그렇지 않으면 파이썬은 이를 튜플로 인식하지 못한다.

```
aTuple = ('apple',)  
print(type(aTuple))      # <class 'tuple'>  
  
# NOT a tuple  
bStr = ('apple')  
print(type(bStr))        # <class 'str'>  
  
# NOT a tuple  
cInt = (10)  
print(type(cInt))        # <class 'int' >
```

class tuple

# 자료형: 튜플 (4/4)

## 예제 0-5: 리스트와 튜플

```
aTuple = (10, 20, 30, 40, 50)
print(aTuple)

# 튜플을 리스트 객체로 변경 후 항목의 값을 변경 및 추가
aList = list(aTuple)
aList[1] = 60
aList.append(70)
print(aList)

aTuple = tuple(aList)
print(aTuple)
```

```
bTuple = ('국어', '영어', '수학')
cTuple = ('과학',)

# 튜플 연결(결합)
bTuple += cTuple
print(bTuple)
```

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1,
Type "help", "copyright", "credits" ,
>>> ===== RESTART: C:\Users\W
(10, 20, 30, 40, 50)
[10, 60, 30, 40, 50, 70]
(10, 60, 30, 40, 50, 70)
('국어', '영어', '수학', '과학')
>>>
```

# 자료형: 딕셔너리 (1/3)

---

## ● 딕셔너리(Dictionary)

○ 연관되어 있는 데이터를 하나의 쌍으로 묶어서 관리한다.

- 키(Key)와 값(Value)의 쌍으로 데이터 값을 저장
- 순서가 지정되고 변경 가능하며, 중복을 허용하지 않는다.
  - 동일한 키를 가진 두 개의 항목이 있을 수 없습니다(유일한 키 값).
  - 딕셔너리가 생성된 후 항목을 변경, 추가 또는 제거할 수 있다.

class dict

# 자료형: 딕셔너리 (2/3)

## ● 딕셔너리(Dirctionary): {}

### ○ 딕셔너리 객체 생성 및 할당

```
# 딕셔너리 객체 생성  
aDict = {}  
bDict = dict() # dict 내장 함수
```

```
# 딕셔너리 객체 생성  
carDict = {  
    'brand': 'HYUNDAI',  
    'model': 'IONIQ',  
    'year': 2025  
}
```

```
# 딕셔너리 객체의 키에 대한 이름으로 참조한다.  
print(type(carDict)) # <class 'dict'>  
print(carDict['brand']) # HYUNDAI  
print(carDict.get('brand')) # HYUNDAI
```

# class dict

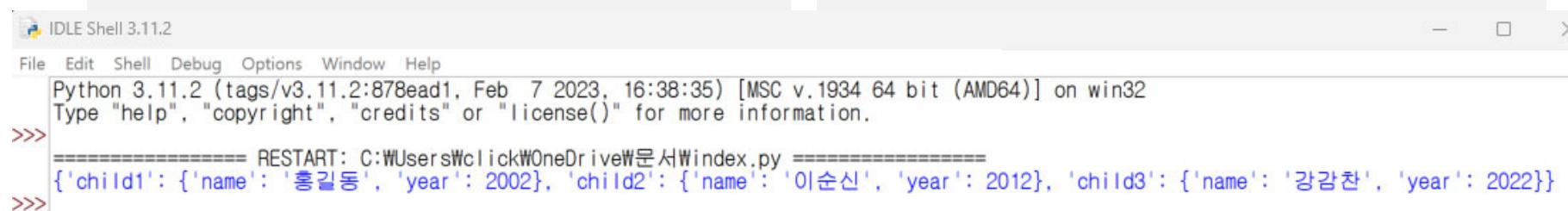
# 자료형: 딕셔너리 (3/3)

## 예제 0-6: 딕셔너리 -- 중첩 딕셔너리

```
myFamily = {
    'child1' : {
        'name' : '홍길동',
        'year' : 2002
    },
    'child2' : {
        'name' : '이순신',
        'year' : 2012
    },
    'child3' : {
        'name' : '강감찬',
        'year' : 2022
    }
}
print(myFamily)
```

```
child1 = {
    'name' : '홍길동',
    'year' : 2002
}
child2 = {
    'name' : '이순신',
    'year' : 2012
}
child3 = {
    'name' : '강감찬',
    'year' : 2022
}

myFamily = {
    'child1' : child1,
    'child2' : child2,
    'child3' : child3
}
print(myFamily)
```



The screenshot shows the Python IDLE shell interface. The title bar says "IDLE Shell 3.11.2". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The Python version is listed as "Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32". The command prompt shows "Type "help", "copyright", "credits" or "license()" for more information." The code is run with "myFamily" and its three children. The output is a dictionary where each child is another dictionary with 'name' and 'year' keys.

```
>>> ===== RESTART: C:\Users\click\OneDrive\문서\Windex.py =====
{'child1': {'name': '홍길동', 'year': 2002}, 'child2': {'name': '이순신', 'year': 2012}, 'child3': {'name': '강감찬', 'year': 2022}}
```

# 자료형: 셋

## ● 셋(Set): { }

### ○ 값은 버리고, 키만 남은 딕셔너리와 같다.

- 딕셔너리와 마찬가지로 각 키들은 반드시 고유해야 한다.
  - 즉, 중복되는 키는 저장되지 않는다.
- 단일 변수에 여러 항목을 지정하는데 사용된다.
- 셋은 순서가 없고, 변경할 수 없고, 인덱싱 되지 않는다.
  - 항목은 변경할 수 없지만, 항목을 제거하고 새 항목을 추가할 수 있다.

### ○ 셋 객체 생성 및 할당

```
newSet = set()          # set 내장 함수  
  
# 셋 객체 생성  
  
carSet = { 'brand' , 'model' , 'year' }  
  
print(type(carSet))    # <class 'set'>
```

class set



# 파이썬 프로그래밍 기초

수식과 연산자

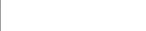
: 연산자 우선 순위와 결합성



# 연산자 우선 순위와 결합성 (1/8)

우선 순위	연산자 설명	연산자	결합성
1	바인딩(Binding) 또는 괄호 표현식 List: [ expressions... ] Dictionary: { key: value... } Set: { expressions... }	() [] { } {} ∅	
2	인덱싱(indexing) 슬라이싱(slicing)	x[ index ] x[ index:index ]	
	호출(call)	x( arguments... )	
	속성 참조(attribute reference)	x.attribute	
3	await 표현식	<b>await</b>	
4	지수 연산자	**	
5	양수 / 음수	+ -	
	1의 보수 연산(NOT)	~	
6	산술 연산자	곱셈 / 행렬 곱셈 * @	
		나눗셈 / // %	
		덧셈 / 뺏셈 + -	
7			
8	시프트 연산자	<< >>	
9	비트 논리 연산자	논리곱(AND) &	
		배타적 논리합(XOR) ^	
		논리합(OR) 	

# 연산자 우선 순위와 결합성 (2/8)

우선 순위	연산자 설명	연산자	결합성
12	식별 연산자	<b>is, is not</b>	
	멤버 연산자	<b>in, not in</b>	
	비교 연산자	<b>&lt; &gt; &lt;= &gt;=</b>	
		<b>==</b>	
		<b>!=</b>	
13	논리 연산자	<b>not</b>	
14		<b>and</b>	
15		<b>or</b>	
16	조건식	<b>if – else</b>	
17	람다(Lambda) 표현식	<b>lambda</b>	
18	할당 표현식	<b>:=</b>	

[ 출처: "Python 3.11.2 documentation", <https://docs.python.org/3/reference/expressions.html>, 2023. ]



# 연산자 우선 순위와 결합성 (3/8)

## 예제 0-7: 논리 연산자 -- not, and, or

```
a = 5
```

```
print(True and False)          # False
print(True or False)           # True
print(not True)                # False

print(1 and 0)                 # 0
print(5 or 0)                  # 5
print(not -5)                  # False

print('' and 0)                #
print('Python' or 0)           # Python
print(not [])                  # True

print(a<5 and a<10)           # False
print(a<5 or a<10)            # True
print(not(a<5 and a<10))      # True
```

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Win
Python 3.11.2 (tags/v3.11.2b2-759ccf6092, Feb 28 2023, 16:40:45)
Type "help", "copyright", "credits" or "license" for more information
>>> ===== RESTART:
False
True
False
0
5
False

Python
True
False
True
True
>>>
```

# 연산자 우선 순위와 결합성 (4/8)

## ● 비교 연산자(Comparison operators)

○ == 과 != 의 기본 동작은 객체의 ID를 기반으로 한다.

- 즉, 동일한 객체는 동등하게 비교되어야 한다.
- $a == b$  # a is b

○ 비교 연산자는 임의로 연결할 수 있다.

- $a < b <= c$  # a < b and b <= c # 단, b는 한 번만 평가된다.
- $a < b > c$  # a < b and b > c
- 비교 연산은 전이적이어야 한다.
- $a > b \text{ and } b > c$  # a > c
- $a < b \text{ and } b <= c$  # a < c

# 연산자 우선 순위와 결합성 (5/8)

## 예제 0-8: 비교 연산자 -- >, <, >=, <=, ==, !=

```
a = 10
print(a > 10)          # False
print(a < 10)          # False
print(a >= 10)         # True
print(a <= 10)         # True
```

# == 과 != 의 기본 동작은 객체의 ID를 기반으로 한다.

```
print(10 == 10)        # True
print(10 != 10)        # False
print(10 == '10')       # False
```

```
print(10 == 10.0)      # True
# SyntaxWarning: "is" with a literal. Did you mean "=="?
print(10 is 10.0)      # False
```

```
print("Hello" == 'Hello')    # True
print("Hello" == 'hello')     # False
```

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC
Type "help", "copyright", "credits" or "license()" for more infor
>>>
Warning (from warnings module):
  File "C:\Users\click\OneDrive\문서\Windex.py", line 14
    print(10 is 10.0) # False
SyntaxWarning: "is" with a literal. Did you mean "=="?
>>>
===== RESTART: C:\Users\click\OneDrive\문서\Windex.py
False
False
True
True
True
False
False
True
False
True
False
False
>>>
```

# 연산자 우선 순위와 결합성 (6/8)

## ● 식별 연산자(Identity operators)

### ○ 같은 객체인지 여부를 판단하는 연산자

연산자		사용 예
<b>is</b>	두 변수가 동일한 객체인지 여부	<code>a <b>is</b> b</code>
<b>is not</b>	두 변수가 동일한 객체가 아닌지 여부	<code>a <b>is not</b> b</code>

- 동일한 객체는 동등하게 비교되어야 한다.
- `a is b`                    `# a == b`
- 객체의 ID는 `id` 내장 함수를 사용하여 확인할 수 있다.



# 연산자 우선 순위와 결합성 (7/8)

## ● 멤버 연산자(Membership operators)

### ○ 객체에 표시되는지 판단하는 연산자

연산자		사용 예
<b>in</b>	지정된 값이 존재하는지 여부	a <b>in</b> b
<b>not in</b>	지정된 값이 존재하지 않는지 여부	a <b>not in</b> b

#### # 문자열 객체

```
'C' in 'Hi~ Clickseo'          # True  
'C' not in 'Hi~ Clickseo'      # False
```

#### # 리스트 객체

```
10 in [10, 20, 30, 40, 50]      # True  
10 not in [10, 20, 30, 40, 50]    # False
```

# 연산자 우선 순위와 결합성 (8/8)

## ● 비트 연산자

- 비트 단위의 정밀한 데이터 제어를 위해 사용

“정수 계산이 가능한 정수형 데이터에서만 동작”

- (주의) 피연산자로 float과 같은 부동형을 사용할 수 없다.

구분	연산자	의미
비트 단위 논리 연산	$\sim a$	1의 보수 연산(NOT)
	$a \& b$	a와 b의 비트 단위 논리곱(AND)
	$a   b$	a와 b의 비트 단위 논리합(OR)
	$a ^ b$	a와 b의 비트 단위 배타적 논리합(XOR)
시프트 연산	$a << n$	a의 각 비트를 왼쪽으로 n 만큼 이동
	$a >> n$	a의 각 비트를 오른쪽으로 n 만큼 이동



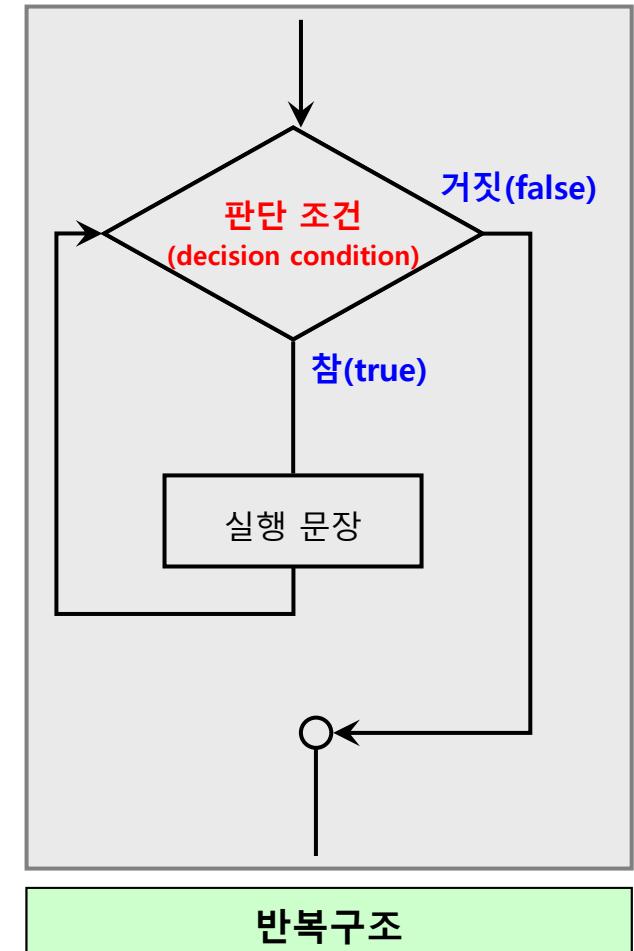
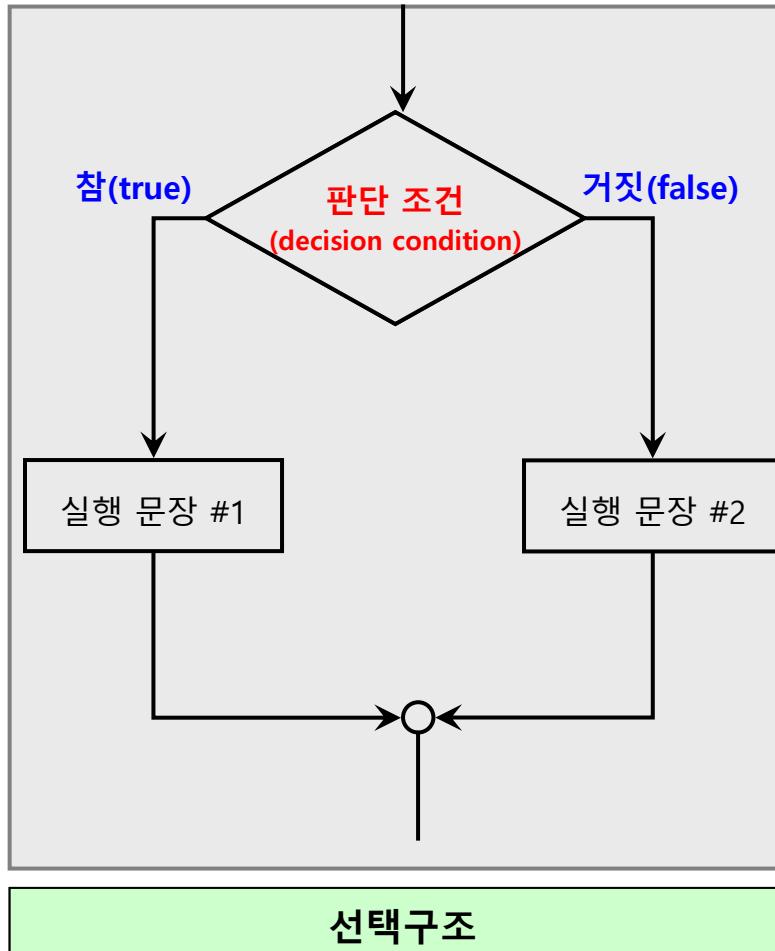
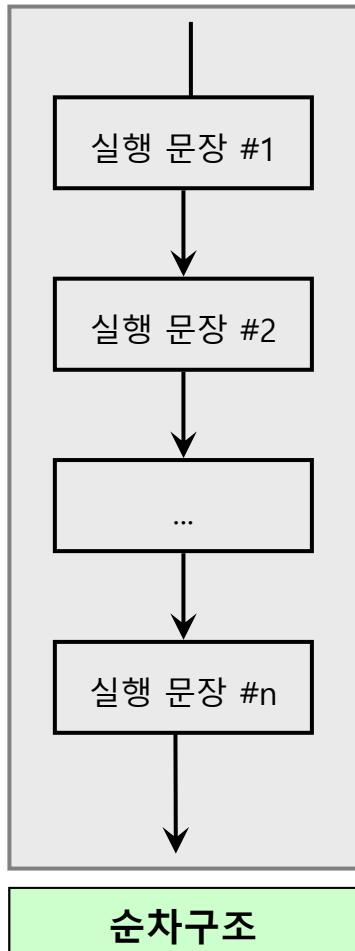
# 파이썬 프로그래밍 기초

제어 흐름



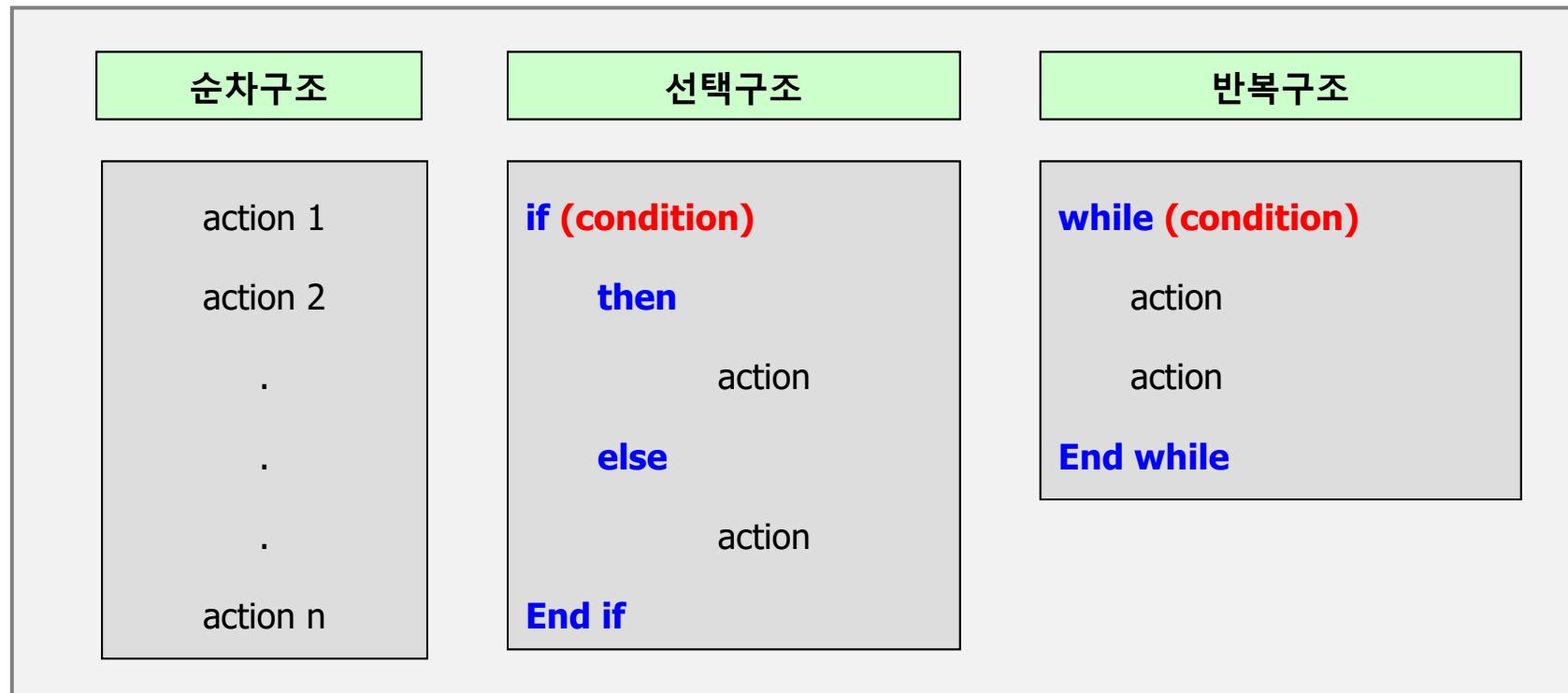
# 제어 흐름 (1/2)

## ● 순서도(Flowchart): 알고리즘을 그림으로 표현



# 제어 흐름 (2/2)

- **의사코드(pseudo-code)**: 영어와 비슷한 자연어로 표현
  - 특정 프로그래밍 언어의 문법에 따라 쓰인 것이 아니라, 일반적인 프로그래밍 언어와 형태가 유사하다.
    - 특정 언어로 프로그램을 작성하기 전에 알고리즘의 모델을 대략적으로 모델링



# 제어 흐름: 선택 구조 (1/6)

## 예제 0-9: 이중 선택 -- if-else

```
print('학생 성적(0 ~ 100)을 입력하세요.')
num = int(input('num: '))
```

```
# 짹수와 홀수 판단: 단순 if-else
if num % 2 == 0 :
    print(f'{num}은 짹수입니다.')
else :
    print(f'{num}은 홀수입니다.)
```

```
# Pass, Non-Pass 판단: 복합문
if num >= 60 :
    print('Pass, ', end='')
    print('축하 드립니다.')
else :
    print('Non-Pass, ', end='\t')
    print('내년에 다시 봄여. ^..~')
```

```
if num%2 == 0 : s = '짜수'
else : s = '홀수'
```

```
# 조건부 표현식(Conditional Expression)
# 변수 = 참(값) if 조건문 else 거짓(값)
s = '짜수' if num%2 == 0 else '홀수'
print(f'{num}은 {s}입니다.)
```

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb
Type "help", "copyright", "credits" or "li
>>> ===== RESTART: C:\Users\click\
학생 성적(0 ~ 100)을 입력하세요.
num: 98
98은 짹수입니다.
Pass, 축하 드립니다.

>>> ===== RESTART: C:\Users\click\
학생 성적(0 ~ 100)을 입력하세요.
num: 59
59은 홀수입니다.
Non-Pass, 내년에 다시 봄여.
```

# 제어 흐름: 선택 구조 (2/6)

## 예제 0-10: 이중 선택 -- 중첩 if문

```
num = int(input('임의의 정수를 입력하세요: '))

if num >= 0 :
    if num == 0 :
        print('0입니다.')
    else :
        print('양수입니다.')
else :
    print('음수입니다.)
```

중첩 if문



IDLE Shell 3.11.2

File Edit Shell Debug Options Window Help

Python 3.11.2 (tags/v3.11.2:878ead1, Feb  
Type "help", "copyright", "credits" or "li  
==== RESTART: C:\Users\click\W  
임의의 정수를 입력하세요: 10  
양수입니다.

==== RESTART: C:\Users\click\W  
임의의 정수를 입력하세요: -10  
음수입니다.

==== RESTART: C:\Users\click\W  
임의의 정수를 입력하세요: 0  
0입니다.

```
num = int(input('임의의 정수를 입력하세요: '))
if num > 0 : print('양수입니다.')
elif num == 0 : print('0입니다.')
else : print('음수입니다.)
```

다중 선택

# 제어 흐름: 선택 구조 (3/6)

## ● match 문

### ○ Python 3.10에서 도입(switch문과 비슷하다.)

- 일치하는 첫 번째 패턴만 실행되며, 값에서 변수로 구성 요소(시퀀스 요소 또는 개체 특성)를 추출할 수도 있다.
- 표면적으로 C/C++, Java 또는 JavaScript(및 기타 여러 언어)의 **switch** 문과 유사하지만, **Rust** 또는 **Haskell**과 같은 언어의 패턴 일치와 더 유사하다.

### ○ match, case 및 \_ 는 소프트 키워드이다.

- **soft keywords (Python 3.10의 새로운 기능)**
  - 일부 식별자는 특정 컨텍스트에서만 예약된다.



[ 출처: "Python 3.11.2 documentation", <https://docs.python.org/3/tutorial/controlflow.html#match-statements>, 2023. ]

# 제어 흐름: 선택 구조 (4/6)

## 예제 0-11: 다중 선택 -- 딕셔너리로 구현한 switch-case

```
# dict: Python 3.10이전에 딕셔너리로 구현한 switch-case
```

```
scores = {  
    10: 'A',  
    9: 'A',  
    8: 'B',  
    7: 'C',  
    6: 'D',  
}
```

```
score = int(input('학생 성적(0 ~ 100)을 입력하세요: '))
```

```
# 딕셔너리의 get 메소드
```

```
# dict.get(key, 0)      # 0은 Key를 찾을 수 없을 때 나오는 값
```

```
num = score // 10
```

```
grade = scores.get(num, 'F')
```

```
print(f'당신의 학점은 {grade} 입니다.')
```

```
IDLE Shell 3.11.2  
File Edit Shell Debug Options Window Help  
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 1 2023, 14:35:22)  
Type "help", "copyright", "credits" or "license" for more information.  
=====  
=>>> RESTART: C:\Users\click\...  
학생 성적(0 ~ 100)을 입력하세요: 98  
당신의 학점은 A 입니다.  
=====  
=>>> RESTART: C:\Users\click\...  
학생 성적(0 ~ 100)을 입력하세요: 75  
당신의 학점은 C 입니다.  
=====  
=>>> RESTART: C:\Users\click\...  
학생 성적(0 ~ 100)을 입력하세요: 59  
당신의 학점은 F 입니다.
```

# 제어 흐름: 선택 구조 (5/6)

## 예제 0-12: 다중 선택 -- match 문

#01

```
# match: Python 3.10에서 도입된 match-case (switch-case와 비슷하다.)  
score = int(input('학생 성적(0 ~ 100)을 입력하세요: '))  
  
num = score // 10  
  
match num:  
    case 10 | 9: grade = 'A' # case 조건에 | 를 사용하여 OR를 표현  
    # case 9: grade = 'A'  
    case 8: grade = 'B'  
    case 7: grade = 'C'  
    case 6: grade = 'D'  
    case default: grade = 'F'  
    # case _: grade = 'F'  
  
print(f'당신의 학점은 {grade} 입니다.')
```



```
IDLE Shell 3.11.2  
File Edit Shell Debug Options Window Help  
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 1 2023, 16:37:46)  
Type "help", "copyright", "credits" or "license" for more information.  
==== RESTART: C:\Users\click\python\ex0-12.py  
>>> 학생 성적(0 ~ 100)을 입력하세요: 98  
당신의 학점은 A 입니다.  
==== RESTART: C:\Users\click\python\ex0-12.py  
>>> 학생 성적(0 ~ 100)을 입력하세요: 75  
당신의 학점은 C 입니다.  
==== RESTART: C:\Users\click\python\ex0-12.py  
>>> 학생 성적(0 ~ 100)을 입력하세요: 59  
당신의 학점은 F 입니다.
```

# 제어 흐름: 선택 구조 (6/6)

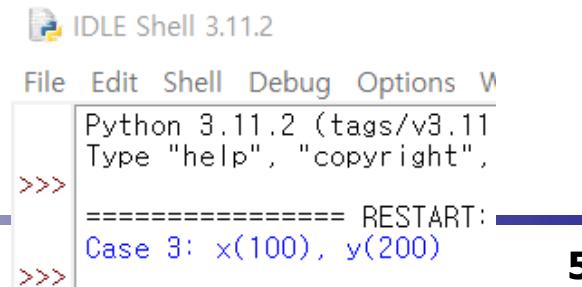
## 예제 0-12: 다중 선택 -- match 문

#02

```
# 변수를 바인딩 (Binding)하는데 사용할 수 있다.  
# print(f'y:{y}')           # NameError: name 'y' is not defined  
flag = False
```

**match (100, 200):**

```
    case (100, 300):          # Mismatch: 200 != 300  
        print('Case 1: x(100), y(300)')  
    # case (100, 200):         # Matches  
    #     print(f'Case 2: x(100), y(200)')  
    case (100, 200) if flag:   # Successful match, but guard fails  
        print('Case 2')  
    case (100, y):            # Matches and binds y to 200  
        print(f'Case 3: x(100), y({y})')  
    case _: # Pattern not attempted  
        print('Case 4, I match anything!!!!')  
# print(f'y:{y}')           # y:200
```



```
IDLE Shell 3.11.2  
File Edit Shell Debug Options V  
Python 3.11.2 (tags/v3.11  
Type "help", "copyright",  
>>>  
===== RESTART:  
>>> Case 3: x(100), y(200)
```

# 제어 흐름: 반복 구조 (1/5)

## ● 반복문의 필요성: while 문

### ○ 반복되는 문장 출력

```
# 문자열 객체 'Hi~ Clickseo' 를 5번 연속하여 출력
# print('Hi~ Clickseo\n' * 5)
print('Hi~ Clickseo')
print('Hi~ Clickseo')
print('Hi~ Clickseo')
print('Hi~ Clickseo')
print('Hi~ Clickseo')
```

The screenshot shows the Python 3.11.2 shell window. At the top, it says "File Edit Shell Debug Options Win". Below that, it says "Python 3.11.2 (tags/v3.11.2)". Then, there's a command prompt "Type 'help', 'copyright'", followed by "==== RESTART:". The output shows five lines of "Hi~ Clickseo" printed sequentially.

```
# while문: 조건식
```

```
num = 1           # 초기화
while num <= 5 : # 조건문
    print('Hi~ Clickseo')
    num=num+1      # 증감문
else : print('반복문 종료!!!')
```

```
# 무한 루프
```

```
while True:
    pass
```

# 제어 흐름: 반복 구조 (2/5)

## ● 반복문의 필요성: for 문

### ○ 반복되는 문장 출력

```
# 문자열 객체 'Hi~ Clickseo' 를 5번 연속하여 출력  
# print('Hi~ Clickseo\n' * 5)  
print('Hi~ Clickseo')  
print('Hi~ Clickseo')  
print('Hi~ Clickseo')  
print('Hi~ Clickseo')  
print('Hi~ Clickseo')
```

```
# for문: 리스트 객체 활용  
for i in [ 1,2, 3, 4, 5 ] :  
    print('Hi~ Clickseo')
```

```
# for문: range 객체 활용  
for i in range(5) :  
    print('Hi~ Clickseo')
```

# 제어 흐름: 반복 구조 (3/5)

## ● 반복문의 필요성: for 문

### ○ range 클래스: 원하는 범위의 수열을 생성

# range 클래스

range(5) # 0, 1, 2, 3, 4

range(0, 5) # 0, 1, 2, 3, 4

range(0, 5, 1) # 0, 1, 2, 3, 4

range(5, 10) # 5, 6, 7, 8, 9

range(-5, 0) # -5, -4, -3, -2, -1

range(1, 10, 2) # 1, 3, 5, 7, 9

# 인자는 모두 정수형 자료만 가능하다.

# range(2.5) # TypeError 발생

# range 객체 유형 변경: list, tuple

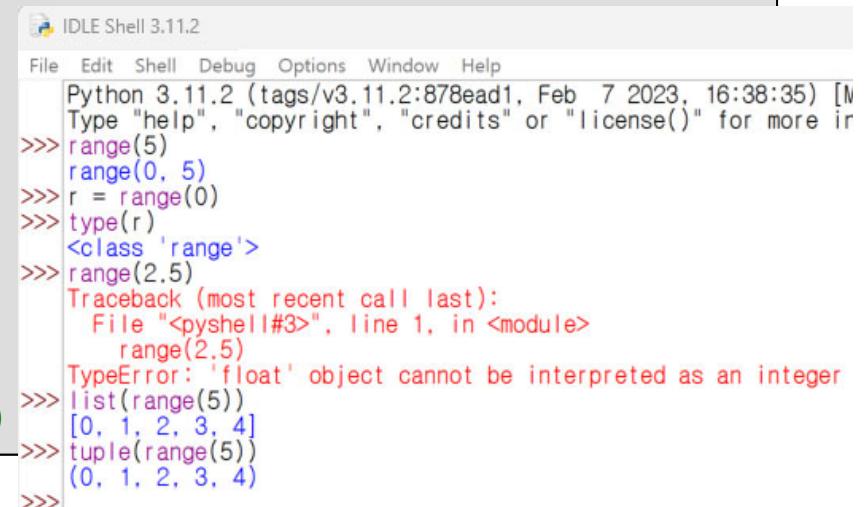
list(range(1, 10, 2)) # [1, 3, 5, 7, 9]

tuple(range(10, 0, -2)) # (10, 8, 6, 4, 2)

# range 클래스

class range(stop)

class range(start = 0, stop[, step = 1])



```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [N
Type "help", "copyright", "credits" or "license()" for more information
>>> range(5)
range(0, 5)
>>> r = range(0)
>>> type(r)
<class 'range'>
>>> range(2.5)
Traceback (most recent call last):
  File "<pyshell#3>", line 1, in <module>
    range(2.5)
TypeError: 'float' object cannot be interpreted as an integer
>>> list(range(5))
[0, 1, 2, 3, 4]
>>> tuple(range(5))
(0, 1, 2, 3, 4)
>>>
```

# 제어 흐름: 반복 구조 (4/5)

- 반복문의 필요성: for 문

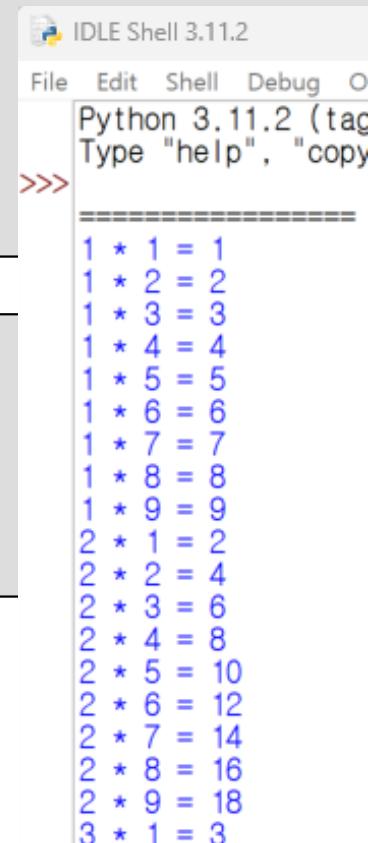
- 구구단 출력: 중첩 for 문

```
# 구구단 출력
```

```
for i in [ 1, 2, 3, 4, 5, 6, 7, 8, 9 ]:  
    for j in [ 1, 2, 3, 4, 5, 6, 7, 8, 9 ]:  
        print(f'{i} * {j} = {i*j}')
```

```
# 구구단 출력
```

```
for i in range(1,10):  
    for j in range(1, 10):  
        print(f'{i} * {j} = {i*j}')
```



The screenshot shows the Python IDLE Shell interface. The title bar says "IDLE Shell 3.11.2". The menu bar includes File, Edit, Shell, Debug, and Help. The main window has a command prompt starting with ">>>". Below the prompt, there is a separator line followed by a list of multiplication results from 1\*1 to 3\*1. The results are displayed in blue text.

```
Python 3.11.2 (tag  
Type "help", "copy"  
=====
```

1 * 1 = 1
1 * 2 = 2
1 * 3 = 3
1 * 4 = 4
1 * 5 = 5
1 * 6 = 6
1 * 7 = 7
1 * 8 = 8
1 * 9 = 9
2 * 1 = 2
2 * 2 = 4
2 * 3 = 6
2 * 4 = 8
2 * 5 = 10
2 * 6 = 12
2 * 7 = 14
2 * 8 = 16
2 * 9 = 18
3 * 1 = 3

## 중첩 for 문

# 제어 흐름: 반복 구조 (5/5)

## ● 점프문

### ○ break : 반복문 종료

- 반복문의 조건 검사를 거짓(false)으로 하는 것과 동일한 의미이다.
- 만약 중첩된 반복문을 사용한다면, 현재 위치한 가장 내부의 반복문만 종료한다.

```
for i in range(1, 11) :  
    if i==5 :  
        break  
    print(f'i: {i}')  
else : print('반복문 종료')
```

The image shows two side-by-side Python IDLE shells. The left shell shows the code execution step-by-step:  
1. The first four iterations (i=1 to 4) are printed.  
2. At i=5, the word "break" is reached, and the loop exits.  
3. The word "else" is reached, followed by the output "반복문 종료".  
The right shell shows the continuation of the loop from where it left off:  
1. It prints the remaining iterations from i=6 to 10.  
2. Finally, it prints the string "반복문 종료".

### ○ continue : 조건식 종료

- 반복문을 종료 하는 것이 아니라, 조건식으로 제어가 이동된다.

```
for i in range(1, 11) :  
    if i==5 :  
        continue  
    print(f'i: {i}')  
else : print('반복문 종료')
```

The image shows two side-by-side Python IDLE shells. The left shell shows the code execution step-by-step:  
1. The first four iterations (i=1 to 4) are printed.  
2. At i=5, the word "continue" is reached, and the loop skips the rest of the current iteration.  
3. The loop continues to the next iteration (i=6).  
4. It prints the remaining iterations from i=6 to 10.  
5. Finally, it prints the string "반복문 종료".

# 예외 처리 (1/8)

## ● 예외 처리: 전통적인 스타일의 예외 처리

### ○ 예외 처리를 하지 않은 프로그램

```
print('두 개의 정수를 입력하세요.')

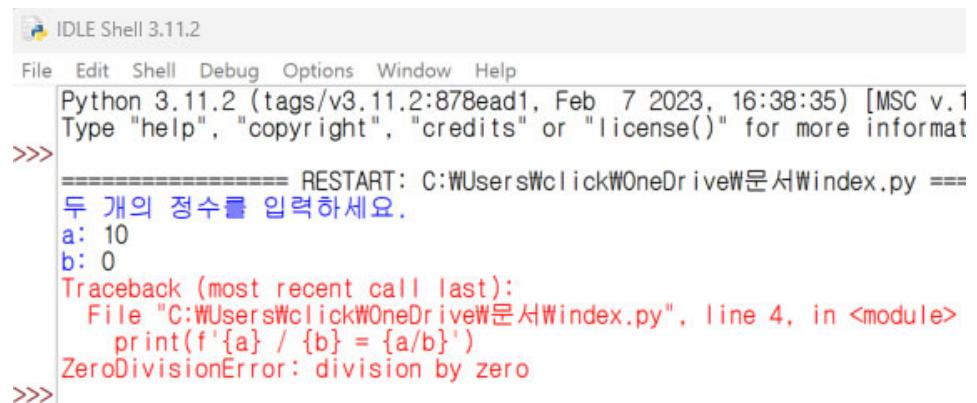
a = int(input('a: '))
b = int(input('b: '))
print(f'{a} / {b} = {a / b}')
```

```
# 전통적인 스타일의 예외 처리

if b == 0: print('입력 오류: 다시 실행하세요.')
else:      print(f'{a} / {b} = {a / b}')
```

- ZeroDivisionError: division by zero

## 전통적인 스타일의 예외 처리



```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC v.1
Type "help", "copyright", "credits" or "license()" for more information
>>> ===== RESTART: C:\Users\click\OneDrive\문서\Windex.py ====
두 개의 정수를 입력하세요.
a: 10
b: 0
Traceback (most recent call last):
  File "C:\Users\click\OneDrive\문서\Windex.py", line 4, in <module>
    print(f'{a} / {b} = {a/b}')
ZeroDivisionError: division by zero
>>>
```

# 예외 처리 (2/8)

## ● 예외 처리: try, except

### ○ try, except 문

- try 블록 수행 중 오류가 발생하면 except 블록이 수행된다.

The screenshot shows a Python script and its execution in the IDLE Shell. The script prompts for two numbers and performs division. It includes a try-except block to handle a ZeroDivisionError.

```
try: ...
except [발생오류 [as 오류변수]]:
    ...

print('두 개의 정수를 입력하세요.')
try:
    a = int(input('a: '))
    b = int(input('b: '))
    print(f'{a} / {b} = {a / b}')
except:
    print('입력오류, 다시 실행하세요.')

# 특정 예외만 처리하도록 발생 오류 명시
# except [발생오류 [as 오류변수]]:
except ZeroDivisionError as e:
    print(f'오류: {e}')
```

The shell output shows:

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window
Python 3.11.2 (tags/v3.11.2:87)
Type "help", "copyright", "cre
=====
RESTART: C:/...
두 개의 정수를 입력하세요.
a: 10
b: 3
10 / 3 = 3.333333333333335
=====
RESTART: >>>
두 개의 정수를 입력하세요.
a: 10
b: 0
입력오류, 다시 실행하세요.
```

# 예외 처리 (3/8)

## 예제 0-13: 예외 처리 -- try, except, else, finally

```
sList = [10, 20, 30, 40, 50]

print('두 개의 정수를 입력하세요.')

try:
    a = int(input('a: '))
    b = int(input('b: '))
    res = a / b

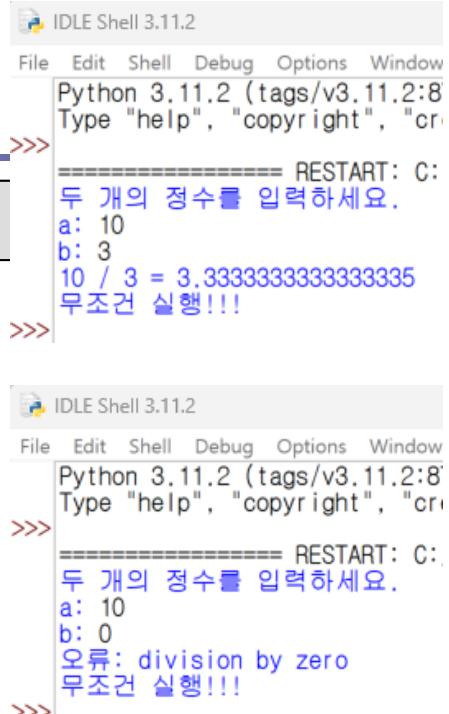
    # 여러 개의 오류 처리

    except ZeroDivisionError as e: print(f'오류: {e}') # 예외의 에러 메시지 받아오기
    except IndexError as e:         print(f'오류: {e}') # 예외의 에러 메시지 받아오기
    except FileNotFoundError:       pass                 # 오류 회피

    else:                         # try문 수행 중 오류가 없으면 else절이 수행된다.
        print(f'{a} / {b} = {res}')

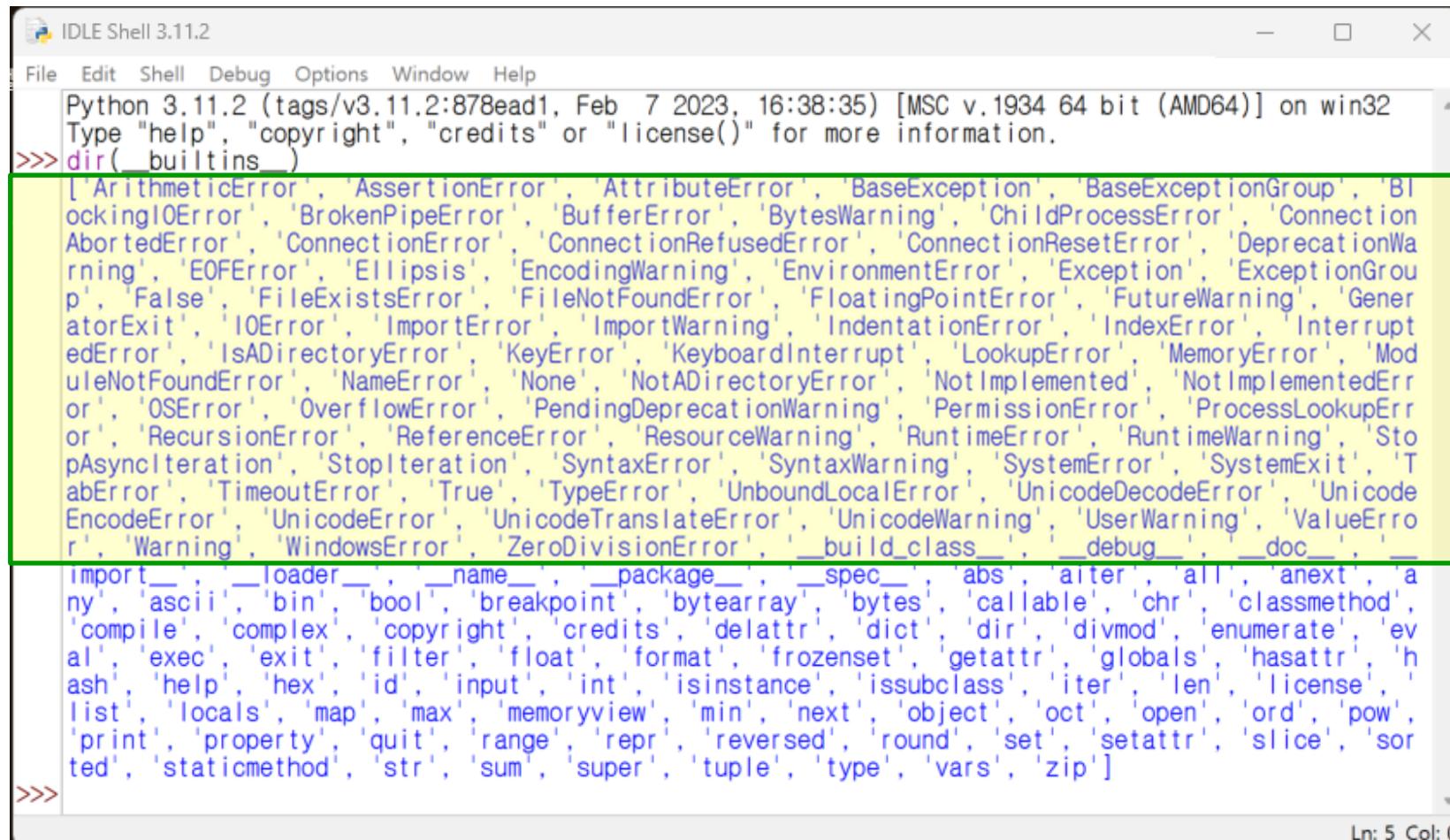
    finally:                      # 무조건 실행: Clean-up 작업 정의
        print('무조건 실행!!!!')


```



# 예외 처리 (4/8)

## ● 파이썬 내장 객체: \_\_builtins\_\_



The screenshot shows the Python 3.11.2 IDLE Shell interface. The command `>>> dir(__builtins__)` has been entered, and the resulting list of built-in exception and utility functions is displayed. A green rectangular box highlights the first part of the output, which lists various exception types. The full output is as follows:

```
'ArithError', 'AssertionError', 'AttributeError', 'BaseException', 'BaseExceptionGroup', 'BlockingIOError', 'BrokenPipeError', 'BufferError', 'BytesWarning', 'ChildProcessError', 'ConnectionAbortedError', 'ConnectionError', 'ConnectionRefusedError', 'ConnectionResetError', 'DeprecationWarning', 'EOFError', 'Ellipsis', 'EncodingWarning', 'EnvironmentError', 'Exception', 'ExceptionGroup', 'False', 'FileExistsError', 'FileNotFoundException', 'FloatingPointError', 'FutureWarning', 'GeneratorExit', 'IOError', 'ImportError', 'ImportWarning', 'IndentationError', 'IndexError', 'InterruptedError', 'IsADirectoryError', 'KeyError', 'KeyboardInterrupt', 'LookupError', 'MemoryError', 'ModuleNotFoundError', 'NameError', 'None', 'NotADirectoryError', 'NotImplemented', 'NotImplementedError', 'OSError', 'OverflowError', 'PendingDeprecationWarning', 'PermissionError', 'ProcessLookupError', 'RecursionError', 'ReferenceError', 'ResourceWarning', 'RuntimeError', 'RuntimeWarning', 'StopAsyncIteration', 'StopIteration', 'SyntaxError', 'SyntaxWarning', 'SystemError', 'SystemExit', 'TabError', 'TimeoutError', 'True', 'TypeError', 'UnboundLocalError', 'UnicodeDecodeError', 'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError', 'UnicodeWarning', 'UserWarning', 'ValueError', 'Warning', 'WindowsError', 'ZeroDivisionError', '__build_class__', '__debug__', '__doc__', '__import__', '__loader__', '__name__', '__package__', '__spec__', 'abs', 'alter', 'all', 'anext', 'any', 'ascii', 'bin', 'bool', 'breakpoint', 'bytearray', 'bytes', 'callable', 'chr', 'classmethod', 'compile', 'complex', 'copyright', 'credits', 'delattr', 'dict', 'dir', 'divmod', 'enumerate', 'eval', 'exec', 'exit', 'filter', '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', 'reversed', 'round', 'set', 'setattr', 'slice', 'sorted', 'staticmethod', 'str', 'sum', 'super', 'tuple', 'type', 'vars', 'zip']
```

Ln: 5 Col: 0

# 예외 처리 (5/8)

---

## ● 예외 처리(Exceptions Handling)

### ○ 예외(exception): 코드를 실행하는 중에 발생한 에러

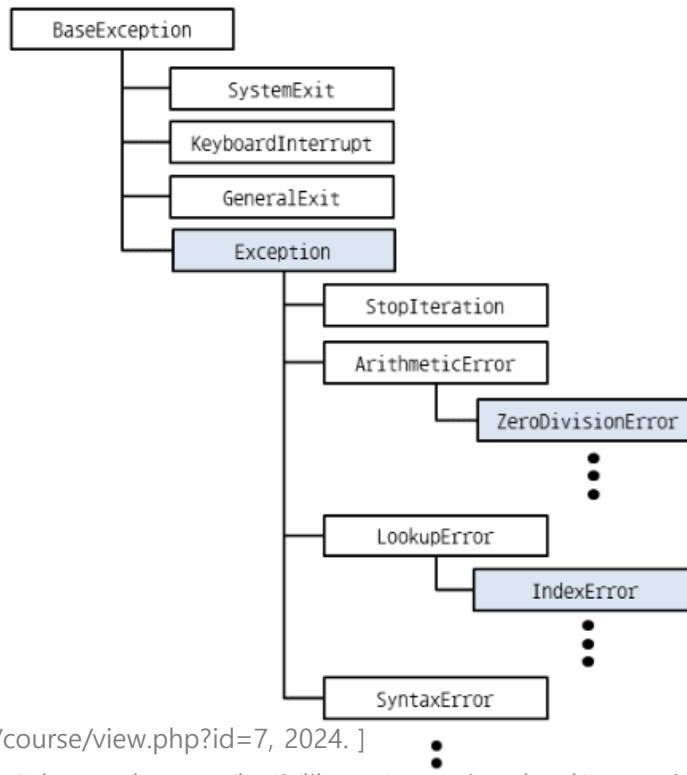
- **ValueError:** 값의 형식이 잘못된 경우 발생하는 오류
- **TypeError:** 연산 중 자료형이 다른 경우 발생하는 오류
- **ZeroDivisionError:** 숫자를 나누는 연산을 할 때 나누는 수가 0인 경우 발생하는 오류
  
- **IndexError:** 인덱싱을 할 때 얻을 수 없는 인덱스 번호를 가리키는 경우 발생하는 오류
- **KeyError:** 딕셔너리에서 없는 키를 입력할 경우 발생하는 오류
  
- **FileNotFoundException:** 존재하지 않는 파일을 불러올 때 발생하는 오류



# 예외 처리 (6/8)

## ● 예외 처리: 예외 계층도

- 파이썬 예외(Exception) 계층도: 예외도 클래스 상속으로 구현
  - 새로운 예외를 만들 때는 `Exception`을 상속받아서 구현한다.



[ 출처: "파이썬 코딩 도장", <https://dojang.io/course/view.php?id=7>, 2024. ]

[ 출처: "Python 3.13.2 documentation", <https://docs.python.org/ko/3/library/exceptions.html#exception-hierarchy>, 2025. ]

# 예외 처리 (7/8)

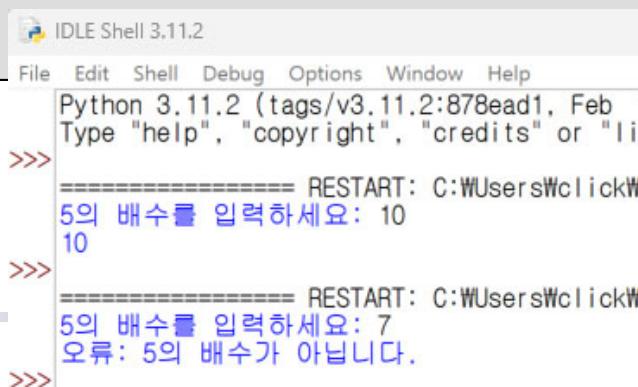
## ● 예외 처리: raise

- **raise:** 기본적인 예외 또는 사용자 정의 예외를 발생시킬 수 있다.

```
raise ExceptionType("Error message")
```

```
# 예외 처리: raise

try:
    num = int(input('5의 배수를 입력하세요: '))
    if num % 5 != 0:
        raise Exception('5의 배수가 아닙니다.') # 사용자 정의 예외를 발생시킨다.
    print(num)
except Exception as e: # 예외가 발생했을 때 실행된다.
    print(f'오류: {e}')
```



The screenshot shows a Python IDLE Shell window. The code in the shell is as follows:

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb
Type "help", "copyright", "credits" or "li
>>> ===== RESTART: C:\Users\click\w
5의 배수를 입력하세요: 10
10
>>> ===== RESTART: C:\Users\click\w
5의 배수를 입력하세요: 7
오류: 5의 배수가 아닙니다.
>>>
```

The output shows that when the user inputs 10, it prints 10. When the user inputs 7, it prints the error message "오류: 5의 배수가 아닙니다." (Error: 5's multiple).

# 예외 처리 (8/8)

## ● 예외 처리: assert

### ○ assert

```
assert condition, message
```

- condition: 테스트하려는 Bool 표현식
  - condition이 True이면 정상적으로 계속 실행한다.
  - condition이 False이면 message와 함께 AssertionError를 발생시킨다.
- message: AssertionError에 대한 자세한 정보를 제공하는 내용

```
# 예외 처리: assert

a = int(input('a: '))
b = int(input('b: '))
assert b != 0, "division by zero"
print(f'{a} / {b} = {a / b}')
```

The screenshot shows the Python IDLE Shell 3.11.2 interface. A script named 'Windex.py' is running. The first run (top) shows successful division: 'a: 10', 'b: 3', and '10 / 3 = 3.3333333333333335'. The second run (bottom) shows an assertion error when 'b' is 0: 'a: 10', 'b: 0', followed by a traceback indicating the error occurred at line 3: 'assert b != 0, "division by zero"'.



# 파이썬 프로그래밍 기초

## 함수

: 사용자 정의 함수, 지역 변수와 전역 변수



# 사용자 정의 함수 (1/7)

## ● 사용자 정의 함수: 함수 정의 및 호출

### ○ 함수 정의 및 호출

```
# 파이썬 첫 번째 프로그램
```

```
print('Hello World!!!')
```

```
# 함수 정의
```

```
def myFunc():
```

```
    print('Hello World!!!')
```

```
# 함수 호출
```

```
myFunc()
```



The screenshot shows the Python 3.11.2 shell interface. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the command line with three red '>>>' prompts. The first prompt shows the Python version and help information. The second prompt shows the path to the restart file. The third prompt shows the output of the 'Hello World!!!' print statement.

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb
Type "help", "copyright", "credits" or "li
=====
>>> ===== RESTART: C:\Users\clickw
>>> Hello World!!!
```

```
# 함수 정의
```

```
def myFunc():
```

```
    pass
```

```
        # pass 예약어
```

# 사용자 정의 함수 (2/7)

## 예제 0-12: 사용자 정의 함수 -- ADD 함수

```
# 함수 정의
def ADD(a, b):
    tot = a + b
    return tot          # 반환 값: tot
# return a + b

# a, b = map(int, input().split())
print('임의의 정수를 입력하세요... ')
a = int(input('a: '))
b = int(input('b: '))

# 함수 호출
sum = ADD(a, b)
print(f'{a} + {b} = {sum}' )
```

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb
Type "help", "copyright", "credits" or "li
>>> ===== RESTART: C:\Users\click\W
임의의 정수를 입력하세요...
a: 10
b: 20
10 + 20 = 30
>>>
```

# 람다 표현식(Lambda Expression)

# 특정한 기능을 수행하는 함수를 한 줄에 작성 할 수 있다는 특징이 있다.

```
sum = (lambda a, b: a+b)(a, b)
```

# 사용자 정의 함수 (3/7)

## 예제 0-13: 사용자 정의 함수 -- Cal 함수

# 함수의 반환 값은 오직 하나이다.

```
def Cal(a, b) :
    return a+b, a-b, a*b, a/b

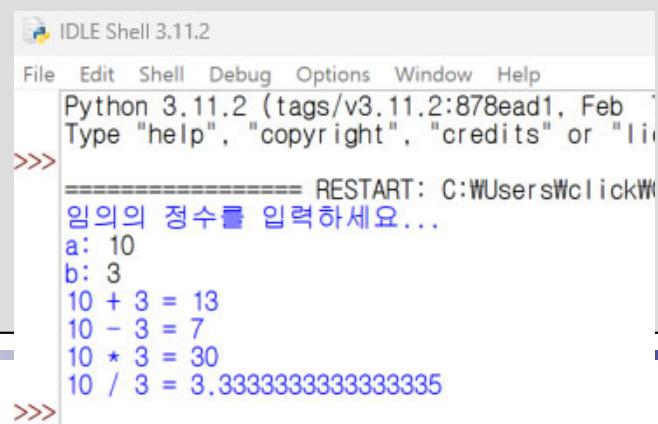
# a, b = list(map(int, input().split()))
print('임의의 정수를 입력하세요... ')
a = int(input('a: '))
b = int(input('b: '))

res = Cal(a, b)  # res: tuple 객체
print(f'{a} {b} = {res}')
```



```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb
Type "help", "copyright", "credits" or "li
>>> ===== RESTART: C:\Users\click\w
임의의 정수를 입력하세요...
a: 10
b: 3
10 3 = (13, 7, 30, 3.3333333333333335)
```

```
# res = Cal(a, b)
# print(f'{a} {b} = {res}')
add, sub, mul, div = Cal(a, b)
print(f'{a} + {b} = {add}')
print(f'{a} - {b} = {sub}')
print(f'{a} * {b} = {mul}')
print(f'{a} / {b} = {div}')
```



```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb
Type "help", "copyright", "credits" or "li
>>> ===== RESTART: C:\Users\click\w
임의의 정수를 입력하세요...
a: 10
b: 3
10 + 3 = 13
10 - 3 = 7
10 * 3 = 30
10 / 3 = 3.3333333333333335
```

# 사용자 정의 함수 (4/7)

## ● 사용자 정의 함수: 기본 매개변수

### ○ 기본 매개 변수(Default Parameter Value)

- 인수 없이 함수를 호출하면 기본값을 사용한다.

```
def myFunc(num = 1):
    for i in range(num):
        print(f'{i}: Hello World!!!')

myFunc()          # myFunc(1)
myFunc(5)
```



```
IDLE Shell 3.11.2
File Edit Shell Debug Options Windows
Python 3.11.2 (tags/v3.11.2b1-501f2cc, Feb 24 2023, 15:37:46) [MSC v.1932 64 bit (AMD64)]
Type "help", "copyright", "credits" or "license" for more information.
>>> ===== RESTART:
0: Hello World!!!
0: Hello World!!!
1: Hello World!!!
2: Hello World!!!
3: Hello World!!!
4: Hello World!!!
>>>
```

```
def ADD(a, b = 20):
    return a + b

# sum = ADD()          # TypeError
sum = ADD(10)          # sum = ADD(10, 50)
sum = ADD(10, 50)
```

# 사용자 정의 함수 (5/7)

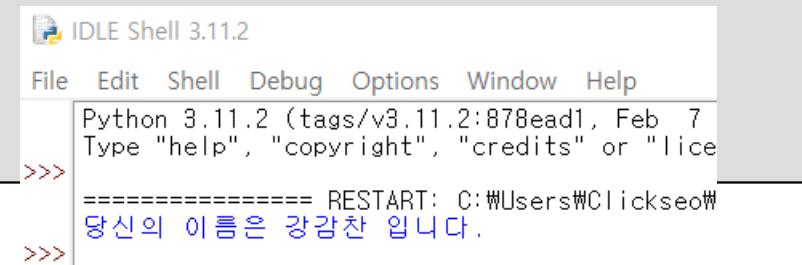
## ● 사용자 정의 함수: 임의의 인수

### ○ 임의의 인수(Arbitrary Arguments): \*args

- 함수에 전달할 인수의 수를 모르는 경우 함수 정의에서 매개변수 이름 앞에 \*를 추가한다.

```
def myFunc(*args):  
    print(f'당신의 이름은 {args[2]} 입니다.')
```

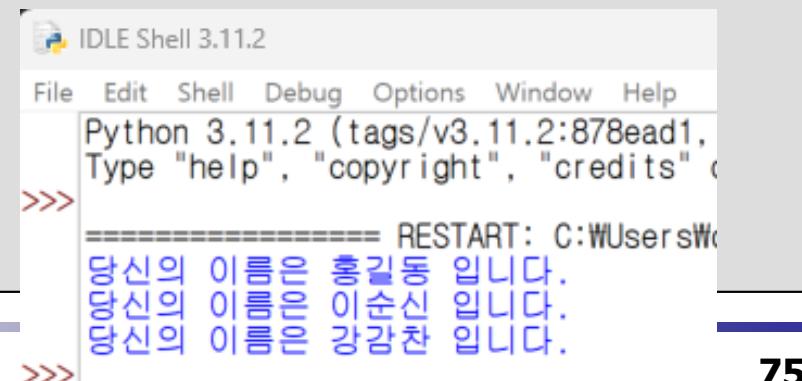
```
myFunc('홍길동', '이순신', '강감찬')
```



```
IDLE Shell 3.11.2  
File Edit Shell Debug Options Window Help  
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7  
Type "help", "copyright", "credits" or "lic  
===== RESTART: C:\Users\Clickseo#  
당신의 이름은 강감찬 입니다.  
>>>
```

```
def myFunc(*args):  
    for i in range(len(args)):  
        print(f'당신의 이름은 {args[i]} 입니다.')
```

```
myFunc('홍길동', '이순신', '강감찬')
```



```
IDLE Shell 3.11.2  
File Edit Shell Debug Options Window Help  
Python 3.11.2 (tags/v3.11.2:878ead1,  
Type "help", "copyright", "credits" o  
===== RESTART: C:\Users\W  
당신의 이름은 홍길동 입니다.  
당신의 이름은 이순신 입니다.  
당신의 이름은 강감찬 입니다.  
>>>
```

# 사용자 정의 함수 (6/7)

## ● 사용자 정의 함수: 예약어 인수

### ○ 예약어 인수 (Keyword Arguments)

```
def myFunc(a, b, c):  
    print(f'a: {a}, b: {b}, c: {c}')  
  
myFunc(10, 20, 30)  
myFunc(c=30, b=20, a=10)
```



The screenshot shows the Python IDLE Shell 3.11.2 interface. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The title bar says "IDLE Shell 3.11.2". The shell window displays the following output:

```
File Edit Shell Debug Options Window Help  
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:37:54)  
Type "help", "copyright", "credits" or "license" for more information  
=====  
RESTART: C:\Users\click\PycharmProjects\untitled\venv\Scripts\python.exe  
a: 10, b: 20, c: 30  
a: 10, b: 20, c: 30
```



# 사용자 정의 함수 (7/7)

## ● 지역 변수(local variable)

### ○ 함수 안에서 선언되는 변수

- 함수 내에서 생성된 변수는 해당 함수의 지역 범위에 속한다.
- 또한 해당 함수 내에서만 사용할 수 있습니다.

```
# 함수 정의: myFunc
def myFunc(num) :      # 지역 변수: i, num
    for i in range(num) :
        print(f'{i}: Hello World!!!')

# NameError: name 'num' is not defined. Did you mean: 'sum'?

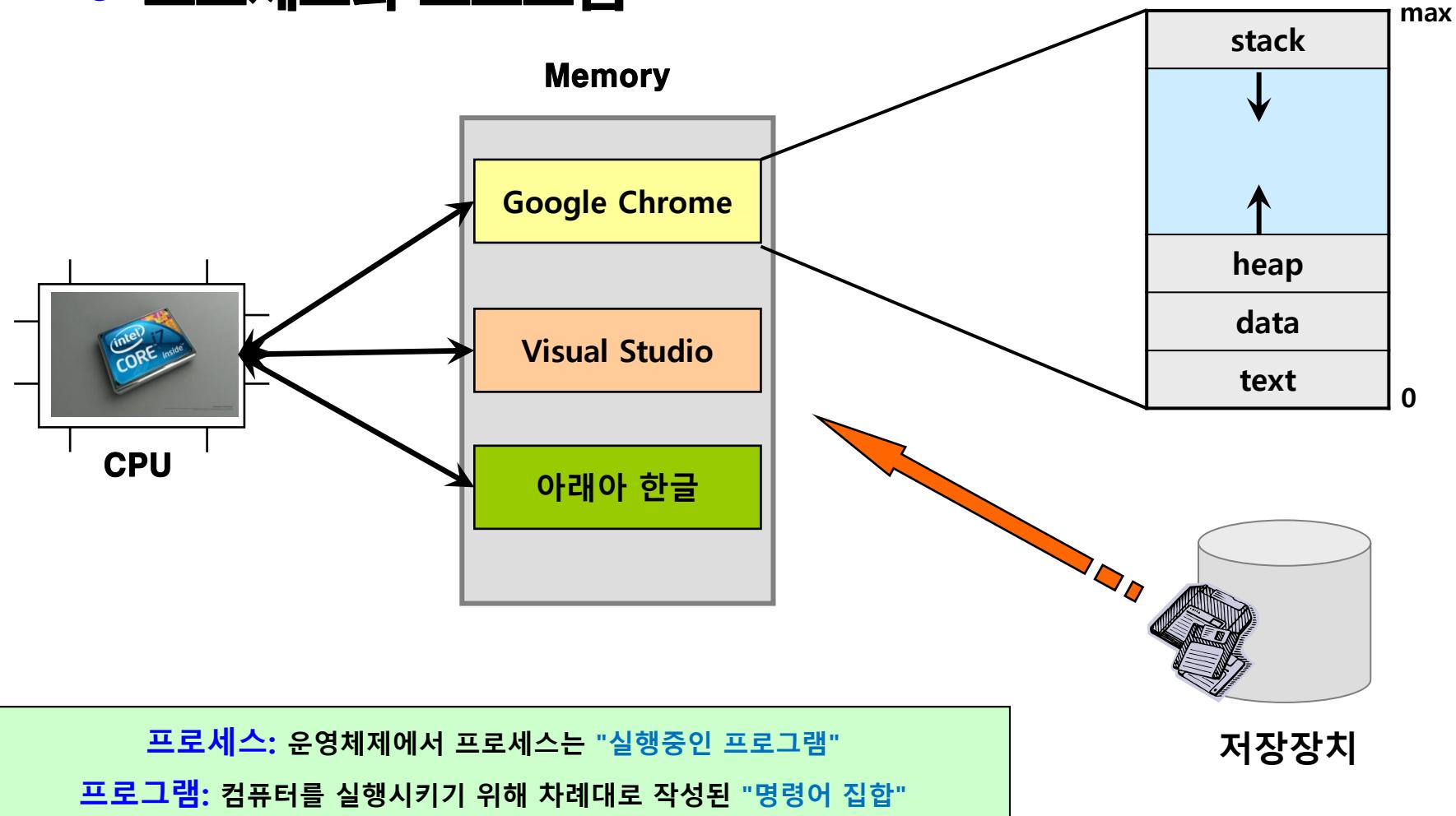
# 함수 호출
myFunc(5)
```



```
IDLE Shell 3.11.2
File Edit Shell Debug Options Windows
Python 3.11.2 (tags/v3.11.2b1-501f2cc, Feb 24 2023, 15:37:46)
Type "help", "copyright", "credits" or "license" for more information
>>> ===== RESTART:
0: Hello World!!!
1: Hello World!!!
2: Hello World!!!
3: Hello World!!!
4: Hello World!!!
>>>
```

# 지역 변수와 전역 변수 (1/6)

## ● 프로세스와 프로그램



# 지역 변수와 전역 변수 (2/6)

## ● 지역 변수(local variable)

### ○ 함수 안에서 선언되는 변수

- 함수 내에서 생성된 변수는 해당 함수의 지역 범위에 속한다.
- 또한 해당 함수 내에서만 사용할 수 있습니다.

```
# 함수 정의: myFunc
def myFunc(num):      # 지역 변수: i, num
    for i in range(num):
        print(f'{i}: Hello World!!!')

# NameError: name 'num' is not defined. Did you mean: 'sum'?

# 함수 호출
myFunc(5)
```



```
IDLE Shell 3.11.2
File Edit Shell Debug Options Windows
Python 3.11.2 (tags/v3.11.2b1-501f0876-2022-10-11 14:48:53)
Type "help", "copyright", "credits" or "license" for more information
>>> ===== RESTART:
0: Hello World!!!
1: Hello World!!!
2: Hello World!!!
3: Hello World!!!
4: Hello World!!!
>>>
```

# 지역 변수와 전역 변수 (3/6)

## ● 지역 변수

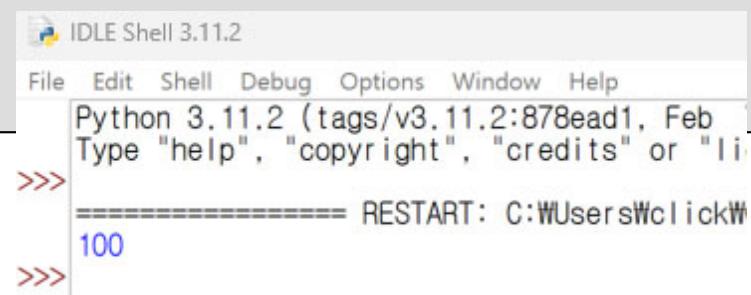
### ○ 함수 내부 함수(Function Inside Function)

- 함수 내부의 모든 함수에서 사용할 수 있다.

```
# 함수 정의: myFunc

def myFunc():
    num = 100          # 지역 변수: num
    def myInnerFunc(): # 함수 내부 함수 정의
        print(num)      # 지역 변수(num) 출력
    myInnerFunc()       # 함수 내부 함수 호출

# 함수 호출
myFunc()
```



# 지역 변수와 전역 변수 (4/6)

## ● 전역 변수(global variable)

### ○ 함수 외부에 선언되는 변수

- 전역 변수는 전역 및 지역의 모든 범위 내에서 사용할 수 있습니다.

```
# 전역 변수: num  
  
num = 100  
  
# 함수 정의: myFunc  
  
def myFunc():  
    print(num)          # 전역 변수(num) 출력  
  
# 함수 호출  
  
myFunc()  
print(num)          # 전역 변수(num) 출력
```



# 지역 변수와 전역 변수 (5/6)

## ● 전역 변수: 지역 변수와 전역 변수

### ○ 지역 변수와 전역 변수

- 함수 내부와 외부에서 동일한 변수 이름

```
# 함수 정의: myFunc
def myFunc(num):      # 지역 변수: num
    print(num)         # 지역 변수(num) 출력

# 전역 변수: num
num = 100
print(num)             # 전역 변수(num) 출력

myFunc(5)
```



```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb
Type "help", "copyright", "credits" or "li
>>>
=====
RESTART: C:\Users\swclick\w
100
5
>>>
```

# 지역 변수와 전역 변수 (6/6)

## ● 전역 변수: global 예약어

### ○ global 예약어

- global 예약어를 사용하는 경우 변수는 전역 범위에 속한다.

```
# 함수 정의: myFunc
def myFunc():
    global num          # 전역 변수: num
    num = 100

# 함수 호출
myFunc()

# 전역 변수(num) 출력
print(num)
```



```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb
Type "help", "copyright", "credits" or "li
>>>
=====
RESTART: C:\Users\click\w
100
>>>
```



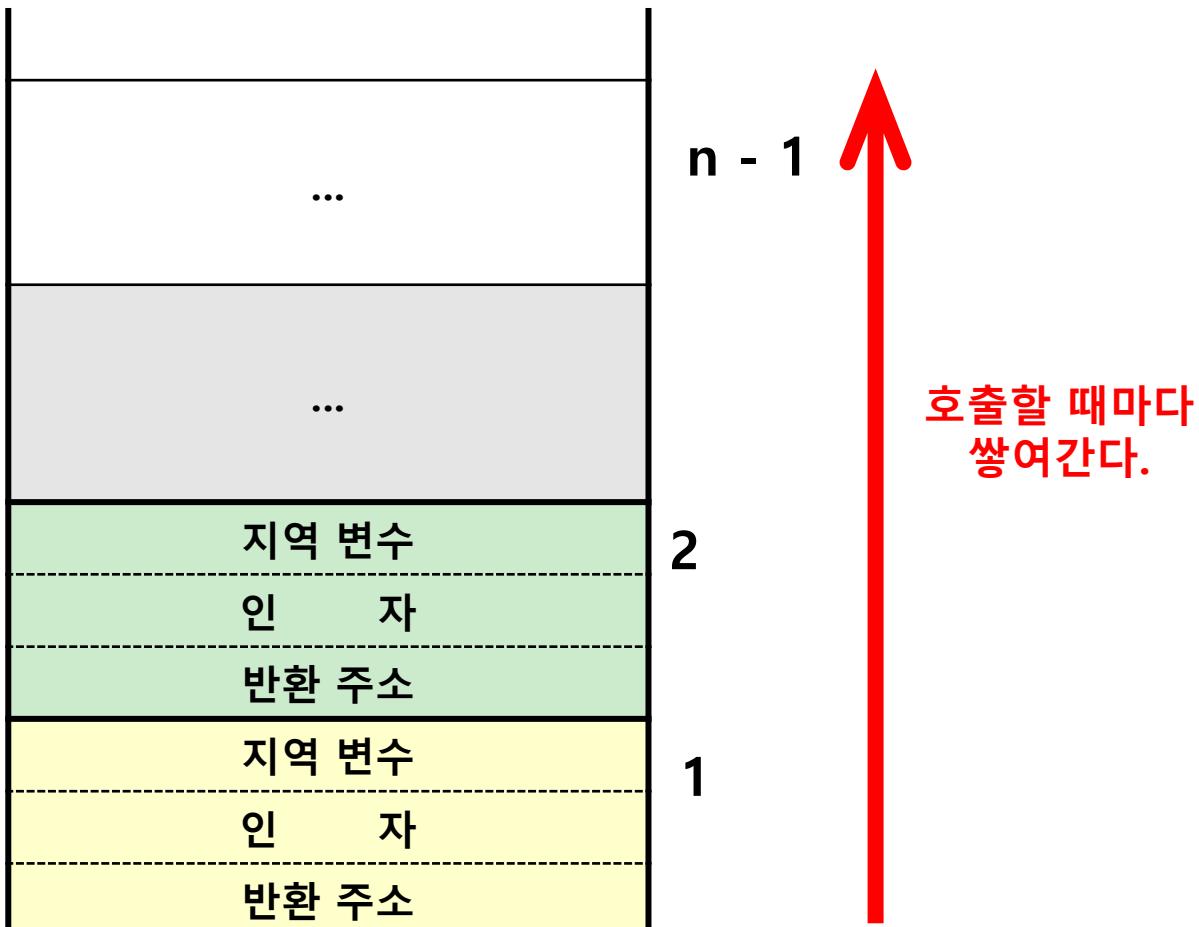
# 파이썬 프로그래밍 기초

함수: 재귀 함수, 내장 함수



# 재귀 함수

- 함수 호출 시 동작 과정: 스택 구조



# 반복적.재귀적 용법 (1/4)

---

## ● 재귀적 해법

- 큰 문제에 닮음 꼴의 작은 문제가 깃든다.
- 잘 쓰면 보약, 잘못 쓰면 맹독
  - 관계중심으로 파악함으로써 문제를 간명하게 볼 수 있다.
  - 재귀적 해법을 사용하면 심한 중복 호출이 일어나는 경우가 있다.
- 재귀적 해법이 바람직한 예
  - 계승(factorial) 구하기
  - 퀸 정렬, 병합 정렬 등의 정렬 알고리즘
  - 그래프의 깊이 우선 탐색(DFS, Depth First Search)
- 재귀적 해법이 치명적인 예
  - 피보나치 수 구하기
  - 행렬 곱셈 최적순서 구하기

# 반복적.재귀적 용법 (2/4)

## ● 계승(Factorial) 구하기

### ○ 반복적 정의

- 반복 함수가 반복적으로 정의된다.
  - 함수 정의는 매개변수를 포함하나 함수 자체는 포함하지 않는다.

$$\text{factorial}(n) = \begin{cases} 1 & \text{if } n = 0 \\ n * (n-1) * (n-2) \dots 3 * 2 * 1 & \text{if } n > 0 \end{cases}$$

### ○ 재귀적 정의

- 함수가 자기 자신을 포함한다.

$$\text{factorial}(n) = \begin{cases} 1 & \text{if } n = 0 \\ n * \text{factorial}(n-1) & \text{if } n > 0 \end{cases}$$

$\Theta(n)$

# 반복적.재귀적 용법 (3/4)

## 예제 0-14: 계승(Factorial) 구하기

```
# 재귀적 용법: Factorial
def Factorial(num):
    if num==0:          # 재귀 함수 탈출(종료) 조건
        return 1
    return num * Factorial(num-1)

if __name__ == '__main__':
    num = int(input('임의의 정수: '))
    print(f'Factorial {num}: {Factorial(num)}')
    # import math
    # print(f'Factorial {num}: {math.factorial(num)}')
```

IDLE Shell 3.10.4  
File Edit Shell Debug Options Window  
Python 3.10.4 (tags/v3.10.4:  
Type "help", "copyright", "c  
===== RESTART:  
=>>> 임의의 정수: 5  
=>>> Factorial 5: 120

```
# 반복적 용법: Factorial
def Factorial(num):
    res = 1
    for i in range(1, num+1):
        res = res * i
    return res
```



# 반복적.재귀적 용법 (4/4)

## 예제 0-15: 1부터 n까지의 합

```
# 재귀적 용법: 1부터 n까지의 합
```

```
def SUM(num:int) -> int:  
    if(num < 2) : return 1  
    return num + SUM(num-1)
```

```
if __name__ == '__main__' :  
    num = int(input('임의의 정수 입력: '))  
    print(f'{SUM(num)}', end='')
```

```
IDLE Shell 3.10.4  
File Edit Shell Debug Options Window I  
Python 3.10.4 (tags/v3.10.4:9d3  
Type "help", "copyright", "cred  
===== RESTART: C:\W  
임의의 정수 입력: 10  
55
```

```
# 반복적 용법: 1부터 n까지의 합
```

```
def SUM(num:int) -> int:  
    tot = 0  
    for i in range(1, num+1):  
        tot += i  
    return tot          # return sum(range(1, num+1))  
    # return num * (num+1) // 2
```

## 알고리즘 분석

$$sum(n) = 1 + 2 + 3 + \dots + (n - 1) + n$$

# O(n)

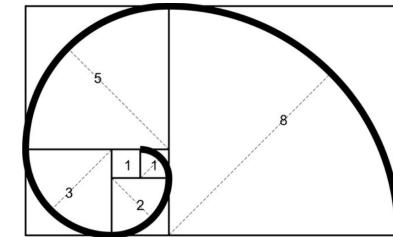
# O(1)

# 피보나치 수열 (1/3)

## ● 피보나치 수열(Fibonacci Sequence)

### ○ 피보나치(Fibonacci)

- 1,200년 경에 활동한 이탈리아 수학자



“토끼 한 마리가 매년 새끼 한 마리를 낳는다.

새끼는 한 달 후부터 새끼를 낳기 시작한다.

최초 토끼 한 마리가 있다고 하면...

한 달 후에 토끼는 두 마리가 되고 두 달 후에는 세 마리가 되고...”

// 재귀적 용법: 피보나치 수열

**Fibonacci(num)**

{

**if (num = 1 or num = 2)**

**then return 1;**

**else**

**return (Fibonacci(num - 1) + Fibonacci(num - 2));**

}

$$f_n = f_{n-1} + f_{n-2} \quad (n \geq 3)$$

$$f_1 = f_2 = 1 \quad (n = 1, 2)$$

“아주 간단한 문제지만...

동적 프로그래밍의 동기와 구현이 다 포함되어 있다.”

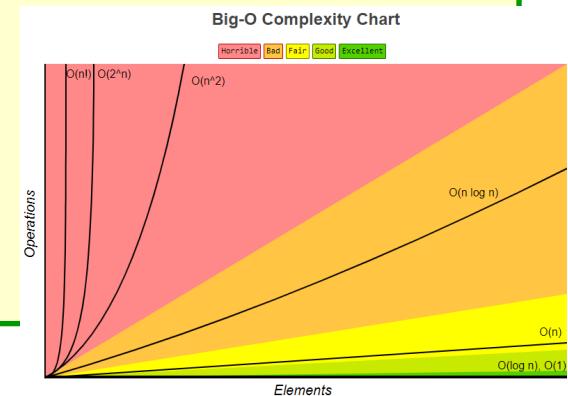
# 피보나치 수열 (2/3)

## 연습문제 0-16: 피보나치 수열 -- 재귀적 용법

| Python

```
# 재귀적 용법: 피보나치 수열
def Fibo(num):
    # 재귀 함수: 탈출 조건
    if num==1 or num==2:
        return 1
    return Fibo(num-1) + Fibo(num-2) # O(2^n)
```

```
if __name__ == '__main__':
    print('### 피보나치 수열 구하기 ###')
    num = int(input('몇 번째 수열까지 출력할까요: '))
    for i in range(1, num+1):
        if i%5: print(f'{Fibo(i):8}', end=' ')
    else: print(f'{Fibo(i):8}')
```



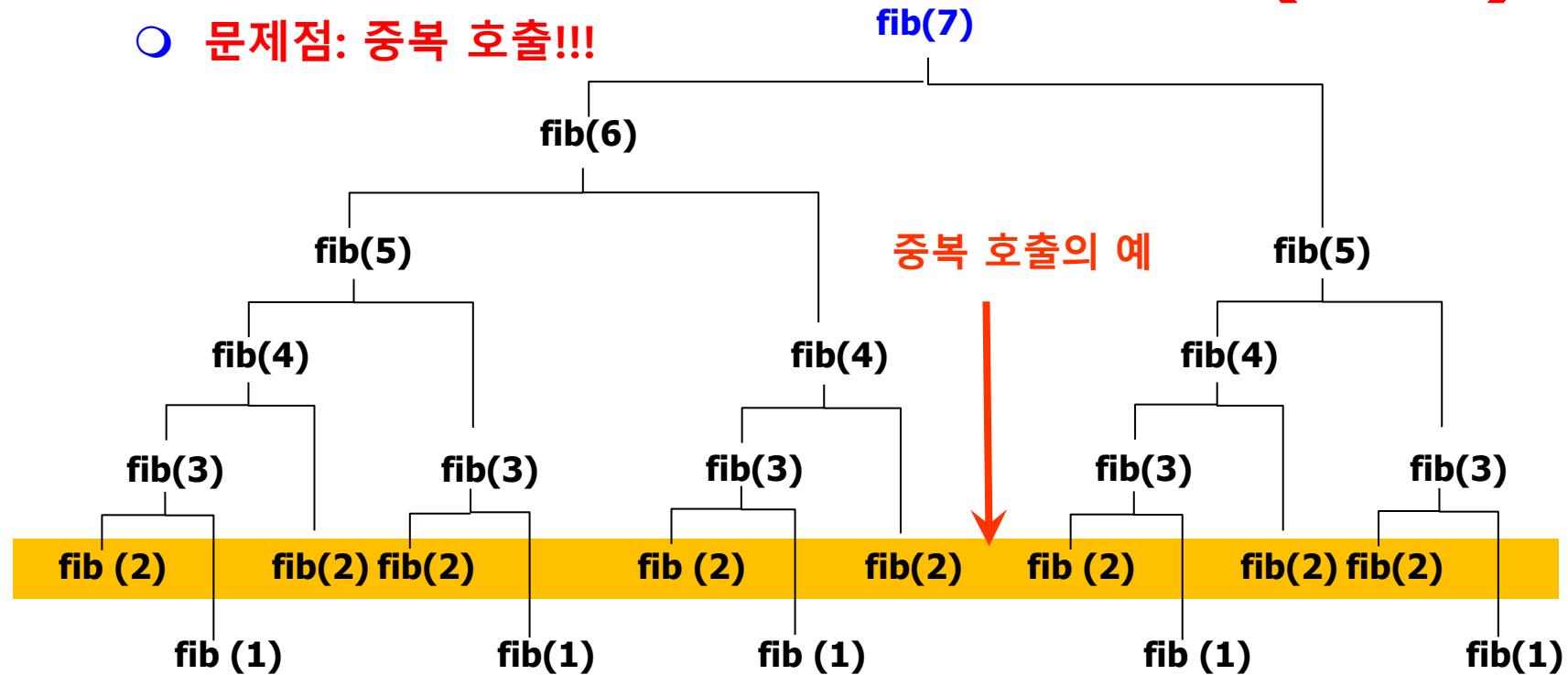
```
IDLE Shell 3.10.4
File Edit Shell Debug Options Window Help
Python 3.10.4 (tags/v3.10.4:9d3f8120, Mar 23 21)
Type "help", "copyright", "credits" or "license" for more information.
>>> ===== RESTART: C:/Users/Clickseo/I
### 피보나치 수열 구하기 ###
몇 번째 수열까지 출력할까요: 35
      1      1      2      3      5
      8     13     21     34     55
     89    144    233    377    610
    987   1597   2584   4181   6765
   10946  17711  28657  46368  75025
  121393 196418 317811 514229 832040
 1346269 2178309 3524578 5702887 9227465
>>>
```

# 피보나치 수열 (3/3)

- 피보나치 수열: 재귀적 용법

- 문제점: 중복 호출!!

$O(2^n)$



“엄청난 중복 호출이 존재한다.”

--> 재귀적 알고리즘은 지수함수에 비례하는 시간이 든다.

# 피보나치 수열: 동적 프로그래밍 (1/2)

## ● 동적 프로그래밍의 적용 조건

### ○ 최적 부분 구조(Optimal Substructure)

- 큰 문제의 해답에 그보다 작은 문제의 해답이 포함되어 있다.
  - 최적 부분 구조를 가진 문제의 경우에는 재귀 호출을 이용하여 문제를 풀 수 있다.

### ○ 재귀 호출 시 중복(overlapping recursive calls)

- 재귀적으로 구현했을 때 중복 호출로 심각한 비효율이 발생한다.

동적 프로그래밍이 그 해결책 !!!

- 위의 두 성질이 있는 문제에 대해 적절한 저장 방법으로 중복 호출의 비효율을 제거한 것을 동적 프로그래밍이라고 한다.

# 피보나치 수열: 동적 프로그래밍 (2/2)

## ● 피보나치 수열: 동적 프로그래밍

Fibonacci(n)

```
{  
    f[1] ← f[2] ← 1;  
    for i ← 3 to n  
        f[i] ← f[i - 1] + f[i - 2];  
  
    return f[n];  
}
```

fibo

1	1	?	?	?	?	?
---	---	---	---	---	---	---

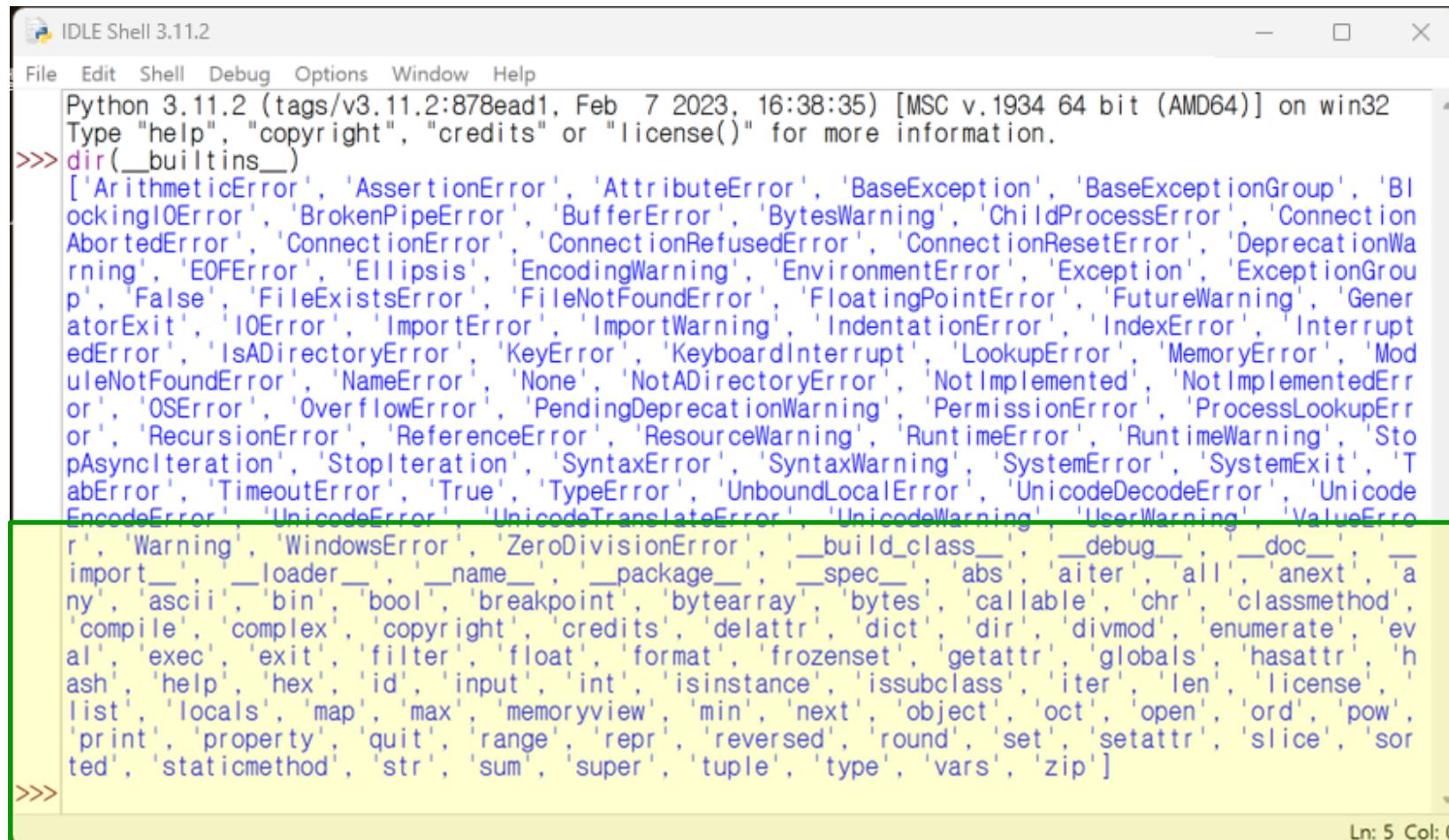
$\Theta(n)$

Fibonacci(n)

```
{  
    first ← second ← 1;  
    for i ← 3 to n  
        res ← first + second;  
        return res;  
}
```

# 내장 함수 (1/2)

## ● 파이썬 내장 객체: \_builtins\_



```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> dir(__builtins__)
['ArithError', 'AssertionError', 'AttributeError', 'BaseException', 'BaseExceptionGroup', 'BlockingIOError', 'BrokenPipeError', 'BufferError', 'BytesWarning', 'ChildProcessError', 'ConnectionAbortedError', 'ConnectionError', 'ConnectionRefusedError', 'ConnectionResetError', 'DeprecationWarning', 'EOFError', 'Ellipsis', 'EncodingWarning', 'EnvironmentError', 'Exception', 'ExceptionGroup', 'False', 'FileExistsError', 'FileNotFoundException', 'FloatingPointError', 'FutureWarning', 'GeneratorExit', 'IOError', 'ImportError', 'ImportWarning', 'IndentationError', 'IndexError', 'InterruptedError', 'IsADirectoryError', 'KeyError', 'KeyboardInterrupt', 'LookupError', 'MemoryError', 'ModuleNotFoundError', 'NameError', 'None', 'NotADirectoryError', 'NotImplemented', 'NotImplementedError', 'OSError', 'OverflowError', 'PendingDeprecationWarning', 'PermissionError', 'ProcessLookupError', 'RecursionError', 'ReferenceError', 'ResourceWarning', 'RuntimeError', 'RuntimeWarning', 'StopAsyncIteration', 'StopIteration', 'SyntaxError', 'SyntaxWarning', 'SystemError', 'SystemExit', 'TabError', 'TimeoutError', 'True', 'TypeError', 'UnboundLocalError', 'UnicodeDecodeError', 'UnicodeEncodeError', 'UnicodeError', 'UnicodeTranslateError', 'UnicodeWarning', 'UserWarning', 'ValueError', 'Warning', 'WindowsError', 'ZeroDivisionError', '__build_class__', '__debug__', '__doc__', '__import__', '__loader__', '__name__', '__package__', '__spec__', 'abs', 'aiter', 'all', 'anext', 'any', 'ascii', 'bin', 'bool', 'breakpoint', 'bytearray', 'bytes', 'callable', 'chr', 'classmethod', 'compile', 'complex', 'copyright', 'credits', 'delattr', 'dict', 'dir', 'divmod', 'enumerate', 'eval', 'exec', 'exit', 'filter', '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', 'reversed', 'round', 'set', 'setattr', 'slice', 'sorted', 'staticmethod', 'str', 'sum', 'super', 'tuple', 'type', 'vars', 'zip']
>>>
```

Ln: 5 Col: 0

# 내장 함수 (2/2)

## ● 내장 함수(Built-in Functions)

Built-in Functions			
A abs() aiter() all() any() anext() ascii()	E enumerate() eval() exec()	L len() list() locals()	R range() repr() reversed() round()
B bin() bool() breakpoint() bytearray() bytes()	F filter() float() format() frozenset()	M map() max() memoryview() min()	S set() setattr() slice() sorted() staticmethod() str() sum() super()
C callable() chr() classmethod() compile() complex()	G getattr() globals()	N next()	T tuple() type()
D delattr() dict() dir() divmod()	H hasattr() hash() help() hex()	O object() oct() open() ord()	V vars()
	I id() input() int() isinstance() issubclass() iter()	P pow() print() property()	Z zip()
			_import_()

[ 출처: "Built-in Functions", Python 3.13.2 documentation, <https://docs.python.org/3/library/functions.html>, 2025. ]

# 내장 함수: iterable (1/6)

## ● 내장 함수: iterable

### ○ 반복 가능한 객체(iterable)

- `all(iterable)`

– iterable의 모든 요소 true인 경우(또는 iterable이 비어 있는 경우) True를 반환한다.

```
def all(iterable):  
    for element in iterable:  
        if not element:  
            return False  
    return True
```

```
def any(iterable):  
    for element in iterable:  
        if element:  
            return True  
    return False
```

The screenshot shows the Python IDLE shell interface. It displays two sets of code examples for the `all()` and `any()` functions. The top set for `all()` shows it returning `True` for non-empty iterables and `False` for empty ones. The bottom set for `any()` shows it returning `True` as soon as it finds a truthy value in the iterable, and `False` if it iterates through the entire iterable without finding one.

```
IDLE Shell 3.11.2  
File Edit Shell Debug Options Window Help  
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 1 2023, 10:00:29)  
Type "help", "copyright", "credits" or "license()" for more information.  
>>> all('test')  
True  
>>> all([1,2,3,4,5])  
True  
>>> all([0,1,2,3,4,5])  
False  
>>> all({9:'A', 8:'B', 7:'C', 6:'D'})  
True  
>>> all({9:'A', 8:'B', 7:'C', 6:'D', 0:'F'})  
False  
  
File Edit Shell Debug Options Window Help  
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 10:00:29)  
Type "help", "copyright", "credits" or "license()" for more information.  
>>> any('012345')  
True  
>>> any([0,1,2,3,4,5])  
True  
>>> any({9:'A', 8:'B', 7:'C', 6:'D', 0:'F'})  
True  
>>>
```

- `any(iterable)`

– iterable의 요소 중 하나라도 true이면 True를 반환한다. 또한 iterable이 비어 있으면 False를 반환한다.

# 내장 함수: iterable (2/6)

## ● 내장 함수: iterable

### ○ 반복 가능한 객체: 수학 함수

# 총합 계산

- `sum(iterable, /, start=0)`

# 최댓값 계산

- `max(iterable, *, key=None)`
- `max(iterable, *, default, key=None)`
- `max(arg1, arg2, *args, key=None)`

# 최솟값 계산

- `min(iterable, *, key=None)`
- `min(iterable, *, default, key=None)`
- `min(arg1, arg2, *args, key=None)`



# 내장 함수: iterable (3/6)

## ● 내장 함수: iterable

### ○ 반복 가능한 객체: 정렬, 역방향

# 새로운 정렬된 목록 반환

- **sorted(iterable, /, \*, key=None, reverse=False)**

# 역방향 반복자를 반환

- **reversed(seq)**

```
aList = [50, 30, 90, 10, 20, 80]
bList = ['Spring', 'Summer', 'Fall', 'Winter']

print( sorted(aList) )
print( sorted(bList) )

print( reversed(aList) )
print( reversed(bList) )

print( list(reversed(aList)) )
print( list(reversed(bList)) )
```

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:00:00)
Type "help", "copyright", "credits" or "license()" for more information
>>> ===== RESTART: C:\Users\Clickseo\OneDrive\
[10, 20, 30, 50, 80, 90]
['Fall', 'Spring', 'Summer', 'Winter']
<list_reverseiterator object at 0x000002B3ECE22380>
<list_reverseiterator object at 0x000002B3ECE22380>
[80, 20, 10, 90, 30, 50]
['Winter', 'Fall', 'Summer', 'Spring']
```

# 내장 함수: iterable (4/6)

## ● 내장 함수: iterable

### ○ 반복 가능한 객체: 열거, 병렬 반복

# 열거 객체 반환

- `enumerate(iterable, start=0)`

# 여러 반복 가능 항목을 병렬로 반복하여 각 항목이 포함된 튜플을 생성한다.

- `zip(*iterables, strict=False)`

```
seasons = ['Spring', 'Summer', 'Fall', 'Winter']

print( enumerate(seasons) )
print( list(enumerate(seasons)) )
print( list(enumerate(seasons, start=1)) )

for item in zip([1, 2, 3], ['sugar', 'spice', 'everything nice']):
    print(item)
```

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.
Type "help", "copyright", "credits" or "license()" for more information
>>> ===== RESTART: C:\Users\Clickseo\OneDrive\문서\index.py
<enumerate object at 0x000002D09B4C7CE0>
[(0, 'Spring'), (1, 'Summer'), (2, 'Fall'), (3, 'Winter')]
[(1, 'Spring'), (2, 'Summer'), (3, 'Fall'), (4, 'Winter')]
(1, 'sugar')
(2, 'spice')
(3, 'everything nice')
>>>
```

# 내장 함수: iterable (5/6)

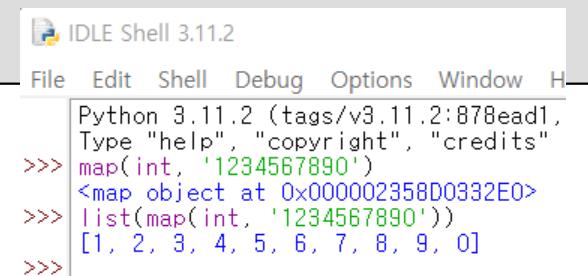
## ● 내장 함수: iterable

### ○ 반복 가능한 객체: map

# iterable의 모든 항목에 함수를 적용하여 결과를 산출하는 반복자를 반환

- `map(function, iterable, *iterables)`

```
>>> map(int, '1234567890')
>>> list( map(int, '1234567890') )
```



```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window H
Python 3.11.2 (tags/v3.11.2:878ead1,
Type "help", "copyright", "credits"
>>> map(int, '1234567890')
<map object at 0x000002358D0332E0>
>>> list(map(int, '1234567890'))
[1, 2, 3, 4, 5, 6, 7, 8, 9, 0]
>>>
```

```
print('두 개의 정수를 입력하세요: ', end=' ')
x, y = map(int, input().split())
# x, y = list(map(int, input().split()))
# map 내장 함수: 반복 가능한 객체(iterable)의 요소를 지정된 함수로 처리해주는 함수
```

# 내장 함수: iterable (6/6)

## ● 내장 함수: iterable

### ○ 내장 함수: filter

# 반복 가능한 객체에서 특정 조건에 맞는 요소만 가져온다.

- `filter(function, iterable)`

```
def f(num):
    return 5 < num < 10          # return 5 < x and x < 10

aList = [10, 5, 1, 8, 2, 3, 9, 12, 7, 16]
print( list(filter(f, aList)) )
```

```
IDLE Shell 3.11.2
File Edit Shell Debug Options
Python 3.11.2 (tags/v3.11-1619.v3.11.2)
Type "help", "copyright",
>>>
=====
RESTART:
[8, 9, 7]
```



# 내장 함수: iterator (1/5)

## ● 내장 함수: iterator

### ○ 반복자(iterator): 값을 차례대로 꺼낼 수 있는 객체

- 조작 함수: \_iter\_



The screenshot shows the Python 3.11.2 IDLE Shell interface. The user has run several `dir()` commands to list the methods available for different built-in types: `str`, `list`, `dict`, and an unnamed object. The `__iter__` method is highlighted in yellow in three of the four lists, indicating it is a common feature across these types.

```
*IDLE Shell 3.11.2*
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> dir(str)
['__add__', '__class__', '__contains__', '__delattr__', '__dir__', '__doc__', '__eq__', '__format__',
 '__ge__', '__getattribute__', '__getitem__', '__getnewargs__', '__getstate__', '__gt__', '__hash__',
 '__init__', '__init_subclass__', '__iter__', '__le__', '__len__', '__lt__', '__mod__', '__mul__',
 '__ne__', '__new__', '__reduce__', '__reduce_ex__', '__repr__', '__rmod__', '__rmul__', '__setat-
tr__', '__sizeof__', '__str__', '__subclasshook__', 'capitalize', 'casefold', 'center', 'count',
 'encode', 'endswith', 'expandtabs', 'find', 'format', 'format_map', 'index', 'isalnum', 'isalpha',
 'isascii', 'isdecimal', 'isdigit', 'isidentifier', 'islower', 'isnumeric', 'isprintable', 'isspace',
 'istitle', 'isupper', 'join', 'ljust', 'lower', 'lstrip', 'maketrans', 'partition', 'removeprefix',
 'removesuffix', 'replace', 'rfind', 'rindex', 'rjust', 'rpartition', 'rsplit', 'rstrip', 'split',
 'splitlines', 'startswith', 'strip', 'swapcase', 'title', 'translate', 'upper', 'zfill']

>>> dir(list)
['__add__', '__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__dir__',
 '__doc__', '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getstate__',
 '__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']

>>> dir(dict)
['__class__', '__class_getitem__', '__contains__', '__delattr__', '__delitem__', '__dir__', '__doc__',
 '__eq__', '__format__', '__ge__', '__getattribute__', '__getitem__', '__getstate__', '__gt__',
 '__hash__', '__init__', '__init_subclass__', '__ior__', '__iter__', '__le__', '__len__', '__lt__',
 '__ne__', '__new__', '__or__', '__reduce__', '__reduce_ex__', '__repr__', '__reversed__', '__ror__',
 '__setattr__', '__setitem__', '__sizeof__', '__str__', '__subclasshook__', 'clear', 'copy', 'from-
keys', 'get', 'items', 'keys', 'pop', 'popitem', 'setdefault', 'update', 'values']

>>>
```

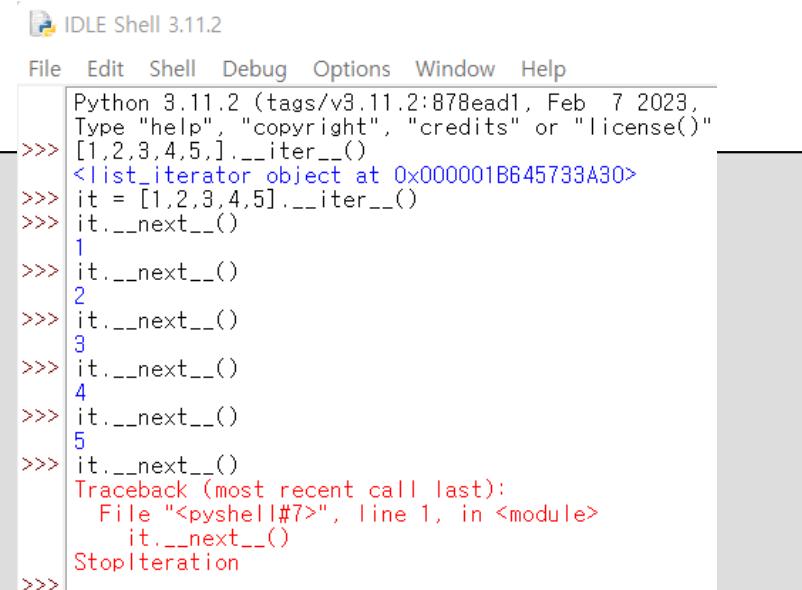
# 내장 함수: iterator (2/5)

## ● 내장 함수: iterator

### ○ 반복자: `__iter__`, `__next__`

```
>>> [1,2,3,4,5].__iter__()

>>> it = [1,2,3,4,5].__iter__()
>>> it.__next__()          # 1
>>> it.__next__()          # 2
>>> it.__next__()          # 3
>>> it.__next__()          # 4
>>> it.__next__()          # 5
>>> it.__next__()          # StopIteration 예외 발생
```



```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023,
Type "help", "copyright", "credits" or "license()"
[1,2,3,4,5].__iter__()
<list_iterator object at 0x000001B645733A30>
it = [1,2,3,4,5].__iter__()
it.__next__()
1
it.__next__()
2
it.__next__()
3
it.__next__()
4
it.__next__()
5
it.__next__()
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    it.__next__()
StopIteration
>>>
```

# for와 반복 가능한 객체

```
for i in range(1, 6):  
    print(i)
```

# range에서 `__iter__`로 반복자(iterator)를 얻는다.

# `__next__`로 숫자를 꺼내서 i에 저장한다.

# 지정된 숫자가 6이 되면 `StopIteration`을 발생시켜서 반복을 종료한다.

# 내장 함수: iterator (3/5)

## ● 내장 함수: iterator

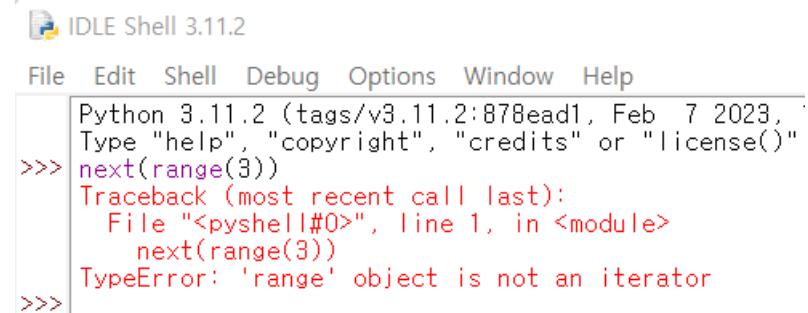
### ○ 반복자: iter, next 내장 함수

- **iter(object)**
- **iter(object, sentinel)**
- **next(iterator)**
- **next(iterator, default)**

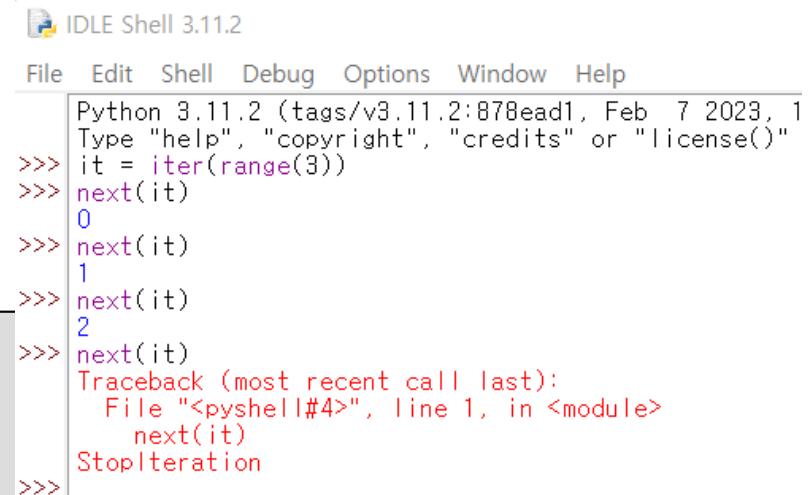
# New in version 3.10.

- **aiter(async\_iterable)**
- awaitable **anext(async\_iterator)**
- awaitable **anext(async\_iterator, default)**

```
>>> it = iter(range(3))
>>> next(it)          # 0
>>> next(it)          # 1
>>> next(it)          # 2
>>> next(it)          # StopIteration 예외 발생
```



```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 1
Type "help", "copyright", "credits" or "license()"
>>> next(range(3))
Traceback (most recent call last):
  File "<pyshell#0>", line 1, in <module>
    next(range(3))
TypeError: 'range' object is not an iterator
>>>
```



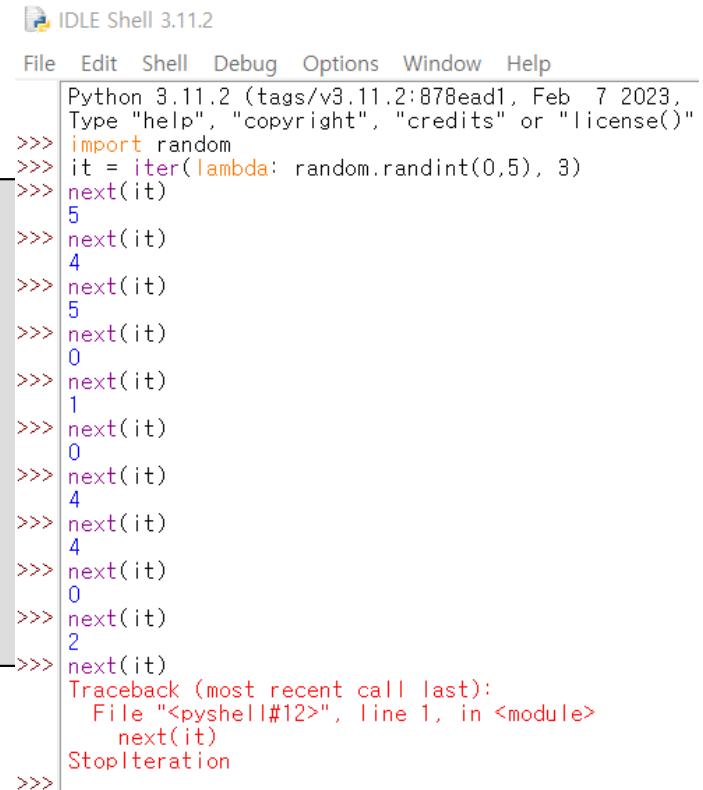
```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 1
Type "help", "copyright", "credits" or "license()"
>>> it = iter(range(3))
>>> next(it)
0
>>> next(it)
1
>>> next(it)
2
>>> next(it)
Traceback (most recent call last):
  File "<pyshell#4>", line 1, in <module>
    next(it)
StopIteration
>>>
```

# 내장 함수: iterator (4/5)

## ● 내장 함수: iterator

### ○ 반복자: iter, next 내장 함수

```
>>> import random
>>> it = iter(lambda: random.randint(0,5), 3)
>>> next(it)          # 5
>>> next(it)          # 4
>>> next(it)          # 5
>>> next(it)          # 0
>>> (생략)
>>> next(it)      # StopIteration 예외 발생
```



```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023,
Type "help", "copyright", "credits" or "license()"
>>> import random
>>> it = iter(lambda: random.randint(0,5), 3)
>>> next(it)
5
>>> next(it)
4
>>> next(it)
5
>>> next(it)
0
>>> next(it)
1
>>> next(it)
0
>>> next(it)
4
>>> next(it)
4
>>> next(it)
0
>>> next(it)
2
>>> next(it)
Traceback (most recent call last):
  File "<pyshell#12>", line 1, in <module>
    next(it)
StopIteration
>>>
```

```
# for와 반복 가능한 객체
import random
for i in iter(lambda: random.randint(0,5), 3):
    print(i)
```

# 내장 함수: iterator (5/5)

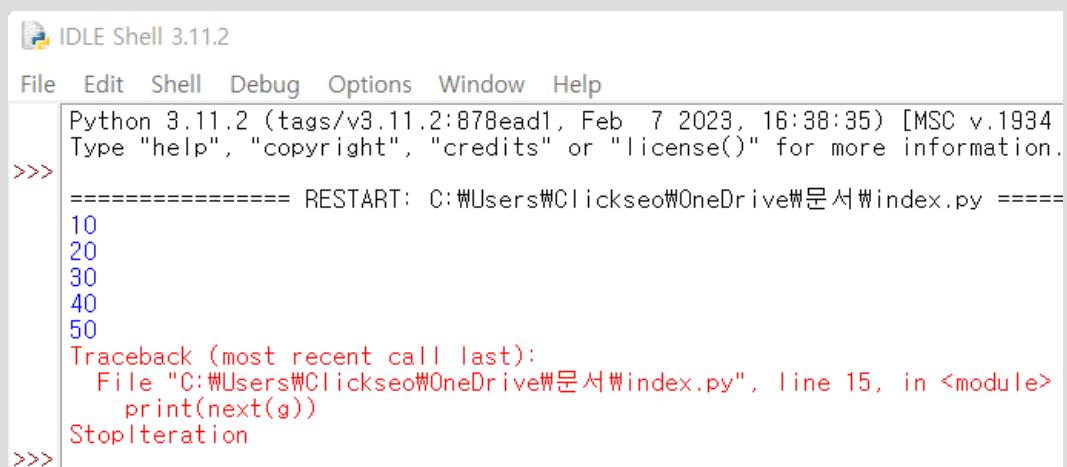
## ● 내장 함수: iterator

### ○ 발생기(generator): 반복자를 생성해 주는 함수

- 발생기(generator)로 생성한 객체는 반복자(iterator)와 마찬가지로 `next` 함수 호출 시 그 값을 차례대로 얻을 수 있다.

```
# 제너레이터 표현식 (generator expression)
# g = (i for i in range(10, 40, 10))
def myGen():
    # for i in range(10, 40, 10):
    #     yield i
    yield 10
    yield 20
    yield 30

# 제너레이터 객체 생성
g = myGen()
print(next(g))
print(next(g))
print(next(g))
print(next(g))      # StopIteration 예외 발생
```



```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934
Type "help", "copyright", "credits" or "license()" for more information.

>>> ===== RESTART: C:\Users\Clickseo\OneDrive\문서\index.py =====
10
20
30
40
50
Traceback (most recent call last):
  File "C:\Users\Clickseo\OneDrive\문서\index.py", line 15, in <module>
    print(next(g))
StopIteration
```

# 자료형과 자료구조



- 파이썬 프로그래밍 기초

- 자료형과 자료구조

백문이 불여일타(百聞而不如一打)

- 문자열: 문자열 조작 함수

- 리스트와 튜플

- 딕셔너리와 셋

- 객체지향 프로그래밍

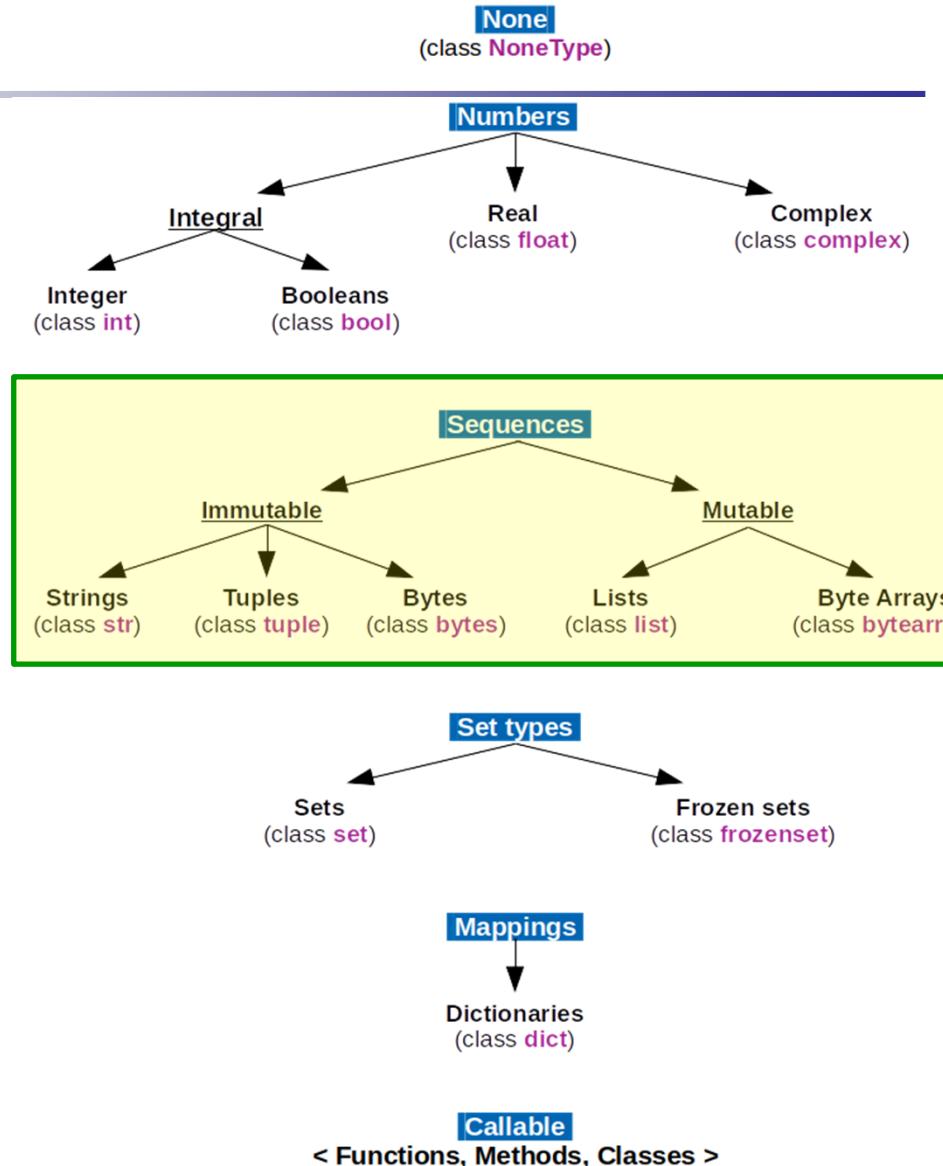
- 파이썬 라이브러리



# 자료형

## ● 자료형(Data Types)

- **NoneType**: None
- 논리형: bool
- 숫자형
  - 정수: int
  - 실수: float
  - 복소수: complex
- 문자열: str
- 리스트(List)
- 튜플(Tuple)
- 사전(Dictionary)
- 집합(Set)





# 자료형과 자료구조

문자열: 문자열 조작 함수



# 문자열의 이해

## ● 문자열(String)

○ 문자열(string)은 문자들의 나열(sequence of characters)이다.

- 문자열을 만드는 방법
  - 큰 따옴표
  - 작은 따옴표

```
print('Hello World!!!')
print("Hello World!!!")
class str
# 문자열 사용시 주의할 점
print("Hello World!!!") # SyntaxError: unterminated string literal
```

- 큰따옴표(")로 시작했다가 작은따옴표(')로 끝내면 문법적인 오류이다.
- **str** 내장 함수: 문자열 형태로 객체를 반환한다.
  - class str(object='')
  - class str(object=b'', encoding='utf-8', errors='strict')

# b를 붙이면 바이트(bytes) 객체가 된다.

# 문자열 보간: 형식 문자열

## ● 문자열 보간: 형식 문자열

### ○ 다양한 형식을 사용하여 데이터 값을 문자열을 출력할 수 있다.

- 옛 스타일: % -- 파이썬 2.3에서 지원
  - `formatString % data` 형식
  - 형식 문자열(format string) 안에 끼워 넣을 데이터를 표시하는 형식
  - 변환 코드: %d, %o, %x, %c, %s, %f, %e, %g, %%
- 새 스타일: {}, 문자열 조작 함수(`format`) -- 파이썬 2.6 이상에서만 지원
  - `formatString.format(data)` 형식
  - `format` 함수의 인수는 형식 문자열 내의 {} 순서대로 나타낸다.
  - 왼쪽 정렬(<, 기본값), 오른쪽 정렬(>), 가운데 정렬(^)
- 최신 스타일: f-문자열 -- 파이썬 3.6 이상에서만 지원
  - 첫 번째 인용 부호(문자열) 앞에 문자 `f` 또는 `F`를 붙이면 형식 문자열 기능을 사용한다.
  - 변수 이름이나 식을 중괄호 안에 포함해 값을 문자열로 가져온다.
  - `format` 함수가 없고 형식 문자열에 빈 괄호 {}와 위치 괄호{1}가 없다.
    - » `format` 함수에서 수행할 수 있는 정의를 {} 안에 사용할 수 있다(코드 읽기 쉬움).
  - f-문자열은 파이썬 3.8부터 이름과 값을 쉽게 출력할 수 있다(디버깅이 편리).

# 문자열 조작 함수: 검색 및 정렬 (1/2)

## ● 문자열 조작 함수: 문자열 검색

### ○ 문자열 검색

함 수	세부 내용	
<b>count</b>	string. <b>count</b> (value, start, end)	문자열에서 지정된 값이 발생한 횟수를 반환한다.
<b>index</b>	string. <b>index</b> (value, start, end)	지정된 값에 대한 문자열을 검색하고, 찾은 위치를 반환한다.
<b>rindex</b>	string. <b>rindex</b> (value, start, end)	지정된 값에 대한 문자열을 검색하고, 찾은 위치의 마지막 위치를 반환한다.
<b>find</b>	string. <b>find</b> (value, start, end)	지정된 값에 대한 문자열을 검색하고, 찾은 위치를 반환한다.
<b>rfind</b>	string. <b>rfind</b> (value, start, end)	지정된 값에 대한 문자열을 검색하고, 찾은 위치의 마지막 위치를 반환한다.
<b>startswith</b>	string. <b>startswith</b> (value, start, end)	문자열이 지정된 값으로 시작하면 True를 반환한다.
<b>endswith</b>	string. <b>endswith</b> (value, start, end)	문자열이 지정된 값으로 끝나면 True를 반환한다.

- **index, rindex** 메소드: 값을 찾을 수 없는 경우 예외를 발생시킨다.
- **find, rfind** 메소드: 값을 찾을 수 없는 경우 -1 값을 반환한다.

# 문자열 조작 함수: 검색 및 정렬 (2/2)

## ● 문자열 조작 함수: 문자열 정렬

### ○ 문자열 정렬 및 채우기

함 수	세부 내용	
<b>center</b>	string. <b>center</b> (length, character)	문자열을 가운데 정렬한다(기본값은 공백).
<b>ljust</b>	string. <b>ljust</b> (length, character)	문자열을 왼쪽 정렬한다(기본값은 공백).
<b>rjust</b>	string. <b>rjust</b> (length, character)	문자열을 오른쪽 정렬한다(기본값은 공백).
<b>zfill</b>	string. <b>zfill</b> (length)	문자열을 시작 부분에 지정된 수의 0 값으로 채운다.

```
aStr = 'Hello World!!!'

print(aStr.center(20))          # 가운데 정렬
print(aStr.ljust(20))           # 왼쪽 정렬
print(aStr.rjust(20))          # 오른쪽 정렬

print(aStr.center(20, '#'))
print(aStr.ljust(20, '#'))
print(aStr.rjust(20, '#'))

print(aStr.zfill(20))
```



# 문자열 조작 함수: 판별 및 변환 (1/4)

## ● 문자열 조작 함수: 문자열 판별

### ○ 문자열 판별

함 수	세부 내용	
<b>islower</b>	string.islower()	문자열의 모든 문자가 <b>소문자</b> 이면 <b>True</b> 를 반환한다.
<b>isupper</b>	string.isupper()	문자열의 모든 문자가 <b>대문자</b> 인 경우 <b>True</b> 를 반환한다.
<b>isalpha</b>	string.isalpha()	문자열의 모든 문자가 <b>알파벳</b> 이면 <b>True</b> 를 반환한다.
<b>isdigit</b>	string.isdigit()	문자열의 모든 문자가 <b>숫자</b> 이면 <b>True</b> 를 반환한다.
<b>isnumeric</b>	string.isnumeric()	문자열의 모든 문자가 <b>숫자</b> 인 경우 <b>True</b> 를 반환한다.
<b>isdecimal</b>	string.isdecimal()	문자열의 모든 문자가 <b>10진수</b> 이면 <b>True</b> 를 반환한다.
<b>isalnum</b>	string.isalnum()	문자열의 모든 문자가 <b>알파벳과 숫자</b> 이면 <b>True</b> 를 반환한다.

# 문자열 조작 함수: 판별 및 변환 (2/4)

## ● 문자열 조작 함수: 문자열 판별

### ○ 문자열 판별

함 수	세부 내용	
<b>isidentifier</b>	<code>string.isidentifier()</code>	문자열이 <b>식별자</b> 이면 <b>True</b> 를 반환한다.
<b>isspace</b>	<code>string.isspace()</code>	문자열의 모든 문자가 <b>공백</b> 이면 <b>True</b> 를 반환한다.
<b>isprintable</b>	<code>string.isprintable()</code>	문자열의 모든 문자가 <b>인쇄</b> 가능한 경우 <b>True</b> 를 반환한다.
<b>istitle</b>	<code>string.istitle()</code>	문자열이 <b>제목 규칙</b> 을 따르는 경우 <b>True</b> 를 반환한다.



# 문자열 조작 함수: 판별 및 변환 (3/4)

## ● 문자열 조작 함수: 문자열 변환

### ○ 문자열 변환

함 수	세부 내용	
lower	string.lower()	문자열을 소문자로 변환한다.
upper	string.upper()	문자열을 대문자로 변환한다.
title	string.title()	각 단어의 첫 문자를 대문자로 변환한다.
swapcase	string.swapcase()	대소문자를 소문자로, 소문자를 대문자로 변환한다.

```
aStr = 'Hello World!!!'

print(aStr.lower())
print(aStr.upper())
print(aStr.title())
print(aStr.swapcase())

print(aStr.islower())          # False
print(aStr.isupper())          # False
```



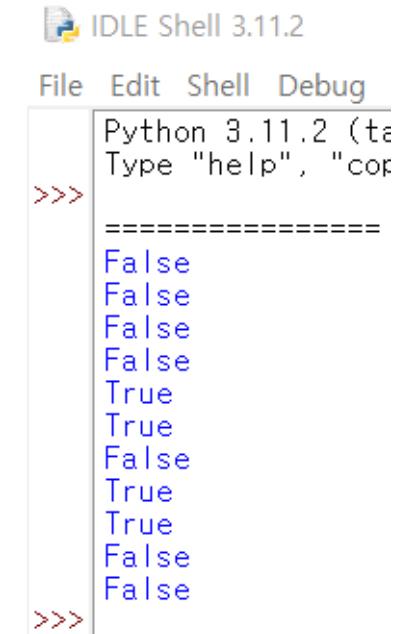
The screenshot shows the Python IDLE Shell interface. The code in the left pane is identical to the one above. The right pane shows the output of the code execution:

```
IDLE Shell 3.11.2
File Edit Shell Debug 
Python 3.11.2 (tag)
Type "help", "copy"
>>>
=====
hello world!!!
HELLO WORLD!!!
Hello World!!!
hELLO WORLD!!!
False
False
>>>
```

# 문자열 조작 함수: 판별 및 변환 (4/4)

## 예제 0-17: 문자열 조작 함수 -- 문자열 판별

```
aStr = 'Hello World!!!'  
print(aStr.islower())          # False  
print(aStr.isupper())          # False  
print(aStr.isalpha())          # False  
print(aStr.isalnum())          # False  
  
bStr = 'HELLO WORLD!!!!'  
print(bStr.isupper())          # True  
  
cStr = 'hello 123'  
print(cStr.islower())          # True  
print(cStr.isdigit())          # False  
  
dStr = '1234567890'  
print(dStr.isdigit())           # True  
print(dStr.isnumeric())        # True  
  
eStr = '-10'  
fStr = '10.5'  
print(eStr.isnumeric())         # False    # print(eStr.isdigit())  
print(fStr.isnumeric())         # False    # print(fStr.isdigit())
```



```
IDLE Shell 3.11.2  
File Edit Shell Debug  
Python 3.11.2 (t  
Type "help", "cop  
=====
```

>>> False  
False  
False  
False  
True  
True  
False  
True  
True  
False  
False  
>>>

# 문자열 조작 함수: 문자열 조작 (1/2)

## ● 문자열 조작 함수: 문자열 공백 삭제 및 변환

### ○ 문자열 공백 삭제 및 변환

함 수	세부 내용	
<b>strip</b>	string. <b>strip</b> (characters)	선행(시작 부분의 공백) 및 후행(끝의 공백) 문자를 제거한다(기본 값은 공백).
<b>lstrip</b>	string. <b>lstrip</b> (characters)	선행(시작 부분의 공백) 문자를 제거한다(기본 값은 공백).
<b>rstrip</b>	string. <b>rstrip</b> (characters)	후행(끝의 공백) 문자를 제거한다(기본 값은 공백).
<b>replace</b>	string. <b>replace</b> (oldvalue, newvalue, count)	지정된 구문을 다른 지정된 구문으로 변경한다.

```
aStr = ',,,Hello,,,World...!!!'

newStr = aStr.strip('.,.!!');      print(newStr)
newStr = aStr.lstrip('.,.!!');    print(newStr)
newStr = aStr.rstrip('.,.!!');    print(newStr)

newStr = aStr.replace('World', 'Clickseo')
print(newStr)
```



```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window
Python 3.11.2 (tags/v3.11.2:87
Type "help", "copyright", "cre
>>>
=====
RESTART: C:\W
Hello...World
Hello...World...!!!
...Hello...World
...Hello...Clickseo...!!!
```

# 문자열 조작 함수: 문자열 조작 (2/2)

## ● 문자열 조작 함수: 문자열 분리 및 결합

### ○ 문자열 분리 및 결합

함 수	세부 내용	
<b>split</b>	string. <b>strip</b> (separator, maxsplit)	지정된 구분 기호에서 문자열을 분할하고 목록을 반환한다.
<b>splitlines</b>	string. <b>lstrip</b> (keeplinebreaks)	줄 바꿈에서 문자열을 분할하고 목록을 반환한다.
<b>join</b>	string. <b>rstrip</b> (iterable)	iterable의 요소를 문자열 끝에 결합한다.

```
aStr = 'Hello World!!!'  
print(aStr.split())  
  
bStr = '2023/12/31'  
print(bStr.split('/'))  
  
cStr = 'Hello\nWorld!!!'  
print(cStr.split('\n'))  
print(cStr.splitlines())  
  
dStr = '#'  
print(dStr.join('Hello World!!!!'))
```



The screenshot shows the Python IDLE Shell 3.11.2 interface. The code is run in the shell, and the output is displayed. The output shows the results of the string manipulation functions: `split()`, `splitlines()`, and `join()`.

```
IDLE Shell 3.11.2  
File Edit Shell Debug Options Window  
Python 3.11.2 (tags/v3.11.2:8  
Type "help", "copyright", "cr  
>>> ===== RESTART: C:  
['Hello', 'World!!!!']  
['2023', '12', '31']  
['Hello', 'World!!!!']  
['Hello', 'World!!!!']  
H#e#l#l#o# #W#o#r#l#d#!!#!#!  
>>>
```



# 자료형과 자료구조

리스트와 튜플

: 리스트와 튜플 조작 함수



# 리스트의 이해 (1/5)

## ● **리스트(List): [ ]**

### ○ 여러 개의 자료들을 모아서 하나의 묶음으로 저장하는 것

- C/C++, Java 같은 프로그래밍 언어에서는 리스트가 없다.
  - 하지만, 리스트와 비슷한 개념인 배열(Array)을 사용한다.

### ○ 리스트 객체 생성

```
aList = []                      # 빈 리스트 객체 생성  
bList = [ 10, 20, 30, 40, 50 ]    # 리스트 객체 생성
```

- **list** 내장 함수: 다른 데이터 유형(문자열, 튜플, 딕셔너리 등)을 리스트로 변환한다.
  - class list
  - class list(iterable)

**class list**

```
temp = list('1234567890')      # 리스트 객체 생성  
temp = ( 10, 20, 30, 40, 50 )    # 튜플 객체 생성  
list(temp)                      # 튜플 객체를 리스트 객체로 변환
```

# 리스트 표현식 (1/4)

## ● 리스트 표현식: List Comprehension

- 순차적인 리스트를 한 줄로 만드는 간단한 방법

```
# 리스트 표현식: List Comprehension
```

```
 newList = [ expression for item in iterable if condition == True ]
```

- item
- iterable: list, tuple, set 등과 같은 iterable 객체
- condition: True로 평가되는 항목만 필터링
- expression: 조작할 수 있는 결과에 대한 표현식

```
# 리스트 객체 생성
```

```
sList = [ '국어', '영어', '수학', '과학' ]
```

```
# List Comprehension
```

```
 newList = [ s for s in sList ]
```

```
 print(newList)
```

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb
Type "help", "copyright", "credits" or "li
>>> ===== RESTART: C:\Users\click\w
['국어', '영어', '수학', '과학']
>>> id(sList)
2349505886400
>>> id(newList)
2349462107264
>>> sList is newList
False
>>>
```

# 리스트 표현식 (2/4)

## 예제 0-18: 리스트 표현식

#01

```
# 리스트 객체 생성
sList = [ '국어', '영어', '수학', '과학' ]

# 반복문: for 문
aList = []
for s in sList:
    aList.append(s)
print(aList)

# List Comprehension
bList = [ s for s in sList ];
print(bList)

# iterable 객체: list, tuple, set 등
cList = [ a for a in range(10) ]
print(cList)

dList = [ b for b in range(10) if b < 5 ]
print(dList)
```

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 1 Type "help", "copyright", "credits" or "li
>>>
=====
RESTART: C:\Users\click\...
['국어', '영어', '수학', '과학']
['국어', '영어', '수학', '과학']
[0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
[0, 1, 2, 3, 4]
```

# 리스트 표현식 (3/4)

예제 0-18: 리스트 표현식

#02

```
# 리스트 객체 생성  
sList = [ '국어', '영어', '수학', '과학' ]  
  
# 반복문: for 문  
aList = []  
for s in sList:  
    if '학' in s:  
        aList.append(s)  
print(aList)  
  
# List Comprehension  
bList = [ s for s in sList if '학' in s ]  
print(bList)  
  
# List Comprehension  
cList = [ s for s in sList if s != '수학' ]  
print(cList)
```



```
IDLE Shell 3.11.2  
File Edit Shell Debug Options Window Help  
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  
Type "help", "copyright", "credits" or "li  
>>> ===== RESTART: C:\Users\click\w  
['수학', '과학']  
['수학', '과학']  
['국어', '영어', '과학']  
>>>
```

# 리스트 표현식 (4/4)

예제 0-18: 리스트 표현식

#03

# 리스트 객체 생성

```
fruits = [ 'apple', 'banana', 'cherry', 'kiwi', 'mango' ]
```

# List Comprehension

```
aList = [ s.upper() for s in fruits ]  
print(aList)
```

```
bList = [ 'hello' for s in fruits ]
```

```
print(bList)
```

```
cList = [ s if s != 'banana' else 'orange' for s in fruits ]
```

```
print(cList)
```

The screenshot shows the Python IDLE Shell interface. The title bar says "IDLE Shell 3.11.2". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the following Python session:

```
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:37:43)
Type "help", "copyright", "credits" or "license()" to get help.  

>>> ===== RESTART: C:\Users\click\OneDrive\...  

['APPLE', 'BANANA', 'CHERRY', 'KIWI', 'MANGO']  

['hello', 'hello', 'hello', 'hello', 'hello']  

['apple', 'orange', 'cherry', 'kiwi', 'mango']  

>>>
```

# 리스트 조작 함수 (1/15)

## ● 리스트 조작 함수: 리스트 항목 추가 또는 삭제

### ○ 리스트 항목 추가 또는 삭제

함 수	세부 내용	
index	list.index(elmnt)	리스트에서 지정된 항목의 위치를 반환한다.
count	list.count(value)	리스트에서 지정된 값의 개수를 반환한다.
append	list.append(elmnt)	리스트 맨 뒤에 새로운 항목을 추가한다.
insert	list.insert(pos, elmnt)	리스트의 지정된 위치에 새로운 항목을 추가한다.
extend	list.extend(iterable)	리스트의 맨 마지막에 새로운 리스트를 추가(연결)한다. 즉, 리스트의 '+' 연산과 동일하다.
remove	list.remove(elmnt)	리스트에서 지정된 항목을 삭제한다. 단, 첫 번째로 검색된 항목(값)만 지운다.
pop	list.pop(pos)	리스트에서 지정된 항목을 삭제한다(기본값은 -1).
clear	list.clear()	리스트의 모든 내용을 삭제한다.

# 리스트 조작 함수: 복사 (1/5)

## ● 리스트 조작 함수: 리스트 복사

- **copy** 함수: 리스트의 모든 항목(값)을 새로운 리스트에 복사한다.

함 수	세부 내용	
copy	list.copy()	리스트의 모든 항목을 새로운 리스트에 복사한다.

```
# 리스트 객체 생성 후 모든 원소를 새로운 리스트에 복사
sList = [ '국어', '영어', '수학', '과학' ]
temp = sList.copy()                                # 새로운 리스트 객체 temp 생성
```

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

# sList의 복사본인 새로운 리스트(aList, bList, cList) 객체 생성
aList = sList[:]
bList = sList.copy()
cList = list(sList)
```

# 리스트 조작 함수: 복사 (2/5)

## ● 리스트 조작 함수: 리스트 복사

- **deepcopy** 함수: 깊은 복사(deep copy)를 수행한다.
  - 얕은 복사(shallow copy)

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

```
# sList의 복사본인 새로운 리스트 aList, bList, cList 객체 생성(얕은 복사)
```

```
aList = sList[:]  
bList = sList.copy()  
cList = list(sList)
```

- 깊은 복사(deep copy)

```
# sList의 복사본인 새로운 리스트 dList 객체 생성(깊은 복사)
```

```
import copy  
dList = copy.deepcopy(sList)
```

# 리스트 조작 함수: 복사 (3/5)

## ● 리스트 조작 함수: 다차원 리스트

### ○ 2차원 리스트: 1차원 리스트를 여러 개 연결 한 리스트

```
# 2차원 리스트 객체 생성 및 초기화
```

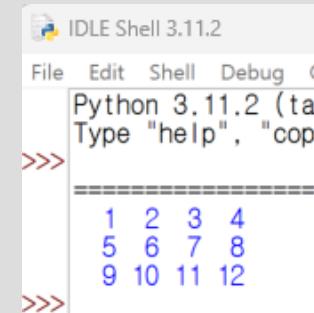
```
sList = [ 1,2,3,4,5,6,7,8,9,10,11,12 ]           # sList[i]
table = [ [1,2,3,4], [5,6,7,8], [9,10,11,12] ]    # table[i][j]
```

```
# 2차원 리스트의 각 원소 접근
```

```
# table[0]
table[0][0]      # 1
table[0][1]      # 2
table[0][2]      # 3
table[0][3]      # 4
```

```
# table[1]
table[1][0]      # 5
table[1][1]      # 6
table[1][2]      # 7
table[1][3]      # 8
```

```
# table[2]
table[2][0]      # 9
table[2][1]      # 10
table[2][2]      # 11
table[2][3]      # 12
```



The screenshot shows the Python 3.11.2 shell interface. The command line shows '>>>' followed by the code's output:

```
1 2 3 4
5 6 7 8
9 10 11 12
```

```
for i in range(len(table)):
    for j in range(len(table[i])):
        print(f'{table[i][j]:3}', end=' ')
    else: print()
[ print(item) for item in table ]
```

# 리스트 조작 함수: 복사 (4/5)

## 예제 0-19: 다차원 리스트 -- 깊은 복사(deep copy)

# 얕은 복사(shallow copy)

```
table = [ [1,2,3,4], [5,6,7,8], [9,10,11,12] ]
copyTable = table.copy()

print(f'table : {table}')
print(f'copyTable: {copyTable}')

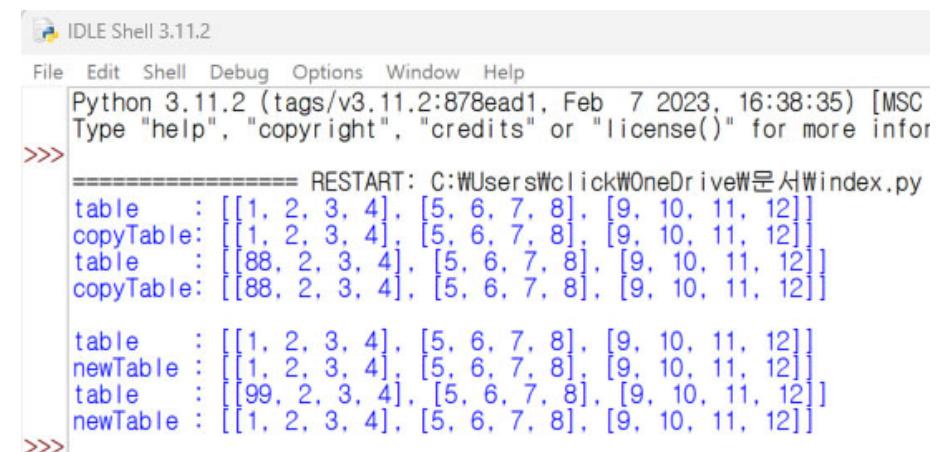
table[0][0] = 88
print(f'table : {table}')
print(f'copyTable: {copyTable}\n')
```

# 깊은 복사(deep copy)

```
table = [ [1,2,3,4], [5,6,7,8], [9,10,11,12] ]
import copy
newTable = copy.deepcopy(table)

print(f'table : {table}')
print(f'newTable : {newTable}')

table[0][0] = 99
print(f'table : {table}')
print(f'newTable : {newTable}')
```



```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC
Type "help", "copyright", "credits" or "license()" for more information
>>> ===== RESTART: C:\Users\click\OneDrive\문서\index.py
table : [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
copyTable: [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
table : [[88, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
copyTable: [[88, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]

table : [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
newTable : [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
table : [[99, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
newTable: [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
```

# 리스트 조작 함수: 복사 (5/5)

## 예제 0-20: 다차원 리스트 -- 3차원 리스트

# 2차원 리스트

aTable = []

```
aTable.append([1,2,3,4])
aTable.append([5,6,7,8])
aTable.append([9,10,11,12])
```

import copy

bTable = copy.deepcopy(aTable)

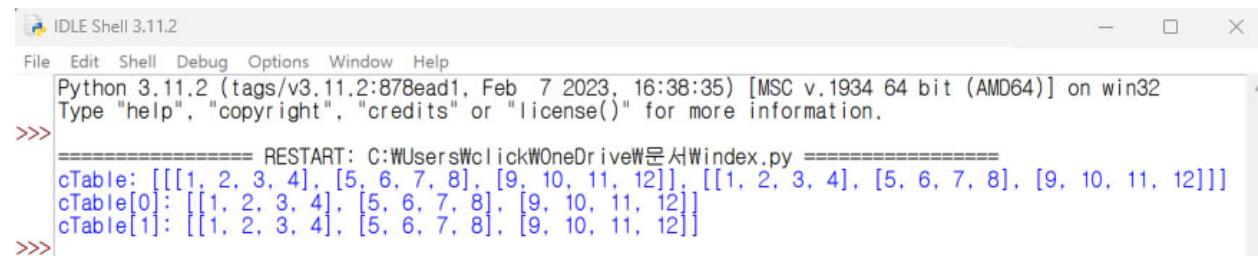
# 3차원 리스트

cTable = []

cTable.append(aTable)

cTable.append(bTable)

```
print(f'cTable    : {cTable}')
print(f'cTable[0] : {cTable[0]}')
print(f'cTable[1] : {cTable[1]}'')
```



```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> ===== RESTART: C:\Users\click\OneDrive\문서\Windex.py =====
cTable: [[[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]], [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]]
cTable[0]: [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]
cTable[1]: [[1, 2, 3, 4], [5, 6, 7, 8], [9, 10, 11, 12]]]
```

# 리스트 조작 함수: 정렬과 역순(1/3)

## ● 리스트 조작 함수: 정렬과 역순

### ○ 리스트 정렬과 역순

함 수	세부 내용	
	list.sort()	리스트 항목을 내부적으로 정렬한다( <b>기존 순서 변경</b> ).
sort	list.sort(reverse=True   False, key=myFunc) reverse=False (기본값) / reverse=True (내림차순) key : 정렬 기준을 지정한다.	
reverse	list.reverse()	리스트 항목의 순서를 반대로 바꾼다( <b>기존 순서 변경</b> ).

### ○ sorted 내장 함수

- 리스트의 항목을 정렬하여, 새로운 리스트를 반환한다(**복사본 생성**).
  - 단, 기존 리스트 항목들의 순서는 변경되지 않는다.

# 리스트 조작 함수: 정렬과 역순(2/3)

예제 0-21: 리스트 조작 함수 -- 리스트 정렬(sort)과 역순

```
import random

print('\t### 리스트 항목 정렬: sort ###')
sList = []
for i in range(10) :
    num = random.randrange(1, 101) # 1 ~ 100 사이의 난수 발생
    sList.append(num)
print(f'sList: {sList}')

# 오름차순: 리스트 항목 정렬
# sList.sort(reverse=False)
sList.sort()
print(f'sList: {sList}')

# 내림차순: 리스트 항목 정렬
sList.sort(reverse=True)
print(f'sList: {sList}')

# 리스트 항목의 순서를 반대로 변경한다.
sList.reverse()
print(f'sList: {sList}')
```



The screenshot shows the Python 3.11.2 IDLE shell interface. The command prompt shows the script being run, followed by the sorted lists and their reversals.

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 11:47:43)
Type "help", "copyright", "credits" or "license()"
>>> ===== RESTART: C:\Users\click\OneDrive\...
    ### 리스트 항목 정렬: sort ####
sList: [40, 22, 50, 3, 23, 100, 99, 58, 24, 21]
sList: [3, 21, 22, 23, 24, 40, 50, 58, 99, 100]
sList: [100, 99, 58, 50, 40, 24, 23, 22, 21, 3]
sList: [3, 21, 22, 23, 24, 40, 50, 58, 99, 100]
>>>
```

# 리스트 조작 함수: 정렬과 역순(3/3)

예제 0-22: 리스트 조작 함수 -- 리스트 정렬(sorted)

```
import random

print('\n\t### 리스트 항목 정렬: sorted ###')
sList = []
for i in range(10) :
    num = random.randint(1, 100)      # 1 ~ 100 사이의 난수 발생
    sList.append(num)
print(f'sList: {sList}')

# 오름차순: 리스트 항목 정렬
# aList = sorted(sList, reverse=False)
aList = sorted(sList)
print(f'aList: {aList}')

# 내림차순: 리스트 항목 정렬
bList = sorted(sList, reverse=True)
print(f'bList: {bList}')

print(f'sList: {sList}')
```



# 자료 구조: 스택과 큐 (1/2)

## 예제 0-23: 스택의 이해 -- list 내장 클래스

# 스택(Stack): 후입선출(LIFO, Last-In-First-Out)

```
s = []          # 빈 스택 생성: 빈 리스트 객체
```

```
s.append(10)
```

```
s.append(20)
```

```
s.append(30)
```

```
print(f'stack is empty: {len(s) == 0}')
```

```
print(f'stack size      : {len(s)}')
```

```
while s:
```

```
    # print(f'top element: {s.pop()}')
```

```
    print(f'top element: {s[-1]}')
```

```
    s.pop()
```

IDLE Shell 3.11.2

File Edit Shell Debug Options W

```
Python 3.11.2 (tags/v3.11-  
Type "help", "copyright",  
>>>  
===== RESTART:  
stack is empty: False  
stack size      : 3  
top element: 30  
top element: 20  
top element: 10  
>>>
```

# 자료 구조: 스택과 큐 (2/2)

## 예제 0-24: 큐의 이해 -- list 내장 클래스

# 큐(Queue): 선입선출(FIFO, First-In First-Out)

q = [] # 빈 큐 생성: 빈 리스트 객체

```
q.append(10)
q.append(20)
q.append(30)
```

```
print(f'queue is empty: {len(q) == 0}')
print(f'queue size : {len(q)}')
```

```
while q:
    # print(f'front element: {q.pop(0)}')
    print(f'front element: {q[0]!r}')
    q.pop(0)
```

```
IDLE Shell 3.11.2
File Edit Shell Debug Options W
Python 3.11.2 (tags/v3.11.
Type "help", "copyright",
>>> ===== RESTART:
queue is empty: False
queue size : 3
front element: 10
front element: 20
front element: 30
>>>
```

# 튜플의 이해 (1/4)

## ● 튜플(Tuple): ()

### ○ 튜플은 순서가 있고 변경할 수 없다.

- 불변(immutable) 데이터 유형
  - 즉, 튜플이 생성된 후에는 항목을 변경, 추가 또는 제거할 수 없다.
  - 튜플은 리스트의 append, insert 등과 같은 튜플 조작 함수가 거의 없다.
- Tuple Comprehension은 없다.
  - 가변적인 데이터 유형(리스트, 딕셔너리와 셋)에는 Comprehension이 있다.
  - 문자열과 튜플은 불변 유형이기 때문에 다른 방법으로 만들어져야 한다.
- 튜플의 다양한 장점
  - 튜플은 리스트보다 더 적은 공간을 사용한다.
  - 프로그래머의 실수로 튜플의 항목이 손상될 염려가 없다.
  - 튜플을 딕셔너리의 키로 사용할 수 있다.
  - 네임드 튜플(named tuple)은 객체의 단순한 대안이 될 수 있다.

# 튜플의 이해 (2/4)

## ● 튜플(Tuple): 튜플 생성 및 초기화

### ○ 튜플 생성 및 초기화

class tuple

```
aTuple = ()
```

# 빈 튜플 생성

```
bTuple = ( 10, 20, 30, 40, 50 )
```

# 튜플 생성 및 초기화

# 튜플을 생성할 때는 괄호를 생략 할 수 있다.

```
cTuple = 10, 20, 30, 40, 50
```

### ○ tuple 내장 함수

- 다른 데이터 유형(문자열, 리스트, 딕셔너리 등)을 튜플로 변환한다.

```
temp = tuple('1234567890')
```

# 튜플 객체 생성

```
temp = [ 10, 20, 30, 40, 50 ]
```

# 리스트 객체 생성

```
tuple(temp)
```

# 리스트를 튜플 객체로 변환

# 튜플의 이해 (3/4)

## ● 튜플(Tuple): 튜플 생성 및 초기화

### ○ 하나의 항목을 가진 튜플 생성

- 항목이 하나만 있는 튜플을 만들려면 항목 뒤에 쉼표를 추가해야 합니다. 그렇지 않으면 파이썬은 이를 튜플로 인식하지 못한다.

```
aTuple = ('apple',)  
print(type(aTuple))      # <class 'tuple'>  
  
# NOT a tuple  
bStr = ('apple')  
print(type(bStr))        # <class 'str'>  
  
# NOT a tuple  
cInt = (10)  
print(type(cInt))        # <class 'int' >
```

class tuple

# 튜플의 이해 (4/4)

## 예제 0-25: 리스트와 튜플

```
aTuple = (10, 20, 30, 40, 50)
print(aTuple)

# 튜플을 리스트 객체로 변경 후 항목의 값을 변경 및 추가
aList = list(aTuple)
aList[1] = 60
aList.append(70)
print(aList)

aTuple = tuple(aList)
print(aTuple)
```

```
bTuple = ('국어', '영어', '수학')
cTuple = ('과학',)

# 튜플 연결(결합)
bTuple += cTuple
print(bTuple)
```

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1,
Type "help", "copyright", "credits" ,
>>> ===== RESTART: C:\Users\W
(10, 20, 30, 40, 50)
[10, 60, 30, 40, 50, 70]
(10, 60, 30, 40, 50, 70)
('국어', '영어', '수학', '과학')
>>>
```

# 튜플 조작 함수

- **튜플 조작 함수: 항목의 위치 및 개수**

- **튜플 항목의 위치 및 개수**

함 수	세부 내용	
<b>index</b>	<code>tuple.index(elmnt)</code>	튜플에서 지정된 항목의 <b>위치</b> 를 반환한다.
<b>count</b>	<code>tuple.count(value)</code>	튜플에서 지정된 값의 <b>개수</b> 를 반환한다.





# 자료형과 자료구조

딕셔너리와 셋

: 딕셔너리와 셋 조작 함수



# 딕셔너리의 이해 (1/3)

## ● 딕셔너리(Dictionary)

○ 연관되어 있는 데이터를 하나의 쌍으로 묶어서 관리한다.

- 키(Key)와 값(Value)의 쌍으로 데이터 값을 저장
- 순서가 지정되고 변경 가능하며, 중복을 허용하지 않는다.
  - 동일한 키를 가진 두 개의 항목이 있을 수 없습니다(유일한 키 값).
  - 딕셔너리가 생성된 후 항목을 변경, 추가 또는 제거할 수 있다.

class dict

# 딕셔너리의 이해 (2/3)

## ● 딕셔너리(Dirctionary): { }

### ○ 딕셔너리 객체 생성 및 할당

```
# 딕셔너리 객체 생성
aDict = {}
bDict = dict() # dict 내장 함수

# 딕셔너리 객체 생성
carDict = {
    'brand': 'HYUNDAI',
    'model': 'IONIQ',
    'year': 2020
}

# 딕셔너리 객체의 키에 대한 이름으로 참조한다.
print(type(carDict)) # <class 'dict'>
print(carDict['brand']) # HYUNDAI
print(carDict.get('brand')) # HYUNDAI
```

# class dict

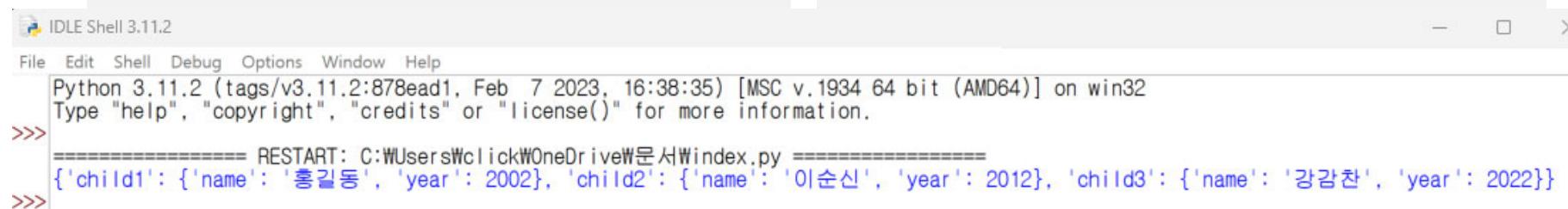
# 딕셔너리의 이해 (3/3)

## 예제 0-26: 딕셔너리 -- 중첩 딕셔너리

```
myFamily = {
    'child1' : {
        'name' : '홍길동',
        'year' : 2005
    },
    'child2' : {
        'name' : '이순신',
        'year' : 2015
    },
    'child3' : {
        'name' : '강감찬',
        'year' : 2025
    }
}
print(myFamily)
```

```
child1 = {
    'name' : '홍길동',
    'year' : 2005
}
child2 = {
    'name' : '이순신',
    'year' : 2015
}
child3 = {
    'name' : '강감찬',
    'year' : 2025
}

myFamily = {
    'child1' : child1,
    'child2' : child2,
    'child3' : child3
}
print(myFamily)
```



```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> ===== RESTART: C:\Users\click\OneDrive\문서\Windex.py =====
>>> {'child1': {'name': '홍길동', 'year': 2002}, 'child2': {'name': '이순신', 'year': 2012}, 'child3': {'name': '강감찬', 'year': 2022}}
```

# 딕셔너리 조작 함수 (1/9)

## ● 딕셔너리 조작 함수: 객체 생성

### ○ 딕셔너리 객체 생성

메소드	세부 내용	
<b>fromkeys</b>	dictionary. <b>fromkeys</b> (keys, value)	<p>지정된 키와 값이 있는 딕셔너리 객체를 반환한다.</p> <p>keys: (필수) 새로운 딕셔너리 객체의 키를 지정 value: (선택) 모든 키의 값(기본값은 없음)</p>
<b>setdefault</b>	dictionary. <b>setdefault</b> (keyname, value)	<p>1) 키가 존재하지 않는 경우: 지정된 값과 함께 키를 <b>추가</b>한다.</p> <p>2) 키가 존재할 경우: 지정된 키의 값을 <b>반환</b>한다.</p> <p>keyname: (필수) 값을 반환하려는 항목의 키 이름. value: (선택) 기본 값은 없음.</p> <ul style="list-style-type: none"><li>- 키가 있는 경우 매개변수는 효과가 없다.</li><li>- 키가 존재하지 않으면 매개변수의 값이 키의 값이 된다.</li></ul>

# 딕셔너리 조작 함수 (2/9)

## 예제 0-27: 딕셔너리 조작 함수 -- 딕셔너리 객체 생성

# 모두 값이 0인 3개의 키가 있는 딕셔너리 객체 생성

```
keys = ('key1', 'key2', 'key3')
```

```
values = 0
```

```
aDict = dict.fromkeys(keys, values)
```

```
print(aDict)
```

# 값을 지정하지 않고, 3개의 키가 있는 딕셔너리 객체 생성

```
keys = ('key1', 'key2', 'key3')
```

```
bDict = dict.fromkeys(keys)
```

```
print(bDict)
```

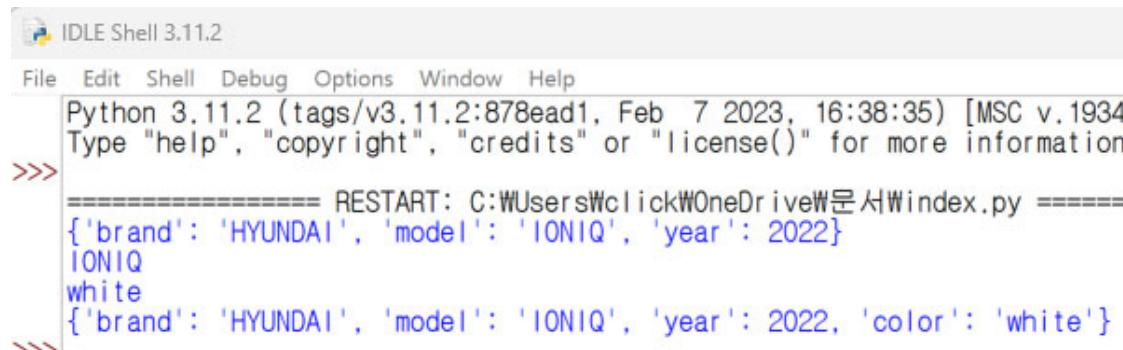
The screenshot shows the Python IDLE Shell 3.11.2 interface. The command line displays the following output:

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 1
Type "help", "copyright", "credits" or "license()"
>>> ===== RESTART: C:\Users\click\OneDrive\
{'key1': 0, 'key2': 0, 'key3': 0}
{'key1': None, 'key2': None, 'key3': None}
```

# 딕셔너리 조작 함수 (3/9)

## 예제 0-28: 딕셔너리 조작 함수 -- 지정된 키 값 반환 및 추가

```
carDict = {  
    'brand': 'HYUNDAI',  
    'model': 'IONIQ',  
    'year': 2020  
}  
  
print(carDict)  
  
# 1) 키가 존재하지 않는 경우: 지정된 값(white)과 함께 키(color)를 추가  
# 2) 키가 존재할 경우: 지정된 키(model)의 값을 반환  
  
value = carDict.setdefault('model', 'IONIQ 9');     print(value)  
value = carDict.setdefault('color', 'white');         print(value)  
  
print(carDict)
```



The screenshot shows the Python IDLE Shell 3.11.2 interface. The code is run in a terminal window, and the output shows the initial dictionary, followed by the addition of a new key-value pair ('color', 'white'), and finally the updated dictionary.

```
IDLE Shell 3.11.2  
File Edit Shell Debug Options Window Help  
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC v.1934  
Type "help", "copyright", "credits" or "license()" for more information.  
===== RESTART: C:\Users\click\OneDrive\문서\Windex.py =====  
{'brand': 'HYUNDAI', 'model': 'IONIQ', 'year': 2020}  
IONIQ  
white  
{'brand': 'HYUNDAI', 'model': 'IONIQ', 'year': 2020, 'color': 'white'}
```

# 딕셔너리 조작 함수 (4/9)

## ● 딕셔너리 조작 함수: 항목 순회

### ○ 딕셔너리 항목 순회

- 모든 키와 값 그리고 키-값 쌍 얻기

함 수	세부 내용	
<b>keys</b>	dictionary.keys()	딕셔너리의 <b>키 목록</b> 을 반환한다.
<b>values</b>	dictionary.values()	딕셔너리의 <b>값 목록</b> 을 반환한다.
<b>items</b>	dictionary.items()	딕셔너리의 <b>모든 키와 값의 쌍</b> 을 반환한다( <b>반환 유형은 튜플</b> ).

```
carDict = {
    'brand': 'HYUNDAI',
    'model': 'IONIQ',
    'year': 2020
}

print(carDict.keys())
print(carDict.values())
print(carDict.items())
```

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934
Type "help", "copyright", "credits" or "license()" for more information
>>> ===== RESTART: C:\Users\Clickseo\OneDrive\문서\index.py =====
dict_keys(['brand', 'model', 'year'])
dict_values(['HYUNDAI', 'IONIQ', 2020])
dict_items([('brand', 'HYUNDAI'), ('model', 'IONIQ'), ('year', 2020)])
```

```
# 딕셔너리의 모든 키 얻기
# 딕셔너리의 모든 값 얻기
# 딕셔너리의 모든 키-값 쌍 얻기
```

# 딕셔너리 조작 함수 (6/9)

## ● 딕셔너리 조작 함수: 항목 추가(수정) 또는 삭제

### ○ 딕셔너리 항목 추가(수정) 또는 삭제

함 수	세부 내용	
<b>get</b>	dictionary. <b>get</b> (keyname, value)	딕셔너리에 지정된 키가 있는 항목의 값을 반환한다. keyname: (필수) 값을 반환하려는 항목의 키 이름. defaultvalue: (선택) 지정된 키가 없을 경우 반환할 값.
<b>update</b>	dictionary. <b>update</b> (iterable)	지정된 키-값 쌍으로 새로운 항목을 <b>추가</b> 하거나 <b>변경</b> 한다.
<b>pop</b>	dictionary. <b>pop</b> (keyname, defaultvalue)	지정된 키가 있는 항목을 <b>삭제</b> 한다( <b>삭제한 항목의 값을 반환</b> ). keyname: (필수) 제거하려는 항목의 키 이름. defaultvalue: (선택) 지정된 키가 없을 경우 반환할 값.
<b>popitem</b>	dictionary. <b>popitem</b> ()	마지막으로 추가된 항목을 <b>삭제</b> 한다. ( <b>삭제된 항목은 튜플 유형으로 반환된다.</b> )
<b>clear</b>	dictionary. <b>clear</b> ()	모든 항목을 <b>삭제</b> 한다( <b>빈 딕셔너리 객체</b> ).

### ○ del 내장 함수

- 지정된 키 이름의 항목을 삭제하거나, 딕셔너리 객체를 삭제한다.

# 딕셔너리 조작 함수 (7/9)

## 예제 0-29: 딕셔너리 조작 함수 -- 딕셔너리 항목 접근 및 추가(수정)

```
carDict = {  
    'brand': 'HYUNDAI',  
    'model': 'IONIQ',  
    'year': 2020  
}
```

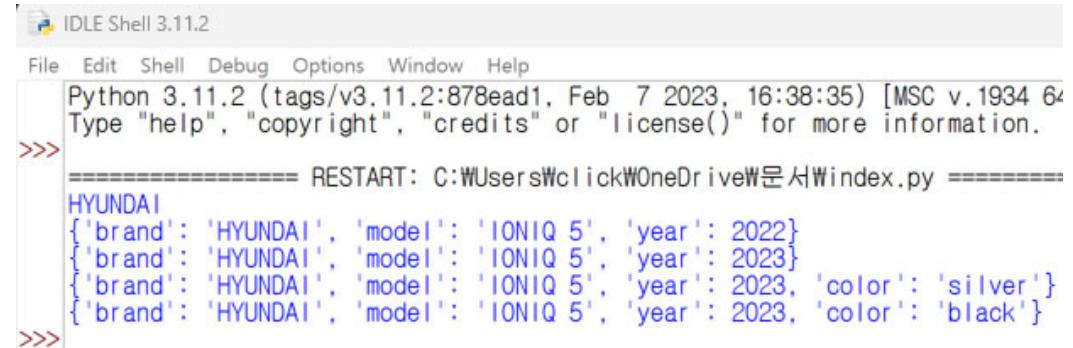
```
# 지정된 키가 있는 항목의 값을 반환  
print(carDict.get('brand'))
```

```
# 키의 이름을 참조하여 특정 항목의 값 변경(수정)
```

```
carDict['model'] = 'IONIQ 9';  
carDict.update({'year': 2025});  
  
print(carDict)  
print(carDict)
```

```
# 새로운 항목 추가
```

```
carDict['color'] = 'silver';  
carDict.update({'color': 'black'});  
  
print(carDict)  
print(carDict)
```



```
IDLE Shell 3.11.2  
File Edit Shell Debug Options Window Help  
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC v.1934 64  
Type "help", "copyright", "credits" or "license()" for more information.  
=====  
==== RESTART: C:\Users\click\OneDrive\문서\index.py =====  
HYUNDAI  
{'brand': 'HYUNDAI', 'model': 'IONIQ 5', 'year': 2022}  
{'brand': 'HYUNDAI', 'model': 'IONIQ 5', 'year': 2023}  
{'brand': 'HYUNDAI', 'model': 'IONIQ 5', 'year': 2023, 'color': 'silver'}  
{'brand': 'HYUNDAI', 'model': 'IONIQ 5', 'year': 2023, 'color': 'black'}
```

# 딕셔너리 조작 함수 (8/9)

## 예제 0-30: 딕셔너리 조작 함수 -- 딕셔너리 항목 삭제

```
carDict = {  
    'brand': 'HYUNDAI',  
    'model': 'IONIQ',  
    'year': 2020  
}
```

```
carDict['color'] = 'silver';
```

# 딕셔너리의 항목 삭제

```
print(carDict.popitem());  
print(carDict.pop('model'));  
del carDict['year'];
```

```
carDict.clear();
```

```
del carDict
```

```
# print(carDict)
```

print(carDict) # 마지막으로 삽입된 항목 제거  
print(carDict) # 지정된 키 이름을 가진 항목 제거  
print(carDict) # 지정된 키 이름을 가진 항목 제거

```
print(carDict) # 빈 딕셔너리 객체
```

# 딕셔너리 객체를 완전히 삭제

```
# NameError: name 'carDict' is not defined
```

The screenshot shows the Python 3.11.2 IDLE shell interface. The code defines a dictionary `carDict` with keys 'brand', 'model', 'year', and 'color'. It then adds 'color' to 'silver'. The first three deletion statements (popitem(), pop('model'), and del) remove items from the dictionary. The final clear() statement makes it empty. The print statements at the bottom show the state of the dictionary after each operation.

```
IDLE Shell 3.11.2  
File Edit Shell Debug Options Window Help  
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:04)  
Type "help", "copyright", "credits" or "license()" for more information  
=====  
RESTART: C:\Users\click\OneDrive\문서\python\0-30.py  
>>> ('color', 'silver')  
{'brand': 'HYUNDAI', 'model': 'IONIQ', 'year': 2022}  
IONIQ  
{'brand': 'HYUNDAI', 'year': 2022}  
{'brand': 'HYUNDAI'}  
[]  
>>>
```

# 딕셔너리 조작 함수 (9/9)

## ● 딕셔너리 조작 함수: 복사

### ○ 딕셔너리 복사

함 수	세부 내용	
copy	dictionary. <b>copy()</b>	지정된 딕셔너리의 <b>복사본</b> 을 반환한다.

```
carDict = {  
    'brand': 'HYUNDAI',  
    'model': 'IONIQ',  
    'year': 2020  
}
```

# carDict의 복사본인 새로운 딕셔너리(aDict, bDict) 객체 생성

aDict = carDict.**copy()**

bDict = **dict**(carDict)

# 딕셔너리 내장 함수

# 셋의 이해

## ● 셋(Set): { }

### ○ 값은 버리고, 키만 남은 딕셔너리와 같다.

- 딕셔너리와 마찬가지로 각 키들은 반드시 고유해야 한다.
  - 즉, 중복되는 키는 저장되지 않는다.
- 단일 변수에 여러 항목을 지정하는데 사용된다.
- 셋은 순서가 없고, 변경할 수 없고, 인덱싱 되지 않는다.
  - 항목은 변경할 수 없지만, 항목을 제거하고 새 항목을 추가할 수 있다.

### ○ 셋 객체 생성 및 할당

```
newSet = set()
```

# set 내장 함수

# 셋 객체 생성

```
carSet = { 'brand', 'model', 'year' }
```

```
print(type(carSet))
```

class set

# <class 'set'>

# 셋 관련 함수 (1/5)

## ● 셋 조작 함수: 항목 추가/삭제와 복사

### ○ 셋 항목 추가 또는 삭제

함 수	세부 내용	
<b>add</b>	<code>set.add(elmnt)</code>	셋에 항목을 <b>추가</b> 한다. 단, 항목이 이미 존재하는 경우에는 항목을 추가하지 않는다.
<b>remove</b>	<code>set.remove(item)</code>	셋에서 지정된 항목을 <b>삭제</b> 한다. 지정된 항목이 존재하지 않으면 오류가 발생한다( <b>KeyError</b> ).
<b>discard</b>	<code>set.discard(value)</code>	셋에서 지정된 항목을 <b>삭제</b> 한다.
<b>pop</b>	<code>set.pop()</code>	셋에서 임의의 항목을 <b>삭제</b> 한다.
<b>clear</b>	<code>set.clear()</code>	셋에서 모든 항목을 <b>삭제</b> 한다.

### ○ 셋 복사

함 수	세부 내용	
<b>copy</b>	<code>set.copy()</code>	셋의 모든 항목을 <b>복사</b> 하여 새로운 셋을 반환한다.

## 셋 관련 함수 (2/5)

### 예제 0-31: 셋 조작 함수 -- 셋 항목 순회 및 복사

```
aSet = {'apple', 'banana', 'cherry'}  
print(aSet)  
  
# 셋의 항목 순회: 반복문  
for key in aSet:  
    print(key)  
  
# 셋의 항목 순회: Set Comprehension  
{ print(key) for key in aSet }  
  
# 셋 복사: copy 조작 함수  
# copySet = set(newSet)  
copySet = aSet.copy();           print(copySet)  
  
# 셋 복사: Set Comprehension  
newSet = { s for s in aSet };   print(newSet)
```



```
IDLE Shell 3.11.2  
File Edit Shell Debug Options Window Help  
Python 3.11.2 (tags/v3.11.2:878ead1,  
Type "help", "copyright", "credits" ,  
>>>  
===== RESTART: C:\Users\W  
'cherry', 'banana', 'apple'  
cherry  
banana  
apple  
cherry  
banana  
apple  
{'cherry', 'banana', 'apple'}  
{'cherry', 'banana', 'apple'}  
>>>
```

# 셋 관련 함수 (3/5)

## 예제 0-32: 셋 조작 함수 -- 셋 항목 추가 및 삭제

```
aSet = {'apple', 'banana', 'cherry'}  
print(aSet)
```

```
# 셋에 새로운 항목 추가  
aSet.add('orange');  
print(aSet)
```

```
# 셋의 특정 항목 삭제  
# newSet.remove('Banana') # KeyError: 'Banana'  
aSet.remove('banana');  
print(aSet)
```

```
aSet.discard('Apple');  
aSet.discard('apple');
```

```
# 셋의 특정(마지막) 항목 삭제  
key = aSet.pop();  
print(key)
```

```
# 셋의 모든 항목 삭제  
aSet.clear();  
del aSet  
# print(aSet) # NameError: name 'aSet' is not defined
```



The screenshot shows the Python IDLE Shell interface. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The shell window displays the following session:

```
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  
Type "help", "copyright", "credits" or "li  
>>>  
===== RESTART: C:\Users\click\w  
{'apple', 'cherry', 'banana'}  
{'apple', 'orange', 'cherry', 'banana'}  
{'apple', 'orange', 'cherry'}  
{'apple', 'orange', 'cherry'}  
{'orange', 'cherry'}  
orange  
set()  
>>>
```

# 셋 관련 함수 (4/5)

## ● 셋 조작 함수: 집합 연산

### ○ 셋 집합 연산

함 수	세부 내용	
<b>union</b>	set. <b>union</b> (set1, set2, ..., etc)	다른 두 셋의 <b>합집합</b> 인 셋을 반환한다. 단, 중복된 항목은 제외된다.
<b>intersection</b>	set. <b>intersection</b> (set1, set2, ..., etc)	다른 두 셋의 <b>교집합</b> 인 집합을 반환한다.
<b>difference</b>	set. <b>difference</b> (set)	두 셋의 <b>차집합</b> 을 셋으로 반환한다.
		첫 번째 셋에만 존재하고, 두 번째 셋에는 없는 항목을 반환한다.
<b>symmetric_difference</b>	set. <b>symmetric_difference</b> (set)	두 셋의 <b>대칭 차집합</b> 을 반환한다.
		두 셋의 모든 항목을 포함하지만, 두 셋에 있는 항목은 포함하지 않는 셋을 반환한다.

### • 집합 연산: 연산자

- | : 합집합(union)
- & : 교집합(intersection)
- - : 차집합(difference)
- ^ : 대칭 차집합(symmetric difference)

# 셋 관련 함수 (5/5)

## 예제 0-33: 셋 조작 함수 -- 집합 연산

```
aSet = {10, 20, 30, 40, 50, 60}
```

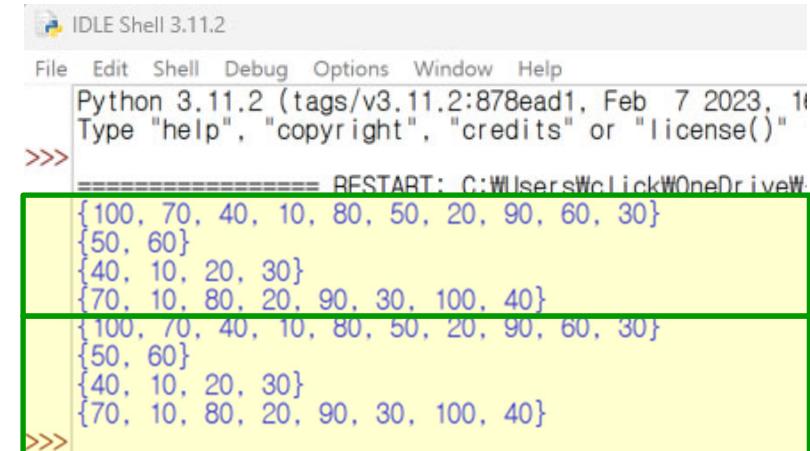
```
bSet = {50, 60, 70, 80, 90, 100}
```

```
# 연산자: |, &, -, ^
```

```
print(aSet | bSet)          # 합집합  
print(aSet & bSet)          # 교집합  
print(aSet - bSet)          # 차집합  
print(aSet ^ bSet)          # 대칭 차집합
```

```
# 셋 함수
```

```
print(aSet.union(bSet))      # 합집합  
print(aSet.intersection(bSet)) # 교집합  
print(aSet.difference(bSet)) # 차집합  
print(aSet.symmetric_difference(bSet)) # 대칭 차집합
```



The screenshot shows the Python IDLE Shell 3.11.2 interface. The code has been run, and the output is displayed in the shell window. The output shows two sets being combined using the union operator (|). The first set is {10, 20, 30, 40, 50, 60} and the second set is {50, 60, 70, 80, 90, 100}. The resulting sets from the union operation are highlighted with a green box. The first result is {100, 70, 40, 10, 80, 50, 20, 90, 60, 30}, and the second result is {100, 70, 40, 10, 80, 50, 20, 90, 60, 30}.

```
IDLE Shell 3.11.2  
File Edit Shell Debug Options Window Help  
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 11:48:57)  
Type "help", "copyright", "credits" or "license()"  
>>> ===== RESTART: C:\Users\Wclick\OneDrive\... =====  
{100, 70, 40, 10, 80, 50, 20, 90, 60, 30}  
{50, 60}  
{40, 10, 20, 30}  
{70, 10, 80, 20, 90, 30, 100, 40}  
{100, 70, 40, 10, 80, 50, 20, 90, 60, 30}  
{50, 60}  
{40, 10, 20, 30}  
{70, 10, 80, 20, 90, 30, 100, 40}  
>>>
```

# 객체지향 프로그래밍



- 파이썬 프로그래밍 기초

- 자료형과 자료 구조

백문이 불여일타(百聞而不如一打)

- 객체지향 프로그래밍

- 클래스와 객체

- 모듈과 패키지

- 파이썬 라이브러리

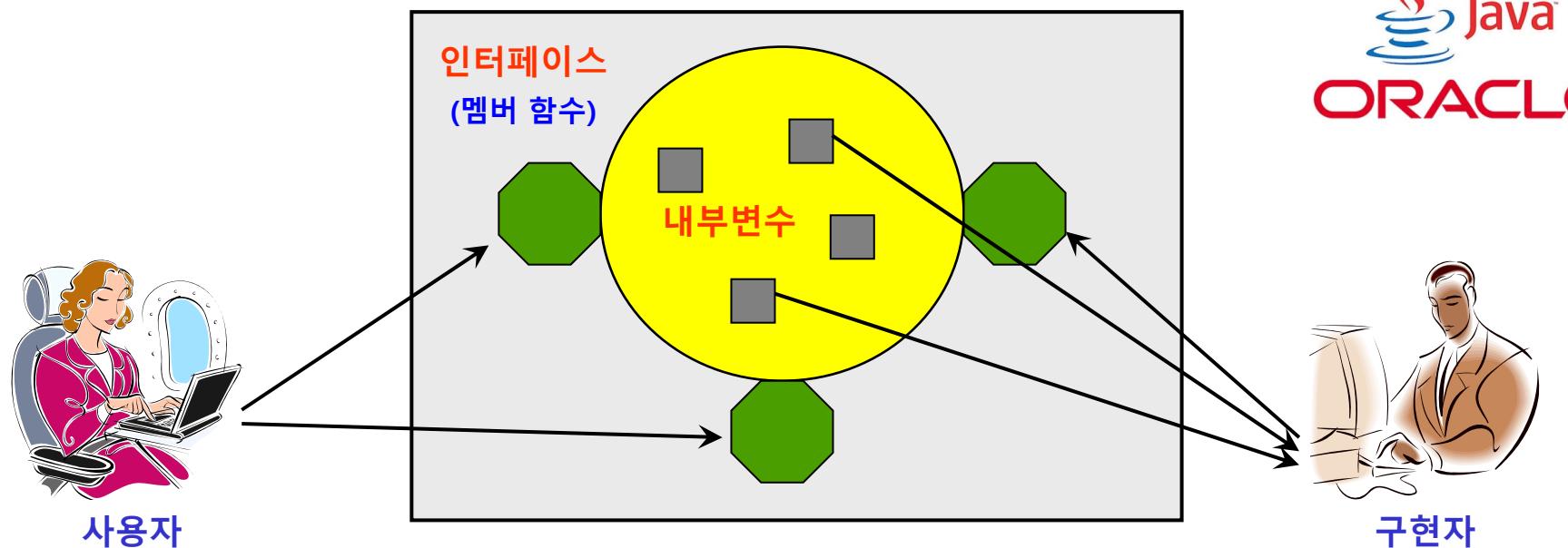


# 객체지향 프로그래밍의 이해 (1/2)

## ● 객체지향 프로그래밍(Object-Oriented Programming)

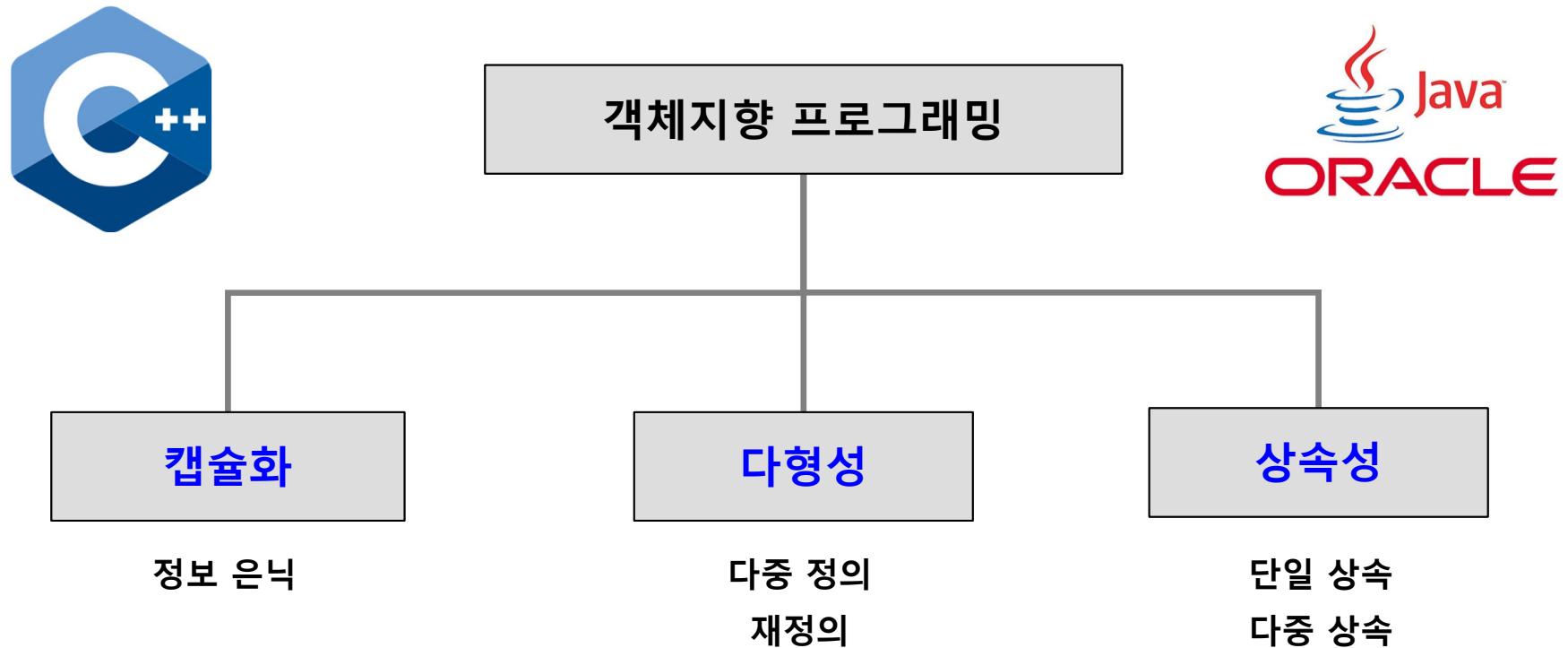
### ○ 객체(Object)들의 모임

- 구성 요소: 클래스, 객체, 메소드, 메시지
- 특징: 캡슐화, 추상화, 다형성, 상속, 인스턴스 등
- C++, JAVA, C# 등



# 객체지향 프로그래밍의 이해 (2/2)

- 객체지향 프로그래밍의 3가지 특징



# 구조체와 멤버 함수 (1/3)

## ● 구조체(Structure): C 프로그래밍 언어

```
// C Programming Language
struct Point {
    int      x;
    int      y;
    void OUTPUT(void) {
        cout << " x : " << x << " y : " << y << endl;
    }
};
```

“구조체 선언 시  
데이터를 조작(접근)하는 함수도 묶을 수는 없을까?”

# 구조체와 멤버 함수 (2/3)

## ● 구조체와 멤버 함수

- C++ 언어에서는 구조체에 함수 정의가 가능하다.

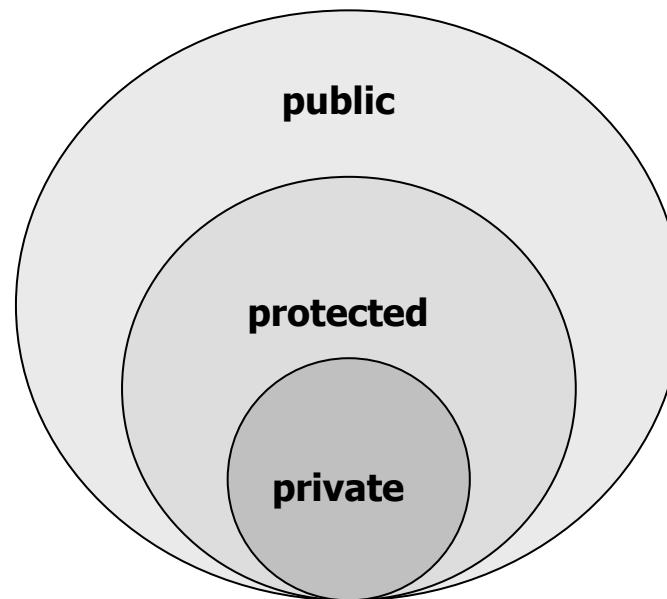
- 구조체에서는 default 접근자가 **public**이지만, 클래스에서는 **private**이 **default**이다.

```
struct Point {  
    // public:  
        // 멤버 변수  
        int      xPos;  
        int      yPos;  
        void    movePosition(int x, int y) {    // 멤버 함수  
            xPos += x;  
            yPos += y;  
        }  
        void    showPosition() {                // 멤버 함수  
            cout << "xPos: " << xPos << ", yPos: " << yPos << endl;  
        }  
};
```

# 구조체와 멤버 함수 (3/3)

## ● 구조체와 클래스의 멤버 접근 권한

- 클래스 내의 멤버 변수 또는 멤버 함수의 접근 권한 부여
  - **public** : 모든 클래스에서 접근 허용(외부 접근)
  - **protected** : 클래스와 상속받은 클래스의 접근 허용
  - **private** : 동일한 클래스의 접근만 허용(내부 접근)



# 클래스와 객체 (1/10)

## ● 클래스(Class)

### ○ 같은 목적을 가진 변수와 들의 집합체(새로운 자료형)

- **변수:** 속성(attribute)
- **함수:** 메소드(method)

클래스 = 멤버 변수 + 멤버 함수

### ○ 객체(Object)

- 클래스를 이용해서 정의된 자료형의 변수 표현(완전한 대상체)
- **인스턴스화**(instantiation): 클래스를 기반으로 객체를 생성하는 것

# 클래스와 객체 (2/10)

- **클래스(Class):** 클래스 정의

- 클래스는 상자를 만드는 틀에 비유할 수 있다.

```
# 자동차(Car) 클래스 정의
```

```
class Car:
```

```
    pass          # 객체를 정의하기 위한 최소한의 정의
```

```
# 자동차(Car) 클래스 정의
```

```
class Car():
```

```
    pass          # 객체를 정의하기 위한 최소한의 정의
```

- **class** 키워드로 클래스를 정의한다.



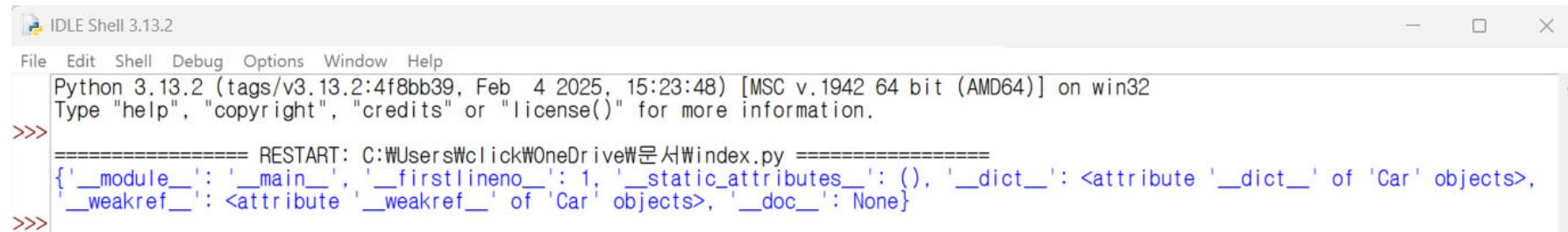
# 클래스와 객체 (3/10)

## 예제 0-34: 클래스 정의 -- 클래스 속성

# 사용자 정의 클래스: 자동차(Car) 클래스 정의

```
class Car:  
    pass
```

```
if __name__ == '__main__':  
    print(Car.__dict__) # 클래스 속성 확인
```



The screenshot shows the Python IDLE Shell 3.13.2 interface. The code in the shell window is:

```
File Edit Shell Debug Options Window Help  
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb 4 2025, 15:23:48) [MSC v.1942 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>> ===== RESTART: C:\Users\click\OneDrive\문서\Windex.py ======
```

The output shows the dictionary of the Car class:

```
{'__module__': '__main__', '__firstlineno__': 1, '__static_attributes__': (), '__dict__': <attribute '__dict__' of 'Car' objects>, '__weakref__': <attribute '__weakref__' of 'Car' objects>, '__doc__': None}
```

# 클래스가 정의될 때 자동으로 부여되는 몇 가지 기본 속성들.

# **\_\_module\_\_** : 클래스가 정의된 모듈의 이름

# **\_\_dict\_\_** : 클래스 속성을 담고 있는 딕셔너리(변수, 메소드, 기타 속성들)

# **\_\_doc\_\_** : 클래스에 작성된 Docstring

# **\_\_weakref\_\_** : 약한 참조(weak reference)를 지원하기 위한 슬롯.

# Python 3.13.2에서 새로 추가된 속성

# **\_\_firstlineno\_\_** : 디버깅이나 디어셈블 과정에서 내부적으로 코드 객체의 줄 번호 확인.

# **\_\_static\_attributes\_\_** : 특정 프레임워크나 확장 기능에서 사용.

# 클래스와 객체 (4/10)

## ● 클래스(Class): 인스턴스화

### ○ 인스턴스화(instantiation)

- 클래스를 기반으로 객체를 생성하는 것

```
# 자동차(Car) 클래스 정의
class Car:
    pass

if __name__ == '__main__':
    # 객체 생성: myCar1, myCar2, myCar3
    myCar1 = Car()
    myCar2 = Car()
    myCar3 = Car()

# isinstance 내장 함수: 특정 클래스의 인스턴스인지 여부 판단
print(isinstance(myCar1, Car))      # True
print(isinstance(myCar2, Car))      # True
print(isinstance(myCar3, Car))      # True
```

# 클래스와 객체 (5/10)

예제 0-35: 클래스와 객체 -- 인스턴스화 및 속성 추가

(1/3)

# \_\_slots\_\_ 에 허용할 속성 이름을 리스트로 넣어준다.

```
# 자동차(Car) 클래스 정의
```

```
class Car:
```

```
    # Docstring: 파이썬 스크립트 내부에 함수, 클래스, 모듈에 대한 정보를 적어 놓은 문자열
```

```
    '''자동차 클래스'''
```

```
    # Docstring
```

```
    __slots__ = ['model', 'color', 'speed']
```

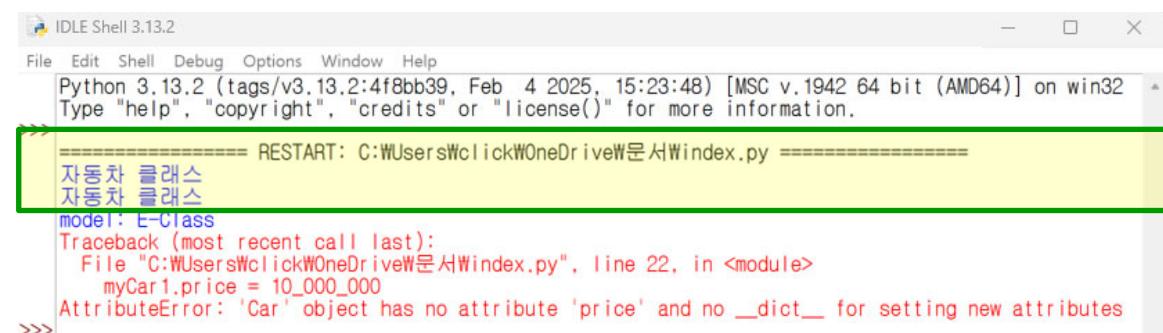
```
    # model, color, speed만 생성 허용
```

```
if __name__ == '__main__':
```

```
    myCar1 = Car()
```

```
print(Car.__doc__)
```

```
print(myCar1.__doc__)
```



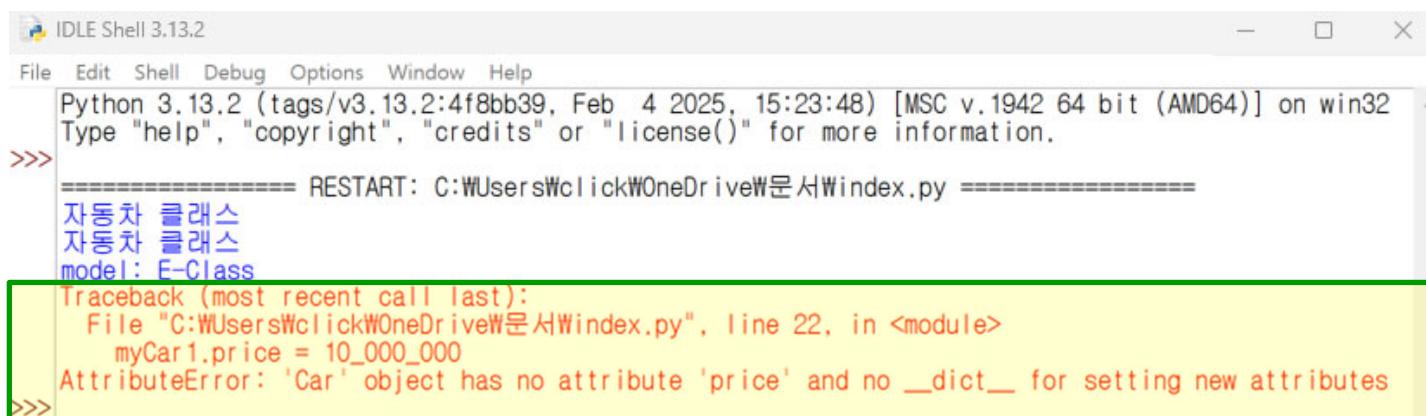
The screenshot shows the Python IDLE Shell interface. The title bar says "IDLE Shell 3.13.2". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The status bar at the bottom indicates "Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb 4 2025, 15:23:48) [MSC v.1942 64 bit (AMD64)] on win32". The shell window displays the following text:  
===== RESTART: C:\Users\click\OneDrive\문서\Windex.py ======  
자동차 클래스  
자동차 클래스  
model: E-Class  
Traceback (most recent call last):  
 File "C:\Users\click\OneDrive\문서\Windex.py", line 22, in <module>  
 myCar1.price = 10\_000\_000  
AttributeError: 'Car' object has no attribute 'price' and no \_\_dict\_\_ for setting new attributes  
>>>

# 클래스와 객체 (6/10)

## 예제 0-35: 클래스와 객체 -- 인스턴스화 및 속성 추가

(2/3)

```
# 객체를 생성한 뒤에 속성 추가  
myCar1.model = 'E-Class'  
print(f'model: {myCar1.model}')  
  
# AttributeError: 'Car' object has no attribute 'price'  
# and no __dict__ for setting new attributes  
# model, color, speed만 생성 허용(다른 속성은 생성 제한)  
myCar1.price = 10_000_000
```



# 클래스와 객체 (7/10)

예제 0-35: 클래스와 객체 -- 인스턴스화 및 속성 추가

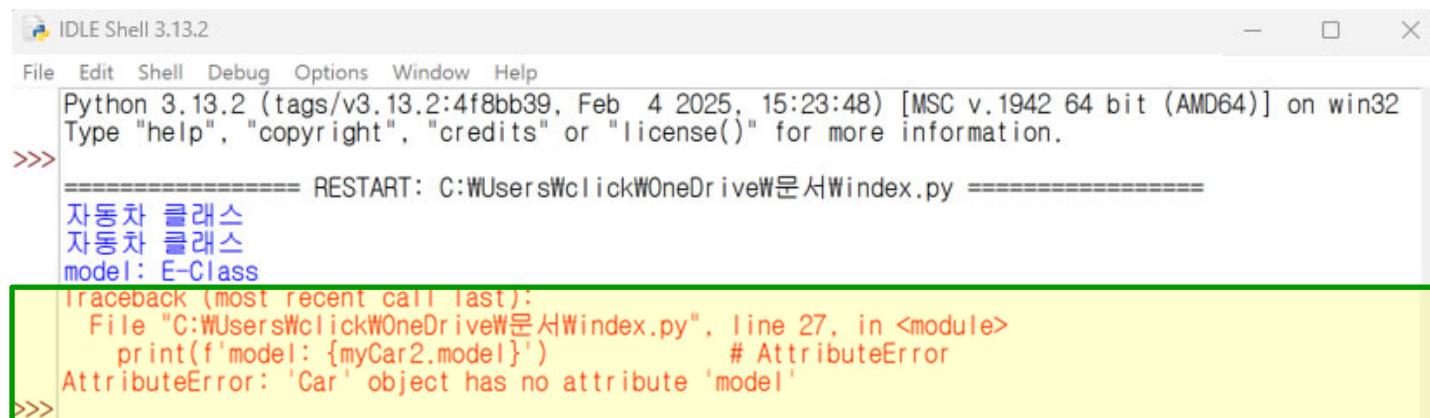
(3/3)

`myCar2 = Car()`

# 다른 인스턴스를 만들 때는 추가 속성이 생성되지 않는다.

# AttributeError: 'Car' object has no attribute 'model'

`print(f'model: {myCar2.model}')`



The screenshot shows the IDLE Shell interface with the title bar "IDLE Shell 3.13.2". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the Python interpreter prompt "Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb 4 2025, 15:23:48) [MSC v.1942 64 bit (AMD64)] on win32". Below the prompt, it says "Type "help", "copyright", "credits" or "license()" for more information." A red box highlights the error message at the bottom of the window:

```
>>> ===== RESTART: C:\Users\click\OneDrive\문서\Windex.py =====
자동차 클래스
자동차 클래스
model: E-Class
>>> traceback (most recent call last):
      File "C:\Users\click\OneDrive\문서\Windex.py", line 27, in <module>
        print(f'model: {myCar2.model}')          # AttributeError
    AttributeError: 'Car' object has no attribute 'model'
```

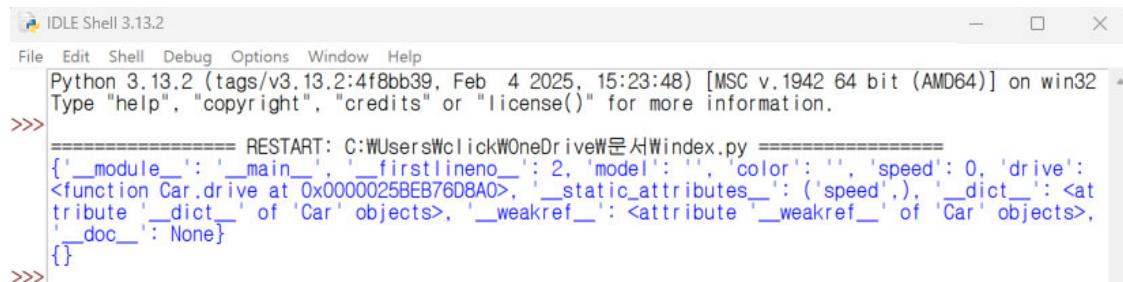
# 클래스와 객체 (8/10)

예제 0-36: 클래스와 객체 -- 인스턴스화, 속성 추가 및 메소드 실행 (1/3)

# 사용자 정의 클래스: 자동차(Car) 클래스 정의

class Car:

```
model = ''  
color = ''  
speed = 0  
  
def drive(self):          # self: 객체 자신을 참조하도록 지정  
    self.speed = 100  
  
  
if __name__ == '__main__':  
    # 객체 생성: myCar1  
    myCar1 = Car()  
  
  
print(Car.__dict__)  
print(myCar1.__dict__)
```



```
IDLE Shell 3.13.2  
File Edit Shell Debug Options Window Help  
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb 4 2025, 15:23:48) [MSC v.1942 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
==== RESTART: C:\Users\click\OneDrive\문서\index.py ======  
>>> {'__module__': '__main__', '__firstlineno__': 2, 'model': '', 'color': '', 'speed': 0, 'drive': <function Car.drive at 0x00000258EB76D8A0>, '__static_attributes__': ('speed',), '__dict__': <attribute '__dict__' of 'Car' objects>, '__weakref__': <attribute '__weakref__' of 'Car' objects>, '__doc__': None}  
>>>
```

# 클래스와 객체 (9/10)

## 예제 0-36: 클래스와 객체 -- 인스턴스화, 속성 추가 및 메소드 실행 (2/3)

# 객체를 생성한 뒤에 속성 추가

myCar1.model = 'E-Class'

# 속성(model): 자동차 모델

myCar1.color = 'blue'

# 속성(color): 자동차 색깔

myCar1.speed = 150

# 속성(speed): 자동차 속도

```
print(f'model: {myCar1.model}')
```

```
print(f'color: {myCar1.color}')
```

```
print(f'speed: {myCar1.speed}')
```

# 메소드(method) 실행: drive()

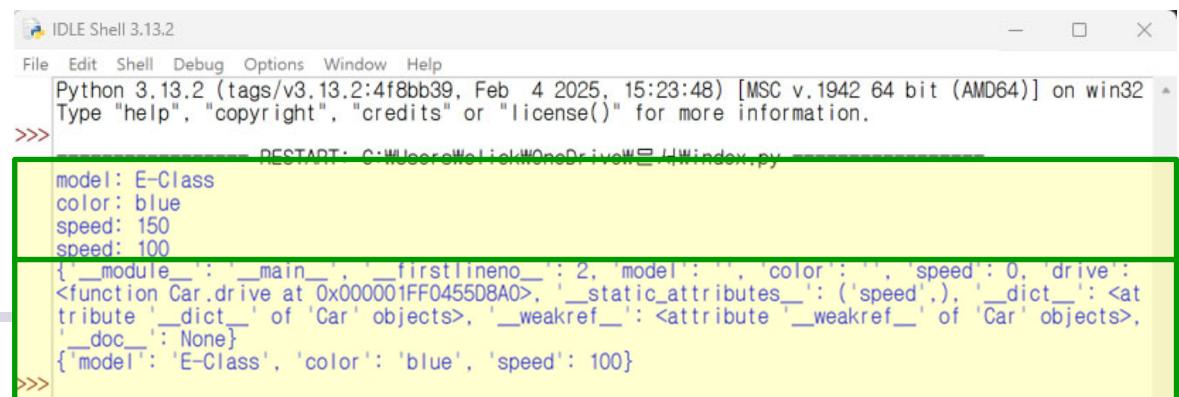
myCar1.drive()

# Car.drive(myCar1)

```
print(f'speed: {myCar1.speed}')
```

```
print(Car.__dict__)
```

```
print(myCar1.__dict__)
```



The screenshot shows the IDLE Shell interface with the following content:

```
IDLE Shell 3.13.2
File Edit Shell Debug Options Window Help
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb 4 2025, 15:23:48) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> RESTART: C:\Users\click\OneDrive\문서\window.py
model: E-Class
color: blue
speed: 150
speed: 100
{'__module__': '__main__', '__lineno__': 2, 'model': '', 'color': '', 'speed': 0, 'drive':
<function Car.drive at 0x000001FF0455D8A0>, '__static_attributes__': ('speed',), '__dict__': <attribute '__dict__' of 'Car' objects>, '__weakref__': <attribute '__weakref__' of 'Car' objects>, '__doc__': None}
{'model': 'E-Class', 'color': 'blue', 'speed': 100}
```

# 클래스와 객체 (10/10)

예제 0-36: 클래스와 객체 -- 인스턴스화, 속성 추가 및 메소드 실행 (3/3)

```
# 객체 생성: myCar2, myCar3
myCar2 = Car()
myCar3 = Car()

print(f'model: {myCar2.model}')
print(f'color: {myCar2.color}')
print(f'speed: {myCar2.speed}')

print(Car.__dict__)
print(myCar1.__dict__)
print(myCar2.__dict__)
print(myCar3.__dict__)
```

```
IDLE Shell 3.13.2
File Edit Shell Debug Options Window Help
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb 4 2025, 15:23:48) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> ===== RESTART: C:\Users\click\OneDrive\문서\Windex.py =====
model: E-Class
color: blue
speed: 150
speed: 100
model:
color:
speed: 0
t __module__ . __main__ . __firsttimed : 2, model : , color : , speed : 0, drive :
<function Car.drive at 0x0000021D7D00908A0>, __static_attributes__ : ('speed',), __dict__ : <attribute '__dict__' of 'Car' objects>, __weakref__ : <attribute '__weakref__' of 'Car' objects>, __doc__ : None}
{'model': 'E-Class', 'color': 'blue', 'speed': 100}
{}
```

# 클래스와 객체: 생성자와 소멸자 (1/6)

## ● 생성자(Constructor)

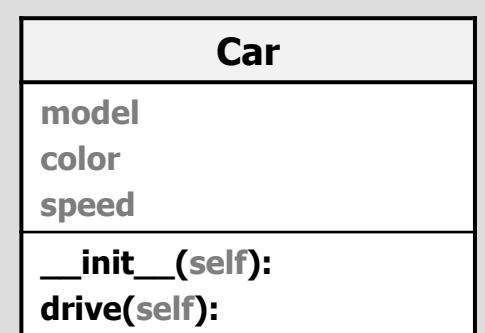
- 객체의 생성과 동시에 호출되는 함수
- \_\_init\_\_ : 객체 초기화
  - 클래스 정의에서 개별 객체를 초기화하는 함수
    - 모든 클래스 정의에서 생성자 함수(\_\_init\_\_)를 가질 필요는 없다.

```
# 자동차(Car) 클래스 정의
class Car:
    def __init__(self):
        self.model = 'IONIQ'
        self.color = 'silver'
        self.speed = 0
    def drive(self):
        self.speed = 100

if __name__ == '__main__':
    # 객체 생성: myCar
    myCar = Car()
```

# 생성자 함수

# 생성자 함수(\_\_init\_\_): 객체 생성과 동시에 호출



# 클래스와 객체: 생성자와 소멸자 (2/6)

## ● 생성자: 객체 초기화

### ○ 매개 변수가 있는 생성자

```
# 자동차(Car) 클래스 정의
```

```
class Car:
```

```
    def __init__(self, model="", color="", speed=0):
```

```
        self.model = model
```

```
        self.color = color
```

```
        self.speed = speed
```

```
    def drive(self):
```

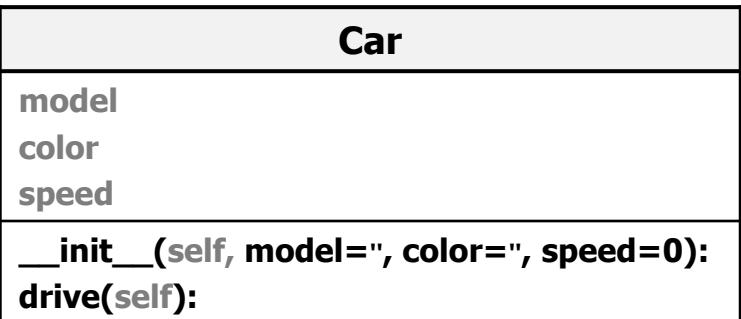
```
        self.speed = 100
```

```
if __name__ == '__main__':
```

```
    # 객체 생성: myCar
```

```
    myCar1 = Car()
```

```
    myCar2 = Car('E-class', 'blue', 0)
```



```
# 자동차(Car) 클래스 정의
```

```
class Car:
```

```
    def __init__(self, *args):
```

```
        self.model = args[0]
```

```
        self.color = args[1]
```

```
        self.speed = args[2]
```

```
    def drive(self):
```

```
        self.speed = 100
```

```
if __name__ == '__main__':
```

```
    # 객체 생성: myCar
```

```
    # myCar = Car('E-class', 'blue', 0)
```

```
    myCar = Car(*['E-class', 'blue', 0])
```

# 클래스와 객체: 생성자와 소멸자 (3/6)

## ● 소멸자(Destructor)

### ○ 객체가 소멸될 때 호출되는 함수

- 객체 소멸 시 다양한 형태의 정리 작업 필요 시...

### ○ `__del__`

```
# 자동차(Car) 클래스 정의
```

```
class Car:
```

```
    def __init__(self, model="", color="", speed=0):          # 생성자 함수
```

```
        self.model = 'iClickseo'
```

```
        self.color = 'black'
```

```
        self.speed = 0
```

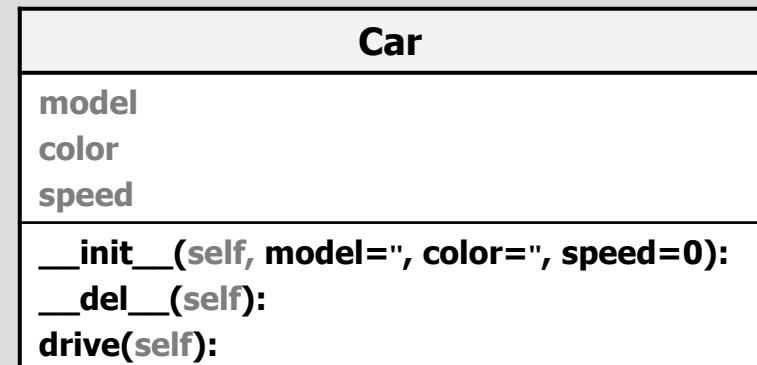
```
    def __del__(self):      # 소멸자 함수
```

```
        pass
```

```
    def drive(self):
```

```
        self.speed = 100
```

```
if __name__ == '__main__':  
    # 객체 생성: myCar  
    myCar = Car()
```



# 클래스와 객체: 생성자와 소멸자 (4/6)

예제 0-37: 클래스와 객체 -- 클래스 변수와 객체 변수

(1/3)

```
# 자동차(Car) 클래스 정의
class Car:
    # 클래스 변수: 모든 객체가 공유하는 변수
    count = 0
    speed = 0

    # 생성자 함수: 객체가 생성될 때 자동 호출되는 함수
    def __init__(self, model="", color="", speed=0):
        # 객체 변수: 각각의 객체가 고유하게 가지는 변수
        self.model = model
        self.color = color
        self.speed = speed
        Car.count += 1

    # 소멸자 함수: 객체가 소멸될 때 자동 호출되는 함수
    def __del__(self):
        Car.count -= 1

    def drive(self):
        self.speed = 100 # km/h
```

# 클래스와 객체: 생성자와 소멸자 (5/6)

예제 0-37: 클래스와 객체 -- 클래스 변수와 객체 변수

(2/3)

```
if __name__ == '__main__':
    # 객체 생성과 클래스 변수
    myCar1 = Car('IONIQ', 'silver', 0)
    print(Car.count)                      # 1

    myCar2 = Car('IONIQ 5', 'white', 0)
    print(Car.count)                      # 2

    myCar3 = Car('IONIQ 9', 'black', 0)
    print(Car.count)                      # 3

    # 클래스 변수와 객체 변수
    myCar1.drive(); print(f'speed: {myCar1.speed}')
    myCar2.drive(); print(f'speed: {myCar2.speed}')
    myCar3.drive(); print(f'speed: {myCar3.speed}')

    # 클래스 변수 (speed)
    print(f'speed: {Car.speed}')
```

```
IDLE Shell 3.13.2
File Edit Shell Debug Options Window Help
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb  4 2025, 15:23:48) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

=====
RESTART: C:\Users\Click\OneDrive\문서\WinIndex.py =====
1
2
3
speed: 100
speed: 100
speed: 100
speed: 0
>>>
```

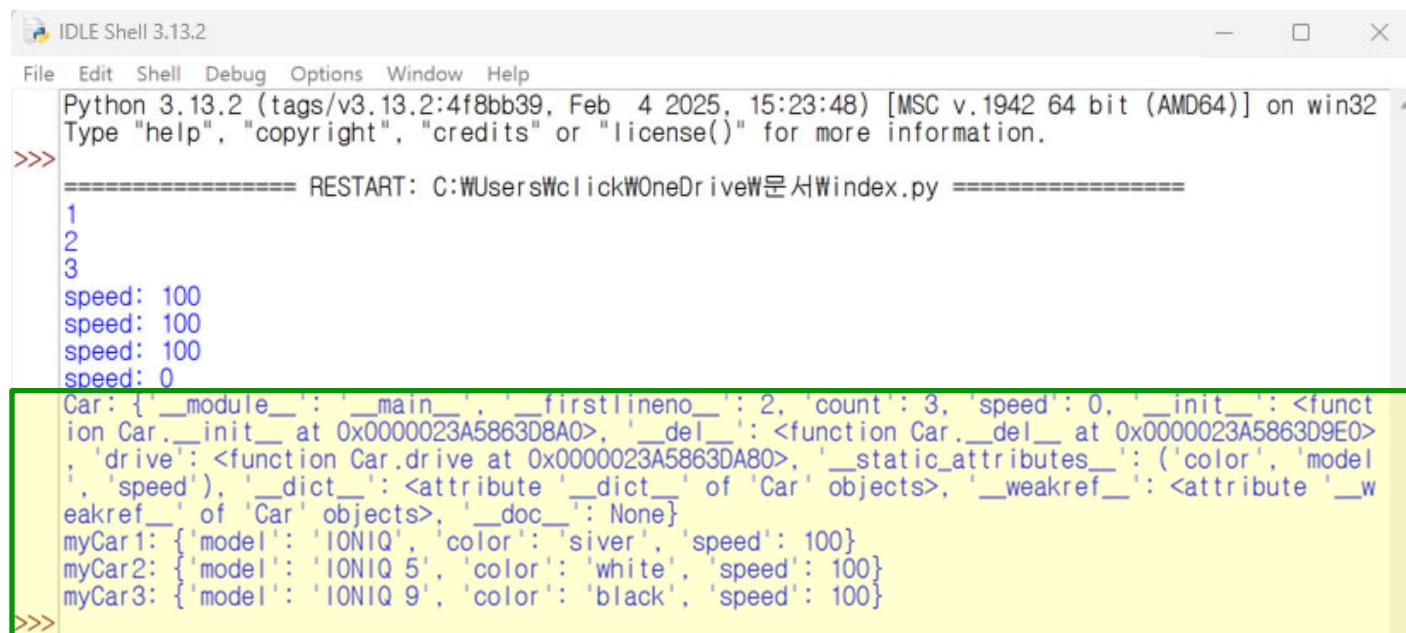
# 클래스와 객체: 생성자와 소멸자 (6/6)

예제 0-37: 클래스와 객체 -- 클래스 변수와 객체 변수

(3/3)

# 객체 속성 확인

```
print(f'Car: {Car.__dict__} ')
print(f'myCar1: {myCar1.__dict__} ')
print(f'myCar2: {myCar2.__dict__} ')
print(f'myCar3: {myCar3.__dict__} )
```



The screenshot shows the Python 3.13.2 IDLE shell interface. The code prints the \_\_dict\_\_ attribute of the Car class and three instances of the myCar class. A green box highlights the output for the Car class, which includes its static attributes (color, model, speed) and its \_\_dict\_\_ attribute, which points to the \_\_dict\_\_ attribute of the 'Car' objects.

```
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb  4 2025, 15:23:48) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> ===== RESTART: C:\Users\click\OneDrive\문서\windex.py =====
1
2
3
speed: 100
speed: 100
speed: 100
speed: 0
Car: {'__module__': '__main__', '__firstlineno__': 2, 'count': 3, 'speed': 0, '__init__': <function Car.__init__ at 0x0000023A5863D8A0>, '__del__': <function Car.__del__ at 0x0000023A5863D9E0>, 'drive': <function Car.drive at 0x0000023A5863DA80>, '__static_attributes__': ('color', 'model', 'speed'), '__dict__': <attribute '__dict__' of 'Car' objects>, '__weakref__': <attribute '__weakref__' of 'Car' objects>, '__doc__': None}
myCar1: {'model': 'IONIQ', 'color': 'siver', 'speed': 100}
myCar2: {'model': 'IONIQ 5', 'color': 'white', 'speed': 100}
myCar3: {'model': 'IONIQ 9', 'color': 'black', 'speed': 100}
```

# 캡슐화와 정보 은닉 (1/5)

## ● 클래스(Class): 멤버 접근 권한

- **클래스 멤버 접근:** 클래스 내의 멤버 변수 또는 멤버 함수의 접근 권한 부여
  - **public** : 모든 클래스에서 접근 허용(**외부 접근**)
    - 해당 속성 앞에 접두사가 없는 형태(예: value)
    - 만약, 접미사의 underscore(\_)가 2개 이상이면 public으로 간주한다(예: \_\_init\_\_ ).
  - **protected** : 클래스와 상속받은 클래스의 접근 허용
    - 해당 속성 앞에 single underscore(\_)를 붙여서 사용한다(예: \_value).
  - **private** : 동일한 클래스의 접근만 허용(**내부 접근**)
    - 해당 속성 앞에 double underscore(\_\_)를 붙여서 사용한다(예: \_\_value).

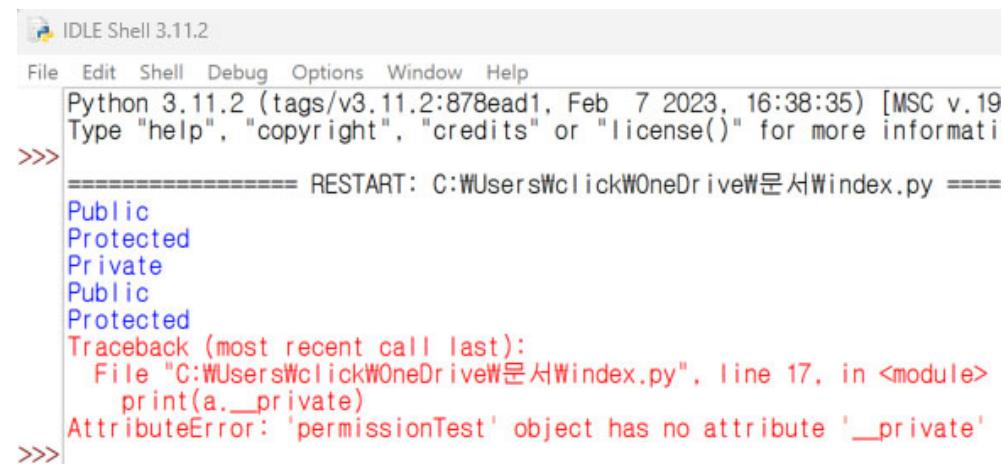


## 캡슐화와 정보 은닉 (2/5)

예제 0-38: 클래스의 멤버 접근 권한 -- public, protected(), private()

```
class PermissionTest:  
    def __init__(self):  
        self.public = 'Public'  
        self._protected = 'Protected'  
        self.__private = 'Private'  
  
    def permissionValue(self):  
        print(self.public)  
        print(self._protected)  
        print(self.__private)  
  
    if __name__ == '__main__':  
        a = PermissionTest()  
        a.permissionValue()  
  
        print(a.public)  
        print(a._protected) # 실제로 제약되지는 않고(강제성이 없음), 일종의 경고 표시로 사용됨.  
  
        # AttributeError: 'permissionTest' object has no attribute '__private'  
        print(a.__private)
```

PermissionTest
public
_protected
__private
permissionValue(self):



The screenshot shows the Python IDLE shell interface. The title bar says "IDLE Shell 3.11.2". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the following code and its output:

```
File Edit Shell Debug Options Window Help  
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC v.1919  
Type "help", "copyright", "credits" or "license()" for more information  
===== RESTART: C:\Users\click\OneDrive\문서\index.py =====  
Public  
Protected  
Private  
Public  
Protected  
Traceback (most recent call last):  
  File "C:\Users\click\OneDrive\문서\index.py", line 17, in <module>  
    print(a.__private)  
AttributeError: 'permissionTest' object has no attribute '__private'
```

===== RESTART: C:\Users\click\OneDrive\문서\index.py =====  
Public  
Protected  
Private  
Public  
Protected  
Traceback (most recent call last):  
 File "C:\Users\click\OneDrive\문서\index.py", line 17, in <module>  
 print(a.\_\_private)  
AttributeError: 'permissionTest' object has no attribute '\_\_private'

# 캡슐화와 정보 은닉 (3/5)

## ● 클래스(Class): 좋은 클래스의 설계

### ○ 캡슐화(Encapsulation)

- 관련 있는 데이터와 함수를 하나의 단위로 묶는다.

### ○ 정보은닉(Information Hiding)

- 모든 멤버 변수를 private으로 선언!!!
  - 객체의 외부에서 객체의 멤버 변수에 직접 접근하지 못하게 하는 것.
  - 오직 객체의 멤버 함수를 통하여 접근하도록 하는 방법

# 클래스 정의: Point

class Point:

def \_\_init\_\_(self):

    self.\_\_x = 0

    self.\_\_y = 0

def showPoint(self):

    print(f'x: {self.\_\_x}, y: {self.\_\_y}')

멤버 변수: private

Point

\_\_x;

\_\_y;

\_\_init\_\_(self):

showPoint(self):

# 캡슐화와 정보 은닉 (4/5)

예제 0-39: 캡슐화와 정보 은닉 -- 반복자(iterator)

(1/2)

```
class MyItertor:  
    def __init__(self, data):  
        self.__data = data  
        self.__pos = 0
```

# 클래스에 \_\_iter\_\_ 함수 구현: 해당 클래스로 생성한 객체는 반복 가능한 객체가 된다.

```
def __iter__(self):  
    return self
```

# 클래스에 \_\_iter\_\_ 함수를 구현할 경우 반드시 \_\_next\_\_ 함수를 구현해야 한다.

# \_\_next\_\_ 함수는 반복 가능한 객체의 값을 차례대로 반환하는 역할을 한다.

```
def __next__(self):  
    if self.__pos >= len(self.__data):  
        raise StopIteration  
    res = self.__data[self.__pos]  
    self.__pos += 1  
    return res
```

# 캡슐화와 정보 은닉 (5/5)

예제 0-39: 캡슐화와 정보 은닉 -- 반복자(iterator)

(2/2)

```
if __name__ == '__main__':
    aIter = MyIterator([10, 20, 30, 40, 50])
    for item in aIter:
        print(item)

# it = iter(aIter)
# print(next(it))      # 10
# print(next(it))      # 20
# print(next(it))      # 30
# print(next(it))      # 40
# print(next(it))      # 50
# print(next(it))      # StopIteration 예외 발생
```

```
IDLE Shell 3.11.2
File Edit Shell Debug Options
Python 3.11.2 (tags/v3.11
Type "help", "copyright",
>>> ===== RESTART:
10
20
30
40
50
>>>
```



# 객체지향 프로그래밍

클래스와 객체

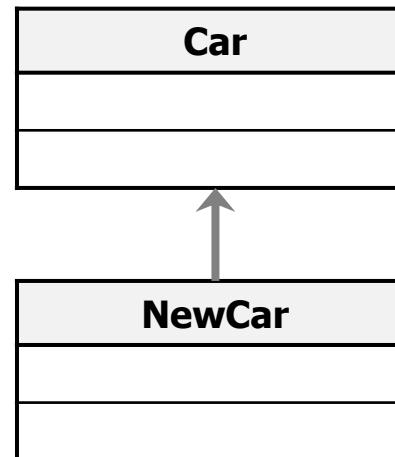
: 상속과 다형성, 다중 상속



# 상속과 다형성 (1/6)

## ● 상속(Inheritance)

- 클래스를 정의할 때 부모 클래스를 지정한다.
  - 자식 클래스는 부모 클래스의 변수와 함수들을 사용할 수 있다.
- 클래스의 상속
  - 기존 클래스에서 일부를 추가하거나 변경하여 새로운 클래스를 생성한다.
  - 코드를 재사용하는 아주 좋은 방법이다.



# 상속과 다형성 (2/6)

## ● 상속: 클래스의 상속

### ○ 부모 클래스와 자식 클래스

- 부모 클래스(Parent Class): 기본 클래스(Base Class), Super Class
- 자식 클래스(Child Class): 파생 클래스(Derived class), Sub Class

### ○ 클래스의 상속 구현

# object class: 모든 클래스의 조상 클래스

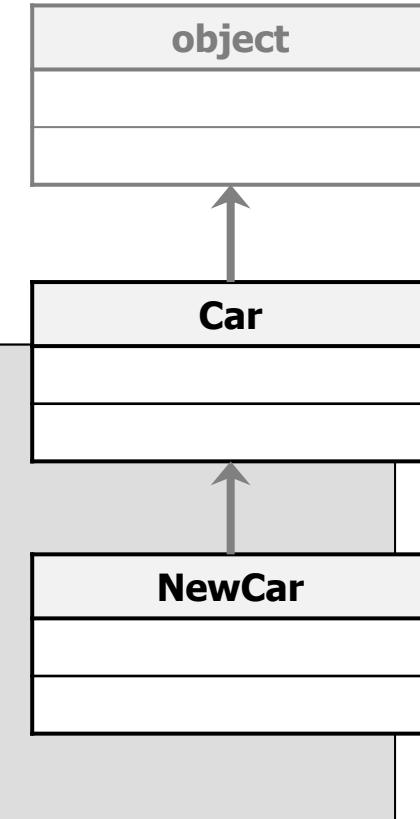
# 자동차(Car) 클래스 정의

```
class Car:  
    pass
```

# 새로운 자동차(NewCar) 클래스 정의(상속)

```
class NewCar(Car):  
    pass
```

- issubclass 내장 함수: 다른 클래스에서 파생되었는지 확인할 수 있다.



# 상속과 다형성 (3/6)

## 예제 0-40: 클래스의 상속 -- Car 클래스와 NewCar 클래스

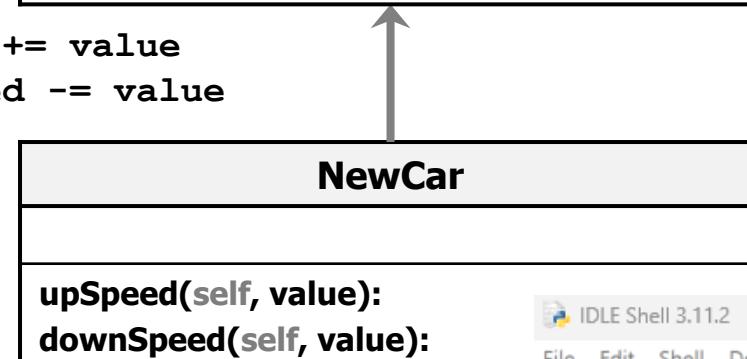
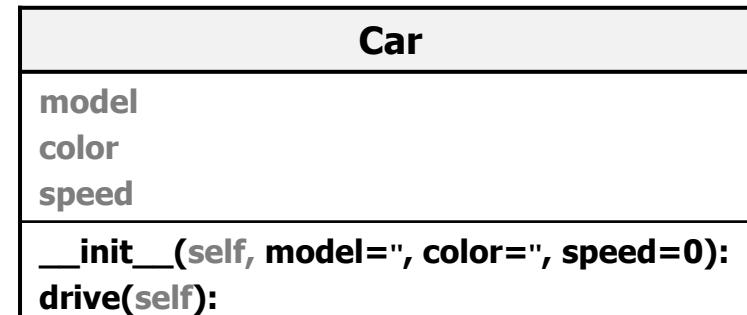
# 자동차(Car) 클래스 정의

```
class Car:  
    def __init__(self, model="", color="", speed=0):  
        self.model = model  
        self.color = color  
        self.speed = speed  
    def drive(self):  
        self.speed = 100
```

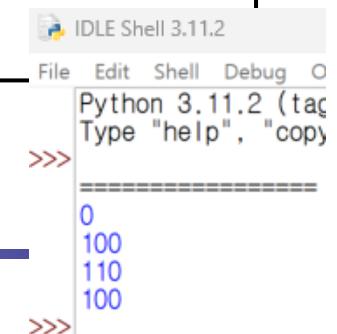
# 새로운 자동차(NewCar) 클래스 정의(상속)

```
class NewCar(Car):  
    def upSpeed(self, value): self.speed += value  
    def downSpeed(self, value): self.speed -= value
```

```
if __name__ == '__main__':  
    # 객체 생성: myCar  
    myCar = NewCar('iClickseo', 'white', 0)  
    print(myCar.speed)  
  
    myCar.drive();  
    myCar.upSpeed(10);  
    myCar.downSpeed(10);
```



```
print(myCar.speed)  
print(myCar.speed)  
print(myCar.speed)
```



# 상속과 다형성 (4/6)

## ● 메소드 재정의(Method Overriding)

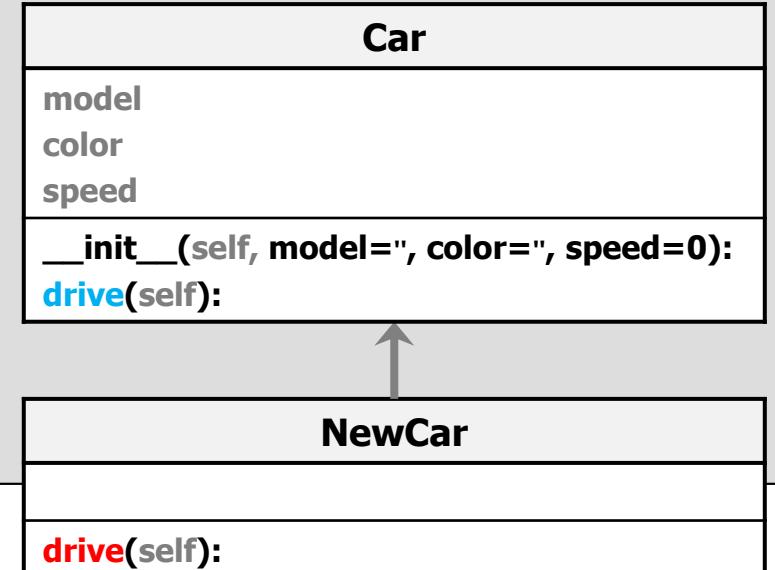
- 상위 클래스의 메소드를 하위 클래스에서 재정의

```
# 자동차(Car)클래스 정의
```

```
class Car:  
    def __init__(self, model="", color="", speed=0):  
        self.model = model  
        self.color = color  
        self.speed = speed  
    def drive(self):  
        self.speed = 100
```

```
# 새로운 자동차(NewCar) 클래스 정의(상속)
```

```
class NewCar(Car):  
    def drive(self):          # 메소드 재정의  
        self.speed = 50
```



# 상속과 다형성 (5/6)

## 예제 0-41: 클래스의 상속과 메소드 재정의

(1/2)

# 자동차(Car) 클래스 정의

class Car:

```
def __init__(self, model='', color='', speed=0):
    self.model = model
    self.color = color
    self.speed = speed
def drive(self):
    self.speed = 100
```

# 새로운 자동차(NewCar) 클래스 정의(상속)

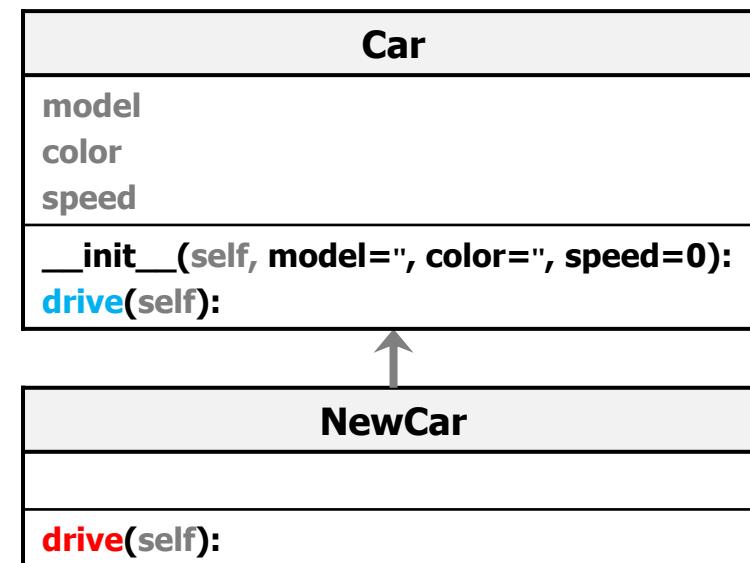
class NewCar(Car):

```
def drive(self):      # 메소드 재정의
    self.speed = 50

if __name__ == '__main__':
    myCar = NewCar('iClickseo', 'white', 0)
    print(myCar.speed)

    myCar.drive()
    print(myCar.speed)
```

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 7 2023, 16:38:35) [MSC v.193
Type "help", "copyright", "credits" or "license()" for more information
>>> ===== RESTART: C:\Users\click\OneDrive\문서\index.py =====
0
50
[<class '__main__.Car'>, <class 'object'>]
[<class '__main__.NewCar'>, <class '__main__.Car'>, <class 'object'>]
>>>
```



# 상속과 다형성 (6/6)

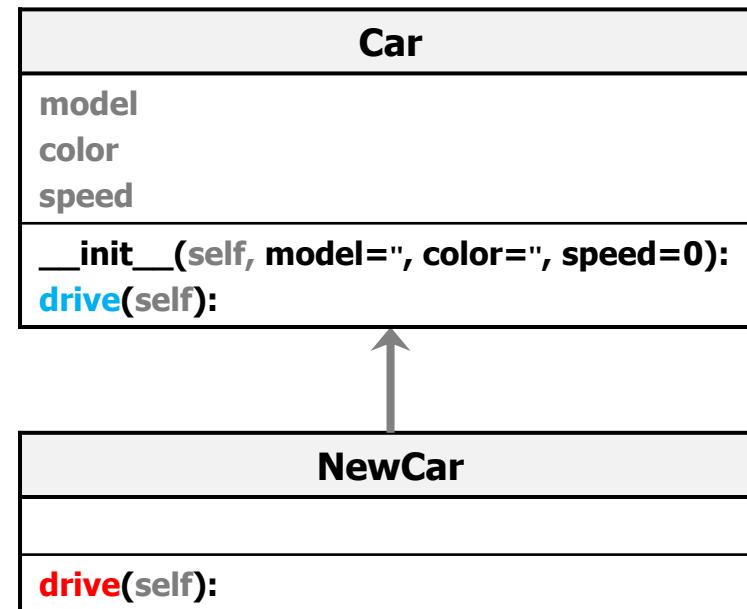
예제 0-41: 클래스의 상속과 메소드 재정의

(2/2)

```
# 메소드 결정 순서: MRO(Method Resolution Order)
print(Car.mro())
# <class '__main__.Car'>, <class 'object'>

print(NewCar.mro())
# <class '__main__.newCar'>, <class '__main__.Car'>, <class 'object'>
```

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window Help
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.193
Type "help", "copyright", "credits" or "license()" for more information
>>> ===== RESTART: C:\Users\click\OneDrive\문서\index.py =====
0
50
[<class '__main__.Car'>, <class 'object'>]
[<class '__main__.newCar'>, <class '__main__.Car'>, <class 'object'>]
>>>
```



# 추상 클래스 (1/3)

## ● 추상 메소드(Abstract Method)

- 상위 클래스에서는 빈 껍질의 메소드만 만들어 놓고, 하위 클래스에서 정의한다.

```
# 자동차(Car) 클래스 정의
```

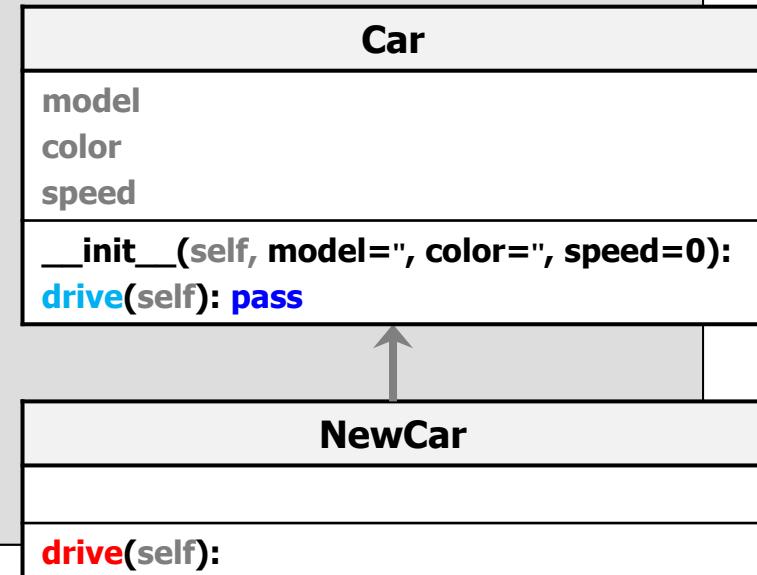
```
class Car:
```

```
    def __init__(self, model, color, speed):  
        self.model = model  
        self.color = color  
        self.speed = speed  
    def drive(self): # 추상 메소드  
        pass
```

```
# 새로운 자동차(NewCar) 클래스 정의(상속)
```

```
class NewCar(Car):
```

```
    def drive(self): # 메소드 재정의  
        self.speed = 50
```



# 추상 클래스 (2/3)

## 예제 0-42: 추상 클래스와 추상 메소드

(1/2)

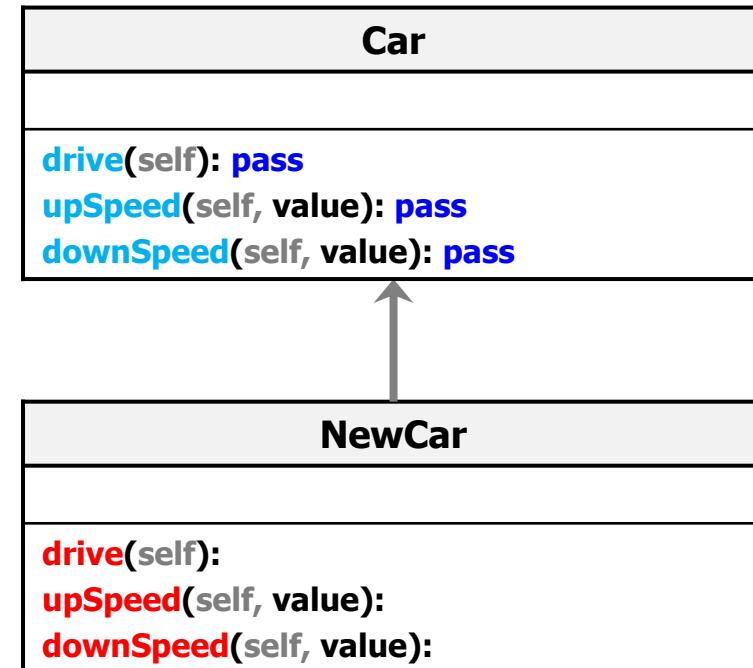
```
# abc: Abstract Base Class
from abc import *

# 추상 클래스 정의: Car 클래스
class Car(metaclass=ABCMeta):
    @abstractmethod
    def drive(self):
        pass

    @abstractmethod
    def upSpeed(self, value):
        pass

    @abstractmethod
    def downSpeed(self, value):
        pass

# 추상 클래스를 상속받았다면...
# @abstractmethod가 붙은 추상 메서드를 모두 구현해야 한다.
# 새로운 자동차(newCar) 클래스 정의: 상속
class NewCar(Car):
    def drive(self):
        self.speed = 50
    def upSpeed(self, value):
        self.speed += value
    def downSpeed(self, value):
        self.speed -= value
```



# 추상 클래스 (3/3)

예제 0-42: 추상 클래스와 추상 메소드

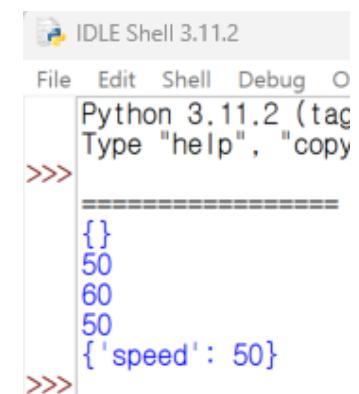
(2/2)

```
if __name__ == '__main__':
    # 추상 클래스는 인스턴스화 할 수 없다.
    # TypeError: Can't instantiate abstract class Car
    #           with abstract methods downSpeed, drive, upSpeed
    # myCar = Car()

    # 객체 생성: myNewCar
    myNewCar = NewCar()
    print(myNewCar.__dict__)

    myNewCar.drive();      print(myNewCar.speed)
    myNewCar.upSpeed(10);  print(myNewCar.speed)
    myNewCar.downSpeed(10); print(myNewCar.speed)

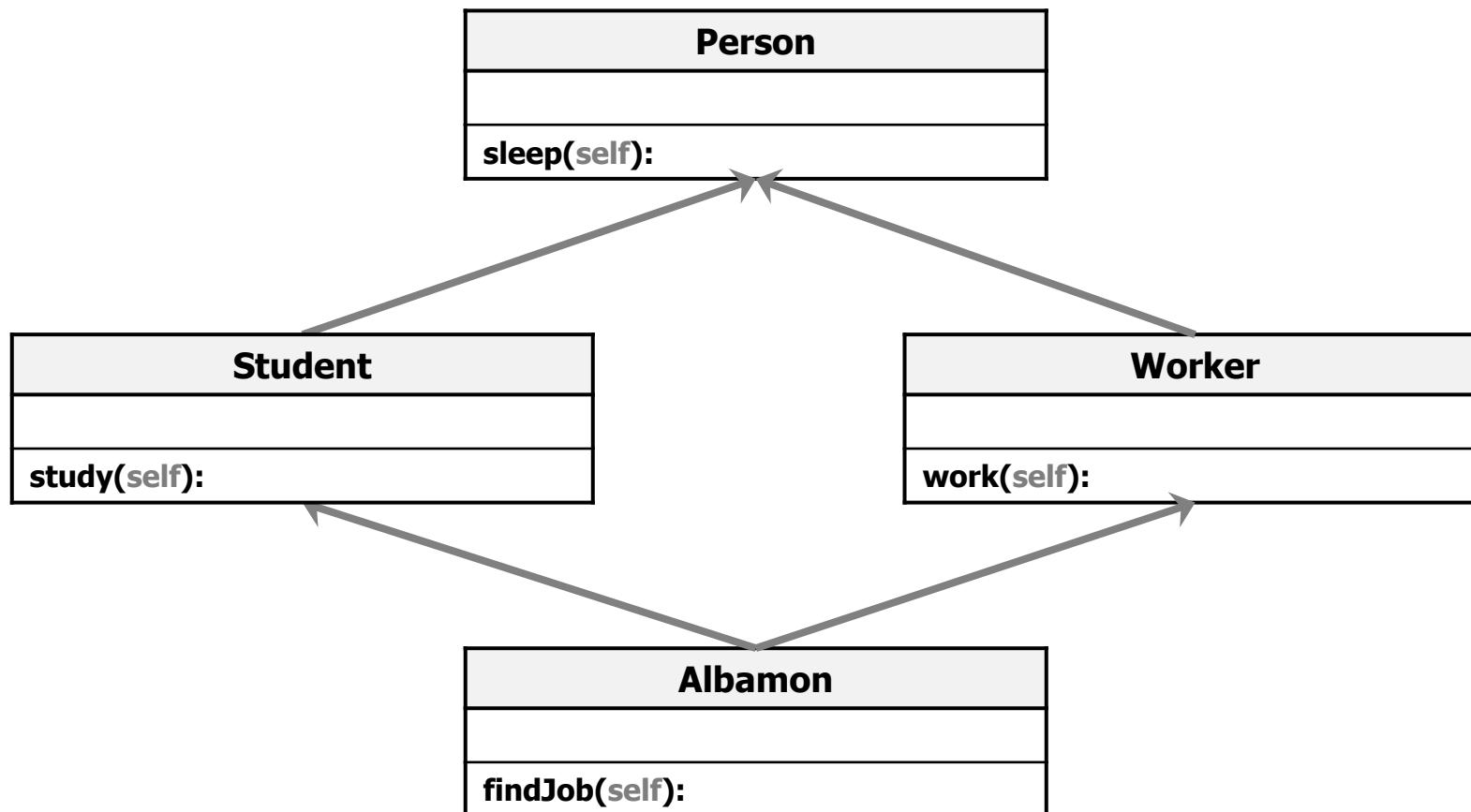
    print(myNewCar.__dict__)
```



# 다중 상속 (1/8)

- **다중 상속**(Multiple Inheritance)

- 2개 이상의 클래스를 상속 받는다.



# 다중 상속 (2/8)

## ● 다중 상속: Albamon

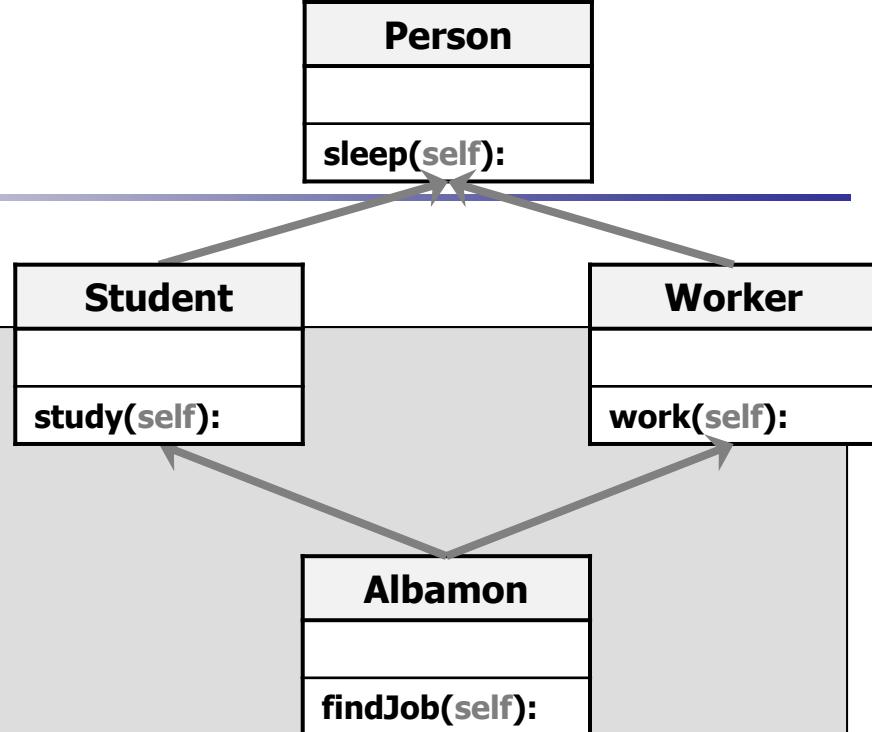
```
# 클래스 정의: Person, Student, Worker
class Person:
    def sleep(self):
        print('sleeping')

class Student(Person):
    def study(self):
        print('studying')

class Worker(Person):
    def work(self):
        print('working')

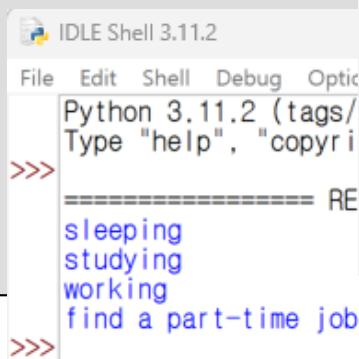
# 다중 상속
class Albamon(Student, Worker):
    def findJob(self):
        print('find a part-time job')

if __name__ == '__main__':
    aAlbamon = Albamon()
    aAlbamon.sleep()
    aAlbamon.study()
    aAlbamon.work()
    aAlbamon.findJob()
```



```
# 클래스의 메소드 결정 순서(MRO)
# print(Albamon.mro())
```

```
# [<class '__main__.Albamon'>,
# <class '__main__.Student'>,
# <class '__main__.Worker'>,
# <class '__main__.Person'>,
# <class 'object'>]
```



# 다중 상속 (3/8)

예제 0-43: 다중 상속 -- 메소드 재정의

(1/2)

# 클래스 정의: Person, Student, Worker

class Person:

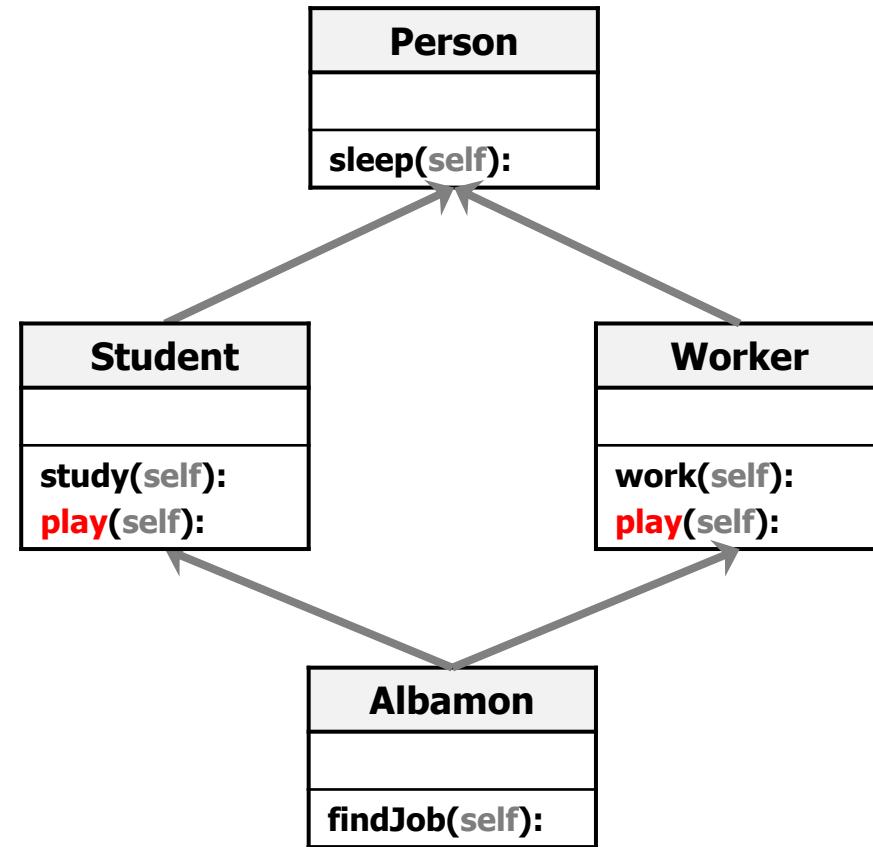
```
def sleep(self):  
    print('sleeping')
```

class Student(Person):

```
def study(self):  
    print('studying')  
  
def play(self):  
    print('play with friends')
```

class Worker(Person):

```
def work(self):  
    print('working')  
  
def play(self):
```



# 다중 상속 (4/8)

예제 0-43: 다중 상속 -- 메소드 재정의

(2/2)

```
# 다중 상속: Albamon
class Albamon(Student, Worker):
    def findJob(self):
        print('find a part-time job')

    if __name__ == '__main__':
        aAlbamon = Albamon()
        aAlbamon.play()

# Student.play(aAlbamon)
# Worker.play(aAlbamon)
```

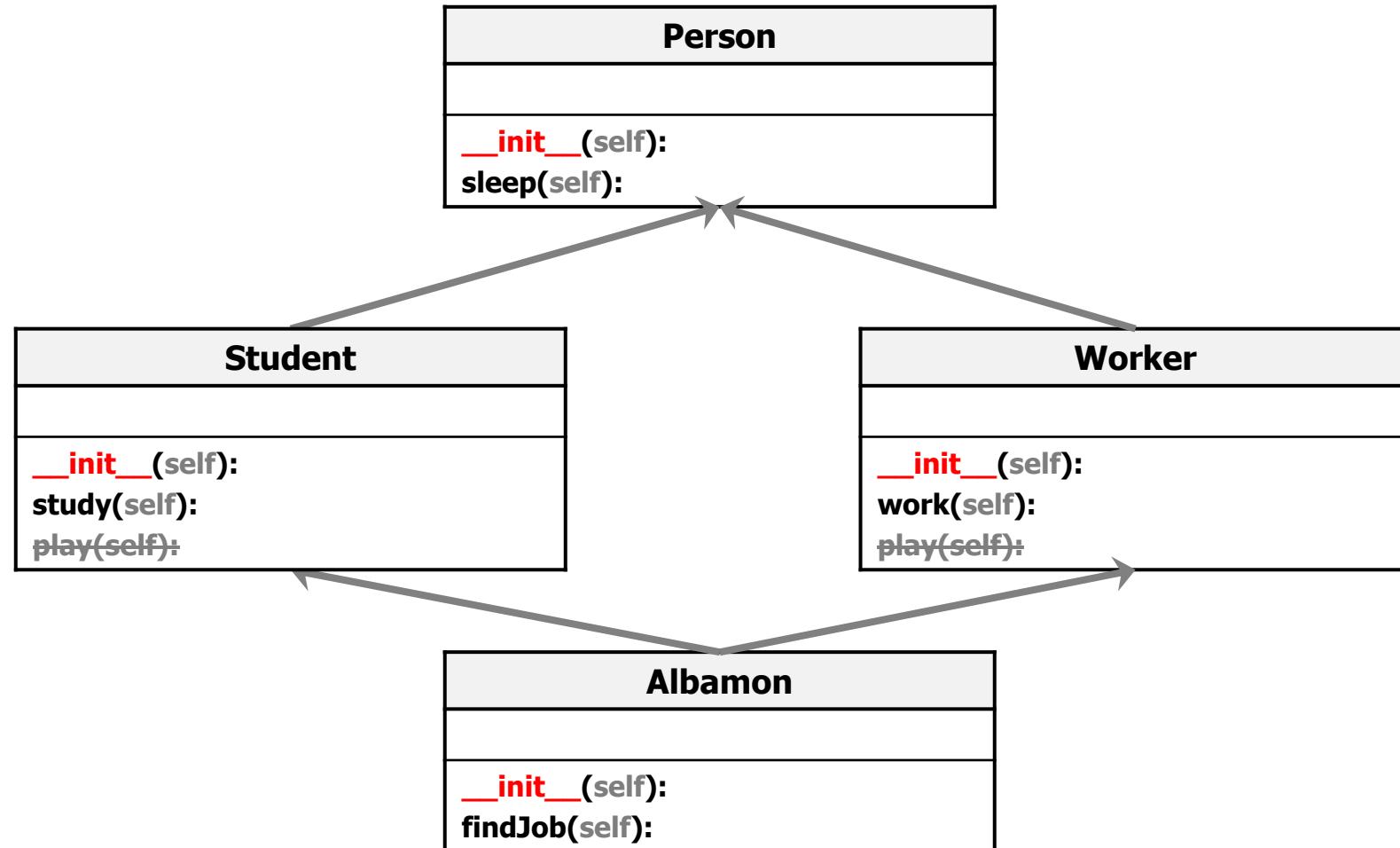
```
# 클래스의 메소드 결정 순서(MRO)
# print(Albamon.mro())

# [<class '__main__.Albamon'>,
# <class '__main__.Student'>,
# <class '__main__.Worker'>,
# <class '__main__.Person'>,
# <class 'object'>]
```



# 다중 상속 (5/8)

- **다중 상속: 다중 상속의 모호성**



# 다중 상속 (6/8)

예제 0-44: 다중 상속 -- 다중 상속의 모호성

#1

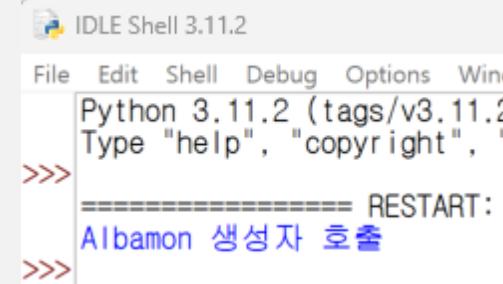
```
# 클래스 정의: Person, Student, Worker
class Person:
    def __init__(self):
        print('Person 생성자 호출!!!!')

class Student(Person):
    def __init__(self):
        print('Student 생성자 호출')

class Worker(Person):
    def __init__(self):
        print('Worker 생성자 호출')

# 다중 상속: Albamon
class Albamon(Student, Worker):
    def __init__(self):
        print('Albamon 생성자 호출')
```

```
if __name__ == '__main__':
    aAlbamon = Albamon()
```



```
# 클래스의 메소드 결정 순서
# print( Albamon.mro() )

# [<class '__main__.Albamon'>,
# <class '__main__.Student'>,
# <class '__main__.Worker'>,
# <class '__main__.Person'>,
# <class 'object'>]
```

# 다중 상속 (7/8)

## 예제 0-44: 다중 상속 -- 다중 상속의 모호성

#2

```
# 클래스 정의: Person, Student, Worker
class Person:
    def __init__(self):
        print('Person 생성자 호출!!!!')

class Student(Person):
    def __init__(self):
        print('Student 생성자 호출')
        Person.__init__(self)

class Worker(Person):
    def __init__(self):
        print('Worker 생성자 호출')
        Person.__init__(self)
```

```
# 다중 상속: Albamon
class Albamon(Student, Worker):
    def __init__(self):
        print('Albamon 생성자 호출')
        Student.__init__(self)
        Worker.__init__(self)
```

```
if __name__ == '__main__':
    aAlbamon = Albamon()
```

```
# 클래스의 메소드 결정 순서
# print( Albamon.mro() )
```

```
# [<class '__main__.Albamon'>,
# <class '__main__.Student'>,
# <class '__main__.Worker'>,
# <class '__main__.Person'>,
# <class 'object'>]
```

The screenshot shows the Python IDLE Shell interface. The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The shell window displays the following text:  
Python 3.11.2 (tags/v3.11.2) Type "help", "copyright", "credits" or "license" for more information  
==== RESTART: Albamon 생성자 호출  
Student 생성자 호출  
Person 생성자 호출!!!  
Worker 생성자 호출  
Person 생성자 호출!!!

# 다중 상속 (8/8)

## 예제 0-44: 다중 상속 -- 다중 상속의 모호성 #3

```
# 클래스 정의: Person, Student, Worker
class Person:
    def __init__(self):
        print('Person 생성자 호출!!!!')

class Student(Person):
    def __init__(self):
        print('Student 생성자 호출')
        # Person.__init__(self)
        super().__init__()

class Worker(Person):
    def __init__(self):
        print('Worker 생성자 호출')
        # Person.__init__(self)
        super().__init__()
```

```
# 다중 상속: Albamon
class Albamon(Student, Worker):
    def __init__(self):
        print('Albamon 생성자 호출')
        # Student.__init__(self)
        # Worker.__init__(self)
        super().__init__()

if __name__ == '__main__':
    aAlbamon = Albamon()

# 클래스의 메소드 결정 순서인
# print( Albamon.mro() )

# [<class '__main__.Albamon'>,
# <class '__main__.Student'>,
# <class '__main__.Worker'>,
# <class '__main__.Person'>,
# <class 'object'>]
```

The screenshot shows the Python IDLE Shell interface. The menu bar includes File, Edit, Shell, Debug, Options, Window, Help, and Restart. The shell window displays the code execution and its output:

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window
Python 3.11.2 (tags/v3.11.2b2, Feb 24 2023, 15:26:46)
Type "help", "copyright", "credits" or "license" for more information
>>> ===== RESTART: C:\Users\Clickseo\PycharmProjects\Python\Ch08\Ex044.py =====
Albamon 생성자 호출
Student 생성자 호출
Worker 생성자 호출
Person 생성자 호출!!!
```



# 객체지향 프로그래밍

모듈과 패키지



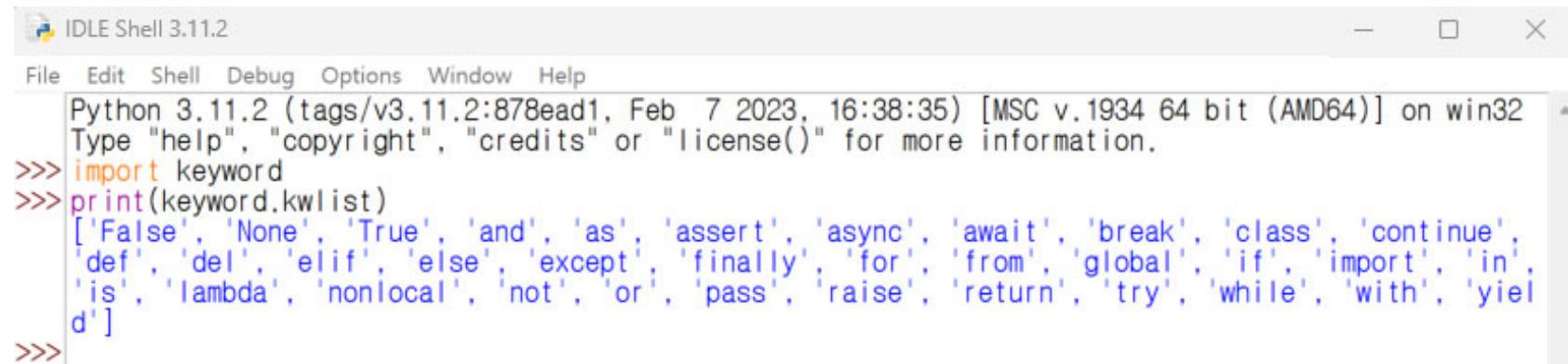
# 모듈의 이해 (2/6)

## ● 모듈: 예약어

### ○ import keyword

- `keyword.kwlist` : 파일 예약어(Keyword)

```
# 파일 예약어 전체 목록 확인  
import keyword  
print(keyword.kwlist)
```



The screenshot shows the Python 3.11.2 IDLE shell interface. The window title is "IDLE Shell 3.11.2". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main console area displays the following output:

```
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> import keyword
>>> print(keyword.kwlist)
['False', 'None', 'True', 'and', 'as', 'assert', 'async', 'await', 'break', 'class', 'continue',
'def', 'del', 'elif', 'else', 'except', 'finally', 'for', 'from', 'global', 'if', 'import', 'in',
'is', 'lambda', 'nonlocal', 'not', 'or', 'pass', 'raise', 'return', 'try', 'while', 'with', 'yield']
```

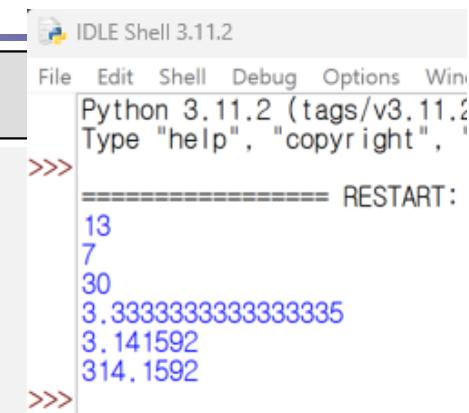
# 모듈의 이해 (3/6)

## 예제 0-45: 모듈 만들기와 불러오기

#1

**# cal.py****# 변수: 원주율(PI)**

PI = 3.141592

**# 클래스: 원의 넓이****class CircleArea:** **def getCircleArea(self, r):**  
 **return PI \* (r \*\* 2)****# 함수: add, sub, mul, div****def add(a, b): return a + b**  
**def sub(a, b): return a - b**  
**def mul(a, b): return a \* b**  
**def div(a, b): return a / b****# index.py****# 모듈 불러오기****import cal****print(cal.add(10, 3))****print(cal.sub(10, 3))****print(cal.mul(10, 3))****print(cal.div(10, 3))****print(cal.PI)****ca = cal.CircleArea()****print(ca.getCircleArea(10))**

IDLE Shell 3.11.2  
File Edit Shell Debug Options Window  
Python 3.11.2 (tags/v3.11.2b2-144-g84c8c1c65, Feb 13 2023, 14:53:49)  
Type "help", "copyright", "credits" or "license" for more information  
==== RESTART: C:\Users\user\PycharmProjects\pythonProject\cal.py ====  
>>> 13  
>>> 7  
>>> 30  
>>> 3.3333333333333335  
>>> 3.141592  
>>> 314.1592  
>>>

# 모듈의 이해 (4/6)

예제 0-45: 모듈 만들기와 불러오기 -- 모듈명 생략

#2

# cal.py

# 변수: 원주율(PI)

PI = 3.141592

# 클래스: 원의 넓이

class CircleArea:

```
    def getCircleArea(self, r):
        return PI * (r ** 2)
```

# 함수: add, sub, mul, div

```
    def add(a, b):    return a + b
    def sub(a, b):    return a - b
    def mul(a, b):    return a * b
    def div(a, b):    return a / b
```

# index.py

# 모듈 일부만 가져오기

# from cal import add, sub, mul, div

# 모듈 전체 가져오기

from cal import \*

```
print(add(10, 3))
```

```
print(sub(10, 3))
```

```
print(mul(10, 3))
```

```
print(div(10, 3))
```

```
print(PI)
```

```
ca = CircleArea()
```

```
print(ca.getCircleArea(10))
```

The screenshot shows the Python 3.11.2 shell interface. The code is being run in a file named 'cal.py'. The output shows the results of the arithmetic operations and the value of PI.

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window
Python 3.11.2 (tags/v3.11.2b2-759d5b1, Feb 28 2023, 16:40:45)
Type "help", "copyright", "credits" or "license" for more information
>>> ===== RESTART: <<<cal.py>>>
13
7
30
3.3333333333333335
3.141592
314.1592
>>>
```

# 모듈의 이해 (5/6)

## 예제 0-45: 모듈 만들기와 불러오기 -- 내장 변수( `__name__` ) #3

# cal.py

```
# 변수: 원주율(PI)
PI = 3.141592

# 클래스: 원의 넓이
class CircleArea:
    def getCircleArea(self, r):
        return PI * (r ** 2)

# 함수: add, sub, mul, div
def add(a, b):    return a + b
def sub(a, b):    return a - b
def mul(a, b):    return a * b
def div(a, b):    return a / b

print(f'cal.py: {__name__}')

print(add(1, 2))
print(sub(1, 2))
print(mul(1, 2))
print(div(1, 2))
```

# index.py

```
# 모듈 일부만 가져오기
# from cal import add, sub, mul, div

# 모듈 전체 가져오기
from cal import *

print(f'index.py: {__name__}')
```

```
print(add(10, 3))
print(sub(10, 3))
print(mul(10, 3))
print(div(10, 3))
```

```
print(PI)
```

```
ca = CircleArea()
print(ca.getCircleArea(10))
```

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Win
Python 3.11.2 (tags/v3.11.2b1-794d5b1, Feb 28 2023, 16:40:45)
Type "help", "copyright", "credits" or "license" for more information
>>> RESTART: cal.py
cal.py: cal
3
-1
2
0.5
>>> RESTART: index.py
index.py: __main__
13
7
30
3.3333333333333335
3.141592
314.1592
>>>
```

# 모듈의 이해 (6/6)

## 예제 0-45: 모듈 만들기와 불러오기 -- 내장 변수( `__name__` ) #4

# cal.py

# 변수: 원주율(PI)

PI = 3.141592

# 클래스: 원의 넓이

class CircleArea:

```
    def getCircleArea(self, r):
        return PI * (r ** 2)
```

# 함수: add, sub, mul, div

```
def add(a, b):    return a + b
def sub(a, b):    return a - b
def mul(a, b):    return a * b
def div(a, b):    return a / b
```

print(f'cal.py: {\_\_name\_\_}')

if \_\_name\_\_ == '\_\_main\_\_':

```
    print(add(1, 2))
    print(sub(1, 2))
    print(mul(1, 2))
    print(div(1, 2))
```

# index.py

# 모듈 일부만 가져오기

# from cal import add, sub, mul, div

# 모듈 전체 가져오기

from cal import \*

print(f'index.py: {\_\_name\_\_}')

if \_\_name\_\_ == '\_\_main\_\_':

```
    print(add(10, 3))
    print(sub(10, 3))
    print(mul(10, 3))
    print(div(10, 3))
```

print(PI)

ca = CircleArea()

print(ca.getCircleArea(10))

The screenshot shows two IDLE shells. The top shell runs the code in index.py, displaying the output of the imported functions and the value of PI. The bottom shell runs the code in cal.py, demonstrating the module's functionality.

```
IDLE Shell 3.11.2
File Edit Shell Debug Options Window
Python 3.11.2 (tags/v3.11.2b1-11815, Feb 28 2023, 16:40:45)
Type "help", "copyright", "credits" or "license" for more information
>>> ===== RESTART: cal.py =====
index.py: __main__
13
7
30
3.141592
314.1592

IDLE Shell 3.11.2
File Edit Shell Debug Options Window
Python 3.11.2 (tags/v3.11.2b1-11815, Feb 28 2023, 16:40:45)
Type "help", "copyright", "credits" or "license" for more information
>>> ===== RESTART: index.py =====
cal.py: __main__
3
-1
2
0.5
```

# 패키지의 이해 (1/2)

## ● 패키지(Package)

- 특정 기능과 관련된 여러 모듈을 묶어 놓은 것.

```
# 패키지 가져오기: 여러 개의 패키지를 가져올 때는 콤마(,)로 구분
```

```
import packageName.moduleName
```

```
packageName.moduleName. 변수
```

```
packageName.moduleName. 함수 ()
```

```
packageName.moduleName. 클래스 ()
```

```
# 패키지 이름 지정
```

```
import packageName.moduleName as Alias
```

```
Alias. 변수
```

```
Alias. 함수 ()
```

```
Alias. 클래스 ()
```

```
# 패키지 일부만 가져오기
```

```
from packageName.moduleName import 변수, 함수, 클래스
```

```
from packageName.moduleName import 변수(함수, 클래스) as Alias
```

# 패키지의 이해 (2/2)

## ● 패키지: 패키지 설계

- 음향 파일과 음향 데이터의 일관된 처리를 위한 모듈들의 집합

```
sound/                               Top-level package
    __init__.py                      Initialize the sound package
    formats/                           Subpackage for file format conversions
        __init__.py
        wavread.py
        wavwrite.py
        aiffread.py
        aiffwrite.py
        auread.py
        auwrite.py
        ...
    effects/                           Subpackage for sound effects
        __init__.py
        echo.py
        surround.py
        reverse.py
        ...
    filters/                           Subpackage for filters
        __init__.py
        equalizer.py
        vocoder.py
        karaoke.py
        ...
```

[ 출처: "Python 3.13.2 documentation", <https://docs.python.org/3/tutorial/modules.html>, 2025. ]

# 파이썬 패키지 관리 (1/2)

## ● pip (패키지 관리자)

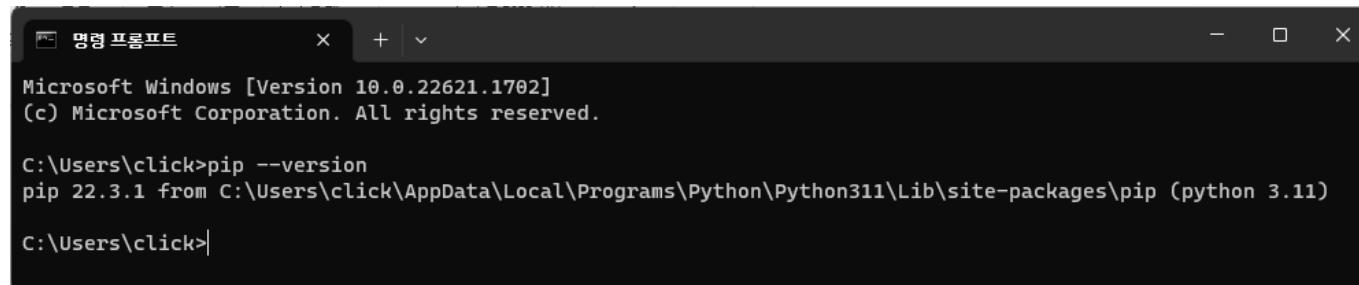
### ○ 파이썬으로 작성된 패키지 관리 시스템

- 소프트웨어 설치 및 관리를 위해 이용한다.
- 대부분의 파이썬 배포판에는 **pip**가 사전 설치된 상태로 제공된다.
  - Python 3.4 이상에는 기본적으로 pip(Python 3의 경우 pip3)가 포함된다.
  - Python 2.7.9 이상(python2 시리즈)

### ○ Command-line interface(CLI)

```
pip <command> [options]
```

- 패키지 설치 및 삭제
  - **pip install** package-name
  - **pip uninstall** package-name



```
명령 프롬프트
Microsoft Windows [Version 10.0.22621.1702]
(c) Microsoft Corporation. All rights reserved.

C:\Users\click>pip --version
pip 22.3.1 from C:\Users\click\AppData\Local\Programs\Python\Python311\Lib\site-packages\pip (python 3.11)

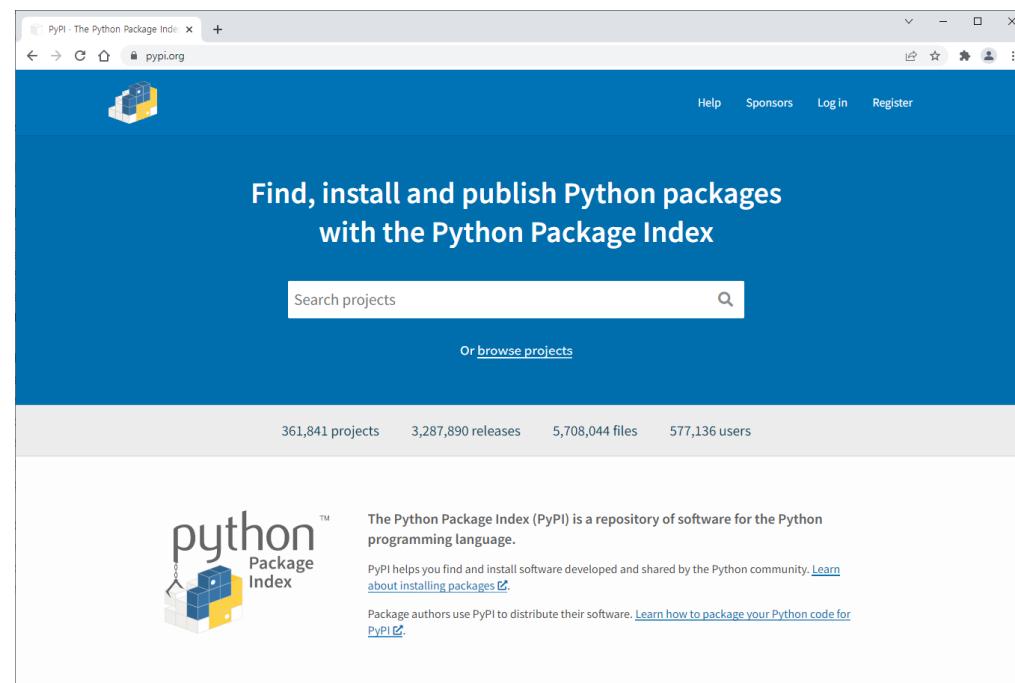
C:\Users\click>
```

# 파이썬 패키지 관리 (2/2)

## ● Python Package Index(PyPI): pypi.org

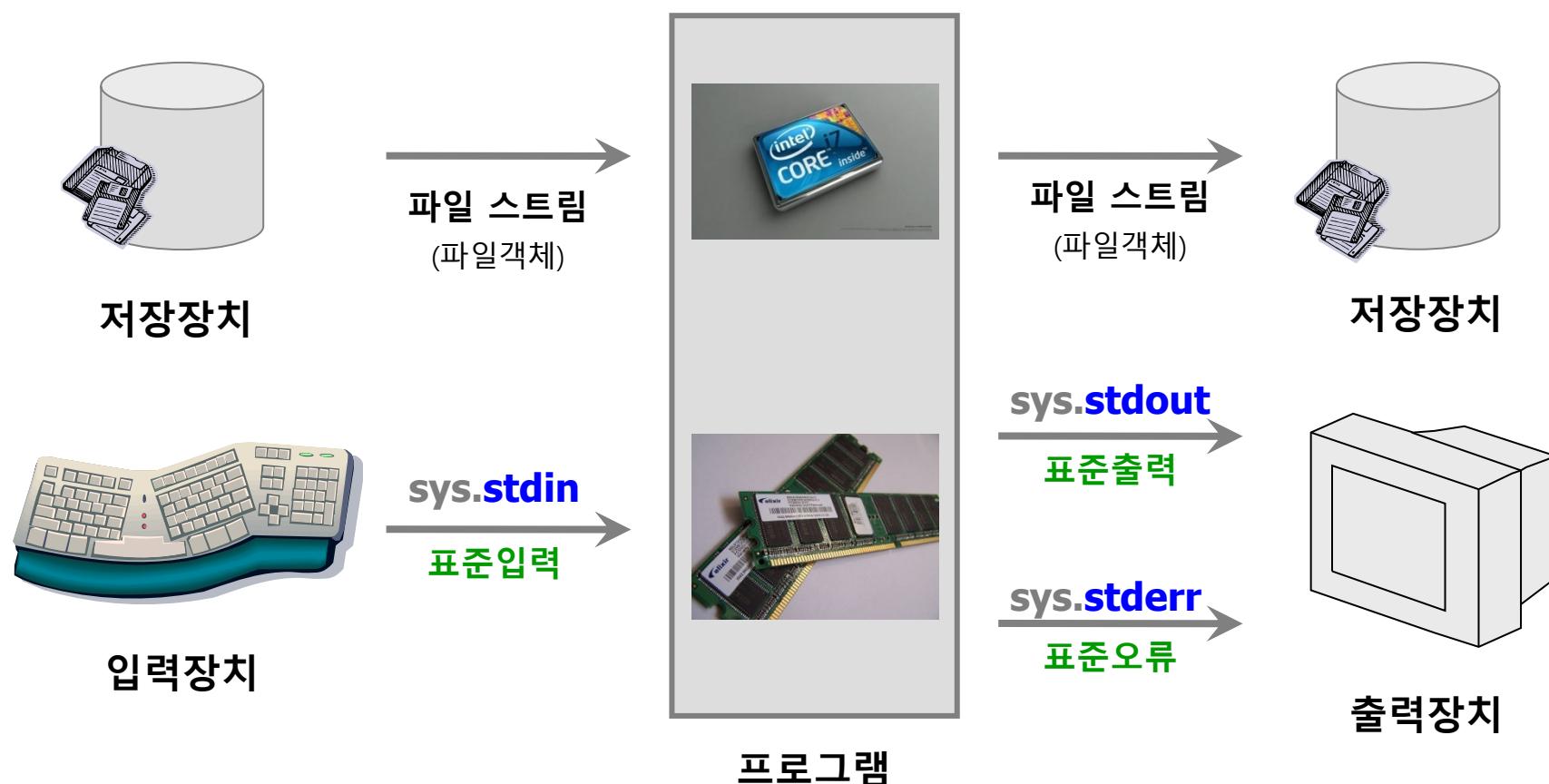
### ○ 파이썬 언어를 위한 소프트웨어 저장소

- Python Software Foundation(PSF)에서 운영한다.
- 파이썬 커뮤니티에서 개발 및 공유하는 소프트웨어를 찾고 설치하는데 유용하다.
  - 또한 패키지 작성자는 PyPI를 사용하여 소프트웨어를 배포할 수 있다.



# 파일 입출력: 입출력 스트림 (1/4)

## ● 표준 입출력과 파일 입출력



# 파일 입출력: open, close (1/4)

## ● 파일 개방: open

○ **open** 내장 함수: 외부 파일과 프로그램 사이에 연결을 만든다.

- 파일 객체를 생성하여 파일 처리에 필요한 정보를 저장한다.

```
open(fileName, mode)
```

- **fileName**: 대상 파일의 경로를 포함한 이름 지정
- **mode**: 파일 개방 모드 설정(r, w, a 등)
- **encoding**: 'UTF8'

## ● 파일 닫기: close

○ **close** 조작 함수: 버퍼 공간과 같은 파일 관련 시스템 자원을 반납한다.

```
fileObject.close()
```

# 파일 입출력: open, close (2/4)

## ● 파일 처리: open / close

# 1) 파일 개방

```
outFile = open('data.txt', 'w')
```

# 2) 파일 입출력 함수: 대상 파일에 대한 입출력 작업 수행

```
print('Hello World!!!', file=outFile)      # 한 줄의 문자열을 파일에 기록
```

text = "'산은 산이요. 물은 물이로다.

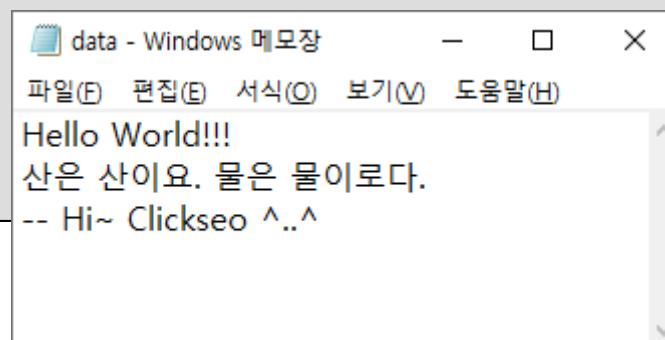
-- Hi~ Clickseo ^..^

'''

```
outFile.write(text)
```

# 3) 파일 닫기

```
outFile.close()
```



# 파일 입출력: open, close (3/4)

## ● 파일 개방: 텍스트 모드

### ○ open 내장 함수의 파일 개방: 텍스트 모드(Text Mode)

모드	의 미	비고
r	<b>읽기 전용(Read Only)</b> 지정된 파일이 존재하지 않으면, <b>FileNotFoundException</b> 발생.	<b>쓰기 불가</b>
w	<b>쓰기 전용(Write Only)</b> 파일이 존재할 경우: 기존 파일의 데이터 삭제 후, 빈 파일을 만든다. 파일이 존재하지 않을 경우: 새로운 파일 생성	<b>읽기 불가</b>
x	<b>쓰기 전용(Write Only)</b> 파일이 존재할 경우: <b>FileExistsError</b> 발생. 파일이 존재하지 않을 경우: 새로운 파일 생성	<b>읽기 불가</b>
a	<b>추가 전용(Append Only)</b> 파일이 존재할 경우: 기존 파일의 맨 끝에서부터 추가한다. 파일이 존재하지 않을 경우: 새로운 파일 생성	<b>읽기 불가</b>
r+	지정된 파일이 존재할 경우, 읽기와 쓰기가 가능하도록 파일을 개방한다.	<b>읽기 + 쓰기</b>
w+	지정된 파일에 대하여 읽기와 쓰기가 모드 가능하도록 파일을 개방한다.	<b>읽기 + 쓰기</b>
a+	지정된 파일에 대하여 읽기와 쓰기가 모드 가능하도록 파일을 개방한다. 단, 지정된 파일의 맨 끝에서부터 읽기와 쓰기가 가능하다.	<b>읽기 + 쓰기</b> <b>이전 부분은 쓰기 불가</b>

# 파일 입출력: open, close (4/4)

## ● 파일 개방: 이진 모드

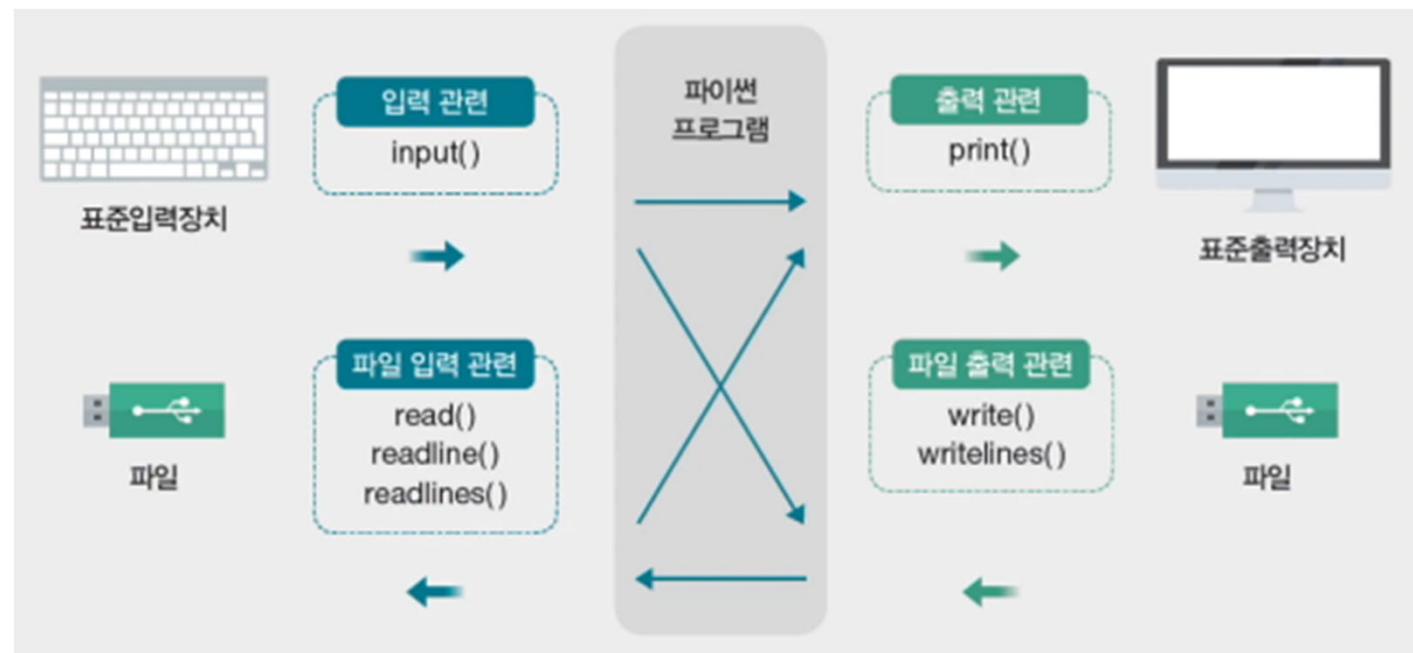
### ○ open 함수의 파일 개방: 이진 모드(Binary Mode)

모 드	의 미
<b>rb</b>	
<b>wb</b>	이진 모드로 파일을 개방하고, 텍스트 모드에서 r, w, a 와 같은 의미
<b>ab</b>	
<b>rb+ / r+b</b>	
<b>wb+ / w+b</b>	이진 모드로 파일을 개방하고, 텍스트 모드에서 r+, w+, a+ 와 동일한 의미
<b>ab+ / a+b</b>	

# 파일 입출력: 파일 입출력 함수 (1/3)

## ● 파일 입출력 함수

### ○ 텍스트 파일과 이진 파일



# 파일 입출력: 파일 입출력 함수 (2/3)

## ● 파일 쓰기: print, write

○ **print** 내장 함수: 표준 출력에 해당 문자열 객체를 출력(기록)한다.

- 파일 객체를 생성하여 파일 처리에 필요한 정보를 저장한다.

```
print('str', sep=' ', end='\\n ', file=None, flush=False)
```

- **sep**: 구분자(separator). 기본값(또는 **None**)은 공백(' ')이다.
- **end**: 마지막 문자열(end string). 기본값(또는 **None**)은 줄 바꿈('\\n')이다.
- **file**: 해당 문자열 객체(str)를 출력할 스트림을 지정한다.
  - » 만약, 생략(또는 **None**)하면 기본 값인 표준 출력(`sys.stdout`)에 내용을 기록한다.

○ **write** 조작 함수

- print** 내장 함수처럼 공백이나 줄 바꿈을 자동으로 추가하지 않는다.
- 파일에 기록한 **bytes**를 반환한다.

```
fileObject.write()
```

# 파일 입출력: 파일 입출력 함수 (3/3)

## ● 파일 읽기: `read`, `readline`, `readlines`

### ○ `read` 조작 함수: 전체 파일 내용을 읽는다.

- 파일에 끝에 도달했을 때, `read` 함수는 빈 문자열("")을 반환한다.
  - 단, 대형 파일로 이 작업을 수행할 때 많은 메모리가 소비되므로 주의해야 한다.

```
fileObject.read()
```

```
fileObject.read(maxLength)
```

### ○ `readline` 조작 함수: 파일을 줄 단위로 읽는다.

### ○ `readlines` 조작 함수

- 전체 파일을 내용을 읽은 후, 각 행을 리스트로 반환한다.

# 파일 입출력: 텍스트 파일 (1/2)

예제 0-46: 텍스트 파일 읽기 -- `read`, `readline` 조작 함수

# 1) 파일 개방

```
inFile = open('data.txt', 'r')
```

# 2) 파일 처리: 파일에서 데이터 읽기

```
# text = inFile.read()      # 파일에서 전체 내용을 읽어 온다.
```

```
text = "
```

```
while True:
```

# 파일에서 최대 10bytes

```
line = inFile.read(10)
```

```
if not line:
```

```
    break
```

```
text += line
```

```
print(f'파일 크기: {len(text)} bytes')
```

```
print(text)
```

# 3) 파일 닫기

```
inFile.close()
```

# 파일에서 줄 단위로 읽어 온다.

```
line = inFile.readline()
```

IDLE Shell 3.11.2

File Edit Shell Debug Options Window Help

Python 3.11.2 (tags/v3.11.2:878ead1  
Type "help", "copyright", "credits"

>>>

===== RESTART: C:\Users\

파일 크기: 53 bytes

Hello World!!!

산은 산이요, 물은 물이로다.

-- Hi~ Clickseo ^..^

>>>

# 파일 입출력: 텍스트 파일 (2/2)

## 예제 0-47: 텍스트 파일 읽기 -- `readlines` 조작 함수

# 1) 파일 개방

```
inFile = open('data.txt', 'r')
```

# 2) 파일 처리: 파일에서 데이터 읽기

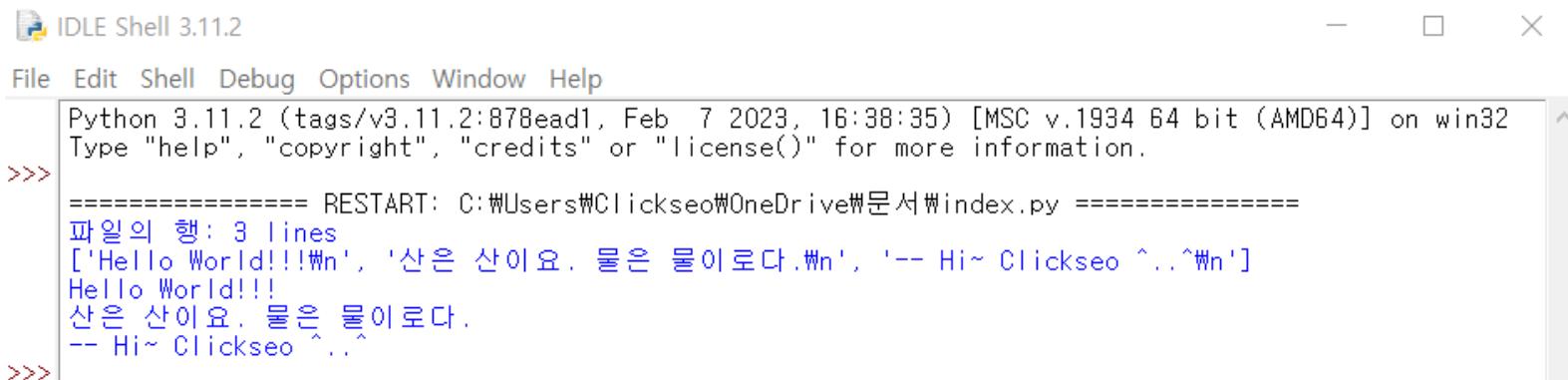
```
lines = inFile.readlines()          # 파일에서 전체 내용을 읽은 후, 각 행을 리스트로 반환
```

```
print(f'파일의 행: {len(lines)} lines')
print(lines)
```

```
for line in lines :
    print(line, end='')
```

# 3) 파일 닫기

```
inFile.close()
```



The screenshot shows the Python IDLE Shell interface. The title bar says "IDLE Shell 3.11.2". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main window displays the following output:

```
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> ===== RESTART: C:\Users\Clickseo\OneDrive\문서\index.py =====
파일의 행: 3 lines
['Hello World!!!\n', '산은 산이요. 물은 물이로다.\n', '-- Hi~ Clickseo ^..^\n']
Hello World!!!
산은 산이요. 물은 물이로다.
-- Hi~ Clickseo ^..^
>>>
```

# 파일 입출력: 이진 파일

## 예제 0-48: 이진 파일 입출력 -- read, write 조작 함수

### # 1) 파일 개방

```
ioFile = open('data.bin', 'wb+')
```

### # 2) 파일 처리: 파일에 데이터 기록(bytes)

```
byteData = bytes(range(0,256))
bytes = ioFile.write(byteData)
print(f'파일에 {bytes} bytes를 기록')
```

### # 파일 객체에서 파일의 작업 위치 값을

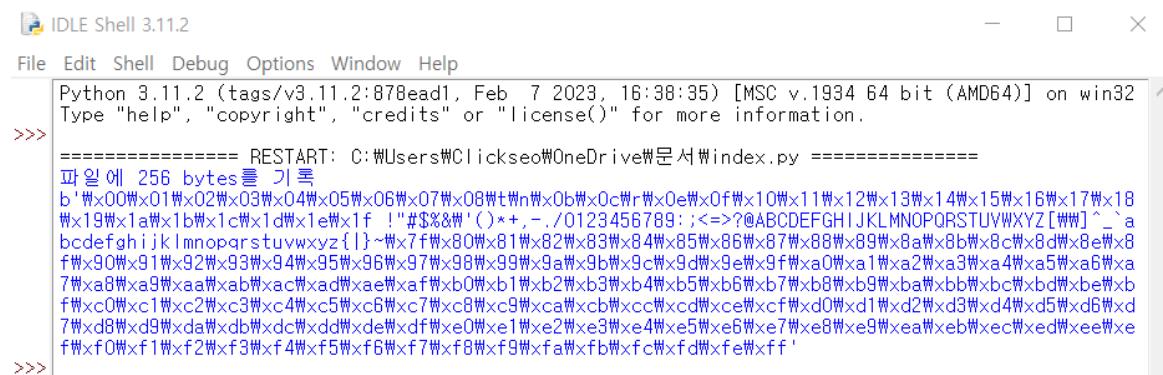
```
ioFile.seek(0,0) # 파일의 맨 앞으로...
```

```
bytes = ioFile.read()
```

```
print(bytes)
```

### # 3) 파일 닫기

```
ioFile.close()
```



# 파일 입출력: 순차 접근과 임의 접근

## ● 랜덤 액세스 함수: seek, tell 조작 함수

- **seek** 조작 함수: 파일 객체의 파일 위치를 임의의 위치로 이동한다.
  - 원하는 자료에 대한 직접 접근(direct access)이 가능하다.

**fileObject.seek(offset, origin)**

- **offset**: 기준점부터 새로운 위치까지 상대적으로 떨어진 거리(**bytes 단위**)
- **origin**: 파일 이동을 위한 기준점

기준점	값	의 미
SEEK_SET	0	파일의 시작 위치를 기준으로 파일의 위치 값을 이동한다( <b>기본값</b> ).
SEEK_CUR	1	현재의 파일의 위치를 기준으로 다음 위치로 이동한다.
SEEK_END	2	파일의 마지막 위치를 기준으로 파일의 위치를 이동한다.

- **tell** 조작 함수: 파일 시작 위치에서 현재 오프셋을 bytes 단위로 반환한다.

**bytes = fileObject.tell()**

# 파이썬 라이브러리



- 파이썬 프로그래밍 기초
- 자료형과 자료 구조
- 객체지향 프로그래밍
- 파이썬 라이브러리

백문이 불여일타(百聞而不如一打)

- 표준 라이브러리
- 외부 라이브러리





# 표준 라이브러리

GUI 프로그래밍

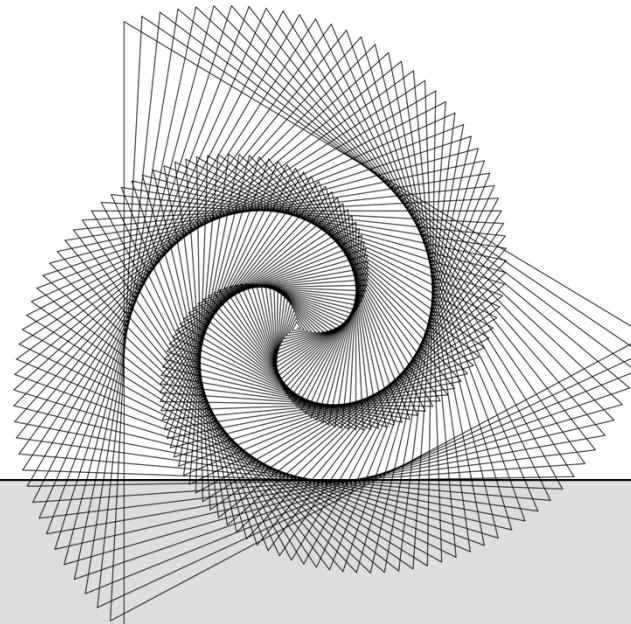
: 터틀 그래픽, Tkinter



# 터틀 그래픽

## ● 터틀 그래픽(Turtle Graphics)

- Turtle Graphics은 간단한 움직임을 반복하는 프로그램을 사용하여 복잡한 모양을 그릴 수 있다.
  - 파이썬 표준 라이브러리에는 **Turtle Graphics** 모듈이 포함되어 있다.



```
# 터틀 그래픽 모듈
```

```
import turtle as Alias  
from turtle import *
```

# 터틀 그래픽: 공개 클래스

## ● 터틀 그래픽: 공개 클래스

### ○ 공개 클래스(Public classes)

# 거북이를 만든다.

- class turtle.RawTurtle(canvas)
- class turtle.RawPen(canvas)
  - canvas – tkinter.Canvas, ScrolledCanvas 또는 TurtleScreen

# bgcolor 등과 같은 화면 지향 메서드를 제공한다.

- class turtle.TurtleScreen(cv) # cv - tkinter.Canvas
- class turtle.Screen # TurtleScreen의 서브 클래스

# 자동 생성된 기본 Screen 객체에 그린다.

- class turtle.Turtle # RawTurtle 서브 클래스
- class turtle.ScrolledCanvas(master)
  - master – ScrolledCanvas
  - 스크롤 막대가 추가된 Tkinter-Canvas를 담을 어떤 Tkinter 위젯
- class turtle.Shape(type\_, data)

# 2차원 벡터 클래스

- class turtle.Vec2D(x, y)

# 터틀 그래픽: 거북이 움직임 (1/7)

## ● 터틀 그래픽: 조작 함수

### ○ 거북이 움직임: 이동 및 그리기 #1

# 지정된 거리(distance)만큼 거북이를 앞으로 움직인다.

- turtle.**forward**(distance)
- turtle.**fd**(distance)

# 지정된 거리(distance)만큼 거북이를 뒤로 움직인다.

- turtle.**back**(distance)
- turtle.**bk**(distance)
- turtle.**backward**(distance)

# 지정된 각도(angle) 단위만큼 거북이를 오른쪽으로 회전한다.

- turtle.**right**(angle)
- turtle.**rt**(angle)

# 지정된 각도(angle) 단위만큼 거북이를 왼쪽으로 회전한다.

- turtle.**left**(angle)
- turtle.**lt**(angle)

```
import turtle
```

```
# 거북이 객체 생성
```

```
t = turtle.Turtle()
```

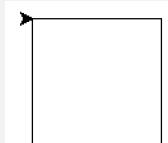
```
# 사각형 그리기
```

```
for _ in range(4):
```

```
    t.forward(100)
```

```
    t.right(90)
```

```
turtle.mainloop()
```



# 터틀 그래픽: 거북이 움직임 (2/7)

## ● 터틀 그래픽: 조작 함수

### ○ 거북이 움직임: 이동 및 그리기 #2

# 거북이를 원점(0,0)으로 이동하고 방향을 시작 방향으로 설정한다.

- turtle.home()

# 거북이를 절대 위치로 움직인다(펜이 내려져 있으면 선을 그린다).

- turtle.goto(x, y=None)
- turtle.setpos(x, y=None)
- turtle.setposition(x, y=None)

# 거북이의 좌표 값을 지정된 값으로 각각 설정한다.

- turtle.setx(x)
- turtle.sety(y)

# 거북이의 방향을 각도(to\_angle)로 설정한다.

- turtle.setheading(to\_angle)
- turtle.seth(to\_angle)

```
import turtle
```

```
t = turtle.Turtle()
```

```
# t.home()
```

```
# t.goto(0, 0)
```

```
# 사각형 그리기
```

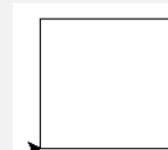
```
t.goto(0, 100)
```

```
t.goto(100, 100)
```

```
t.goto(100, 0)
```

```
t.goto(0, 0)
```

```
turtle.mainloop()
```



# 터틀 그래픽: 거북이 움직임 (3/7)

## ● 터틀 그래픽: 조작 함수

### ○ 거북이 움직임: 이동 및 그리기 #3

# 주어진 반지름(radius)으로 원을 그린다.

- turtle.circle(radius, extent=None, steps=None)

# 색깔(color)을 사용하여 지정된 크기(size)의 원형 점을 그린다.

- turtle.dot(size=None, \*color)

# 거북이 모양의 사본을 현재 거북이 위치에서 캔버스에 찍는다.

- turtle.stamp()

- turtle.clearstamp(stampid)

- turtle.clearstamps(n=None)

# 마지막 거북이의 행동을 (반복적으로) 되돌린다.

- turtle.undo()

# 속도를 지정된(0, 1 ... 10) 범위의 값으로 설정한다.

- turtle.speed(speed=None)

import turtle

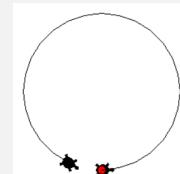
```
t = turtle.Turtle()  
t.shape('turtle')  
t.speed(1)
```

# 원 그리기

```
t.stamp()  
t.dot(10, 'red')  
t.circle(100)
```

t.undo()

turtle.mainloop()



# 터틀 그래픽: 거북이 움직임 (4/7)

## 예제 0-49: 터틀 그래픽 -- 재귀 함수

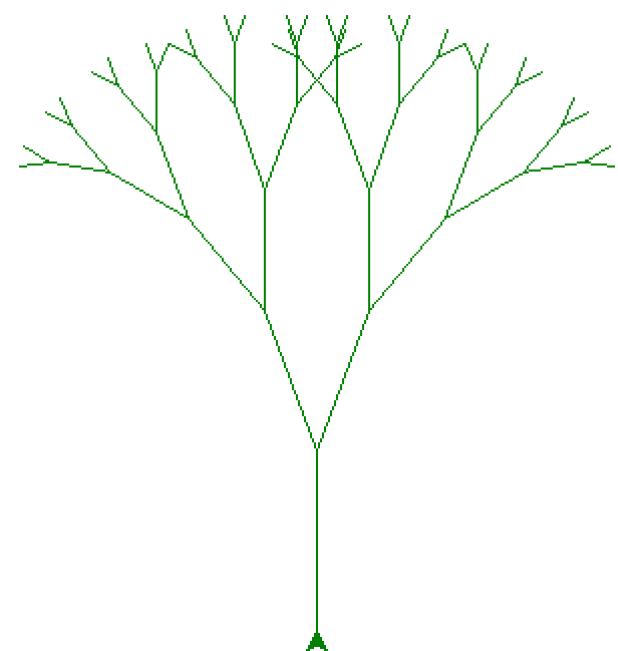
```
import turtle

# 재귀 함수
def tree(length):
    if length < 5 :
        return
    t.forward(length);           # 재귀함수 탈출 조건
    t.right(20)
    tree(length-15);
    t.left(40)
    tree(length-15)
    t.right(20)

# Pen: RawTurtle 자동 생성(스크롤) 캔버스
t = turtle.Pen()               # t = turtle.Turtle()
t.left(90)
t.color('green')
t.speed(1)

tree(90)

turtle.mainloop()
```



# 터틀 그래픽: 거북이 움직임 (5/7)

---

## ● 터틀 그래픽: 조작 함수

### ○ 거북이 상태: 가시성

# 거북이를 보이지 않게 한다.

- `turtle.hideturtle()`
- `turtle.ht()`

# 거북이를 보이게 한다.

- `turtle.showturtle()`
- `turtle.st()`

# 거북이가 보이면 True를, 숨겨져 있으면 False를 반환한다.

- `turtle.isvisible()`

# 터틀 그래픽: 거북이 움직임 (6/7)

---

## ● 터틀 그래픽: 조작 함수

### ○ 거북이 상태: 방향 및 위치

# 거북이의 현재 방향을 반환한다.

- `turtle.heading()`

# 거북이의 현재 위치(x, y)를 반환한다.

- `turtle.position()`
- `turtle.pos()`

# 거북이의 x, y 좌표 값을 각각 반환한다.

- `turtle.xcor()`
- `turtle.ycor()`

# 거북이 위치에서 (x, y), 벡터 또는 다른 거북이로 지정된 위치의 선과 각도 그리고 거리를 단위로 반환한다.

- `turtle.towards(x, y=None)`
- `turtle.distance(x, y=None)`

# 터틀 그래픽: 거북이 움직임 (7/7)

- 터틀 그래픽: 조작 함수

- 측정 설정: 방향 및 위치

- # 각도 측정 단위를 설정한다(즉, 전체 원에 대한 “도(degrees)”의 수를 설정).

- `turtle.degrees(fullcircle=360.0)`

- # 각도 측정 단위를 라디안으로 설정한다.

- `turtle.radians()`



# 터틀 그래픽: 펜 제어

예제 0-50: 터틀 그래픽 .. 막대 그래프 그리기

```
import turtle

def drawBar(height):
    t.begin_fill()
    t.left(90)
    t.forward(height)
    t.write(str(height), font = ('Times New Roman', 16, 'bold'))
    t.right(90)

    t.forward(40)
    t.right(90)
    t.forward(height)
    t.left(90)
    t.end_fill()

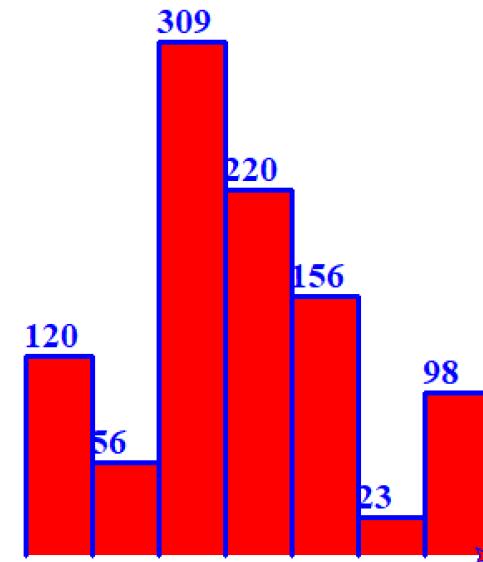
# 거북이 객체 생성
t = turtle.Turtle()

# t.color('blue')
# t.fillcolor('red')
t.color('blue', 'red')
t.pensize(3)

data = [ 120, 56, 309, 220, 156, 23, 98 ]

for d in data :
    drawBar(d)

turtle.mainloop()
```



# 터틀 그래픽: TurtleScreen/Screen (1/3)

## 예제 0-51: 터틀 그래픽 -- 다양한 모양 그리기 #3

```
import turtle

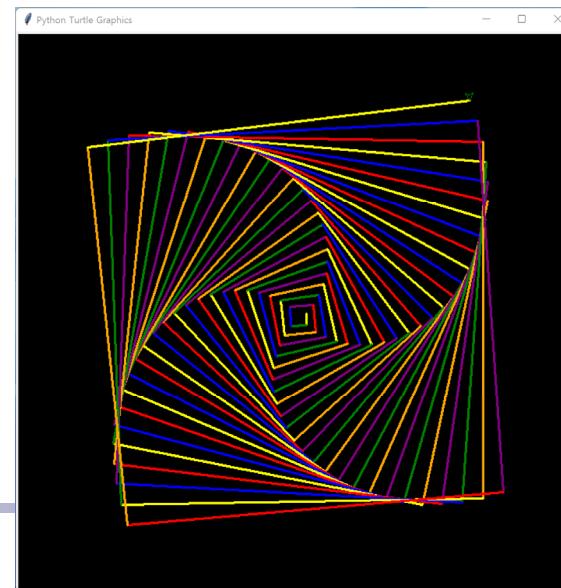
turtle.bgcolor('black')

# 거북이 객체 생성
t = turtle.Turtle()
t.speed(0)
t.width(3)

colors = [ 'red', 'purple', 'blue', 'green', 'yellow', 'orange' ]

length = 10
while length < 500 :
    t.forward(length)
    t.pencolor(colors[length%6])
    t.right(89)
    length += 5

turtle.mainloop()
```



# 터틀 그래픽: TurtleScreen/Screen (2/3)

## 예제 0-52: 터틀 그래픽 -- callback function

```
# 콜백 함수(callback function)
# 이벤트가 발생했을 때, 이벤트를 처리하는 함수
import turtle

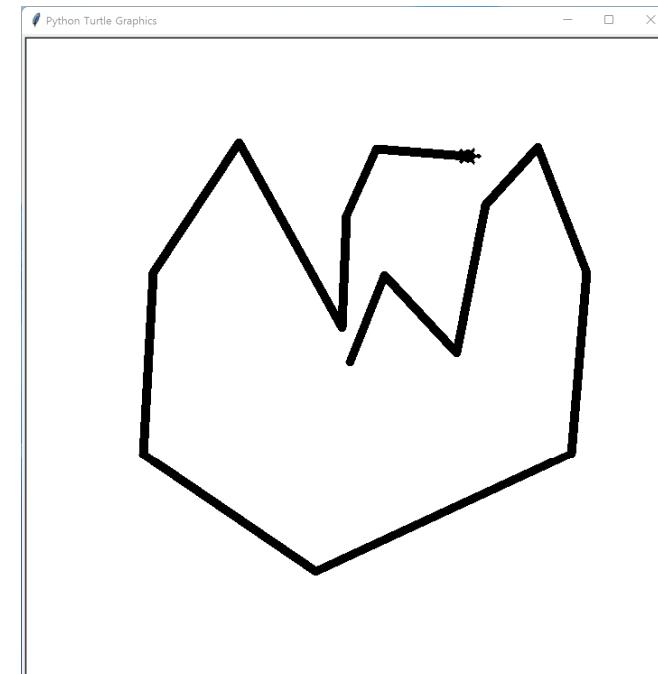
def draw(x, y):
    t.goto(x, y)

# 그림이 그려지는 화면
s = turtle.Screen()

# 마우스 클릭 이벤트 처리 함수 등록
s.onclick(draw)

# 거북이 객체 생성
t = turtle.Turtle()
t.shape('turtle')
t.pensize(10)

# turtle.mainloop()
```



# 터틀 그래픽: TurtleScreen/Screen (3/3)

## 예제 0-53: 터틀 그래픽 -- 다양한 모양 그리기 #2

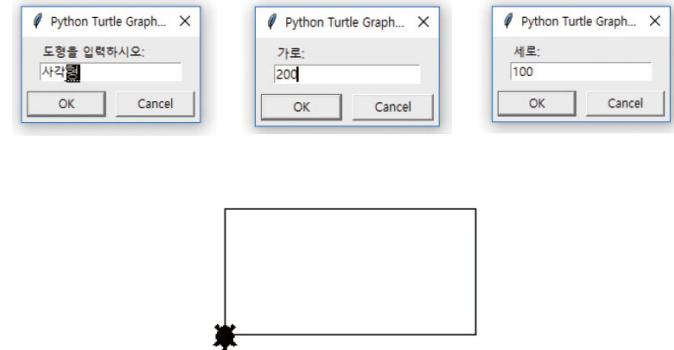
```
import turtle

# 거북이 객체 생성
t = turtle.Turtle()
t.shape('turtle')

s = turtle.textinput("", '도형을 입력하세요(예: 삼각형, 사각형, 원): ')

if s == '사각형' :
    weight=int(turtle.textinput('', '가로: '))
    height=int(turtle.textinput('', '세로: '))
    for i in range(2) :
        t.forward(weight); t.left(90)
        t.forward(height); t.left(90)
elif s == '삼각형' :
    weight=int(turtle.textinput('', '너비: '))
    for i in range(3) :
        t.forward(weight); t.left(120)
elif s == '원' :
    radius=int(turtle.textinput('', '반지름: '))
    t.circle(radius)
else :
    t.write('잘못 입력하셨습니다.')

turtle.mainloop()
```



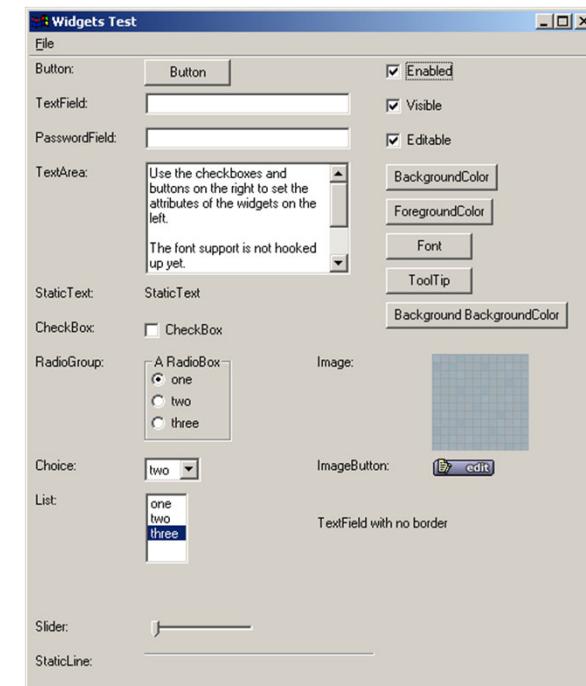
# Tkinter (1/2)

## ● Tkinter(TK Interface): [wiki.python.org/moin/TkInter](http://wiki.python.org/moin/TkInter)

### ○ Tcl/Tk GUI ToolKit에 대한 표준 파이썬 인터페이스

- 파이썬에서 GUI(Graphical User Interface) 프로그래밍을 위하여 가장 일반적으로 사용되는 파이썬 GUI 표준 라이브러리이다.
  - 윈도우와 리눅스, 맥 등에서 모두 동일한 코드로 사용 가능

- Tcl(Tool Command Language)
- Tk(Tool Kit)
- Ttk(Themed Tk): Tk 8.5부터 Tk의 일부로 배포된다.

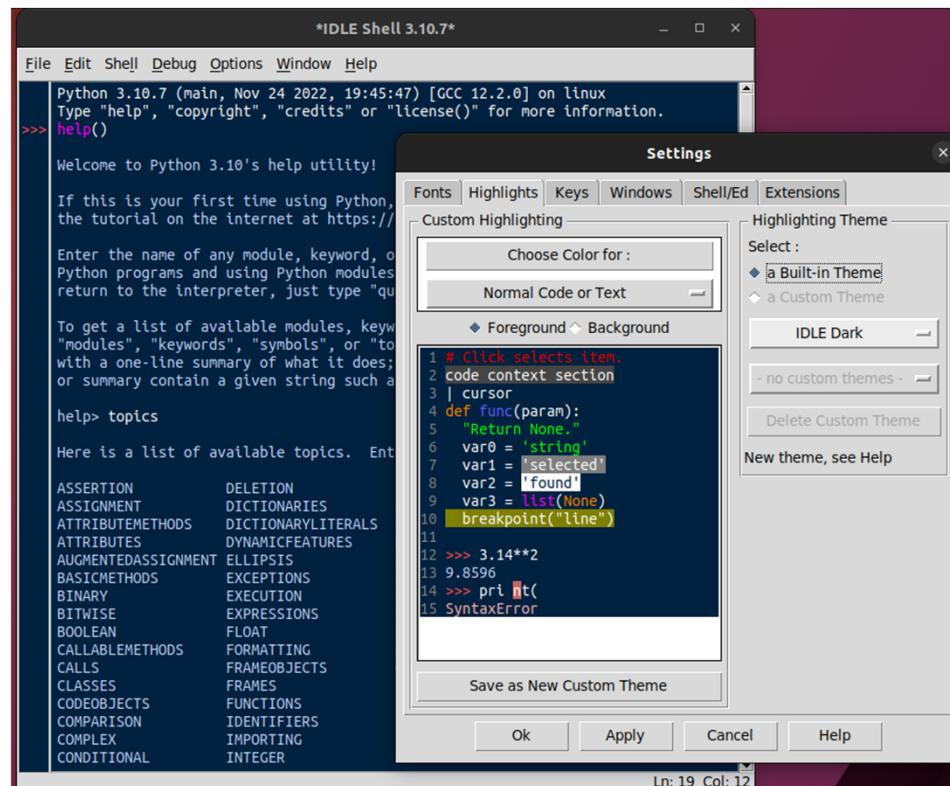


# Tkinter (2/2)

## ● IDLE

### ○ Integrated Development and Learning Environment

- Tkinter를 사용하여 독점적으로 작성된 파이썬의 통합 개발 환경



# Tkinter: Widgets (1/2)

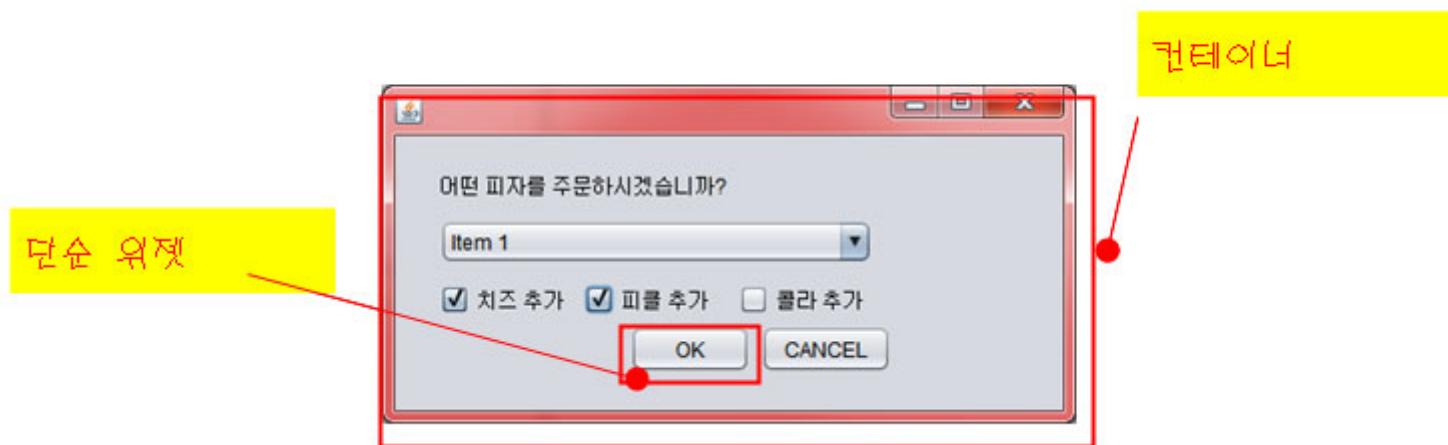
## ● Widgets: 단순 위젯과 컨테이너 위젯

### ○ 단순 위젯

- Button, Canvas, Checkbutton, Entry, Label, Message 등

### ○ 컨테이너 위젯

- 다른 컴포넌트를 안에 포함할 수 있는 컴포넌트
- Frame, Toplevel, LabelFrame, PanedWindow 등



# Tkinter: Widgets (2/2)

## ● 기본 위젯: 기본 윈도우 창

### ○ 기본 윈도우 창 구성

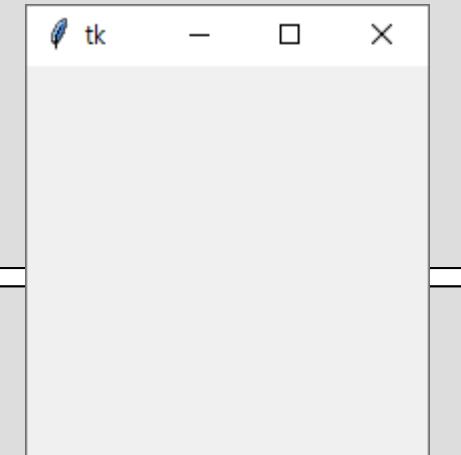
```
# tkinter : 파이썬 GUI 표준 라이브러리
from tkinter import *

# Tk 클래스: 윈도우 창 생성
window = Tk()                      # Tk : Root Window, Base Window
```

# 이 부분에서 화면을 구성하고 처리한다.

# 윈도우 창을 유지한다.  
window.mainloop()

```
# Spyder IDE
import tkinter as tk
window = tk.Tk()
window.mainloop()
```



# Tkinter: 일반 계산기

- 계산기 프로그램: 일반 계산기

- 사용자 인터페이스

- 구현 단계

- 원도우 및 프레임 생성
      - 엔트리 생성 및 화면 표시
      - 버튼 생성 및 격자 모양으로 배치
    - 버튼 이벤트 처리 함수





# 표준 라이브러리

GUI 프로그래밍  
: GUI Frameworks



# GUI Frameworks

---

- **GUI Frameworks**

- PyQt
- PyGObject: PyGTK
- PySide
- wxPython
- Kivy
- Libavg
- PySimpleGUI
- Pyforms
- Wax
- PyGUI



# PyQt (1/2)

- **PyQt:** [riverbankcomputing.com/software/pyqt/](http://riverbankcomputing.com/software/pyqt/)

- Python wrapper for the Qt toolkit
  - Qt의 레이아웃에 Python 코드를 연결하여 GUI 프로그램을 만들 수 있게 해주는 프레임워크를 의미한다.
- 1998년, 영국 회사인 **Riverbank Computing**에서 처음 출시
  - C++ Cross Platform GUI Framework인 Qt를 영국의 Riverbank Computing에서 Python 모듈로 변환해주는 툴을 만들어주며 시작되었다.
- PyQt 릴리즈
  - 1998년, PyQt 0.5.0
  - 2024년 04월, PyQt 6.7.0
- 라이선스: GNU GPL and Commercial
- 프로그래밍 언어: Python, C++



# PyQt (2/2)

- **PyQt: 프로그램 구조**

- PyQt 프로그램 구조: Hello World!!!

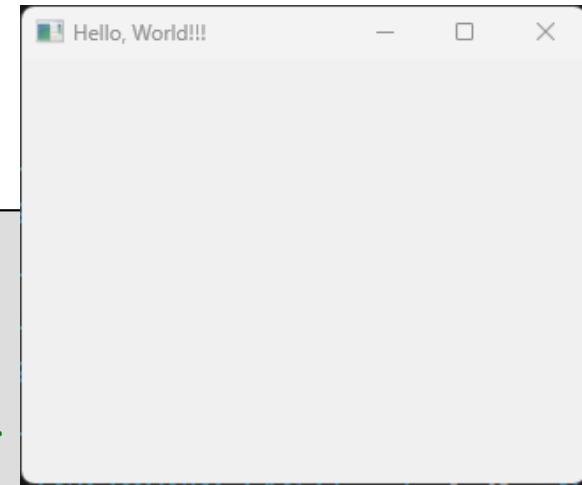
```
# pip install PyQt5
import sys
from PyQt5.QtWidgets import QApplication, QWidget

# 모든 PyQt5 애플리케이션은 애플리케이션 객체를 생성해야 한다.
# app = QApplication(sys.argv)           # PyQt4
app = QApplication([])                  # PyQt5/6

# QWidget: PyQt5의 모든 사용자 인터페이스 객체의 기본 클래스
root = QWidget()

root.resize(320, 240)                  # 위젯의 크기 조정
root.setWindowTitle('Hello, World!!!')  # 창의 제목 설정
root.show()                           # 화면에 위젯을 표시

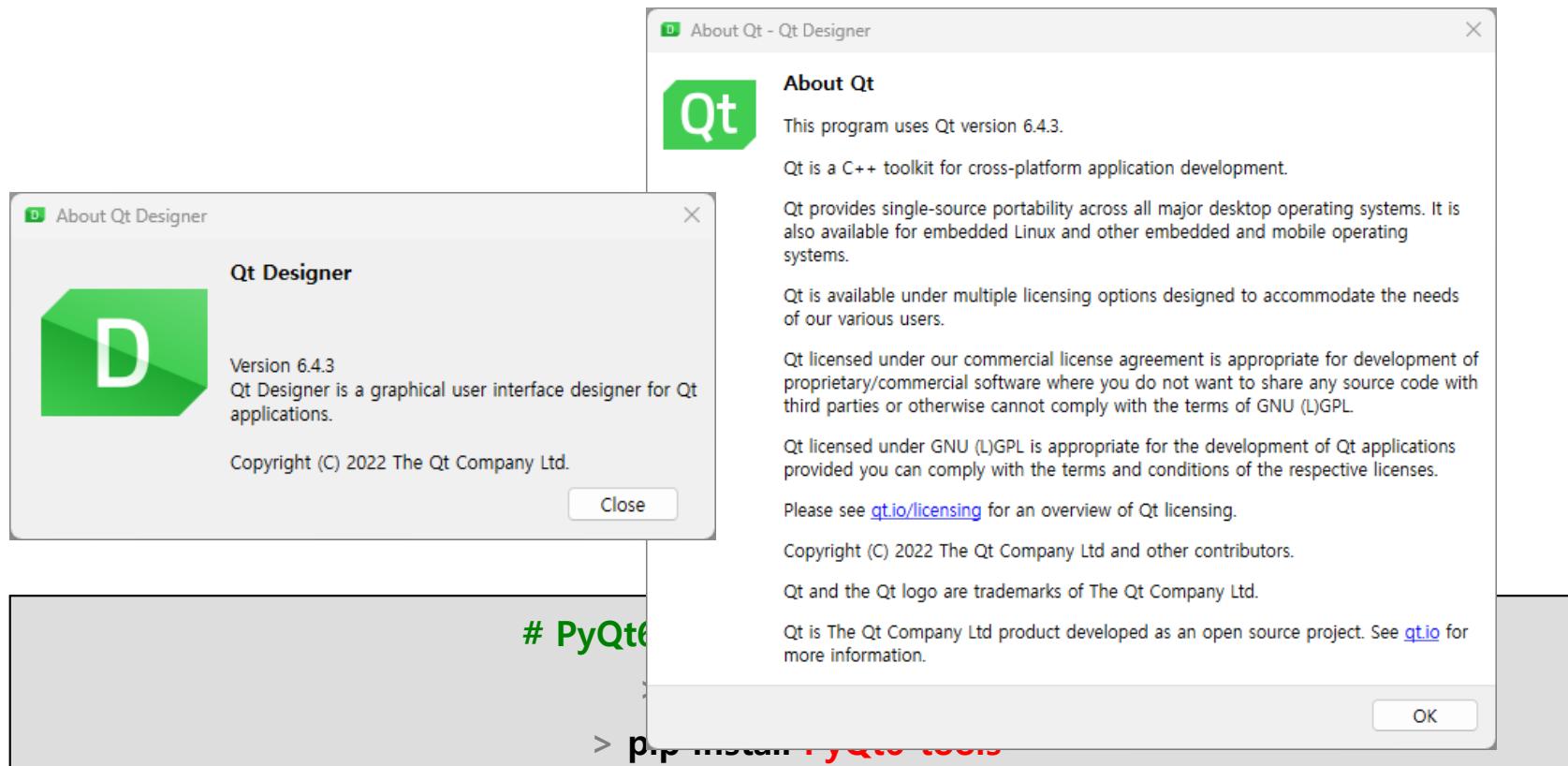
# 응용 프로그램을 mainloop로 진입 시킨다(프로그램을 작동 시킨다).
sys.exit(app.exec_())                 # PyQt4/5
# sys.exit(app.exec())                 # PyQt6
```



# PyQt: Qt Designer (1/6)

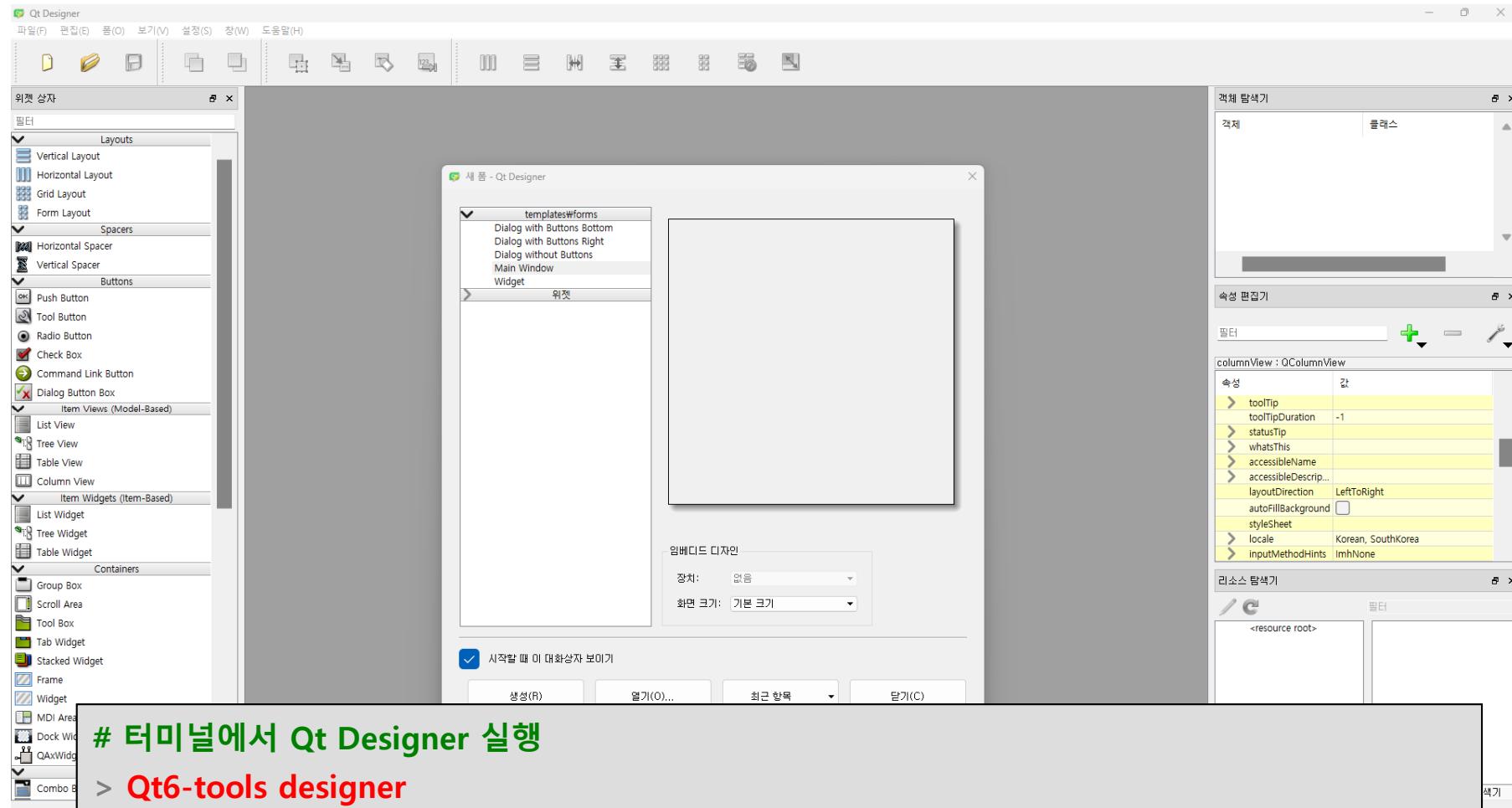
## ● Qt Designer

- PyQt를 이용하여 GUI프로그래밍을 할 때 손쉽게 프로그램의 레이아웃을 편집할 수 있도록 해주는 편집기



# PyQt: Qt Designer (2/6)

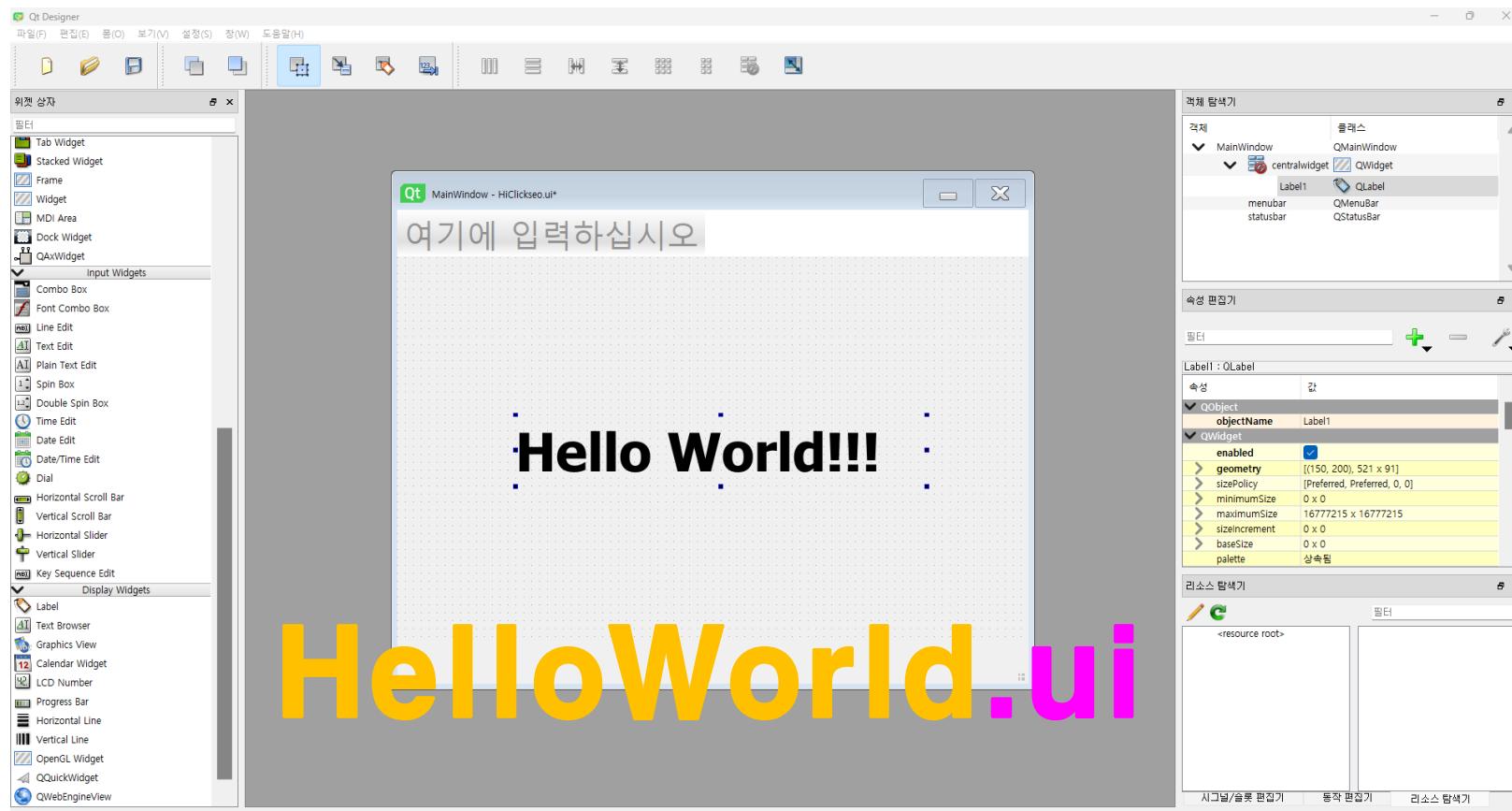
## ● Qt Designer: 시작 화면



# PyQt: Qt Designer (3/6)

## ● Qt Designer: Hello World!!!

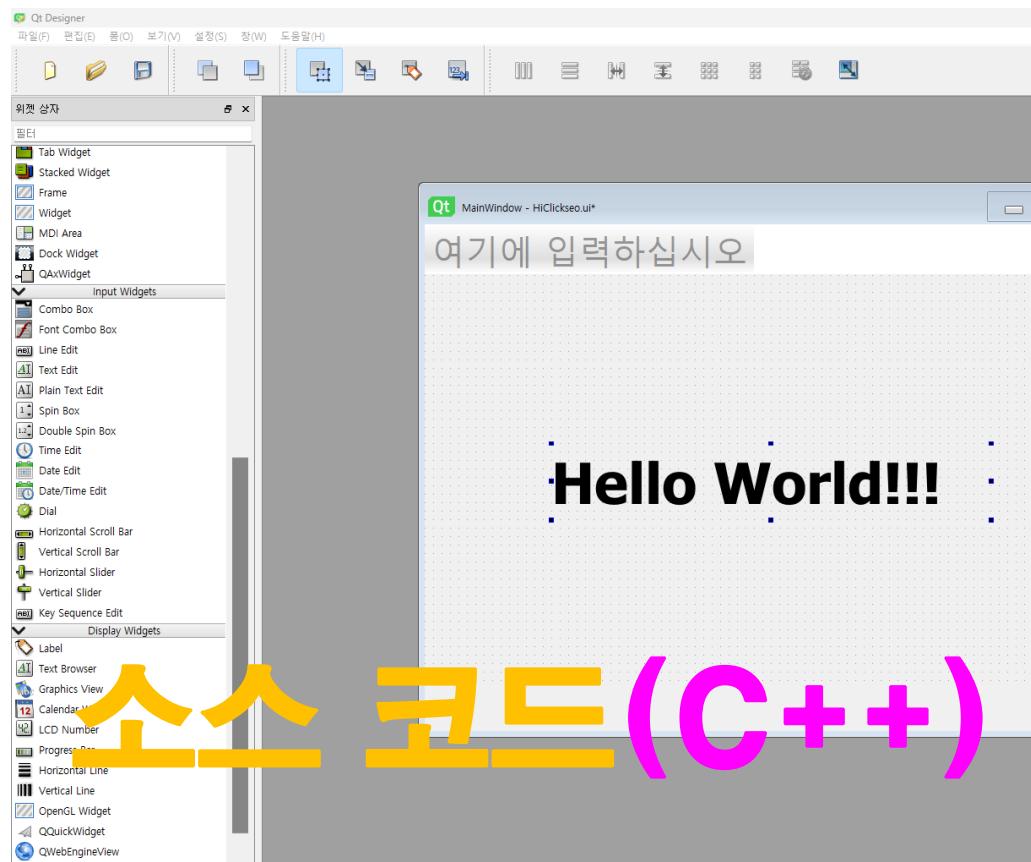
### 1. UI 제작 및 파일 저장: HelloWorld.ui



# PyQt: Qt Designer (4/6)

## ● Qt Designer: Hello World!!!

### 1. UI 제작 및 파일 저장: HelloWorld.ui



```
MainWindow - [코드] - Qt Designer

# ifndef HELLOWORLDBG4484_H
# define HELLOWORLDBG4484_H

#include <QtCore/QVariant>
#include <QtWidgets/QAction>
#include <QtWidgets/QApplication>
#include <QtWidgets/QButtonGroup>
#include <QtWidgets/QHeaderView>
#include <QtWidgets/QLabel>
#include <QtWidgets/QMainWindow>
#include <QtWidgets/QMenuBar>
#include <QtWidgets/QStatusBar>
#include <QtWidgets/QWidget>

QT_BEGIN_NAMESPACE

class Ui_MainWindow
{
public:
    QWidget *centralwidget;
    QLabel *Label1;
    QMenuBar *menubar;
    QStatusBar *statusbar;

void setupUi(QMainWindow *MainWindow)
{
    if (MainWindow->objectName().isEmpty())
        MainWindow->setObjectName(QString::literal("MainWindow"));
    MainWindow->resize(800, 600);
    QFont font;
    font.setPointSize(30);
    MainWindow->setFont(font);
    centralwidget = new QWidget(MainWindow);
    centralwidget->setObjectName(QString::literal("centralwidget"));
    Label1 = new QLabel(centralwidget);
    Label1->setObjectName(QString::literal("Label1"));
    Label1->setEnabled(true);
    Label1->setGeometry(QRect(150, 200, 521, 91));
    QFont font1;
    font1.setFamily(QString::literal("Tahoma"));
    font1.setPointSize(50);
    font1.setBold(true);
    font1.setWeight(75);
    font1.setKerning(true);
    Label1->setFont(font1);
    Label1->setMouseTracking(false);
    MainWindow->setCentralWidget(centralwidget);
    menubar = new QMenuBar(MainWindow);
    menubar->setObjectName(QString::literal("menubar"));
    menubar->setGeometry(QRect(0, 0, 800, 60));
    MainWindow->setMenuBar(menubar);
    statusbar = new QStatusBar(MainWindow);
    statusbar->setObjectName(QString::literal("statusbar"));
    MainWindow->setStatusBar(statusbar);

    retranslateUi(MainWindow);

    QMetaObject::connectSlotsByName(MainWindow);
} // setupUi
```

# PyQt: Qt Designer (5/6)

## ● Qt Designer: Hello World!!!

### 2. UI 와 Python 코드 연결: **HelloWorld.py**

```
import sys
from PyQt5.QtWidgets import *
from PyQt5 import uic

form_class = uic.loadUiType('HelloWorld.ui')[0]    # UI 파일 연결

# 화면을 띄우는데 사용되는 클래스 선언
class WindowClass(QMainWindow, form_class) :
    def __init__(self):
        super().__init__()
        self.setupUi(self)

    if __name__ == '__main__':
        # QApplication: 프로그램을 실행 시켜주는 클래스
        app = QApplication(sys.argv)

        myWindow = WindowClass()          # WindowClass의 객체 생성
        myWindow.show()                  # 프로그램 화면을 보여주는 코드

        # 응용 프로그램을 mainloop 진입 시킨다(프로그램을 작동 시킨다).
        app.exec_()
```

# PyQt: Qt Designer (6/6)

---

- **Qt Designer:** Hello World!!!

- 최종 실행 결과: **HelloWorld!!!**



# PyGObject

- **PyGObject:** [pygobject.readthedocs.io/](https://pygobject.readthedocs.io/)
  - GTK, GStreamer,WebKitGTK, GLib, GIO 등과 같은 **GObject** 기반 라이브러리에 대한 바인딩을 제공하는 파이썬 패키지
    - GNOME용 파이썬 응용 프로그램이나 GTK 기반 파이썬 GUI 응용 프로그램 개발
    - PyGTK는 GTK3 전환과 함께 단계적으로 폐지되고 PyGObject로 대체되었다.
  - PyGObject 릴리즈
    - 2006년 01월, GObject 2.8.0
    - 2023년 03월, GObject 3.44.1
  - 라이선스: GNU LGPL
  - 프로그래밍 언어: Python, C

PyGObject Repository(GNOME GitLab): [gitlab.gnome.org](https://gitlab.gnome.org/GNOME/pygobject)  
<https://gitlab.gnome.org/GNOME/pygobject>





# 외부 라이브러리

인공지능

: TensorFlow, Keras, PyTorch



# 인공지능 (1/3)

## ● TensorFlow: tensorflow.org

- 2015년 11월, Google Brain 팀의 두 번째 머신 러닝 시스템
  - 기계학습(Machine Learning)을 위한 오픈소스SW 라이브러리
    - 2011년부터 Google Brain 팀은 DistBelief를 딥 러닝 신경망(Neural Networks)을 기반으로 한 독점적인 첫 번째 머신 러닝 시스템으로 구축하였다.
    - 2019년 05월, 컴퓨터 그래픽의 딥 러닝을 위한 TensorFlow Graphics 발표
- TensorFlow 릴리즈
  - 2015년 11월, TensorFlow 0.5.0 -- OpenSource 소프트웨어로 전환
  - 2019년 09월, TensorFlow 2.0.0
  - 2024년 03월, TensorFlow 2.16.1
- 라이선스: Apache License 2.0
- 프로그래밍 언어: Python, C++, CUDA



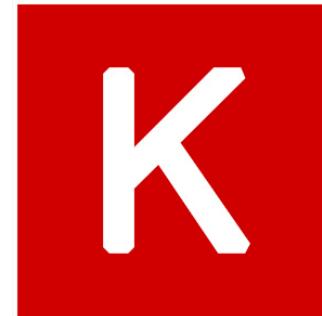
TensorFlow Repository(GitHub): [github.com](https://github.com/tensorflow/tensorflow)  
<https://github.com/tensorflow/tensorflow>



# 인공지능 (2/3)

## ● Keras: keras.io

- 2015년, 인공 신경망을 위한 Python 인터페이스를 제공하는 오픈소스 라이브러리
  - 2017년, Google TensorFlow 핵심 라이브러리에서 Keras를 지원하기로 결정
    - Keras는 TensorFlow 라이브러리의 인터페이스 역할
- Keras 릴리즈
  - 2015년 06월, Keras 0.1.0
  - 2024년 04월, Keras 3.3.3
- 라이선스: Apache License 2.0
- 프로그래밍 언어: Python



Keras Repository(GitHub): [github.com  
<https://github.com/keras-team/keras>](https://github.com/keras-team/keras)



# 인공지능 (3/3)

## ● PyTorch: [pytorch.org](https://pytorch.org)

- 2016년, Meta AI(Facebook)의 인공지능 연구 그룹에 의해 개발
  - 설치가 간편하며, TensorFlow 보다 사용자가 익히기 훨씬 쉽다.
  - 컴퓨터 비전 및 자연어 처리와 같은 애플리케이션에 사용되는 Torch 라이브러리를 기반으로 하는 오픈 소스 기계 학습 프레임워크
    - GPU를 통한 강력한 가속 기능을 갖춘 Tensor Computing(NumPy와 같은)
    - 테이프 기반 자동 미분 시스템에 구축된 심층 신경망
  - Caffe2는 2018년 03월에 PyTorch에 통합 되었다.
- PyTorch 릴리즈
  - 2016년 09월, PyTorch 0.1.1
  - 2024년 04월, PyTorch 3.3.0
- 라이선스: BSD-3 License
- 프로그래밍 언어: Python, C++, CUDA



PyTorch Repository(GitHub): [github.com  
<https://github.com/pytorch/pytorch>](https://github.com/pytorch/pytorch)





# 외부 라이브러리

데이터 분석 및 시각화

: SciPy, NumPy, Pandas, Matplotlib



# 데이터 분석 및 시각화: SciPy

## ● SciPy: [scipy.org](http://scipy.org)

### ○ 2001년,

- 과학과 기술 컴퓨팅에 사용되는 자유-오픈소스 파이썬 라이브러리
- NumPy 배열 객체를 기반으로 하며, Matplotlib, Pandas 및 SymPy 와 같은 도구 및 확장된 과학적 컴퓨팅 라이브러리 세트를 포함하는 NumPy 스택의 일부

### ○ SciPy 릴리즈

- 2001년 08월, SciPy 0.1.0
- 2017년 10월, SciPy 1.0.0
- 2024년 03월, SciPy 1.13.0



### ○ 라이선스: BSD 3-Clause(New BSD) License

### ○ 프로그래밍 언어: Fortran, Python, C/C++

SciPy Repository(GitHub): [github.com](https://github.com/scipy/scipy)  
<https://github.com/scipy/scipy>



# 데이터 분석 및 시각화: NumPy (1/4)

## ● NumPy: [numpy.org](http://numpy.org)

- 2006년, Numeric 에 Numarray 기능을 통합하여 NumPy 개발
  - 수학 및 과학 연산을 위한 과학 컴퓨팅에 사용되는 파이썬 패키지
    - 벡터나 행렬의 연산 등 선형대수학적인 문제나 수치해석과 관련된 문제 해결에 주로 사용
  - 1995년, Numeric -- **Numeric v24.2**(2005.11: 마지막 릴리즈)
    - Numarray **v1.5.2**(2006.08: 마지막 릴리즈)

### ○ NumPy 릴리즈

- 2002년 01월, NumPy 0.2.0
- 2006년, 10월, NumPy 1.0
- 2011년, NumPy 1.5 -- **Python 3 지원**
- 2024년 02월, NumPy 1.26.4



### ○ 라이선스: BSD 3-Clause(New BSD) License

### ○ 프로그래밍 언어: Python, C

NumPy Repository(GitHub): [github.com](https://github.com/numpy/numpy)  
<https://github.com/numpy/numpy>



# 데이터 분석 및 시각화: NumPy (2/4)

## ● 배열 생성: array, arange

- array 함수: 리스트와 튜플로부터 배열을 만들 수 있다.

```
# pip install numpy  
import numpy as np  
arr = np.array( [ 2, 4, 6, 8 ] )
```

- arange 함수: 파이썬의 표준 range 내장 함수와 비슷하다.

```
import numpy as np  
  
a = np.arange(10)                      # 0부터 num - 1 까지의 배열을 반환  
b = np.arange(7, 11)  
c = np.arange(7, 11, 2)  
  
f = np.arange(2.0, 9.8, 0.3)           # 부동소수점 숫자도 사용 가능
```

# 데이터 분석 및 시각화: NumPy (3/4)

## ● 배열 생성: zeros, ones, random

- **zeros** 메소드: 모든 값이 0인 배열을 반환한다.

```
import numpy as np  
arr = np.zeros((3, ))  
table = np.zeros((2, 4))
```

- **ones** 메소드: 모든 값이 1인 배열을 반환한다.

```
import numpy as np  
arr = np.ones((3, ))  
table = np.ones((2, 4))
```

- **random** 메소드: 0.0과 0.1 사이의 임의의 값을 채워서 배열을 생성한다.

```
import numpy as np  
arr = np.random.random((3, ))  
table = np.random.random((2, 4))
```

# 데이터 분석 및 시각화: NumPy (4/4)

## ● 배열 연산: 곱셈(\*) 연산자

- 곱셈(\*) 연산자: 연산자 재정의를 통하여, 한 번에 배열의 모든 값을 곱해준다.

```
from numpy import *
arr = arange(4)
print(arr)
```

```
arr *= 3
print(arr)
```

# 배열 arr의 전체 원소에 3을 곱해준다.

- 파이썬 리스트에서 전체 원소에 한꺼번에 곱해주는 과정

```
slist = list(range(4))
print(slist)
```

```
slist = [ num*3 for num in slist ]
print(slist)
```

# 데이터 분석 및 시각화: Pandas (1/2)

- **Pandas:** [pandas.pydata.org](https://pandas.pydata.org)

- 2008년 01월, 데이터 조작 및 분석을 위한 파이썬 라이브러리
  - 특히, 수치 테이블과 시계열을 조작하기 위한 데이터 구조와 작업을 제공한다.

- Pandas 릴리즈

- 2009년 12월, Pandas 0.1
  - 2024년 04월, Pandas 2.2.2

- 라이선스: BSD 3-Clause(New BSD) License

- 프로그래밍 언어: Python, Cython, C



Pandas Repository(GitHub): [github.com  
<https://github.com/pandas-dev/pandas>](https://github.com/pandas-dev/pandas)



# 데이터 분석 및 시각화: Pandas (2/2)

## 예제 0-54: 데이터 프레임 생성 -- Pandas

```
# pip install pandas
```

```
import pandas as pd
```

```
# 데이터 프레임 생성: 직접 생성과 엑셀, CSV 그리고 DB(MySQL 등)에서 불러올 수 있다.
```

```
df = pd.DataFrame({"col":[1,2,3,4]})  
print(df)
```

```
# 데이터 프레임의 데이터만 출력
```

```
print(df.head())
```

```
# 데이터 타입과 NULL(공백)값의 여부 확인
```

```
print(df.info())
```

```
IDLE Shell 3.11.2  
File Edit Shell Debug Options Window Help  
Python 3.11.2 (tags/v3.11.2:878ead1, Feb 1  
Type "help", "copyright", "credits" or "li  
>>> RESTART: C:\Users\slm\clickse  
  
col  
0 1  
1 2  
2 3  
3 4  
  
col  
0 1  
1 2  
2 3  
3 4  
  
<class 'pandas.core.frame.DataFrame'>  
RangeIndex: 4 entries, 0 to 3  
Data columns (total 1 columns):  
 #   Column Non-Null Count Dtype  
 ---  ---  
 0   col      4 non-null      int64  
 dtypes: int64(1)  
 memory usage: 164.0 bytes  
None  
>>>
```

# 데이터 분석 및 시각화: Matplotlib (1/4)

- **Matplotlib:** [matplotlib.org](http://matplotlib.org)

- 2003년, 파이썬 및 Numpy에서 널리 사용되는 그래픽 패키지
  - MATLAB과 유사한 그래프 표시를 가능하게 하는 라이브러리

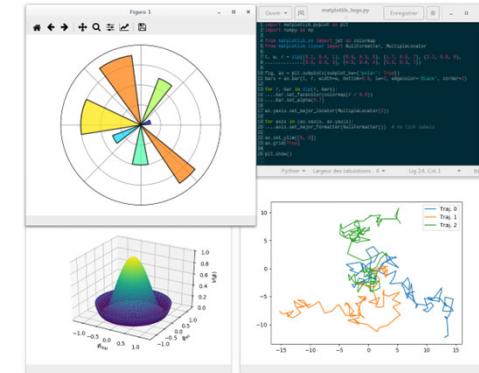
- matplotlib 릴리즈

- 2008년 07월, matplotlib 0.91.2
  - 2012년 11월, matplotlib 1.2.0 -- **Python 3 지원**
  - 2020년 이후 Python2를 지원하지 않는다.
  - 2025년 05월, matplotlib 3.10.3

- 라이선스: **Matplotlib License**

- **BSD** 호환 코드만 사용하며,  
라이선스는 **PSF** 라이센스를 기반으로 한다.

- 프로그래밍 언어: Python



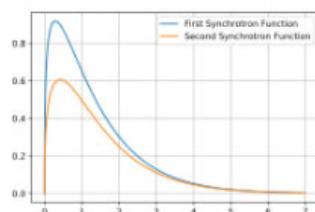
matplotlib Repository(GitHub): [github.com](https://github.com/matplotlib/matplotlib)

<https://github.com/matplotlib/matplotlib>

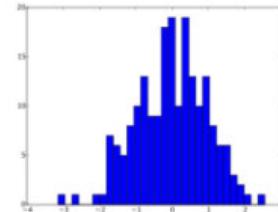
# 데이터 분석 및 시각화: Matplotlib (2/4)

## ● Matplotlib: 그래프

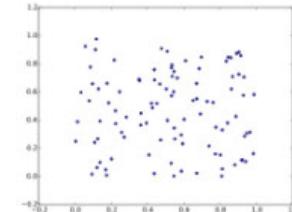
### ○ 다양한 그래프 사용 예



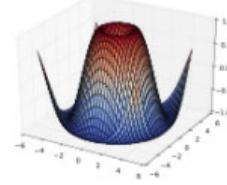
Line plot



Histogram



Scatter plot



3D plot

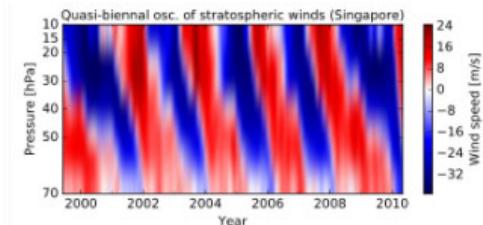
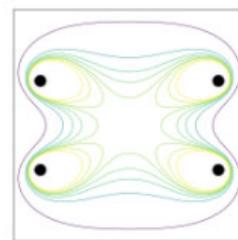
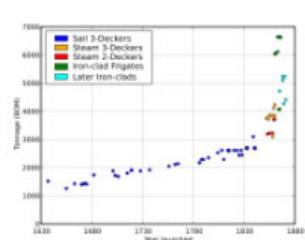


Image plot



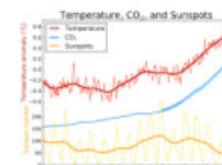
Contour plot



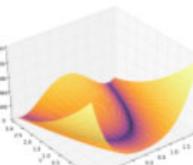
Scatter plot



Polar plot



Line plot



3-D plot

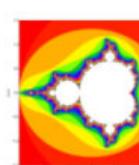


Image plot

[ 이미지 출처: "Matplotlib", WIKIPEDIA. ]

# 데이터 분석 및 시각화: Matplotlib (3/4)

예제 0-55: 그래프 그리기 -- Matplotlib

(1/2)

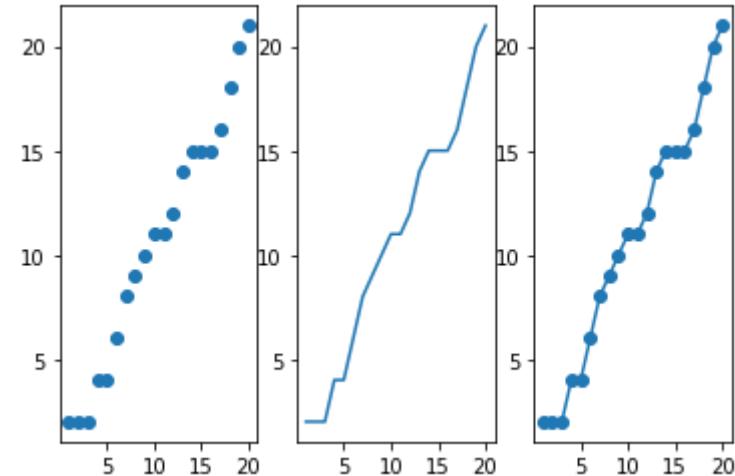
```
# Pyplot은 MATLAB과 유사한 인터페이스를 제공하는 Matplotlib 모듈
# pip install matplotlib
import matplotlib.pyplot as plt
from random import randint

linear = list(range(1,21))
wiggly = list(num + randint(-1,1) for num in linear)

fig, plots = plt.subplots(nrows=1, ncols=3)

ticks = list(range(0, 21, 5))
for plot in plots:
    plot.set_xticks(ticks)
    plot.set_yticks(ticks)

plots[0].scatter(linear, wiggly)
plots[1].plot(linear, wiggly)
plots[2].plot(linear, wiggly, 'o-')
```



# 데이터 분석 및 시각화: Matplotlib (4/4)

예제 0-55: 그래프 그리기 -- Matplotlib

(2/2)

## Run Current File in Interactive Window

The screenshot shows a Jupyter Notebook environment with a dark theme. On the left is a file browser sidebar with icons for files, folders, and other notebook files. The main area has two tabs: 'index.py' (active) and 'Interactive-1'. The code in 'index.py' is as follows:

```
C: > Users > click > OneDrive > 문서 > index.py > ...
1 # Pyplot은 MATLAB과 유사한 인터페이스를 제공하는 Matplotlib
2 # pip install matplotlib
3 import matplotlib.pyplot as plt
4 from random import randint
5
6 linear = list(range(1,21))
7 wiggly = list(num + randint(-1,1) for num in linear)
8
9 fig, plots = plt.subplots(nrows=1, ncols=3)
10
11 ticks = list(range(0, 21, 5))
12 for plot in plots:
13     plot.set_xticks(ticks)
14     plot.set_yticks(ticks)
15
16 plots[0].scatter(linear, wiggly)
17 plots[1].plot(linear, wiggly)
18 plots[2].plot(linear, wiggly, 'o-')
```

The 'Interactive-1' tab shows the output of the code. It includes a status bar at the bottom with information like 'Ln 15, Col 5', 'Spaces: 4', 'UTF-8', and 'Python 3.13.2 64-bit'. A message 'Press Enter to execute.' is displayed at the bottom of the code cell.



# 외부 라이브러리

인터넷과 웹



# 인터넷과 웹 (1/5)

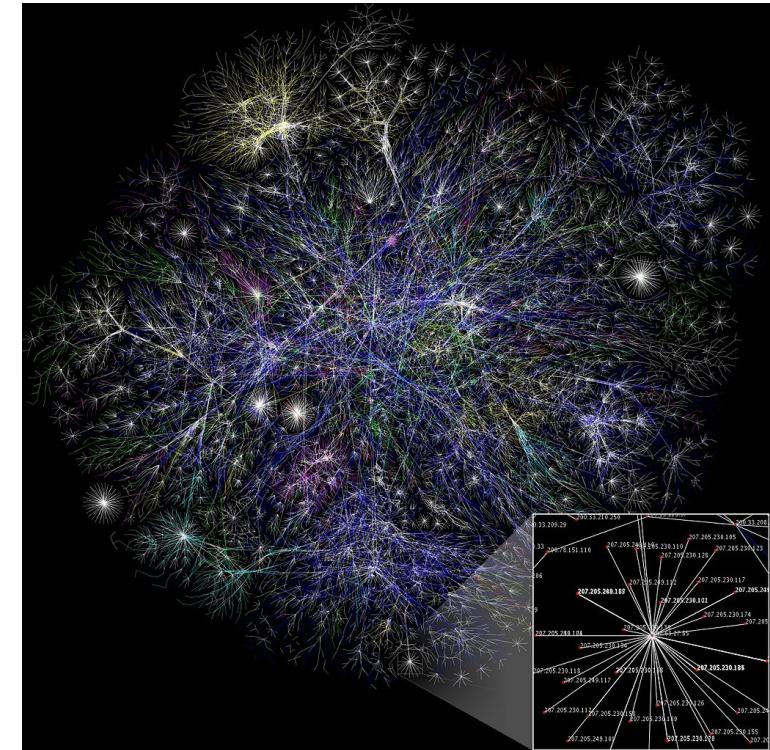
## ● Internet

### ○ International Network

- 네트워크의 네트워크를 구현하여 모든 컴퓨터를 하나의 통신망 안에 연결
- TCP/IP 라는 통신 프로토콜을 이용해 정보를 주고 받는 컴퓨터 네트워크
  - TCP(Transmission Control Protocol)
  - IP(Internet Protocol)

### ○ 인터넷의 특징

- 인터넷의 효용성: 정보 교류, 정보의 바다
- 개방된 통신망
- 독자적인 주소 할당
- 32비트의 고유한 IP 주소



[ 출처: "라우터를 통해 연결된 인터넷 시각화", WIKIPEDIA. ]

## ● Web

### ○ WWW, World Wide Web

- 다양한 형태의 데이터와 정보에 접근할 수 있도록 해 주는 인터넷 서비스
- HTTP(HyperText Transfer Protocol)라는 프로토콜을 사용하는 네트워크
  
- 1989년, 팀 베너스-리(Tim Berners-Lee)에 의해 개발
  - Hyperlink라는 특수한 기능을 사용해 인터넷 공간에서 문서와 문서 사이를 이동할 수 있는 새로운 개념을 제안
  - 1990년, 최초의 웹 브라우저 넥서스(Nexus) 개발
  
- 1993년, Mosaic 웹 브라우저 개발
- 1994년, Netscape Navigator 웹 브라우저
- 1996년, Microsoft Internet Explorer 웹 브라우저



[ 출처: "최초의 그래픽 기반 웹 브라우저인 모자이(Mosaic)", WIKIPEDIA. ]

# 인터넷과 웹 (3/5)

## ● **HTTP**(HyperText Transfer Protocol)

- IETF, HTTP Working Group
- 웹 상에서 정보를 주고받을 수 있는 프로토콜

- 주로 HTML 문서를 주고받는 데 사용한다.
- TCP와 UDP를 사용하고, 80번 포트를 사용한다.
- HTTP 표준과 기술 문서
  - 1996년 05월, **HTTP/1.0** 표준(RFC 1945) 발표
  - 1997년 01월, **HTTP/1.1** 표준(RFC 2068) 발표
    - » 1999년 06월, HTTP/1.1 표준에 대한 개선과 업데이트: RFC 2616
  - 2015년 05월, **HTTP/2** 표준(RFC 7540) 발표
  - **2022년 06월, HTTP/3** 표준(RFC 9114) 발표 -- **HTTP over QUIC(UDP 기반의 프로토콜)**
    - » Google, **QUIC**(Quick UDP Internet Connections) 프로토콜



## ○ **HTTPS**

- HyperText Transfer Protocol over Secure Sockets Layer
- HTTP over TLS, HTTP over SSL, **HTTP Secure**
  - 1994년, Netscape Communications가 Netscape Navigator 웹 브라우저를 위해 개발
  - HTTPS는 소켓 통신에서 일반 텍스트를 이용하는 대신, **SSL**이나 **TLS** 프로토콜을 통해 세션 데이터를 암호화한다.
- HTTPS의 기본 TCP/IP 포트는 **443**이다.



# 인터넷과 웹 (4/5)

## ● URL(Uniform Resource Locator)

### ○ 인터넷에 있는 자료가 가지는 유일한 주소

- 인터넷에 있는 정보의 접근 형식과 자료가 존재하는 위치와 자료의 이름을 표시

**protocol://host.domain[:port]/directory/filename**

### ○ protocol : http, https, ftp, telnet, mailto

### ○ port

- HTTP(Hypertext Transfer Protocol): **80**
- FTP(File Transfer Protocol (Data)): **20**
- FTP(File Transfer Protocol (Control)): **21**
- Telnet(Telnet Protocol): **23**
- SSH(Secure Shell Remote Login Protocol): **22**
- SMTP(Simple Mail Transfer Protocol): **25**
- POP3(Post Office Protocol-Version 3): **110**
- IMAP(Internet Message Access Protocol): **143**
- HTTPS(Hypertext Transfer Protocol Secure): **443**

# 인터넷과 웹 (5/5)

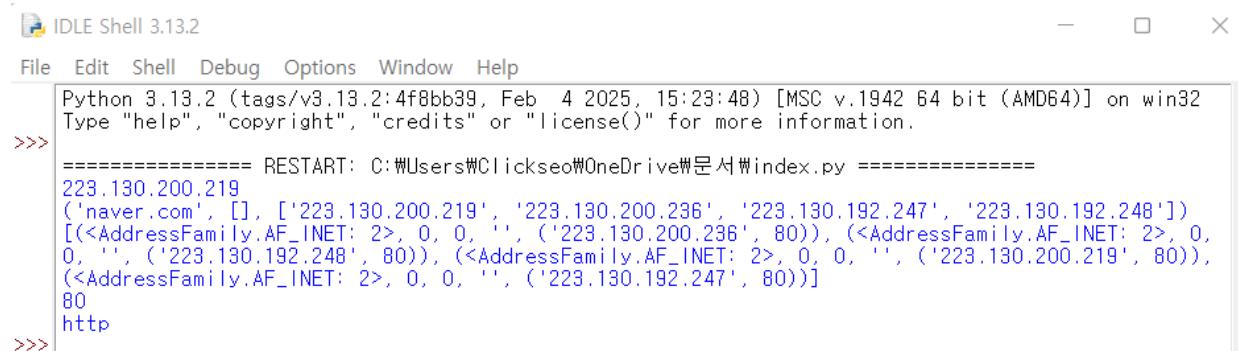
## 예제 0-56: 인터넷 서비스

```
import socket
```

```
# gethostname: 도메인 이름에 대한 IP 주소를 반환
print(socket.gethostname('naver.com'))
print(socket.gethostname_ex('naver.com'))
```

```
# getaddrinfo: IP 주소 검색
print(socket.getaddrinfo('www.naver.com', 80))
```

```
# 서비스 이름과 포트 번호 변환: getservbyname, getservbyport
print(socket.getservbyname('http'))
print(socket.getservbyport(80))
```



The screenshot shows the Python 3.13.2 IDLE Shell interface. The window title is "IDLE Shell 3.13.2". The menu bar includes File, Edit, Shell, Debug, Options, Window, and Help. The main console area displays the following output:

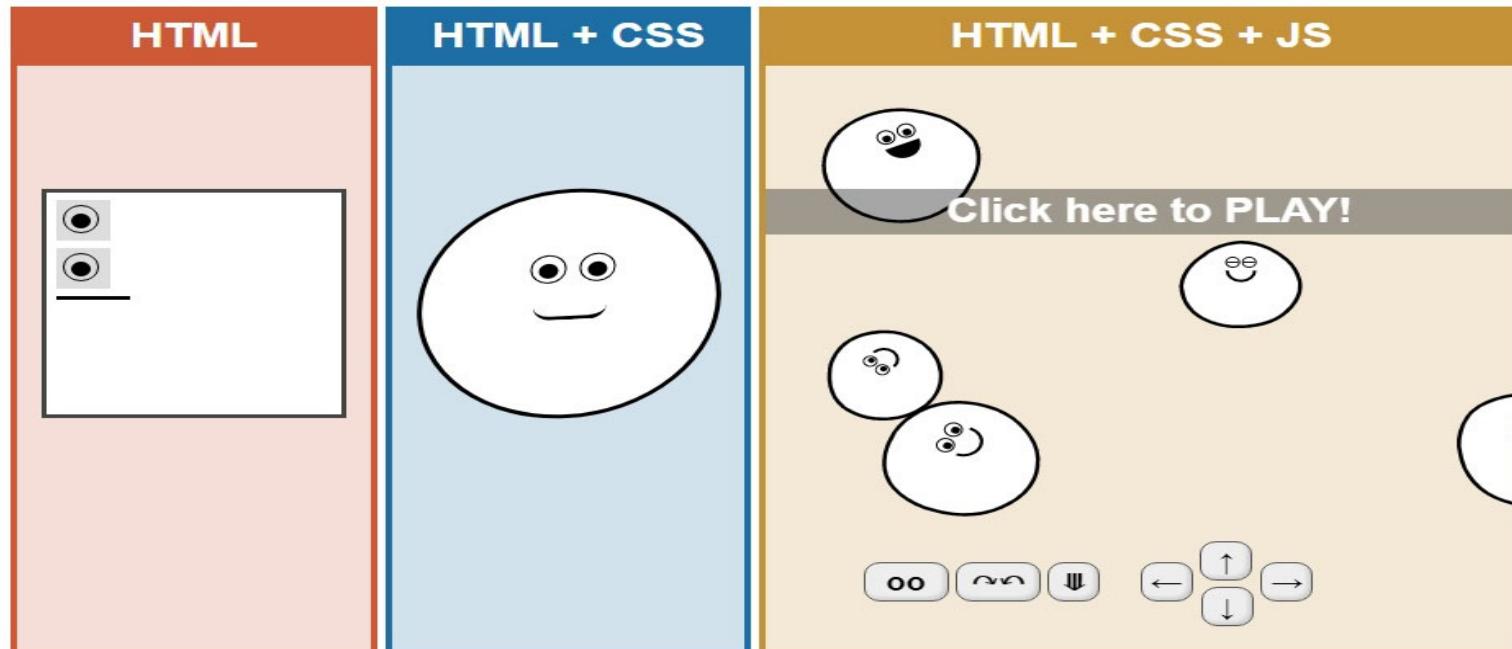
```
Python 3.13.2 (tags/v3.13.2:4f8bb39, Feb  4 2025, 15:23:48) [MSC v.1942 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.

>>> ===== RESTART: C:\Users\Clickseo\OneDrive\문서\index.py =====
223.130.200.219
('naver.com', [], ['223.130.200.219', '223.130.200.236', '223.130.192.247', '223.130.192.248'])
[(<AddressFamily.AF_INET: 2>, 0, 0, '', ('223.130.200.236', 80)), (<AddressFamily.AF_INET: 2>, 0, 0, '', ('223.130.192.248', 80)), (<AddressFamily.AF_INET: 2>, 0, 0, '', ('223.130.200.219', 80)), (<AddressFamily.AF_INET: 2>, 0, 0, '', ('223.130.192.247', 80))]
80
http
>>>
```

# 웹 표준 기술

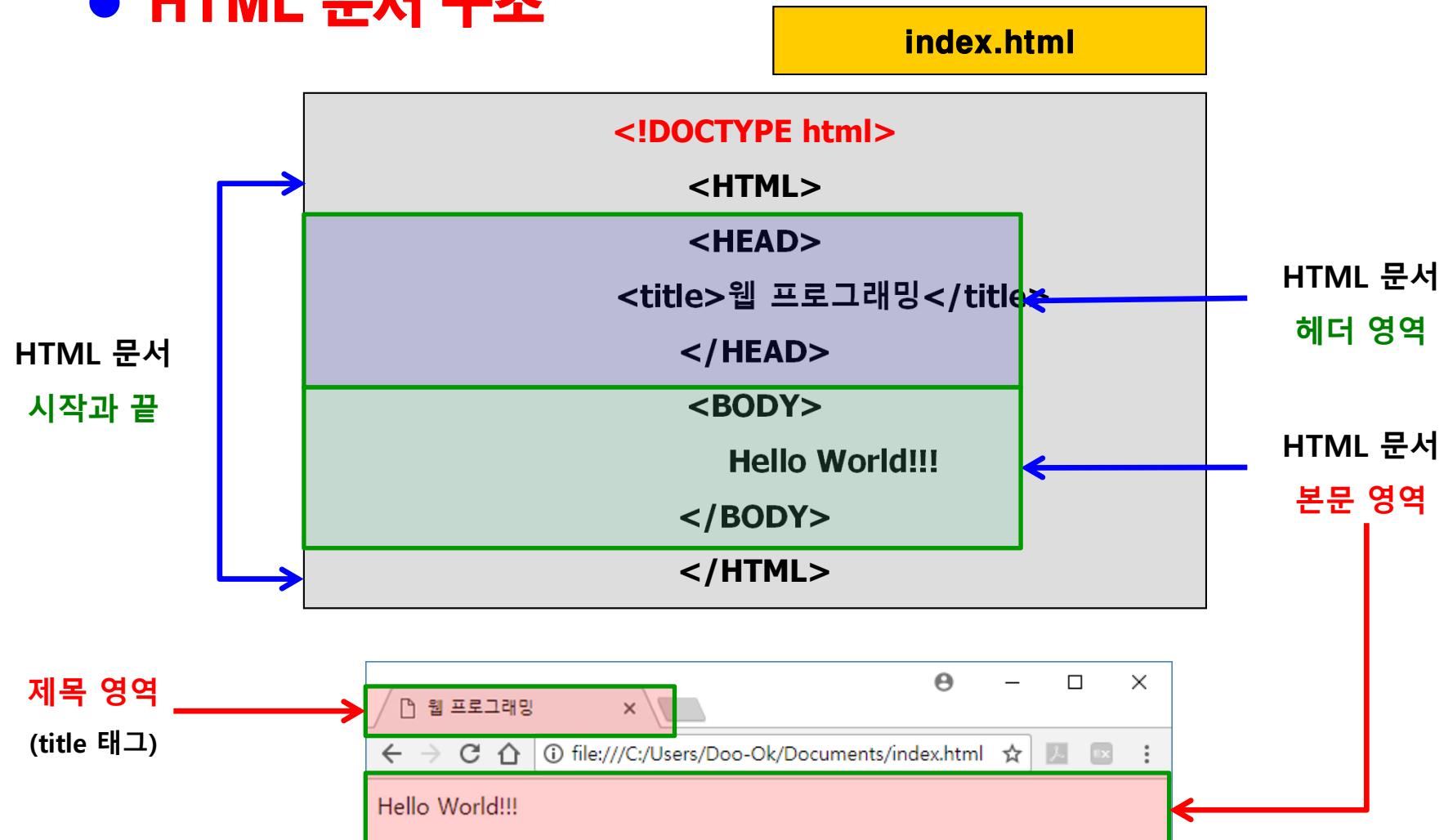
## ● HTML + CSS + JavaScript

- **HTML** : Model(구조, 내용 정의)
- **CSS** : View(레이아웃 지정)
- **JavaScript** : Control(동작 프로그래밍)



# 웹 표준 기술: HTML 문서 구조

## ● HTML 문서 구조



# 웹 크롤링: BeautifulSoup (1/3)

## ● 웹 크롤링(Web Crawling)

- 수 많은 웹사이트들을 체계적으로 돌아다니면서 URL, 키워드 등을 수집하는 것

- 보통 검색 엔진이 웹사이트를 인덱싱하기 위해서 사용된다.
- 크롤링을 하는 프로그램: 크롤러(crawler) 또는 스파이더(spider)

## ○ 웹 스크래핑(Web scraping)

- 웹 사이트 상에서 원하는 정보를 추출하는 기술



# 웹 크롤링: BeautifulSoup (2/3)

예제 0-57: 웹 크롤링 -- 웹 페이지 링크 정보 획득

(1/2)

```
# requests: HTTP(S) 패킷을 생성해서 보내는 방식의 urllib3 기반 Python 라이브러리
# BeautifulSoup(bs4): HTML과 XML 문서를 파싱하기 위한 Python 라이브러리
# pip install requests
# pip install beautifulsoup4

def get_links(url):
    import requests
    from bs4 import BeautifulSoup as soup

    # http://www.clickseo.com
    res = requests.get(url)
    page = res.text
    doc = soup(page, 'html.parser')      # doc = soup(page)

    # links = [element.get('href') for element in doc.select('a')]      # select_one(select)
    links = [ element.get('href') for element in doc.find_all('a') ]      # find(find_all)
    return links
```

# 웹 크롤링: BeautifulSoup (3/3)

예제 0-57: 웹 크롤링 -- 웹 페이지 링크 정보 획득

(2/2)

```
if __name__ == '__main__':
    import sys

# python.exe index.py http://www.clickseo.com
for url in sys.argv[1:]:
    print('Links in', url)
    for num, link in enumerate(get_links(url), start=1):
        print(num, link)
    print()
```

The screenshot shows a terminal window with the following interface elements at the top:

- PROBLEMS (2)
- OUTPUT
- DEBUG CONSOLE
- TERMINAL
- PORTS
- JUPYTER

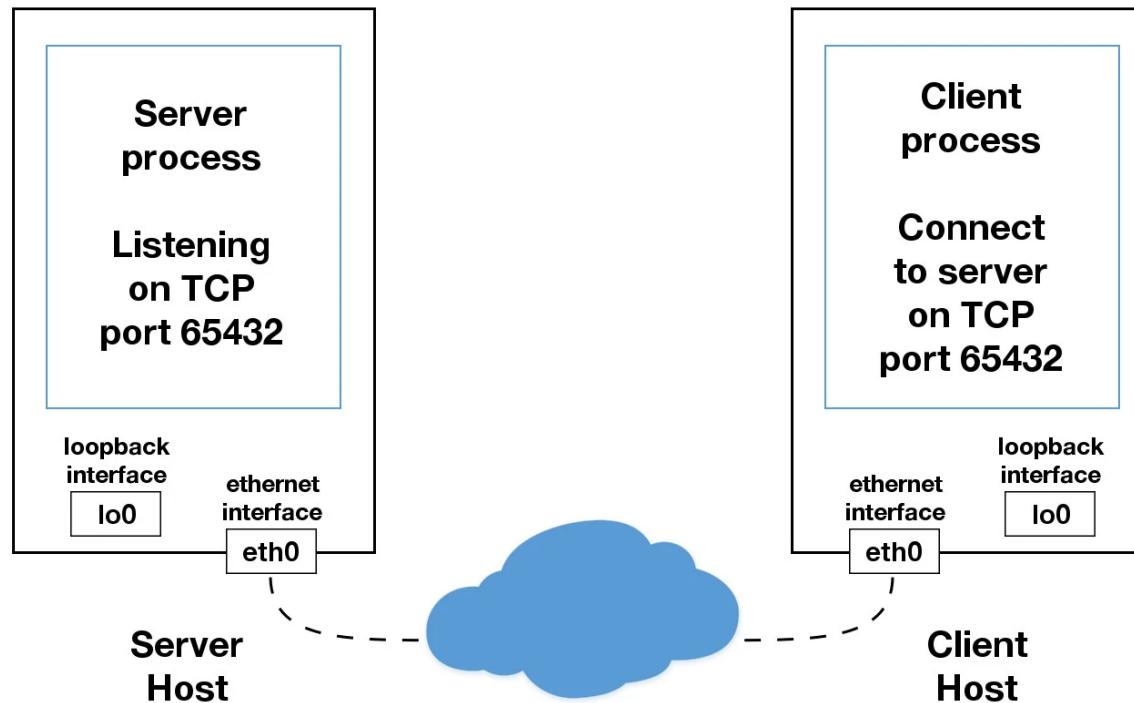
The terminal window displays the following output:

```
PS C:\Users\Clickseo\OneDrive\문서> python.exe .\index.py http://clickseo.com
Links in http://clickseo.com
1 .
2 ./?p=whoami
3 ./?p=opensource
4 ./?p=programming
5 ./?p=system
6 ./?p=ict
7 ./?p=link
8 http://www.youtube.com/channel/UCRxt1jvpf5LG_0MAaTUeXqw
9 http://www.facebook.com/ClickseoInsight

PS C:\Users\Clickseo\OneDrive\문서> []
```

# 소켓 프로그래밍

## ● 클라이언트와 서버 통신

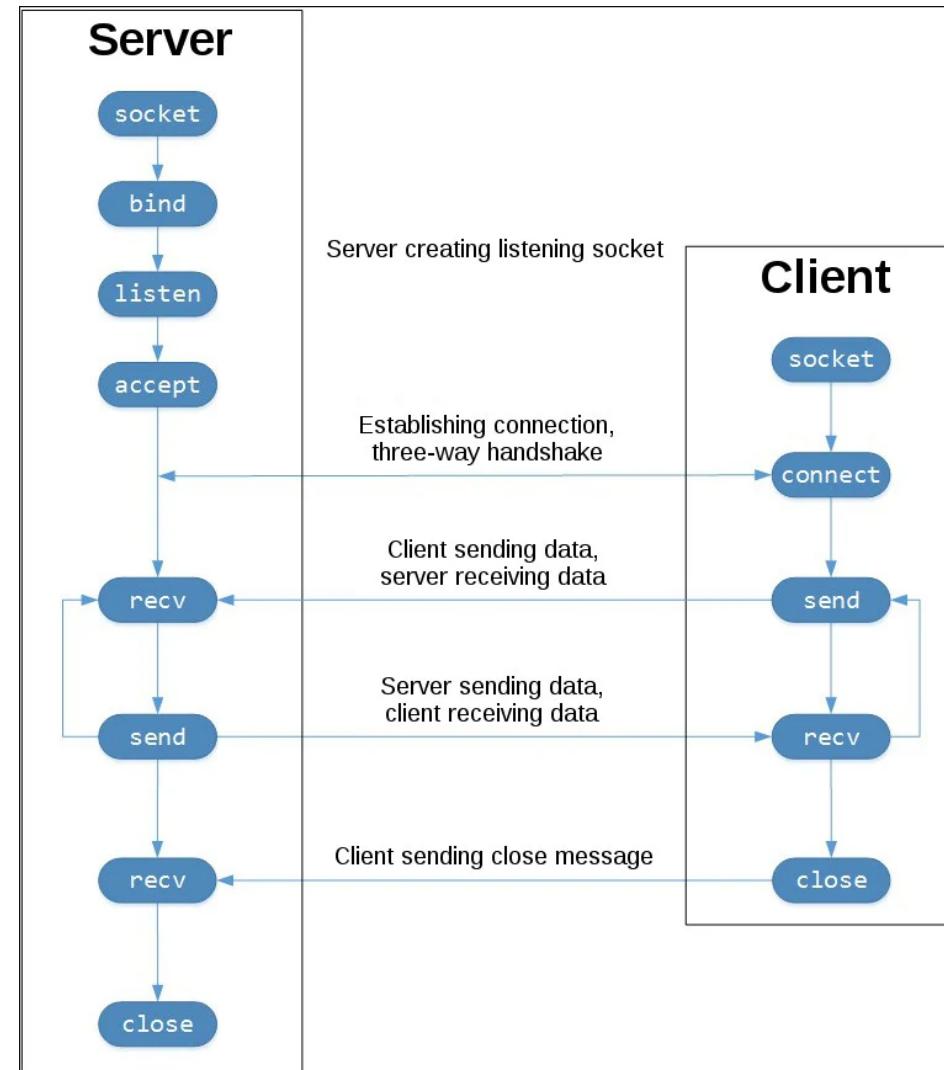


- **loopback 인터페이스:** IPv4(127.0.0.1) 또는 IPv6(::1)
- **ethernet naming:** eth0, eth1, ...

# 클라이언트-서버 모델: TCP (1/3)

## ● TCP 소켓 흐름

### ○ 클라이언트와 서버



# 클라이언트-서버 모델: TCP (2/3)

예제 0-58: TCP 기반 echo 서버/클라이언트 프로그램

tcp\_server.py

```
import socket
from datetime import datetime

address = ('localhost', 6789)
max_size = 1000

print('Starting the server at', datetime.now())
print('Waiting for a client to call.')

server = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
server.bind(address)
server.listen(5)

client, addr = server.accept()
data = client.recv(max_size)

print('At', datetime.now(), client, 'said', data)
client.sendall(b'Hi~ Clickseo')

client.close()
server.close()
```

# 클라이언트-서버 모델: TCP (3/3)

예제 0-58: TCP 기반 echo 서버/클라이언트 프로그램

tcp\_client.py

```
import socket
from datetime import datetime

address = ('localhost', 6789)
max_size = 1000

print('Starting the client at', datetime.now())

client = socket.socket(socket.AF_INET, socket.SOCK_STREAM)
client.connect(address)

client.sendall(b'^..^')
data = client.recv(max_size)

print('At', datetime.now(), 'someone replied', data)

client.close()
```

# 클라이언트-서버 모델: UDP (1/2)

예제 0-59: UDP 기반 echo 서버/클라이언트 프로그램

udp\_server.py

```
import socket
from datetime import datetime

server_address = ('localhost', 6789)
max_size = 4096

print('Starting the server at', datetime.now())
print('Waiting for a client to call.')

server = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
server.bind(server_address)

data, client = server.recvfrom(max_size)

print('At', datetime.now(), client, 'said', data)
server.sendto(b'Hi~ Clickseo', client)

server.close()
```

# 클라이언트-서버 모델: UDP (2/2)

예제 0-59: UDP 기반 echo 서버/클라이언트 프로그램

udp\_client.py

```
import socket
from datetime import datetime

server_address = ('localhost', 6789)
max_size = 4096

print('Starting the client at', datetime.now())

client = socket.socket(socket.AF_INET, socket.SOCK_DGRAM)
client.sendto(b'^..^', server_address)

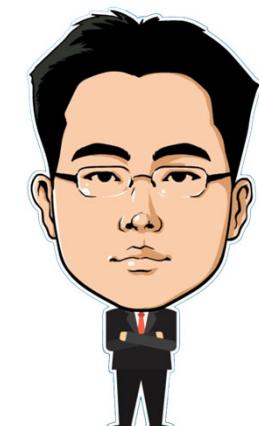
data, server = client.recvfrom(max_size)

print('At', datetime.now(), server, 'said', data)

client.close()
```

# 참고문헌

- [1] Bill Lubanovic, 최길우 역, “처음 시작하는 파이썬: 파이썬 패키지를 활용한 모던 컴퓨팅 입문”, 2판, 한빛미디어, 2020.
- [2] 남재윤, “파이썬 코딩 도장: 프로그래밍은 공부가 아닙니다. 연습입니다.”, 개정판, 길벗, 2023.
- [3] 박응용, “Do it! 점프 투 파이썬: 이미 200만명이 이 책으로 프로그래밍을 시작했다！”, 개정판, 이지스퍼블리싱, 2019.
- [4] “Python 3.12.2 documentation”, Python Software Foundation, 2024 of viewing the site, <https://docs.python.org/3/>.
- [5] “파이썬 코딩 도장”, 코딩 도장, 2024 of viewing the site, <https://dojang.io/course/view.php?id=7>.
- [6] “점프 투 파이썬”, WikiDocs, 2024 of viewing the site, <https://wikidocs.net/book/1>.



이 강의자료는 저작권법에 따라 보호받는 저작물이므로 무단 전제와 무단 복제를 금지하며,  
내용의 전부 또는 일부를 이용하려면 반드시 저작권자의 서면 동의를 받아야 합니다.

Copyright © Clickseo.com. All rights reserved.