

실습

실습에 앞서 해당 URL에 접속해주시고 깃허브 회원가입 및 로그인 부탁드립니다

접속URL:

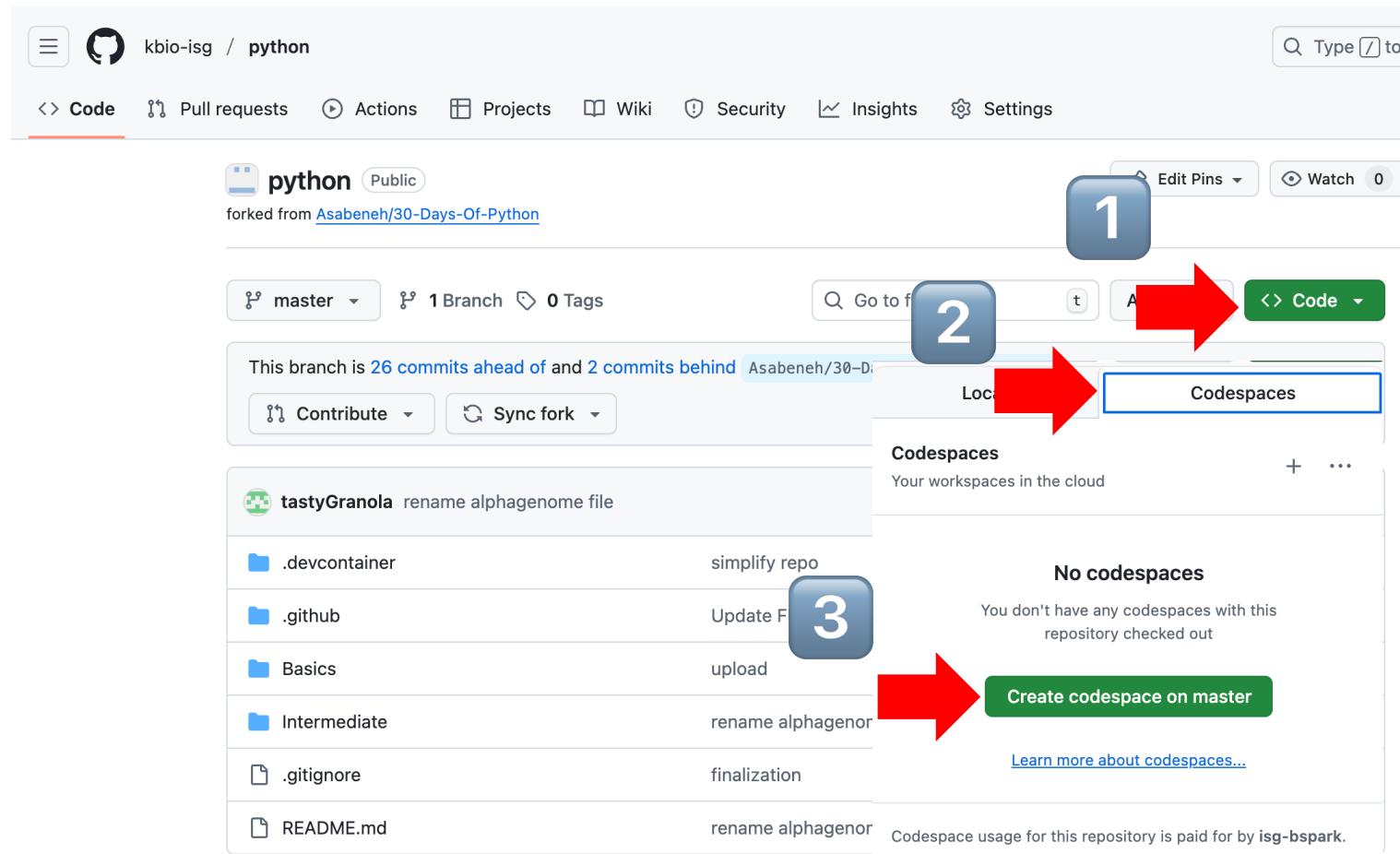
github.com/kbio-isg/python

QR:

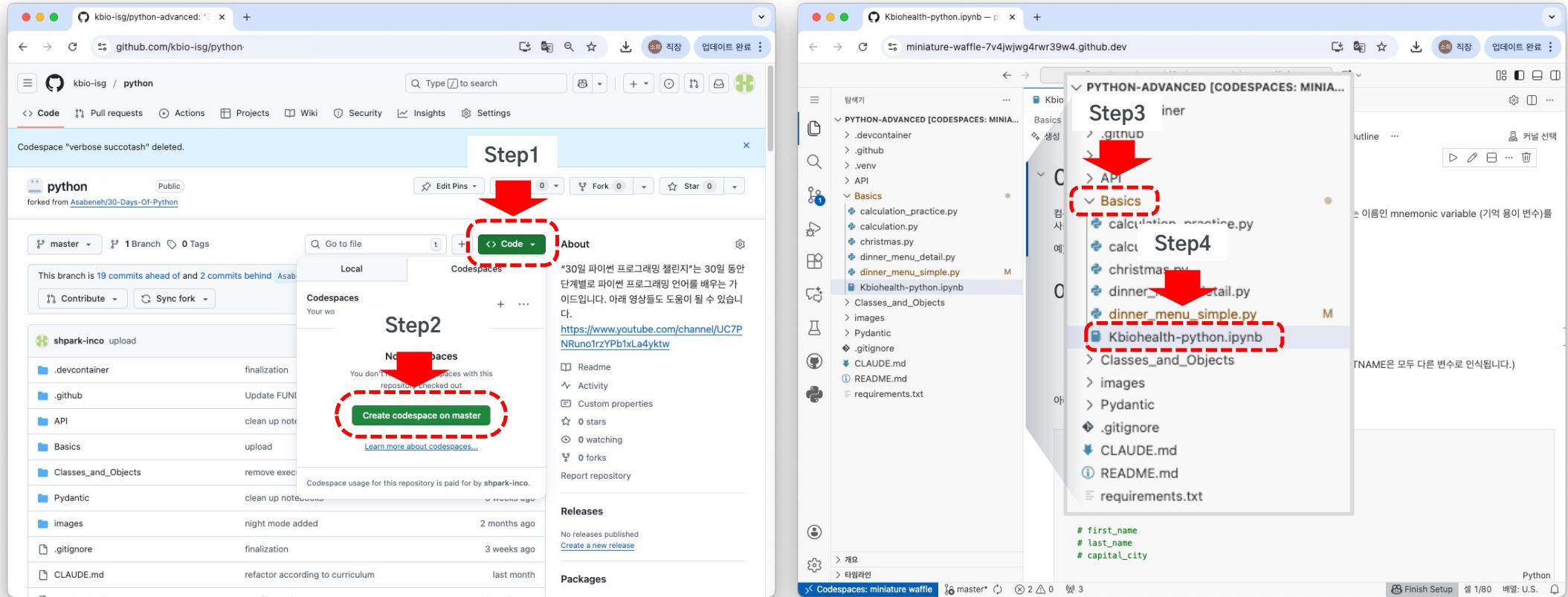


The screenshot shows the GitHub repository page for `kbio-isg / python`. The repository is public and was forked from `Asabeneh/30-Days-Of-Python`. The main branch is `master`, which has 26 commits ahead of and 2 commits behind the original repository. The commit history includes several changes such as renaming files, updating FUNDING.yml, and uploading content. The repository has 0 forks and 0 stars. On the right side, there is an `About` section with a link to a YouTube channel. The top navigation bar includes links for Platform, Solutions, Resources, Open Source, Enterprise, Pricing, and a search bar. The top right corner features a red box around the `Sign in` and `Sign up` buttons.

실습



[실습] github.com/kbio-isg/python 접속



KBIOHEALTH-Ver.202511-01

[AI · 바이오 융합 기본과정] Python 초·중급

2025-11-25 (주)인실리코젠

INSILICOGEN
www.insilicogen.com

본 문서의 모든 콘텐츠는 저작권법의 보호를 받는 저작물로 별도의 저작권 표시 또는 다른 출처를 명시한 경우를 제외하고는 (주)인실리코젠에 저작권이 있습니다.
저작권 표시 또는 기타 소유권 표시를 삭제해도 안되며, 당사와의 협의 또는 허락 없이 무단 복제, 변경, 배포를 금지합니다.
저작권 관련 문의사항이 있으시면 info@insilicogen.com으로 연락바랍니다.
© 2025 INSILICOGEN, INC. ALL RIGHTS RESERVED.

Python 초급

01 파이썬 개요

- 프로그래밍 언어
- 파이썬이란?
- 파이썬의 규칙

02 파이썬 자료형

- 기본 자료형
- 배열형 자료형
- 집합형 자료형

03 파이썬 기본 문법

- 제어문
- 반복문

04 모듈과 패키지

- 라이브러리
- 모듈/패키지

05 함수

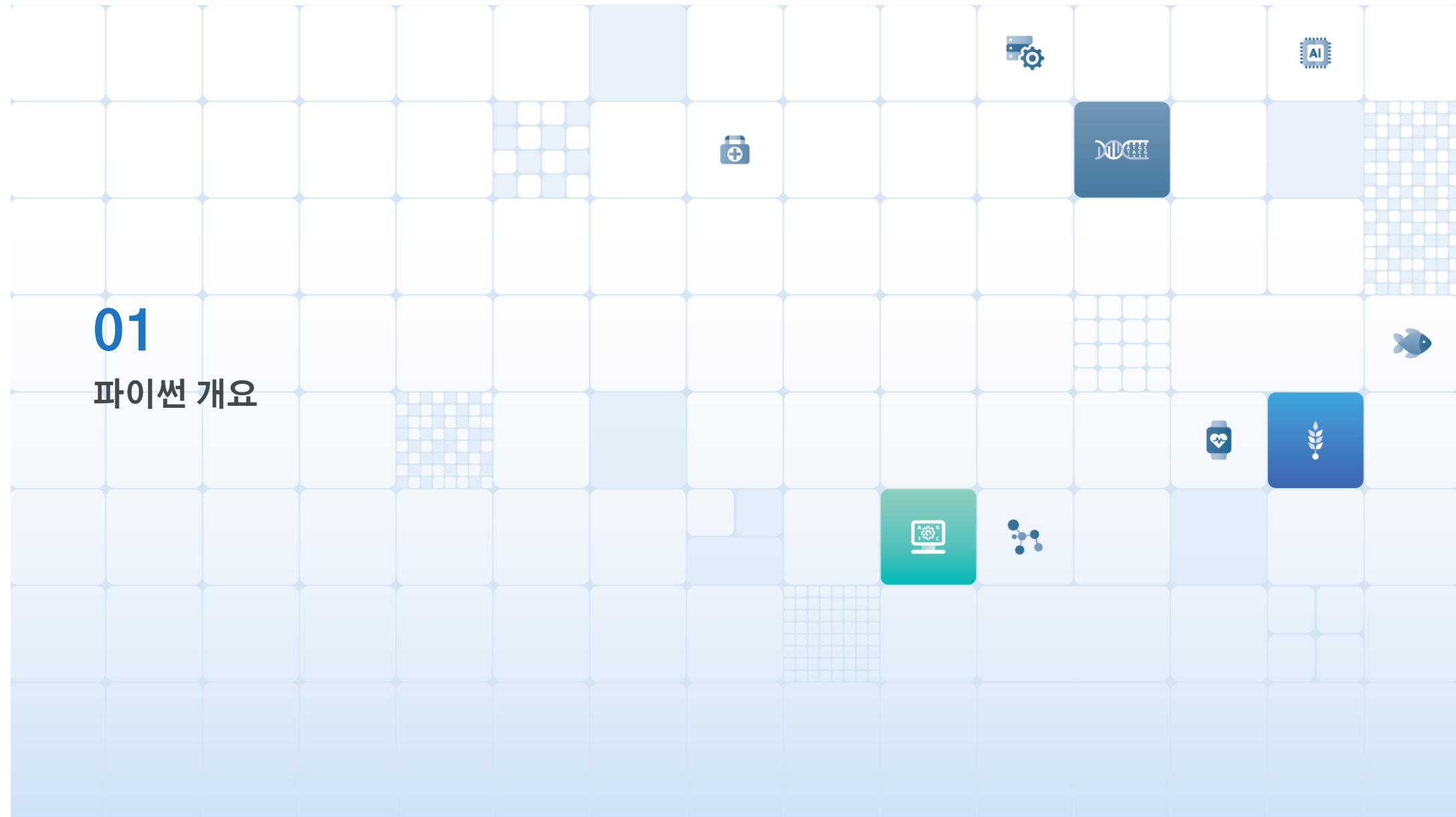
- 함수
- 인수와 변수

06 파일 다루기

- 파일 입출력
- 표준 입출력

01

파이썬 개요



프로그래밍 언어



프로그래밍 언어

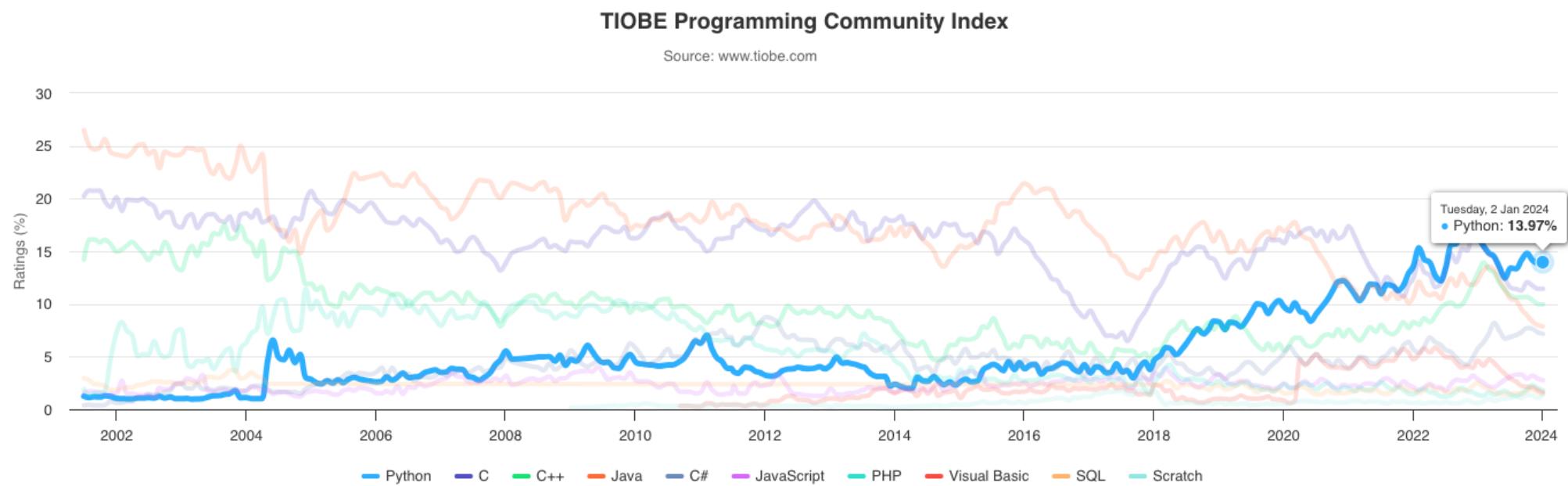


프로그래밍 언어 순위 통계 사이트

<https://www.tiobe.com/tiobe-index/>

Jan 2024	Jan 2023	Change	Programming Language	Ratings	Change
1	1		Python	13.97%	-2.39%
2	2		C	11.44%	-4.81%
3	3		C++	9.96%	-2.95%
4	4		Java	7.87%	-4.34%
5	5		C#	7.16%	+1.43%
6	7	▲	JavaScript	2.77%	-0.11%
7	10	▲	PHP	1.79%	+0.40%
8	6	▼	Visual Basic	1.60%	-3.04%
9	8	▼	SQL	1.46%	-1.04%

프로그래밍 언어



파이썬이란?

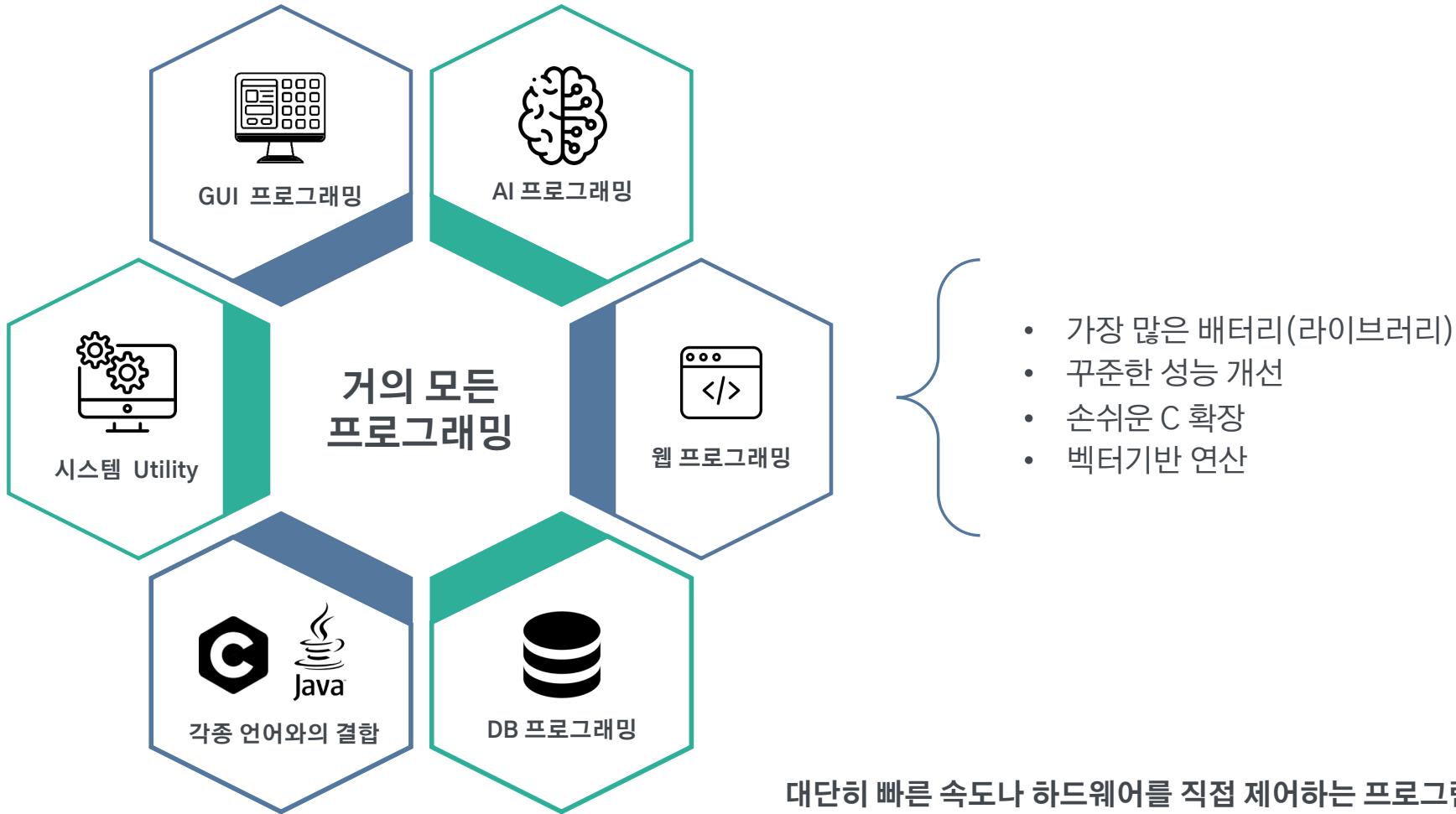


파이썬(Python)은 1991년 프로그래머인
귀도 반 로섬(Guido van Rossum)이 발표한 고급 프로그래밍 언어로,
플랫폼 독립적이며 인터프리터식, 객체지향적, 동적 타이핑, 함수형
언어이다.

파이썬이라는 이름은 귀도가 좋아하는 코미디
〈Monty Python's Flying Circus〉에서 따온 것이다.



파이썬이란?



파이썬이란?

1. **다중패러다임** - 좋은 것을 모두 모음 “**절차적**”, “**명령형**”, “**함수형**”, “**객체지향형**”
2. **플랫폼 독립** - 윈도우, 유닉스, 맥 등 다른 운영체제에서도 동일하게 사용
3. **인터프리터식** - 컴파일 필요 없음. 상호 반응형 환경 제공
4. **객체지향적** - 코드를 객체 단위로 구성하여 더욱 현실에 가까운 모델링. 모듈성과 재사용성을 높여 유지관리 용이.
5. **동적타이핑** - 변수의 데이터 타입을 미리 선언하지 않아도, 실행되면서 결정됨.
6. **함수형** - 입력과 출력으로 일관화 함.

→ 대규모 커뮤니티에서 컴퓨터 과학의 좋은 것들을 빠르게 도입하여 최신의 개념, 기술들을 반영함

[파이썬의 철학과 규칙] Zen of Python (총 19개)

>>> import this

Beautiful is better than ugly.

못생긴 것보다 아름다운 것이 낫다.

Special cases aren't special enough to break the rules. Although practicality beats purity.

규칙을 깨야 할 정도로 특별한 경우는 없다. 비록 실용성이 이상을 능가한다 하더라도.

Explicit is better than implicit.

암시적인 것보다 명시적인 것이 낫다.

Errors should never pass silently. Unless explicitly silenced.

오류는 결코 조용히 지나가지 않는다. 알고도 침묵하지 않는 한.

Simple is better than complex.

복잡한 것보다 단순한 것이 더 낫다.

In the face of ambiguity, refuse the temptation to guess..

모호함을 마주하고 추측하려는 유혹을 거절하라.

Complex is better than complicated.

난해한 것보다 복잡한 것이 더 낫다.

There should be one-- and preferably only one --obvious way to do it.

문제를 해결할 하나의 – 바람직하고 유일한 – 명백한 방법이 있을 것이다.

Readability counts.

프로그램은 읽기 쉬워야 한다.

If the implementation is hard to explain, it's a bad idea.

설명하기 어려운 구현이라면 좋은 아이디어가 아니다.

[파이썬의 철학과 규칙] Coding style

PEP 8 - Style Guide for Python Code

PEP:	8
Title:	Style Guide for Python Code
Author:	Guido van Rossum <guido at python.org>, Barry Warsaw <barry at python.org>, Nick Coghlan <ncoghlan at gmail.com>
Status:	Active
Type:	Process
Created:	05-Jul-2001
Post-History:	05-Jul-2001, 01-Aug-2013

Python 코딩 스타일 (PEP8)

- 일반변수는 lowercase_with_underscore
- 클래스는 CamelCase
- 들여쓰기는 스페이스 4칸

[파이썬의 철학과 규칙] Indentation (들여쓰기)

C언어

```
int factorial(int x)
{
    if(x == 0) {
        return 1;
    } else {
        return x * factorial(x - 1);
    }
}
```

Python

```
def factorial(x):
    if x == 0:
        return 1
    else:
        return x * factorial(x - 1)
```

{ } 대신 들여쓰기가 문법요소

[파이썬 실행] 파이썬 변수 (Python variable)

- 변수(Variable)란 어떤 값(value)을 저장한 공간을 말하며 변수 선언은 다음과 같이 할 수 있다.
- 변수 선언을 통해 각각의 값에 고유한 변수명(one, two)을 부여할 수 있다.

```
>>> one = 'hello!'
>>> two = 'hello! hello!'
>>> three = 3
```

- 아래와 같이 여러 변수를 한번에 선언할 수도 있다.

```
>>> one, two, three = 'hello!', 'hello! hello!', 3
>>> one
'hello!'
>> three
3
```

[파이썬 실행] 파이썬 변수 (Python variable)

파이썬의 변수는 데이터 타입을 가지고 있습니다.

```
>>> type(one)  
<class 'str'>    ----> 문자열
```

```
>>> type(three)  
<class 'int'>    ----> 정수형
```

▶ type() 함수를 이용하여 데이터 타입을 알 수 있습니다.

[파이썬 실행] 파이썬의 실행 방법

- 상호반응형 모드 (Interactive mode)

```
$ python3
Python 3.4.3 (default, Jul 13 2015, 12:18:23)
[GCC 4.2.1 Compatible Apple LLVM 6.1.0 (clang-602.0.53)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello world")
Hello world
```

- 프로그램 소스 코드를 .py 파일에 저장

```
$ vi mycode.py
print("Hello world")
$ python mycode.py
Hello world
```

[파이썬 실행] 파이썬의 실행 방법

- 상호반응형 모드 (Interactive mode)

```
$ python3
Python 3.4.3 (default, Jul 13 2015, 12:18:23)
[GCC 4.2.1 Compatible Apple LLVM 6.1.0 (clang-602.0.53)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello world")
Hello world
```

- 프로그램 소스 코드를 .py 파일에 저장

```
$ vi mycode.py
print("Hello world")
$ python mycode.py
Hello world
```

[실습] Jupyter notebook 기초

The screenshot shows a Jupyter Notebook interface with a sidebar on the left containing project files and a main workspace on the right.

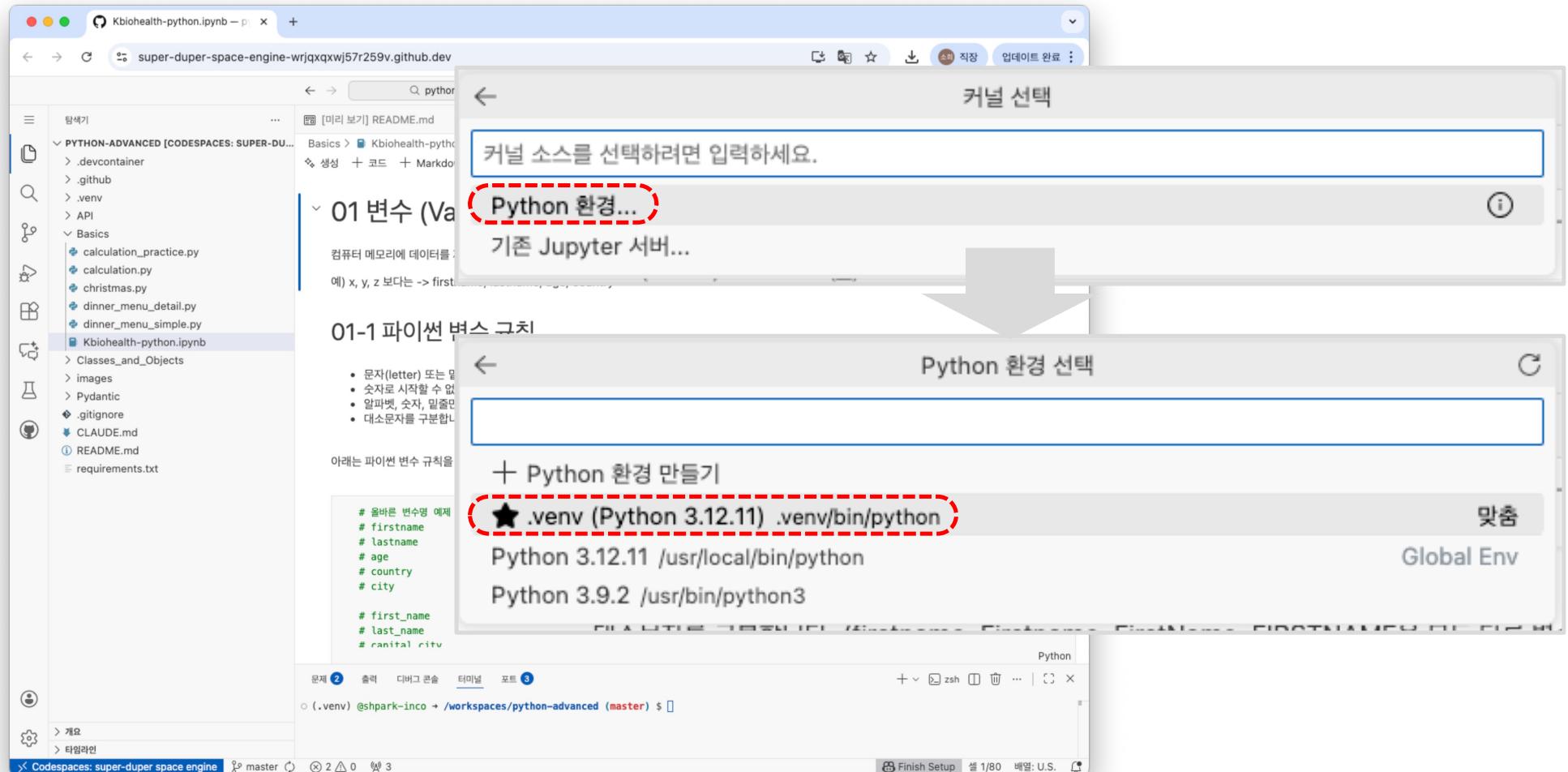
마크다운 셀 더블클릭시: A red dashed box highlights the title "01 변수 (Variables)" in a Markdown cell. A large grey arrow points from the right towards this cell, labeled "입력 창".

마크다운 셀: A red dashed box highlights the content of the Markdown cell, which includes a section titled "# 01-1 파이썬 변수 규칙" and a note about mnemonic variables.

코드 셀: A blue dashed box highlights the content of a Python code cell, which contains a dictionary definition for a person's information.

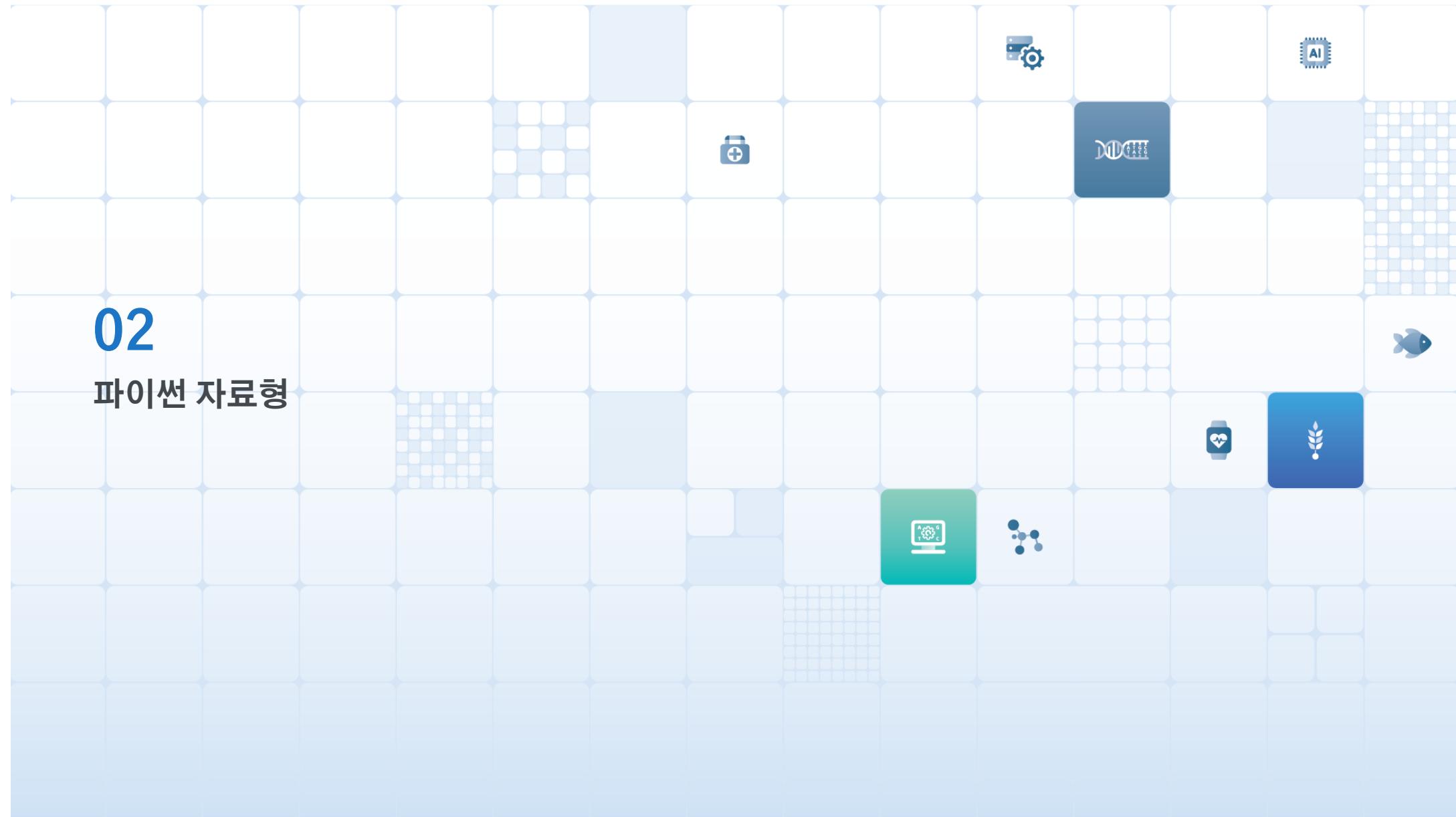
설명	옵션 종류
셀 실행	[Shift][Enter] [Ctrl][Enter]
셀 삽입	A 또는 B
셀 삭제	DD
마크다운 셀로 변경	M
코드 셀로 변경	Y

[실습] 실습 환경 세팅



02

파이썬 자료형



[기본 자료형]

구분	자료형	설명	비고
기본 자료형	Boolean (불린)	True 혹은 False로 정의되는 데이터 유형	a = True
	Numbers (숫자형)	정수, 실수, 분수, 복소수 등의 수로 표현되는 데이터 유형	a = 1.0
	String (문자열)	문자로 이루어진 데이터 유형	a = 'Hello'
배열형 자료형	List (목록)	숫자, 문자, 목록 등을 순서대로 나열한 것	a = ['my', 1, 'one']
	Tuple (튜플)	숫자, 문자, 목록 등을 순서대로 나열한 것이며 변경할 수 없음. (immutable)	a = ('my', 1, 'one')
집합형 자료형	Set (세트)	숫자, 문자, 목록 등을 순서없이 모아놓은 것	a = set([1, 2, 3])
	Dictionary (사전)	키-값의 형태로 순서없이 모아놓은 것	a = {'one': 1, 'two': 2}

[기본 자료형]

구분	자료형	설명	비고
기본 자료형	Boolean (불린)	True 혹은 False로 정의되는 데이터 유형	a = True
	Numbers (숫자형)	정수, 실수, 분수, 복소수 등의 수로 표현되는 데이터 유형	a = 1.0
	String (문자열)	문자로 이루어진 데이터 유형	a = 'Hello'
배열형 자료형	List (목록)	숫자, 문자, 목록 등을 순서대로 나열한 것	a = ['my', 1, 'one']
	Tuple (튜플)	숫자, 문자, 목록 등을 순서대로 나열한 것이며 변경할 수 없음. (immutable)	a = ('my', 1, 'one')
집합형 자료형	Set (세트)	숫자, 문자, 목록 등을 순서없이 모아놓은 것	a = set([1, 2, 3])
	Dictionary (사전)	키-값의 형태로 순서없이 모아놓은 것	a = {'one': 1, 'two': 2}

[기본 자료형] 숫자형 연산

- 정수(integer), 실수(real number), 복소수(complex)가 있으며, 연산자로 연산이 가능합니다.

```
# 정수  
>>> x = 10  
>>> print(x, "is of type", type(x))  
10 is of type <class 'int'>
```

```
# 실수  
>>> x = 4.0  
>>> print(x, "is of type", type(x))  
4.0 is of type <class 'float'>
```

```
# 복소수 (허수를 'j'로 표현)  
>>> x = 1+3j  
>>> print(x, "is of type", type(x))  
1+3j is of type <class 'complex'>  
>>> print(x, "is complex number?", isinstance(x,complex))  
(1+3j) is complex number? True
```

#	연산자	설명
1	+	덧셈
2	-	뺄셈
3	*	곱셈
4	**	지수
5	/	나눗셈
6	//	나눗셈의 몫
7	%	나눗셈의 나머지

isinstance(value, type) 함수로
특정 값이 특정 타입인지 확인 가능

[기본 자료형] 숫자형 형변환 및 진법 변환

- 정수형과 실수형 간의 형변환을 할 수 있습니다.
- 2진수, 8진수, 16진수로 변환을 할 수 있습니다.

```
# 실수 → 정수로 변환
```

```
>>> int(2.9999)
```

```
2
```

```
>>> int(-2.9999)
```

```
-2
```

```
# 정수 → 실수로 변환
```

```
>>> float(100)
```

```
100.0
```

```
>>> float(-100)
```

```
-100.0
```

```
# 2진수로 변환 (접두어 '0b')
```

```
>>> x = 200
```

```
>>> bin(x)
```

```
'0b11001000'
```

```
# 8진수로 변환 (접두어 '0o')
```

```
>>> oct(x)
```

```
'0o310'
```

```
# 16진수로 변환 (접두어 '0x')
```

```
>>> hex(x)
```

```
'0xc8'
```

```
# 2진수 → 10진수
```

```
>>> x = 0b11001000
```

```
>>> x
```

```
200
```

```
# 8진수 → 10진수
```

```
>>> x = 0o310
```

```
>>> x
```

```
200
```

```
# 16진수 → 10진수
```

```
>>> x = 0xc8
```

```
>>> x
```

```
200
```

```
# 복소수는 정수, 실수로 불가변환할 수 없음.
```

```
>>> int(1+5j)
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
TypeError: can't convert complex to int
```

```
>>> float(1+5j)
```

```
Traceback (most recent call last):
```

```
  File "<stdin>", line 1, in <module>
```

```
TypeError: can't convert complex to float
```

[기본 자료형] 문자열 연산

- 파이썬 문자열은 유니코드 문자로 이루어진 데이터 유형입니다.
- 따옴표(' or ")를 이용하여 생성할 수 있으며, 여러줄로 이루어진 문자열은 따옴표 3개로('' or ''') 생성할 수 있습니다..
- 문자열은 연산이 가능합니다.

```
# 문자열 선언  
>>> 'Hello World'  
'Hello World'  
>>> "Hello"  
'Hello'
```

```
# 여러줄의 문자열 선언  
>>> """This is  
... python string"""  
'This is\npython string'
```

```
# 따옴표가 포함된 문자열  
>>> 'It\'s good!' # 역슬래시(\) 사용  
"It's good!"  
>>> "It's good!"  
"It's good!"
```

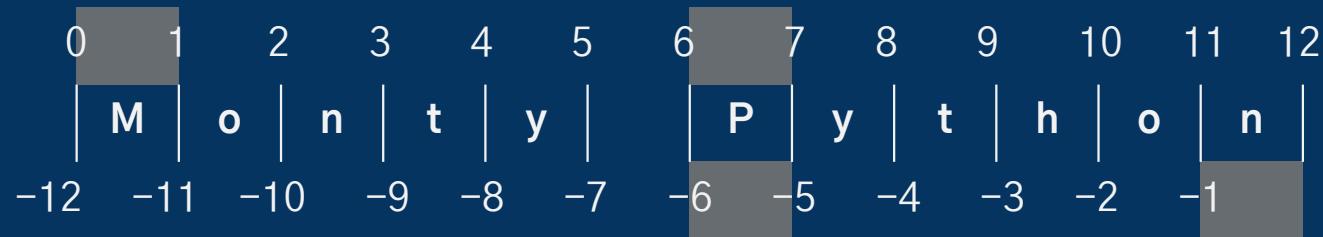
#	연산자	설명
1	+	문자열 합치기
2	*	문자열 곱하기

```
# 문자열 합치기  
>>> a = 'Beautiful is '  
>>> b = 'better than ugly.'  
>>> a + b  
'Beautiful is better than ugly.'
```

```
# 문자열 곱하기  
>>> a * 3  
'Beautiful is Beautiful is Beautiful is '
```

[기본 자료형] 문자열 인덱싱과 슬라이싱 (1/2)

- 파이썬의 문자열은 각각의 문자마다 인덱스를 가지고 있습니다.
- 인덱스는 0부터 시작합니다.

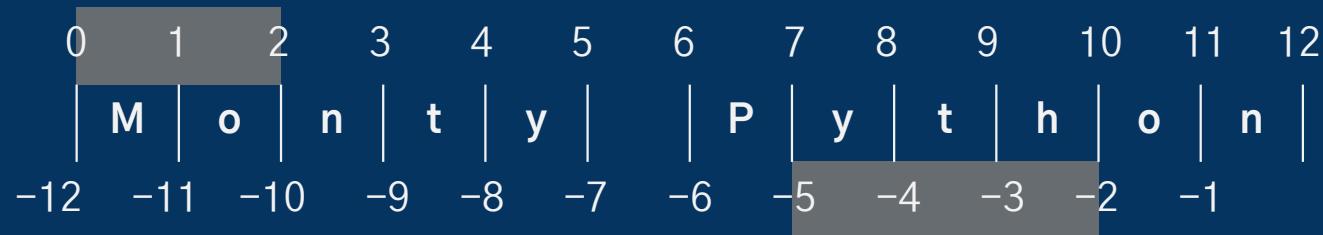


```
>>> a = "Monty Python"  
>>> a[0]  
'M'  
>>> a[-1]  
'n'  
>>> a[6]  
'P'  
>>> a[-6]  
'P'
```

```
# 문자열의 내용은 바꿀 수 없다.  
>>> a[6] = 'X'  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: 'str' object does not support item assignment
```

[기본 자료형] 문자열 인덱싱과 슬라이싱 (2/2)

- 파이썬의 슬라이싱을 이용하면 문자열의 원하는 부분만 얻을 수 있습니다.
- [시작 인덱스:끝 인덱스]와 같은 형태로 사용합니다.



```
>>> a = 'Monty Python'  
>>> a[0:2]  
'Mo'  
  
>>> a[-5:-2]  
'yth'  
  
>>> a[-2:] # 끝 값을 생략하면 마지막 글자까지 나옴.  
'on'  
  
>>> a[:5] # 처음 값을 생략하면 첫 글자부터 나옴.  
'Monty'
```

```
>>> a[:]  
'Monty Python'  
  
>>> a[::-1] # 역순으로 나옴.  
'nohtyP ytnoM'  
  
>>> a[0:10:2] # 0~10까지의 문자를 2문자씩 건너서 가져옴.  
'MnyPt'
```

[기본 자료형] 문자열 관련 함수 및 메소드

- 파이썬은 문자열을 다룰 수 있는 다양한 메소드들이 있습니다.

#	기능	설명
1	.find('x')	문자열에서 원하는 문자 x가 위치한 인덱스 반환
2	.upper()	문자열을 대문자로 반환
3	.lower()	문자열의 소문자를 반환
4	.count('x')	문자열에서 x의 카운트 반환
5	.strip()	문자열에서 양끝의 공백, 탭 등 제거
6	.lstrip()	문자열 왼쪽끝의 공백, 탭 등 제거
7	.rstrip()	문자열 오른쪽끝의 공백, 탭 등 제거
8	.replace('a', 'b')	문자열에서 'a'를 'b'로 치환
9	.title()	단어 앞글자만 대문자로 변환
10	len()	글자수 반환 (공백포함)

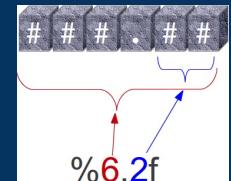
```
>>> a = 'Monty Python'  
>>> a.find('P')  
>>> a.upper()  
>>> a.lower()  
>>> a.count('X')  
>>> a.count('Py')  
>>> a.strip()  
>>> a.lstrip()  
>>> a.rstrip()  
>>> a.replace('Monty ', 'I love ')  
>>> lower_a = a.lower()  
>>> lower_a.title()  
>>> len(a)  
>>> len(a.strip())
```

[기본 자료형] 문자열 포맷팅 (1/2) – old way

- 파이썬의 문자열에 변수를 대입하고 싶을 때 "%" 구문을 이용하여 문자열 포맷팅을 사용할 수 있습니다.

```
print("%s is better than %s." % ('Simple', 'complex'))
```

```
print("%4.2f + %f = %d" % (1.1, 2.2, 1.1 + 2.2))
```



```
>>> print("%s is better than %s." % ('Simple', 'complex'))  
Simple is better than complex.
```

```
>>> print("%s is better than %s." % ('Flat', 'nested'))  
Flat is better than nested.
```

```
>>> print("%4.2f + %f = %d" % (1.1, 2.2, 1.1 + 2.2))  
1.10 + 2.200000 = 3
```

[기본 자료형] 문자열 포맷팅 (2/2) – new way

- 파이썬의 문자열에 변수를 대입하고 싶을 때 “format” 을 이용하여 문자열 포맷팅을 사용할 수 있습니다.

```
print("{0} is better than {1}.".format('Simple', 'complex'))
```

```
print("{0:4.2f} + {1:6.5f} = {2:1.0f}".format(1.1, 2.2, 1.1 + 2.2))
```

```
>>> print("{} is better than {}".format('Simple', 'complex'))
```

```
Simple is better than complex.
```

```
>>> print("{0:4.2f} + {1:6.5f} = {2:1.0f}".format(1.1, 2.2, 1.1 + 2.2))
```

```
1.10 + 2.20000 = 3
```

[기본 자료형] 문자열 포맷팅 (2/2) – new way

일반적인 포맷팅 1

```
>>> print("{} is better than {}".format('Simple', 'complex'))  
Simple is better than complex.
```

순서를 이용한 포맷팅 2

```
>>> print("{0} is better than {1}".format('Simple', 'complex'))  
Simple is better than complex.
```

필드명을 이용한 포맷팅 3

```
>>> print("{a} is better than {b}".format(a='Simple', b='complex'))  
Simple is better than complex.
```

Python 3.6 new function, literal string interpolation

```
>>> name = "Yong"  
>>> score = 95  
>>> f "{name}'s score is {score}. "  
"Yong's score is 95."
```

[배열형 자료형]

구분	자료형	설명	비고
	Boolean (불린)	True 혹은 False로 정의되는 데이터 유형	a = True
기본 자료형	Numbers (숫자형)	정수, 실수, 분수, 복소수 등의 수로 표현되는 데이터 유형	a = 1.0
	String (문자열)	문자로 이루어진 데이터 유형	a = 'Hello'
	List (목록)	숫자, 문자, 목록 등을 순서대로 나열한 것	a = ['my', 1, 'one']
배열형 자료형	Tuple (튜플)	숫자, 문자, 목록 등을 순서대로 나열한 것이며 변경할 수 없음. (immutable)	a = ('my', 1, 'one')
	Set (세트)	숫자, 문자, 목록 등을 순서없이 모아놓은 것	a = set([1, 2, 3])
집합형 자료형	Dictionary (사전)	키-값의 형태로 순서없이 모아놓은 것	a = {'one': 1, 'two': 2}

[배열형 자료형] 리스트 연산

- 리스트는 숫자, 문자, 리스트, 튜플 등을 순서대로 나열한 자료형입니다.
- 어떤 자료형이든 리스트의 원소가 될 수 있고, 여러 자료형으로 이루어진 리스트도 구성할 수 있습니다.
- 리스트로 묶고자하는 요소들을 []로 감싸주고 ','로 구분합니다.

```
# 정수로 구성된 리스트  
>>> a = [1, 2, 3, 4, 10]  
  
# 문자열로 구성된 리스트  
>>> a = ['apple', 'banana', 'melon']  
  
# 다양한 자료형으로 구성된 리스트  
>>> a = [1, 'apple', 0.332, 'banana']  
  
# 리스트를 요소로 가진 리스트  
>>> a = [['apple', 'banana', 'melon'], [1, 3.2, 4, 6]]  
>>> type(a)  
<class 'list'>
```

```
# 리스트 끝에 '*' 연산은 안됨.  
>>> a * b  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: can't multiply sequence by non-int of type 'list'
```

#	연산자	설명	예제
1	+	리스트 합치기	>>> [1, 2, 3] + [4, 5, 6] [1, 2, 3, 4, 5, 6]
2	*	리스트 반복하기	>>> [1, 2, 3] * 3 [1, 2, 3, 1, 2, 3, 1, 2, 3]

```
# 리스트 합치기  
>>> a = ['A', 'B', 'C']  
>>> b = ['G', 'T']  
>>> a + b  
['A', 'B', 'C', 'G', 'T']  
  
# 리스트 반복하기  
>>> a * 2  
['A', 'B', 'C', 'A', 'B', 'C']  
>>> (a * 2) + b  
['A', 'B', 'C', 'A', 'B', 'C', 'G', 'T']
```

[배열형 자료형] 리스트 인덱싱과 슬라이싱 (1/2)

- 리스트도 파이썬 문자열과 마찬가지로 인덱싱을 사용할 수 있습니다.
- 인덱스는 0부터 시작합니다.

	apple	banana	melon	orange
index	[0]	[1]	[2]	[3]
	[-4]	[-3]	[-2]	[-1]

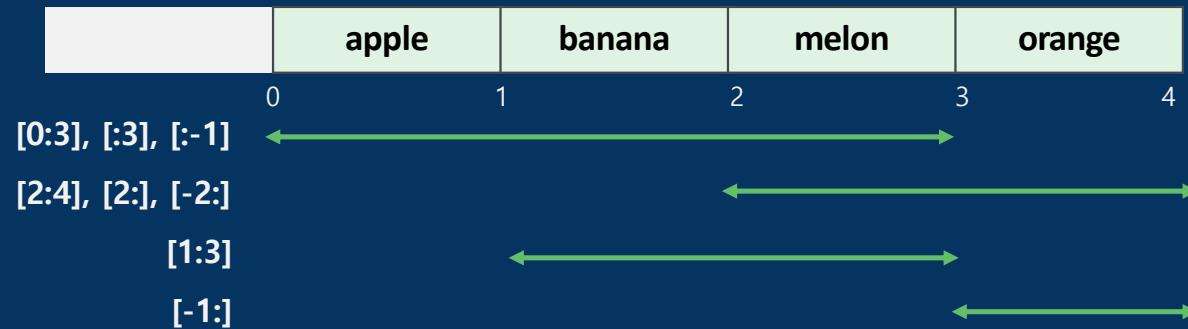
```
>>> a = ['apple', 'banana', 'melon', 'orange']
>>> a[3]
'orange'
>>> a[0]
'apple'
>>> a[-1]
'orange'
>>> a[-4]
'apple'
```

```
# 인덱스로 리스트의 요소 바꾸기
>>> a[0] = 'pineapple'
>>> a
['pineapple', 'banana', 'melon', 'orange']

# 요소 삭제
>>> del a[1]
>>> a
['pineapple', 'melon', 'orange']
```

[배열형 자료형] 리스트 인덱싱과 슬라이싱 (2/2)

- 리스트도 슬라이싱을 사용할 수 있습니다.



```
>>> a = ['apple', 'banana', 'melon', 'orange']
```

```
>>> a[:3]
['apple', 'banana', 'melon']
```

```
>>> a[:-1]
['apple', 'banana', 'melon']
```

```
>>> a[2:4]
['melon', 'orange']
```

```
>>> a[2:]
['melon', 'orange']
```

```
>>> a[1:3]
['banana', 'melon']
# 슬라이싱으로 리스트의 요소 바꾸기
```

```
>>> a[2] = ['APPLE', 'BANANA']
>>> a
['APPLE', 'BANANA', 'melon', 'orange']
```

[배열형 자료형] 리스트 관련 함수 및 메소드 (1/3)

- 파이썬은 리스트를 다룰 수 있는 다양한 메소드들을 제공합니다.

.append('val')

리스트 끝에 'val'을 추가

```
>>> a = ['A', 'B', 'C']
>>> a.append('D')
>>> a
['A', 'B', 'C', 'D']
```

.index('val')

리스트에서 'val'의 인덱스 반환

```
>>> a.index('Good')
5
>>> a.index('A')
1
```

.insert(index, 'val')

주어진 index위치에 'val'을 추가

```
>>> a.insert(0, 'val')
>>> a
['val', 'A', 'B', 'C', 'D']
```

.remove('val')

리스트에서 'val' 삭제

```
>>> a.remove('val')
>>> a
['A', 'B', 'C', 'D', 'Good', 'Bad']
```

.extend([list])

[list]를 리스트 끝에 추가 (리스트 확장)

```
>>> a.extend(['Good', 'Bad'])
>>> a
['val', 'A', 'B', 'C', 'D', 'Good', 'Bad']
```

.sort()

리스트의 요소를 정렬
(요소들이 같은 자료형 일 경우에만 가능)

```
>>> a.sort()
>>> a
['A', 'B', 'Bad', 'C', 'D', 'Good']
```

[배열형 자료형] 리스트 관련 함수 및 메소드 (2/3)

- 파이썬은 리스트를 다룰 수 있는 다양한 메소드들을 제공합니다.

.reverse()

리스트의 요소 순서 반대로

```
>>> a = [5, 'A', 14, 4, 2]
>>> a.reverse()
>>> a
[2, 4, 14, 'A', 5]
```

.pop(index)

리스트에서 index에 해당하는 요소를 반환하고 리스트에서 삭제함.

```
>>> a = [5, 'A', 14, 4, 2]
>>> a.pop(1)
'A'
>>> a = [5, 14, 4, 2]
#인덱스를 주지 않으면 가장 끝 요소가 반환됨.
>>> a.pop()
2
>>> a
[5, 14, 4]
```

.count('val')

리스트에서 'val'의 개수를 반환

```
>>> a = [1, 1, 2, 2, 2, 3, 4, 3, 5]
>>> a.count(1)
2
```

len(list)

리스트 요소 개수 반환

```
>>> a = [1, 1, 2, 2, 2, 3, 4, 3, 5]
>>> len(a)
9
```

[배열형 자료형] 리스트 관련 함수 및 메소드 (3/3)

- 파이썬은 리스트를 다룰 수 있는 다양한 메소드들을 제공합니다.

max(list), min(list)

리스트에서 가장 큰, 혹은 작은 값 반환

```
>>> a = [10, 2, 100, 99, 32]  
>>> max(a)  
100  
>>> min(a)  
2
```

문자열로 이루어진 리스트는 알파벳 순서로 판단

```
>>> b = ['B', 'C', 'FS', 'S']  
>>> max(b)  
'S'  
>>> min(b)  
'B'
```

list(tuple)

튜플을 리스트로 변환

```
>>> a = (1, 2, 3)  
>>> list(a)  
[1, 2, 3]
```

s.split(separator)

문자열s를 separator기준으로 나눠서 리스트로 반환

```
>>> a = "Banana Apple Strawberry Peach"  
  
# 공백(' ')기준으로 나눔.  
>>> a.split(' ')  
['Banana', 'Apple', 'Strawberry', 'Peach']
```

[배열형 자료형] 튜플

- 튜플은 리스트와 비슷하지만 요소의 변경이 불가능하다는 점이 다릅니다.
- 리스트와 마찬가지로 어떤 자료형이든 튜플의 원소가 될 수 있고 여러 자료형으로 이루어진 튜플도 구성할 수 있습니다.
- 튜플로 묶고자 하는 요소들을 ()로 감싸주고 ','로 구분합니다.

```
# 빈 튜플 생성
>>> a = ()

# 정수로 구성된 튜플
>>> a = (1, 2, 3, 4, 10)

# 문자열로 구성된 튜플
>>> a = ('apple', 'banana', 'melon')

# 다양한 자료형으로 구성된 튜플
>>> a = (1, 'apple', 0.332, 'banana')

# 튜플과 리스트를 요소로 가진 튜플
>>> a = ([('apple', 'banana', 'melon'), (1, 3.2, 4, 6))

>>> type(a)
<class 'tuple'>
```

```
# 요소가 하나인 튜플의 자료형은 튜플이 아님.
>>> a = ('A')
>>> type(a)
<class 'str'>
```

```
# 요소가 하나인 튜플을 만들기 위해서는 ','를 붙여야 함.
>>> b = ('A', )
>>> type(b)
<class 'tuple'>
```

[배열형 자료형] 튜플 연산

- 튜플 연산도 리스트 연산과 동일합니다.

#	연산자	설명	예제
1	+	튜플 합치기	>>> (1, 2, 3) + (4, 5, 6) (1, 2, 3, 4, 5, 6)
2	*	튜플 반복하기	>>> (1, 2, 3) * 3 (1, 2, 3, 1, 2, 3, 1, 2, 3)

```
# 튜플 끼리 '*' 연산은 안됨.  
>>> a * b  
Traceback (most recent call last):  
  File "<stdin>", line 1, in <module>  
TypeError: can't multiply sequence by non-int of type 'tuple'
```

```
# 튜플 합치기  
>>> a = ('A', 'B', 'C', 'D')  
>>> b = ('G', 'T')  
>>> a + b  
('A', 'B', 'C', 'D', 'G', 'T')  
>>> a + b + a  
('A', 'B', 'C', 'D', 'G', 'T', 'A', 'B', 'C', 'D')
```

```
# 튜플 반복하기  
>>> a * 2  
('A', 'B', 'C', 'D', 'A', 'B', 'C', 'D')  
>>> (a * 2) + b  
('A', 'B', 'C', 'D', 'A', 'B', 'C', 'D', 'G', 'T')
```

[배열형 자료형] 튜플 인덱싱과 슬라이싱 (1/2)

- 튜플도 파이썬 문자열, 리스트와 마찬가지로 인덱싱을 사용할 수 있습니다.
- 인덱스는 0부터 시작하며 사용방법은 리스트 인덱싱과 동일합니다.

	apple	banana	Melon	orange
index	[0]	[1]	[2]	[3]
	[-4]	[-3]	[-2]	[-1]

```
>>> a = ('apple', 'banana', 'melon', 'orange')

>>> a[3]
'orange'

>>> a[0]
'apple'

>>> a[-1]
'orange'

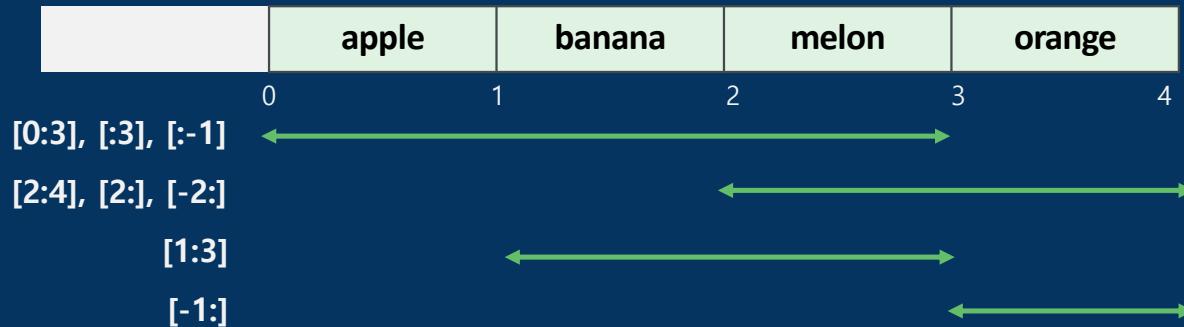
>>> a[-4]
'apple'
```

```
# 요소변경, 삭제는 불가함.
>>> a[0] = 'pineapple'
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment

>>> del a[1]
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: 'tuple' object doesn't support item deletion
```

[배열형 자료형] 튜플 인덱싱과 슬라이싱 (2/2)

- 튜플의 슬라이싱도 리스트와 동일한 방법으로 사용할 수 있습니다.



```
>>> a = ('apple', 'banana', 'melon', 'orange')
```

```
>>> a[:3]
('apple', 'banana', 'melon')
```

```
>>> a[:-1]
('apple', 'banana', 'melon')
```

```
>>> a[2:4]
('melon', 'orange')
```

```
>>> a[2:]
('melon', 'orange')
```

```
>>> a[1:3]
('banana', 'melon')
```

```
>>> a[:]
('apple', 'banana', 'melon', 'orange')
```

[배열형 자료형] 튜플 관련 함수

- 파이썬은 튜플을 다룰 수 있는 다양한 함수들을 제공합니다.
- 단, 값 변경과 관련된 `.append()`, `.insert()` 등은 지원하지 않습니다.

`len(tuple)`

튜플의 요소 수 세기

```
>>> a = ('A', 'B', 'C')  
>>> len(a)  
3
```

`max(tuple), min(tuple)`

튜플에서 가장 큰, 혹은 작은 값 반환

```
>>> a = (10, 2, 100, 99, 32)  
>>> max(a)  
100  
>>> min(a)  
2  
# 문자열로 이루어진 튜플은 알파벳 순서로 판단  
>>> b = ('B', 'C', 'FS', 'S')  
>>> max(b)  
'S'  
>>> min(b)  
'B'
```

`tuple(list)`

리스트를 튜플로 변환

```
>>> a = [1, 2, 3]  
>>> tuple(a)  
(1, 2, 3)
```

`.count('val')`

목록에서 'val'의 개수를 반환

```
>>> a = (1, 1, 2, 2, 2, 3, 4, 3, 5)  
>>> a.count(1)  
2
```

`.index('val')`

목록에서 'val'의 인덱스 반환. 찾는 값이 여러 개일 경우 첫번째 인덱스를 반환

```
>>> a = ('A', 'A', 'B', 'C', 'B')  
>>> a.index('B')  
2
```

[집합형 자료형]

구분	자료형	설명	비고
기본 자료형	Boolean (불린)	True 혹은 False로 정의되는 데이터 유형	a = True
	Numbers (숫자형)	정수, 실수, 분수, 복소수 등의 수로 표현되는 데이터 유형	a = 1.0
	String (문자열)	문자로 이루어진 데이터 유형	a = 'Hello'
배열형 자료형	List (목록)	숫자, 문자, 목록 등을 순서대로 나열한 것	a = ['my', 1, 'one']
	Tuple (튜플)	숫자, 문자, 목록 등을 순서대로 나열한 것이며 변경할 수 없음. (immutable)	a = ('my', 1, 'one')
집합형 자료형	Set (세트)	숫자, 문자, 목록 등을 순서없이 모아놓은 것	a = set([1, 2, 3])
	Dictionary (사전)	키-값의 형태로 순서없이 모아놓은 것	a = {'one': 1, 'two': 2}

[집합형 자료형] 세트

- 세트는 집합유형의 데이터를 쉽게 다룰 수 있도록 만들어진 자료형입니다.
- 리스트(혹은 튜플)을 set()로 감싸서 생성할 수 있습니다.
- 세트는 리스트, 튜플과 달리 순서가 없으며 중복을 허용하지 않습니다.

```
# 빈 세트 생성
>>> a = set()
>>> a
set()

# 정수로 이루어진 세트 생성
>>> a = set([1, 2, 3, 4])
>>> a
{1, 2, 3, 4}

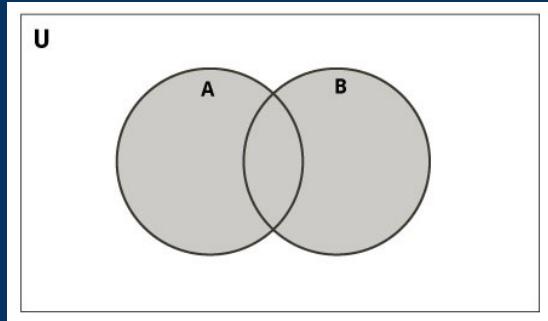
# 혼합된 자료형으로 이루어진 세트 생성
>>> a = set(['Green', 'Blue', 140, 50.3])
>>> a
{'Blue', 50.3, 140, 'Green'}
```

```
# 세트는 중복을 허용하지 않음.
>>> a = set([4, 5, 1, 2, 1, 3, 5, 6, 4, 10])
>>> a
{1, 2, 3, 4, 5, 6, 10}
>>> type(a)
<class 'set'>

# set() 대신 {}를 사용해도 됨.
>>> x = {1, 2, 3, 4, 5}
>>> type(x)
<class 'set'>
```

[집합형 자료형] 세트 집합연산

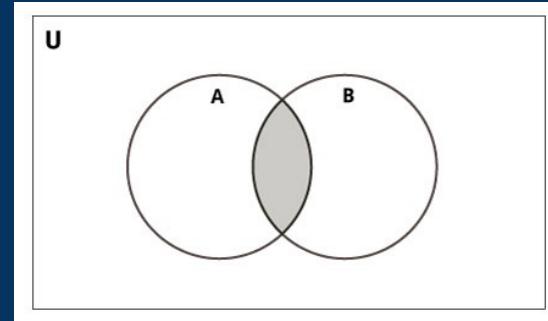
- 합집합 (Union)



'|' 혹은 union() 사용

```
>>> x = {1, 2, 3, 4, 5}  
>>> y = {1, 2, 6, 7, 8}  
>>> x | y  
{1, 2, 3, 4, 5, 6, 7, 8}  
>>> x.union(y)  
{1, 2, 3, 4, 5, 6, 7, 8}  
>>> y.union(x)  
{1, 2, 3, 4, 5, 6, 7, 8}
```

- 교집합 (Intersection)

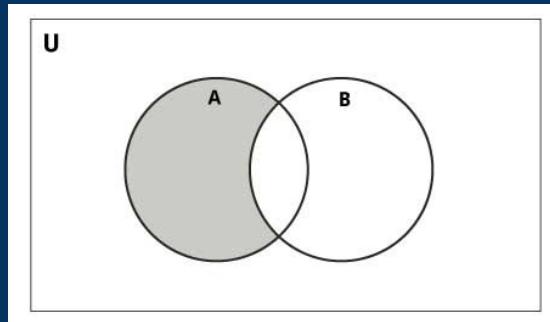


'&' 혹은 intersection() 사용

```
>>> x = {1, 2, 3, 4, 5}  
>>> y = {1, 2, 6, 7, 8}  
>>> x & y  
{1, 2}  
>>> x.intersection(y)  
{1, 2}  
>>> y.intersection(x)  
{1, 2}
```

[집합형 자료형] 세트 집합연산

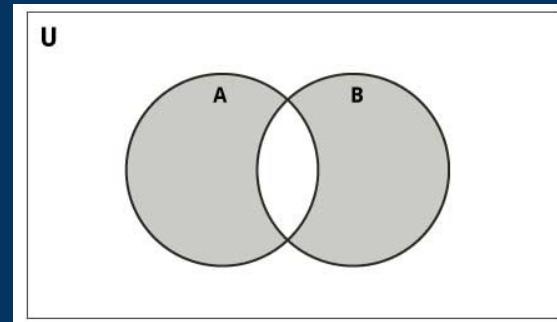
- 차집합 (Difference)



'-' 혹은 `difference()` 사용

```
>>> x = {1, 2, 3, 4, 5}  
>>> y = {1, 2, 6, 7, 8}  
>>> x - y  
{3, 4, 5}  
>>> y - x  
{6, 7, 8}  
>>> x.difference(y)  
{3, 4, 5}  
>>> y.difference(x)  
{6, 7, 8}
```

- 여집합 (Symmetric difference)



'^' 혹은 `symmetric_difference()` 사용

```
>>> x = {1, 2, 3, 4, 5}  
>>> y = {1, 2, 6, 7, 8}  
>>> x ^ y  
{3, 4, 5, 6, 7, 8}  
>>> x.symmetric_difference(y)  
{3, 4, 5, 6, 7, 8}  
>>> y.symmetric_difference(x)  
{3, 4, 5, 6, 7, 8}
```

<https://www.programiz.com/python-programming/set>

[집합형 자료형] 세트 접근 (1/2)

- 세트의 값을 추가하는 방법은 .add()와 .update()가 있습니다.

.add(val)

세트에 value를 추가

```
>>> a = set([ 4, 5, 1, 2, 1, 3, 5, 6, 4, 10])
>>> a
{1, 2, 3, 4, 5, 6, 10}
>>> a.add('A')
>>> a
{1, 2, 3, 4, 5, 6, 10, 'A'}
```

.update(list|tuple|set)

세트에 여러 value들을 추가

```
>>> a = set([ 4, 5, 1, 2, 1, 3, 5, 6, 4, 10])
>>> a.update(['A', 'B', 'C'])
>>> a
{1, 2, 3, 4, 5, 6, 'C', 'B', 10, 'A'}
# 여러 리스트와 세트를 동시에 추가할 수 있음.
>>> b = set(['new', 'items', 'here'])
>>> c = [1, 100, 1000, 'new']
>>> a.update(b, c)
>>>
>>> a
{'items', 1, 'C', 3, 4, 5, 6, 2, 'here', 'B', 10, 100, 1000, 'new', 'A'}
```

[집합형 자료형] 세트 접근 (2/2)

- 세트의 값을 삭제하는 방법은 다음과 같습니다.

.remove(val)

```
>>> a = set([ 4, 5, 1, 2, 1, 3, 5, 6, 4, 10])
>>> a.remove(1)
>>> a
{2, 3, 4, 5, 6, 10}
# 없는 값을 삭제하면 KeyError 발생
>>> a.remove('A')
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
KeyError: 'A'
```

세트에서 특정 value를 삭제

.discard(val)

```
>>> a = set([ 4, 5, 1, 2, 1, 3, 5, 6, 4, 10])
>>> a.remove(1)
>>> a
{2, 3, 4, 5, 6, 10}
# 없는 값을 삭제해도 KeyError 발생 안함.
>>> a.discard('A')
```

세트에서 특정 value를 삭제

.clear()

```
>>> a = set([ 4, 5, 1, 2, 1, 3, 5, 6, 4, 10])
>>> a.clear()
>>> a
set()
```

세트를 비움.

.pop()

```
>>> a = set(['Banana', 'Apple', 'Green', 'Blue'])
>>> a
{'Blue', 'Banana', 'Apple', 'Green'}
>>> a.pop()
'Blue'
>>> a.pop()
'Banana'
>>> a
{'Apple', 'Green'}
```

세트의 요소를 임의의 순서로 하나씩 리턴하고
세트에서 제거함. 더이상 제거할 값이
없으면 KeyError 발생

[집합형 자료형] 세트 관련 함수 및 메서드

- 파이썬의 세트는 다양한 빌트인 함수들을 제공합니다.

frozenset()

set() 대신 frozenset()을 사용하면 값 변경이 불가능한 세트를 생성할 수 있음.

```
>>> a = frozenset([ 4, 5, 1, 2, 1, 3, 5, 6, 4, 10])
>>> a.add('B')
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
AttributeError: 'frozenset' object has no attribute 'add'
```

.copy()

세트 복사본 생성

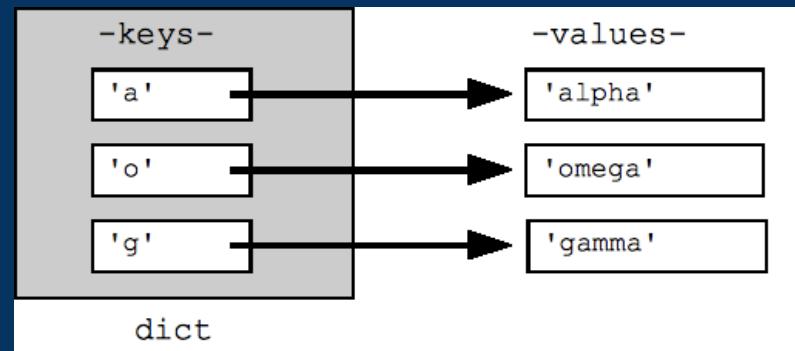
```
>>> a = set([ 4, 5, 1, 2, 1, 3, 5, 6, 4, 10])
>>> a2 = a
>>> a1 = a.copy()
>>> a.clear()
>>> a1
{1, 2, 3, 4, 5, 6, 10}
>>> a2
set()
```

copy()를 사용하지 않으면 예제와 같이 두 변수(a, a2)가 같은 세트를 공유함. 그러므로 a의 값이 바뀌면 a2의 값도 바뀜.

[집합형 자료형] 딕셔너리

- 딕셔너리는 키(key)와 값(value)이 쌍으로 이루어진 순서가 없는 자료형입니다.
- 딕셔너리는 {'key': 'value', ...} 형태로 생성할 수 있습니다.
- 값은 문자열, 숫자열, 사전, 튜플 등 어느 형태든지 가능하며, 키는 중복될 수 없습니다.

```
# 빈 딕셔너리 생성  
>>> d = {}  
  
# 딕셔너리 생성  
>>> d = {'A': 50, 'B': 54, 'C': 93}
```



```
# dict()를 이용하여 딕셔너리 생성  
>>> d = dict([('A', 50), ('B', 54), ('C', 93)])  
  
>>> d  
{'B': 54, 'C': 93, 'A': 50}  
  
>>> d = dict(A=50, B=54, C=93)
```

```
# value에 리스트도 넣을 수 있음.  
>>> d = {'A': [34, 65], 'B': 43}  
  
# value에 딕셔너리도 넣을 수 있음.  
>>> d = {'A': {'one': 1, 'two': 2}, 'B': [1, 2, 3]}
```

[집합형 자료형] 딕셔너리 접근

- 딕셔너리는 리스트나 튜플과 달리 순서가 없기 때문에 인덱스가 없고, key 값을 이용하여 value에 접근할 수 있습니다.
- 딕셔너리의 요소를 추가하고, 편집, 삭제할 수 있습니다.

$$d = \{ \text{A}: 50, \text{B}: 54, \text{C}: 93 \}$$

key value key value key value

```
>>> d['A']
50

>>> d['B']
54

# 없는 key로 접근할 경우 에러 발생
>>> d['D']
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
KeyError: 'D'
```

```
# 'D' 추가
>>> d['D'] = 'New element'
>>> d
{'B': 54, 'C': 93, 'A': 50, 'D': 'New element'}

# 'D'의 value 변경
>>> d['D'] = 'Changed'
>>> d
{'B': 54, 'C': 93, 'A': 50, 'D': 'Changed'}

# 요소 삭제
>>> del d['D']
>>> d
{'B': 54, 'C': 93, 'A': 50}
```

[집합형 자료형] 딕셔너리 관련 함수 및 메소드 (1/3)

- 딕셔너리를 다룰 수 있는 다양한 함수, 메소드들을 제공합니다.

.keys()

사전의 모든 key를 반환

```
>>> d = {'B': 54, 'C': 93, 'A': 50}  
>>> d.keys()  
dict_keys(['B', 'C', 'A'])  
  
# dict_keys 타입으로 반환되므로 list()를 통해 리스트로 변환  
>>> list(d.keys())  
['B', 'C', 'A']  
  
# list(사전)으로도 key값의 리스트를 얻을 수 있음.  
>>> list(d)  
['B', 'C', 'A']
```

.values()

사전의 모든 value를 반환

```
>>> d = {'B': 54, 'C': 93, 'A': 50}  
>>> d.values()  
dict_values([54, 93, 50])  
>>> list(d.values())  
[54, 93, 50]
```

.items()

사전의 모든 key, value를 [(key, value), ...] 형태로 반환

```
>>> d = {'B': 54, 'C': 93, 'A': 50}  
>>> d.items()  
dict_items([('B', 54), ('C', 93), ('A', 50)])  
>>> list(d.items())  
[('B', 54), ('C', 93), ('A', 50)]
```

[집합형 자료형] 딕셔너리 관련 함수 및 메소드 (2/3)

- 딕셔너리를 다룰 수 있는 다양한 함수, 메소드들을 제공합니다.

len()

사전의 'key':'value' 쌍의 개수 반환

```
>>> d = {'B': 54, 'C': 93, 'A': 50}  
>>> len(d)  
3
```

str()

사전을 문자열으로 변환

```
>>> d = {'B': 54, 'C': 93, 'A': 50}  
>>> s = str(d)  
>>> s  
"{'B': 54, 'C': 93, 'A': 50}"
```

.clear()

사전의 모든 요소 지우기 (사전 비우기)

```
>>> d = {'B': 54, 'C': 93, 'A': 50}  
>>> d.clear()  
>>> d  
{}
```

.copy()

사전 복사본 생성

```
>>> d = {'B': 54, 'C': 93, 'A': 50}  
>>> d.copy()  
{'B': 54, 'C': 93, 'A': 50}  
  
>>> d2 = d  
>>> d1 = d.copy()  
>>> d.clear()  
>>> d1 # d의 값이 바뀌어도 변경x  
{'B': 54, 'C': 93, 'A': 50}  
>>> d2 # d의 값이 바뀌면 같이 변경  
{}
```

copy()를 사용하지 않으면 예제와 같이 두 변수(d, d2)가 같은 사전을 공유함. 그러므로 d의 값이 바뀌면 d2의 값도 바뀜.

[집합형 자료형] 딕셔너리 관련 함수 및 메소드 (3/3)

- 딕셔너리를 다룰 수 있는 다양한 함수, 메소드들을 제공합니다.

.get(key, default)

사전으로부터 특정 key의 value를 가져오고
key가 없을 경우 default값을 반환

```
>>> d = {'B': 54, 'C': 93, 'A': 50}  
>>> d.get('C', 'novalue')  
93  
>>> d.get('Q', 'novalue')  
'novalue'
```

dict1.update(dict2)

dict2의 내용을 dict1에 추가

```
>>> d = {'B': 54, 'C': 93, 'A': 50}  
>>> d1 = {'E': 400, 'F': 22}  
>>> d.update(d1)  
>>> d  
{'C': 93, 'Q': 'novalue', 'F': 22, 'B': 'TT', 'E': 400, 'A': 50}
```

.setdefault(key, default)

get()과 비슷하나 key가 없을 경우 default값
을 사전에 추가

```
>>> d.setdefault('A', 'novalue')  
50  
>>> d  
{'B': 54, 'C': 93, 'A': 50}  
>>> d.setdefault('Q', 'novalue')  
'novalue'  
>>> d  
{'B': 54, 'C': 93, 'A': 50, 'Q': 'novalue'}
```

[실습] Jupyter notebook 기초

The screenshot shows a Jupyter Notebook interface with a sidebar on the left containing project files and a main workspace on the right.

Left Sidebar:

- 탐색기
- ...
PYTHON [CODESPACES: DIDACTIC SPACE ORBIT]
- > .devcontainer
- > .github
- Basics
 - calculation_practice.py
 - calculation.py
 - christmas.py
 - dinner_menu_detail.py
 - dinner_menu_simple.py
 - Kbiohealth-python.ipynb
- > Intermediate
- ◆ .gitignore
- ❶ README.md

Right Main Area:

File: Kbiohealth-python.ipynb M X
Basics > Kbiohealth-python.ipynb > M 01 변수 (Variables) > M 01-1 파이썬 변수 규칙 > # 잘못된 변수명 예제 (주석 처리 - 실행하면 오류 발생)
생성 + 코드 + Markdown | 모두 실행 ⚡ Restart 마크다운 셀

마크다운 셀 더블클릭시

01 변수 (Variables)

컴퓨터 메모리에 데이터를 저장합니다. 쉽게 기억되고 그 의미를 연상할 수 있는 이름인 mnemonic variable (기억 용이 변수)를 사용하는 것이 권장됩니다.
예) x, y, z 보다는 -> firstname, lastname, age, country

01-1 파이썬 변수 규칙

- * 문자(letter) 또는 밑줄(_)로 시작해야 합니다.
- * 숫자로 시작할 수 있습니다.
- * 알파벳, 숫자, 밑줄만 사용할 수 있습니다. (A-z, 0-9, 그리고 _)
- * 대소문자를 구분합니다.
(firstname, Firstname, FirstName, FIRSTNAME은 모두 다른 변수로 인식됩니다.)

markdown

코드 셀

아래는 파이썬 변수 규칙을 따르는 변수입니다.

```
# 올바른 변수명 예제
# firstname
# lastname
# age
# country
# city
person_info = {
    'firstname':'Sohee',
    'lastname':'Park',
    'country':'South Korea',
    'city':'Suwon'
}
```

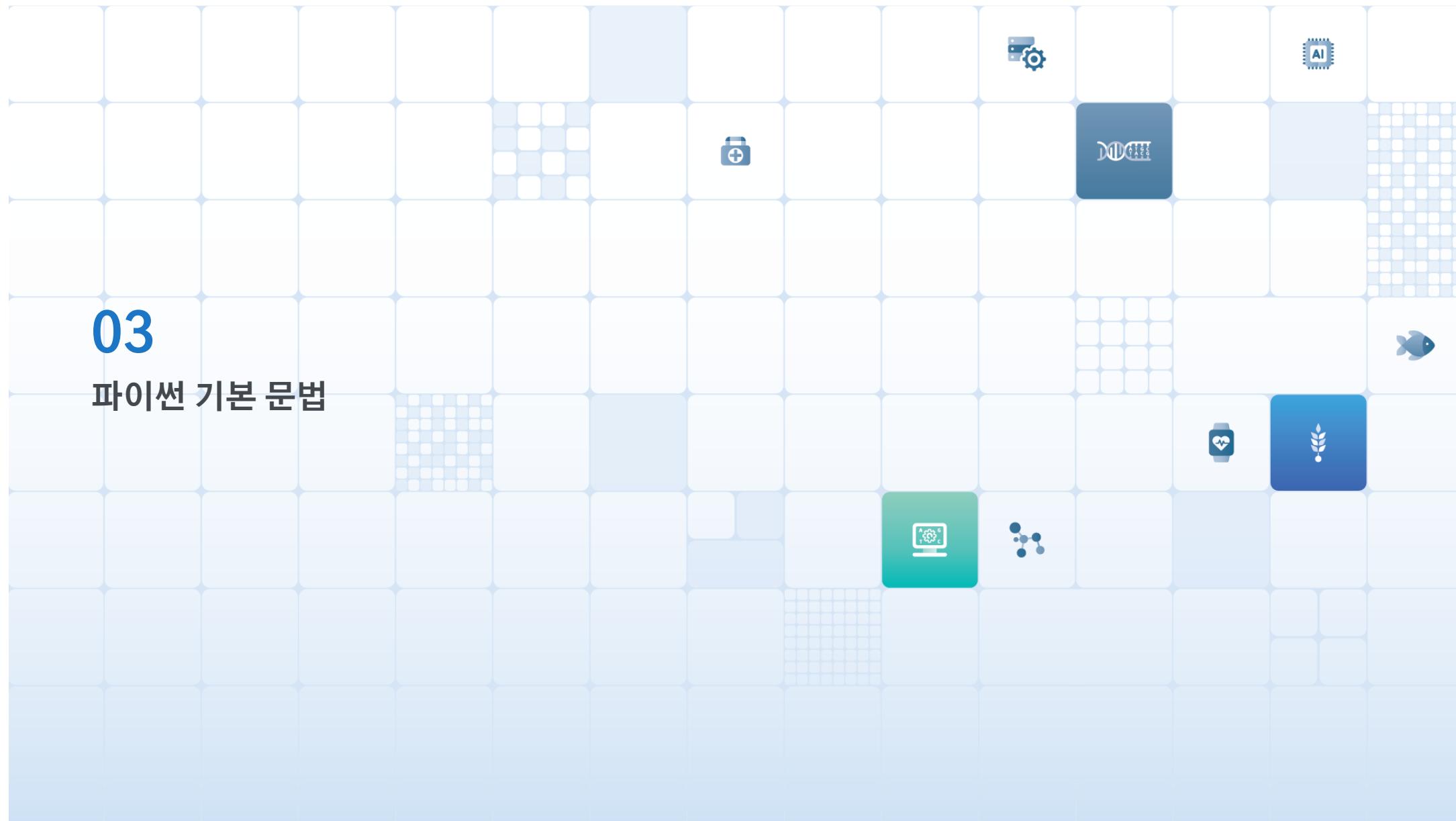
Python

입력 창

설명	옵션 종류
셀 실행	[Shift][Enter] [Ctrl][Enter]
셀 삽입	A 또는 B
셀 삭제	DD
마크다운 셀로 변경	M
코드 셀로 변경	Y

03

파이썬 기본 문법



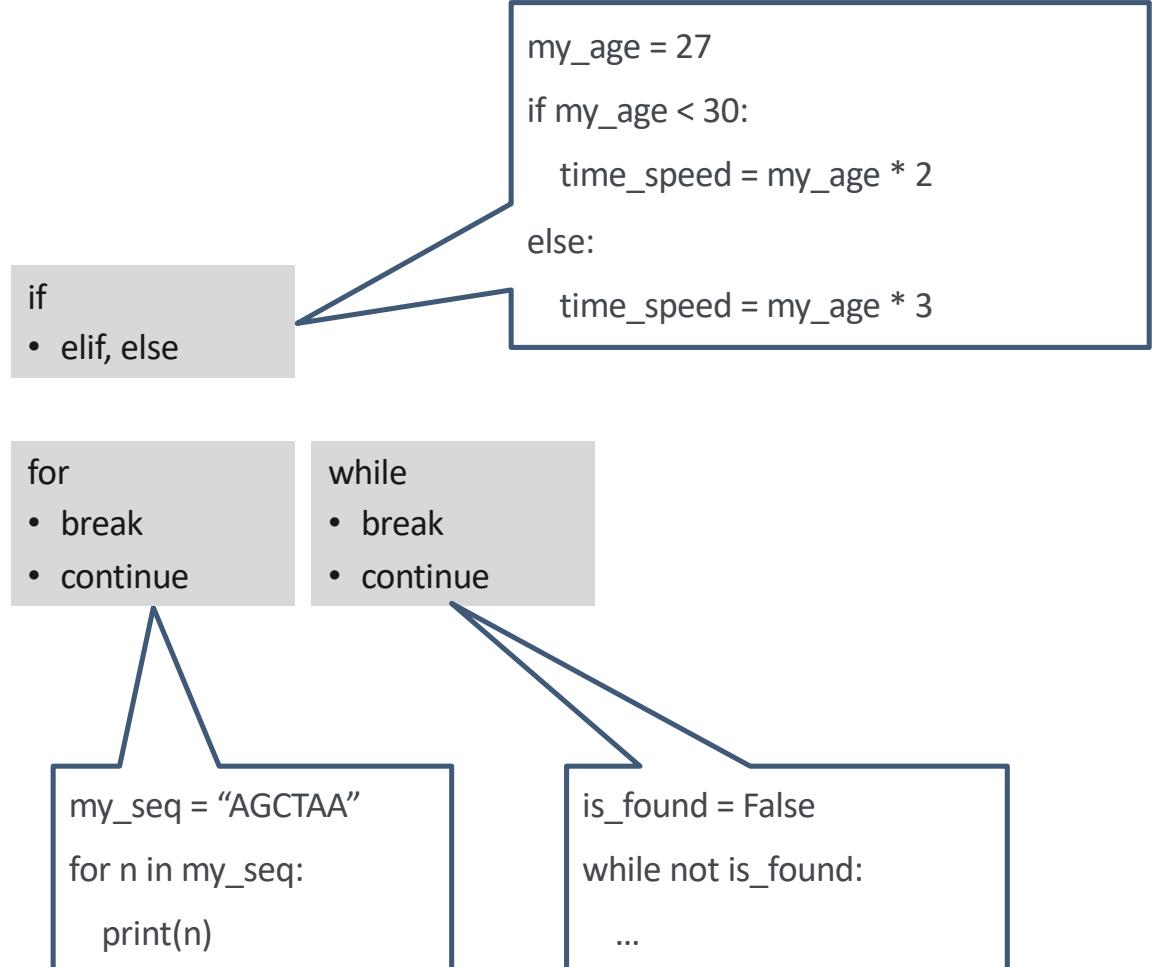
제어문의 필요성

프로그래밍이란,
다양한 상황에 맞게 필요한 업무를 수행하도록
계획안을 만드는 것입니다.

특정 상황에 맞는 일을 하는 것을
조건문 즉, **if 문**이라고 합니다.

특정 상황에 맞도록 반복하는 일도 있습니다.
for 문과 **while 문**으로 반복하는 일을 수행합니다.

제어문에서는 들여쓰기가 중요합니다!
들여쓰기가 시작되면 구문에 진입한 것이고,
들여쓰기가 끝나면 구문에서 빠져나옵니다.



[if 문] if 문의 구조

- if와 else를 이용한 조건문의 기본구조 입니다.
- 다양한 조건을 판단할 때 elif를 사용하며, 이전 조건문이 거짓일 때 수행합니다.

if 조건문1:

```
수행 문장1  
수행 문장2  
...  
...
```

elif 조건문:

```
수행 문장a  
수행 문장b  
...  
...
```

else:

```
수행 문장A  
수행 문장B  
...  
...
```

a의 값 유무를 판단하고 조건에 상응하는 코드 수행

```
>>> a = 3  
>>> if a:  
...     print('there is a')  
... else:  
...     print('there is no a')  
...  
there is a
```

```
>>> x = int(input('Please enter an integer: '))
```

```
Please enter an integer: 42
```

```
>>> if x < 0:  
...     x = 0  
...     print('Negative changed to zero')  
... elif x == 0:  
...     print('Zero')  
... elif x == 1:  
...     print('Single')  
... else:  
...     print('More')  
...  
More
```

[if 문] if 문 연산자 (1/2) – 비교 및 논리 연산자

비교 연산자

#	연산자	Syntax	Method
1	ordering	a < b	a.__lt__(b)
2	ordering	a <= b	a.__le__(b)
3	equality	a == b	a.__eq__(b)
4	difference	a != b	a.__ne__(b)
5	ordering	a >= b	a.__ge__(b)
6	ordering	a > b	a.__gt__(b)

```
>>> money = 2000
>>> if money >= 5000:
...     print('택시를 타고 갑시다')
... else:
...     print('걸어 갑시다')
...
걸어 갑시다
```

논리 연산자

#	연산자	설명
1	x or y	x와 y 둘중 하나만 참이면 참이다
2	x and y	x와 y 모두 참이면 참이다
3	not x	x가 거짓이면 참이다

- True and True → True
- True and False → False
- False and False → False
- False or False → False
- True or False → True

all(), any() 함수 – iterable 모든 요소에 대해 확인

- all : return true if all elements of the iterable are true
- any : return true if any element of the iterable is true

멤버십 연산자

#	x in list	x not in list
1	x in tuple	x not in tuple
2	x in string	x not in string
3	x in list	x not in list

```
>>> 1 in [1, 2, 3]
```

```
True
```

```
>>> 1 not in [1, 2, 3]
```

```
False
```

```
>>> 'a' in ('a', 'b', 'c')
```

```
True
```

```
>>> 'j' not in 'python'
```

```
True
```

반복문의 필요성

아래와 같이 리스트 내의 원소를 한 개씩 출력하고 싶다고 가정해봅시다.
원소의 개수가 100,000개일 때 스크립트의 문장 수는?

→ 거의 똑같은 문장을 100,000개 작성해야 하고, 번거롭습니다.

```
>>> test_list = ['one', 'two', 'three']
>>> print(test_list[0])
one
>>> print(test_list[1])
two
>>> print(test_list[2])
three
```

→ 반복문 (for 문, while 문)을 통해 해소 가능

[for 문] for 문의 구조

- 리스트나 튜플, 문자열의 첫번째 요소부터 마지막 요소까지 차례로 변수에 대입되어 수행할 문장 1, 2 등이 수행됩니다.

for 변수 in 리스트/튜플/문자열:

 수행 문장1

 수행 문장2

 ...

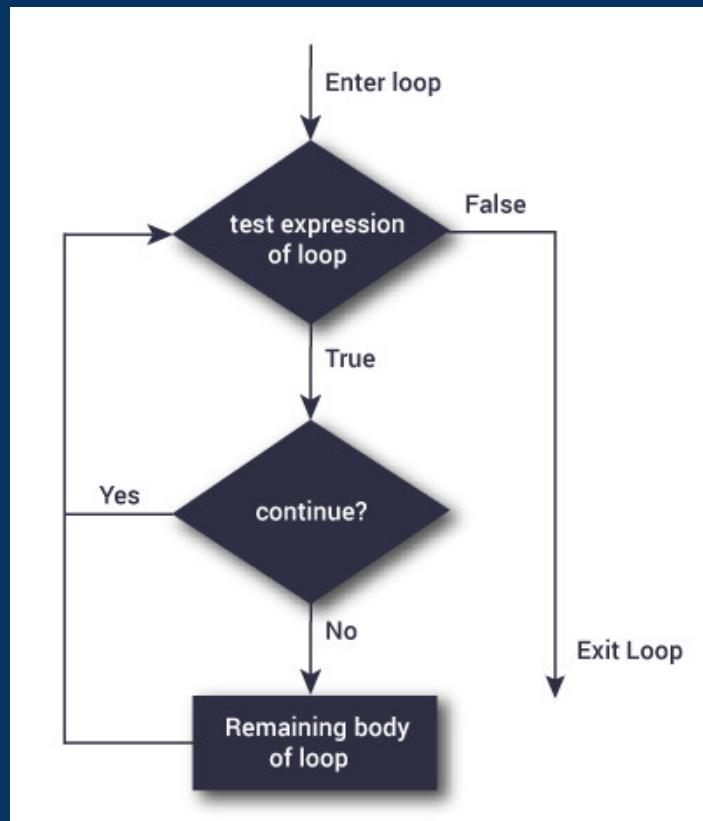
```
# 기본형태
>>> test_list = ['one', 'two', 'three']
>>> for element in test_list:
...     print(element)
...
one
two
three
```

```
# list comprehension
>>> squares = []
>>> for x in range(10):
...     squares.append(x**2)
...
>>> squares
[0, 1, 4, 9, 16, 25, 36, 49, 64, 81]

# the below equivalents to the above
>>> squares = [x**2 for x in range(10)]
```

[for 문] for 문의 흐름 제어 (1/2) – continue

- Continue를 활용하여 60점 이상인 사람에게는 축하메시지를 보내고 나머지 사람에게는 아무런 메시지도 전하지 않는 프로그램을 작성해봅니다.



```
>>> marks = [90, 25, 67, 45, 80]
>>> i = 0
>>> for mark in marks:
...     i += 1
...     if mark < 60:
...         continue
...     print(f'{i}번 학생 축하합니다. 합격입니다.')
.....
1번 학생 축하합니다. 합격입니다.
3번 학생 축하합니다. 합격입니다.
5번 학생 축하합니다. 합격입니다.
```

[for 문] for 문의 흐름 제어 (2/2) – range(), enumerate()

range()

순차적으로 늘어나는 수 만큼 iteration을 수행할 때

```
>>> list(range(5))  
[0, 1, 2, 3, 4]
```

```
>>> for i in range(5):  
...     print(i)  
...  
0  
1  
2  
3  
4
```

앞선 예제를 range()와 len()함수로 구현하면 아래와 같다.

```
>>> marks = [90, 25, 67, 45, 80]  
>>> for i in range(len(marks)):  
...     if marks[i] < 60:  
...         continue  
...     print(f'{i+1}번 학생 축하합니다. 합격입니다.')
```

1번 학생 축하합니다. 합격입니다.
3번 학생 축하합니다. 합격입니다.
5번 학생 축하합니다. 합격입니다.

enumerate()

배열의 인덱스를 함께 리턴 – enumerate(sequence[, start=0])

```
>>> for i, season in enumerate(['spring', 'summer', 'fall', 'winter']):  
...     print(i, season)  
0 spring  
1 summer  
2 fall  
3 winter
```

[while 문] while 문의 구조

- While 문은 반복해서 문장을 수행해야 할 경우에 사용합니다.
- While 문의 조건이 참일 동안 while 문 아래에 속하는 문장들이 반복해서 수행됩니다.

while 조건문:

수행 문장1

수행 문장2

...

```
>>> a = list(range(10))
>>> while len(a) > 5:
...     print(a)
...     print(a.pop())
...else:
...     print('End of while')
```

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

9

[0, 1, 2, 3, 4, 5, 6, 7, 8]

8

[0, 1, 2, 3, 4, 5, 6, 7]

7

[0, 1, 2, 3, 4, 5, 6]

6

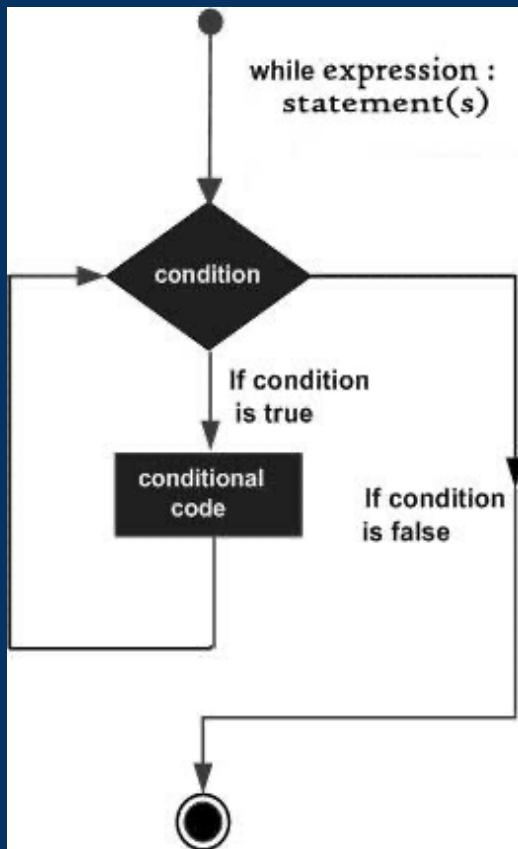
[0, 1, 2, 3, 4, 5]

5

End of while

[while 문] while 문의 구조

- 0부터 5까지 세면서 5와 비교하는 스크립트를 while 문을 통해 나타내면 다음과 같습니다.
- while 문은 적절한 탈출 구문이 없으면 무한루프에 빠집니다.



```
>>> count = 0  
>>> while count < 5:  
...     print(count, ' is less than 5')  
...     count += 1  
...else:  
...     print(count, ' is not less than 5')  
  
0 is less than 5  
1 is less than 5  
2 is less than 5  
3 is less than 5  
4 is less than 5  
5 is not less than 5
```

```
>>> while True:  
...     print('ctrl+c를 눌러야 while문을 빠져나갈 수 있습니다.')  
...  
ctrl+c를 눌러야 while문을 빠져나갈 수 있습니다.  
...
```

[while 문] while 문의 흐름 제어 (1/2) – Break, Continue, Pass

Break

루프 밖 탈출하기

```
number = 0
while number < 10:
    number += 1
    if number == 5:
        break
    print('Number is ' + str(number))
print('out of loop')
```

output (4 이후 루프 밖으로 빠져나옴)
Number is 1
Number is 2
Number is 3
Number is 4
out of loop

Continue

다음 루프로 넘어가기

```
number = 0
while number < 10:
    number += 1
    if number == 5:
        continue
    print('Number is ' + str(number))
print('out of loop')
```

output (5를 건너뜀)
Number is 1
Number is 2
Number is 3
Number is 4
Number is 6
Number is 7
Number is 8
Number is 9
Number is 10
out of loop

Pass

아무것도 안하기

```
number = 0
while number < 10:
    number += 1
    if number == 5:
        pass
    print('Number is ' + str(number))
print('out of loop')
```

output
Number is 1
Number is 2
Number is 3
Number is 4
Number is 5
Number is 6
Number is 7
Number is 8
Number is 9
Number is 10
out of loop

[while 문] while 문의 흐름 제어 (2/2) – Flag

- Flag 변수로 깃발을 정해두고, while 구문에서 바꿔주는 방식으로 프로그래밍 할 수 있습니다.
- 예제) 모든 알파벳이 소문자인 단어가 나올 때 까지 루프

```
>>> words = ['Hello', 'World', 'My', 'python', 'Great']
>>> not_found = True
>>> i = 0
>>> while not_found:
...     word = words[i]
...     if word.lower() == word:
...         not_found = False
...     print(word)
...     i += 1
```

python

[실습] 커멘드 라인에서 파이썬 스크립트를 실행해봅시다.

설명	옵션 종류
현재 디렉토리	pwd
폴더 목록	ls -la
폴더 이동	cd
상위 폴더	cd ..
파일 읽기	cat

리눅스 셸 명령어

cd Basics ...Basics 디렉토리 들어가기

python christmas.py ...for 문

python dinner_menu_simple.py ... list

dinner_menu_detail.py ... dict

python calculation.py ... if

리눅스 셸에서 하나씩 실행

[파이썬 실행] 파이썬의 실행 방법

- 상호반응형 모드 (Interactive mode)

```
$ python3
Python 3.4.3 (default, Jul 13 2015, 12:18:23)
[GCC 4.2.1 Compatible Apple LLVM 6.1.0 (clang-602.0.53)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello world")
Hello world
```

- 프로그램 소스 코드를 .py 파일에 저장

```
$ vi mycode.py
print("Hello world")
$ python mycode.py
Hello world
```

[파이썬 실행] 파이썬의 실행 방법

- 상호반응형 모드 (Interactive mode)

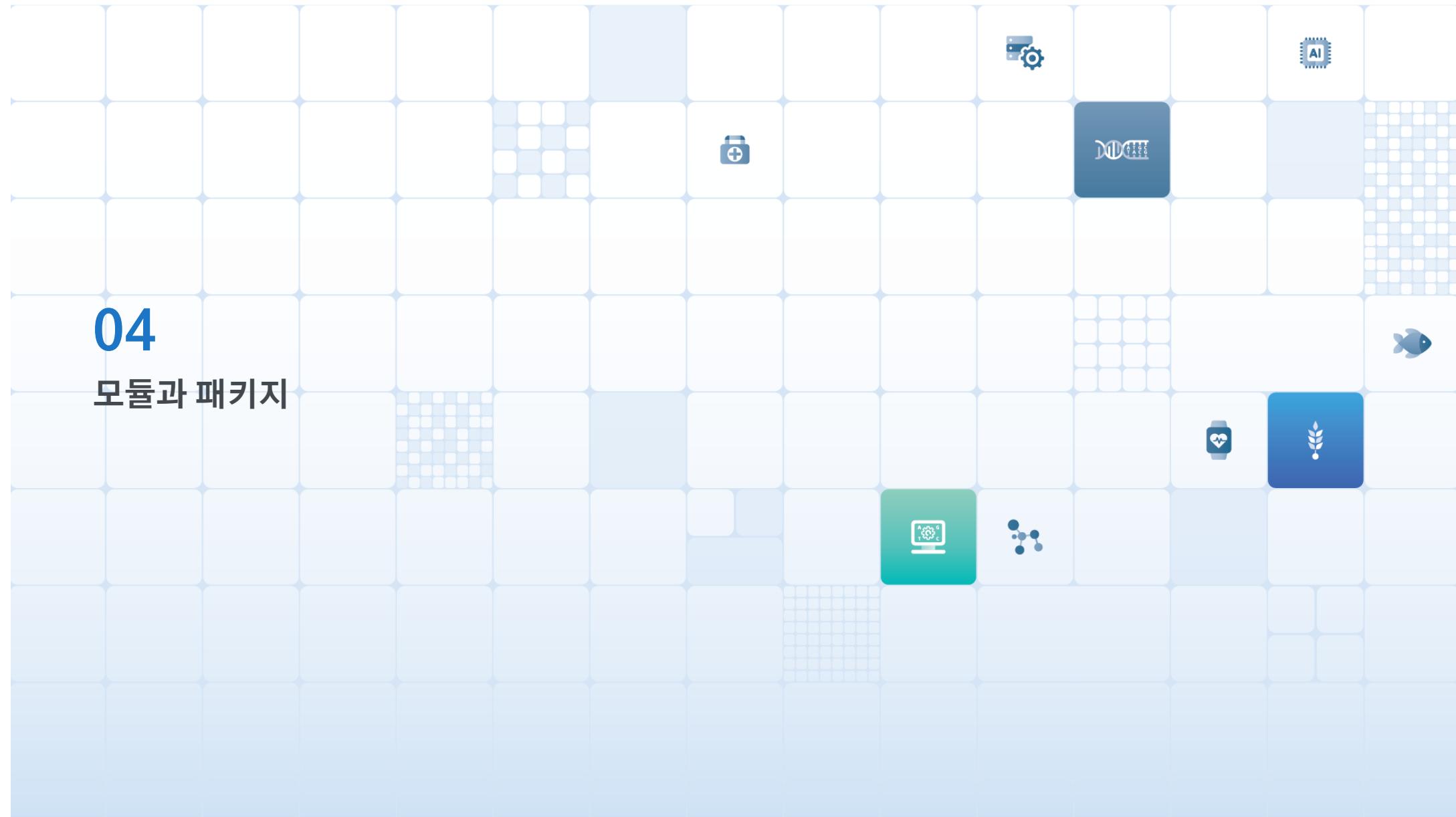
```
$ python3
Python 3.4.3 (default, Jul 13 2015, 12:18:23)
[GCC 4.2.1 Compatible Apple LLVM 6.1.0 (clang-602.0.53)] on darwin
Type "help", "copyright", "credits" or "license" for more information.
>>> print("Hello world")
Hello world
```

- 프로그램 소스 코드를 .py 파일에 저장

```
$ vi mycode.py
print("Hello world")
$ python mycode.py
Hello world
```

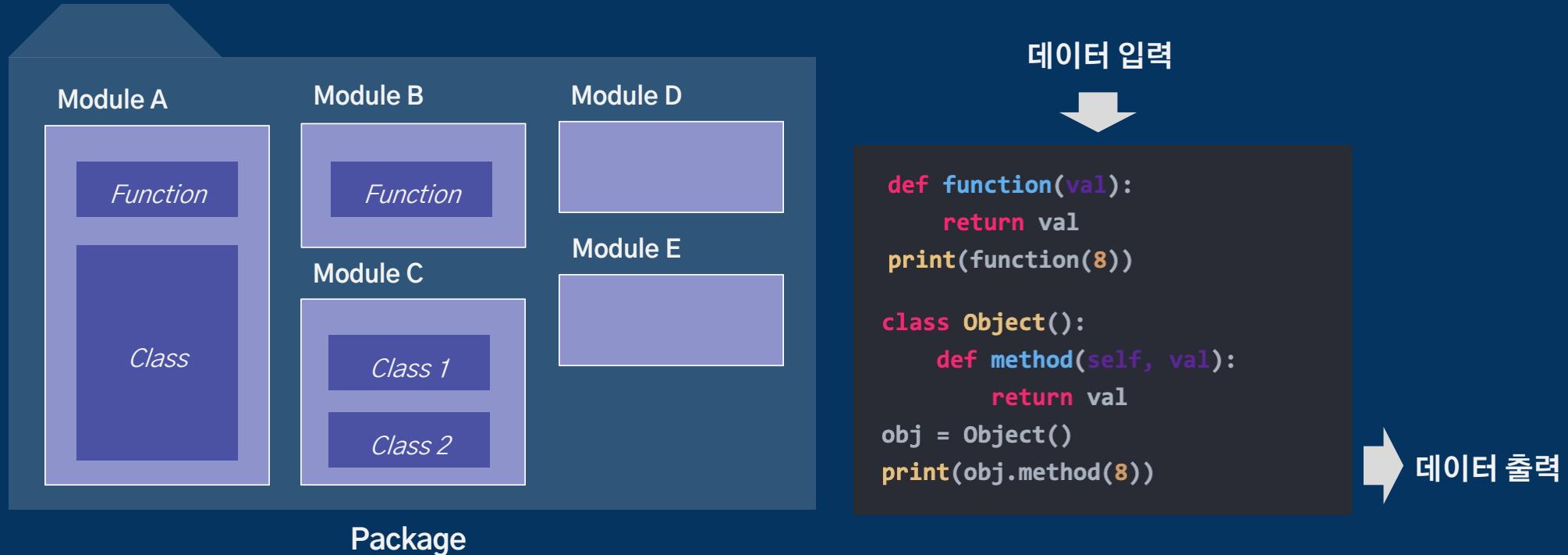
04

모듈과 패키지



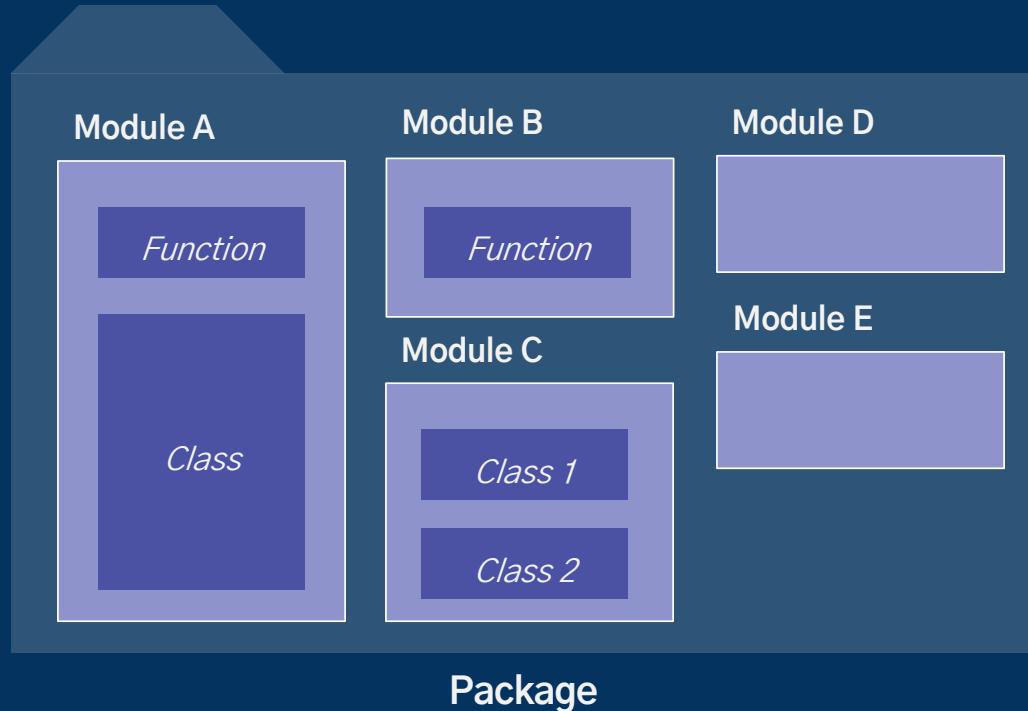
모듈과 패키지

- 필요한 패키지를 불러와 사용할 수 있습니다.
- 나만의 함수를 만들어 활용할 수 있습니다.
- 파이썬 스크립트를 작성하여 데이터 입력하고 원하는 형태로 출력할 수 있습니다.



모듈과 패키지

- 필요한 패키지를 불러와 사용할 수 있습니다.
- 나만의 함수를 만들어 활용할 수 있습니다.
- 파이썬 스크립트를 작성하여 데이터 입력하고 원하는 형태로 출력할 수 있습니다.



데이터 입력



```
def function(val):  
    return val  
print(function(8))  
  
class Object():  
    def method(self, val):  
        return val  
obj = Object()  
print(obj.method(8))
```

데이터 출력

[라이브러리] 라이브러리의 종류

1. 파이썬 기본 제공 함수

- 파이썬에서 기본 함수로 직접 사용 가능합니다.

	함수	설명	코드 예시
1	max/min	둘 이상의 인수 중에서 최대값 또는 최소값을 반환	<code>max(numbers), min(numbers)</code>
2	int	숫자나 문자열을 정수형으로 변환	<code>int(num_str)</code>
3	isinstance	객체가 특정 클래스의 인스턴스인지 확인	<code>isinstance(var, int)</code>
4	map	주어진 함수를 반복 가능한 객체(리스트 등)의 각 요소에 적용하여 결과 반환	<code>list(map(lambda x: x**2, numbers))</code>

2. 내장 라이브러리

- 파이썬에서 기본으로 제공하는 모듈/패키지이며, 파이썬 설치시 함께 제공합니다.

	함수	설명	코드 예시
1	math	수학 관련 함수와 상수를 제공하는 모듈	<code>math.sqrt(num), math.radians(45)</code>
2	datetime	날짜와 시간 조작을 위한 클래스와 함수를 제공하는 모듈	<code>datetime(2000, 5, 15), datetime.now()</code>
3	os	운영 체제와 상호 작용하기 위한 함수들을 제공하는 모듈	<code>os.getcwd(), os.listdir(cwd)</code>
4	sys	파이썬 인터프리터와 관련된 정보를 제공하는 모듈	<code>sys.version, sys.argv</code>
5	re	정규 표현식(Regex) 처리를 위한 모듈 예) 비밀번호 만들때 조건, 엔기서열 등	<code>re.match(pattern, password)</code>

3. 외부 라이브러리

- 외부 모듈/패키지를 별도로 설치해야 합니다.



[라이브러리] 모듈과 패키지의 구성 (1/2)

- 모듈:** 하나의 파이썬 파일(.py)로, 변수, 함수, 클래스 등을 담고 있을 수 있습니다.
- 패키지:** 여러 모듈을 하나의 디렉토리에서 관리하는 방식입니다. 패키지는 디렉토리 내에 `__init__.py` 파일을 포함하여 해당 디렉토리를 패키지로 인식합니다. 패키지는 하위 디렉토리(서브 패키지)를 가질 수도 있습니다.

```
# 변수(variables) 정의
```

```
numberone = 1  
ageofqueen = 78
```

} 변수

```
# 함수(functions) 정의
```

```
def printhello():  
    print("hello")
```

} 함수

```
def timesfour(input):  
    print(input * 4)
```

클래스

module_test.py

```
# 클래스(class) 정의
```

```
class Piano:  
    def __init__(self):  
        self.type = raw_input("What type of piano? ")  
        self.height = raw_input("What height (in feet)? ")  
        self.price = raw_input("How much did it cost? ")  
        self.age = raw_input("How old is it (in years)? ")  
  
    def printdetails(self):  
        print("This piano is a/an " +  
              self.height +  
              " foot",)  
        print(self.type, "piano, " +  
              self.age, "years old and costing " +  
              self.price + " dollars.")
```

module_test.py

https://en.wikibooks.org/wiki/A_Beginner%27s_Python_Tutorial/Importing_Modules

[라이브러리] 모듈과 패키지의 구성 (2/2)

- import 명령어를 이용해서 모듈, 함수, 변수를 불러올 수 있습니다.
- 파일명이 모듈명이 되며, 특정 변수나 함수만 불러오기도 가능합니다.

(1) 모듈 불러오기

```
import 모듈명(또는 파일명)  
import 모듈1, 모듈2, ..., 모듈N
```

모듈 불러오기 (module_test.py)

```
import module_test  
moduletest.ageofqueen  
>>> 78  
moduletest.printhello()  
>>> hello
```

(2) 모듈에서 특정 함수 또는 변수 불러오기

```
from 모듈명 import 함수명(또는 변수명)  
from 모듈명 import 함수1, 함수2, ..., 함수N  
from 모듈명 import 함수명(또는 변수명) as Alias명  
from 모듈명 import *
```

모듈에서 특정 변수나 함수만 불러오기

```
from module_test import ageofqueen, timesfour  
timesfour(40)  
>>> 160  
ageofqueen  
>>> 78
```

[모듈/패키지] 파이썬 내장 라이브러리 (1/3)

1. math

- 수학적 연산에 필요한 기능들을 모아놓은 모듈입니다.
- 참고: <https://docs.python.org/3/library/math.html>

```
# math 모듈 임포트
import math
n = math.factorial(3)
n
>>> 6

# math 모듈의 factorial() 함수만 임포트
from math import factorial
n = factorial(3)
n
>>> 6
factorial(4)
>>> 24
```

[모듈/패키지] 파이썬 내장 라이브러리 (2/3)

2. os

- 파일을 읽기/쓰기, 환경변수, 파일/디렉토리를 제어할 수 있는 기능을 가진 모듈입니다.
- 참고: <https://docs.python.org/3/library/os.html>

```
# 시스템 환경변수 확인
```

```
import os  
os.environ  
  
>>> environ({'__CF_USER_TEXT_ENCODING': '0x1F5:0x3:0x33',  
           'LDLIBRARY_PATH': ... 'pw_pctjw000gn/T/,'})
```

```
# 디렉토리 위치 변경&확인
```

```
os.getcwd()  
>>> '/Users/koreabio/Downloads'  
os.chdir('..')  
os.getcwd()  
>>> '/Users/koreabio'
```

```
# 시스템 명령어 실행
```

```
f = os.popen('pwd')  
f.read()  
>>> '/Users/koreabio\n'
```

```
# 디렉토리 생성, 삭제, 이름변경
```

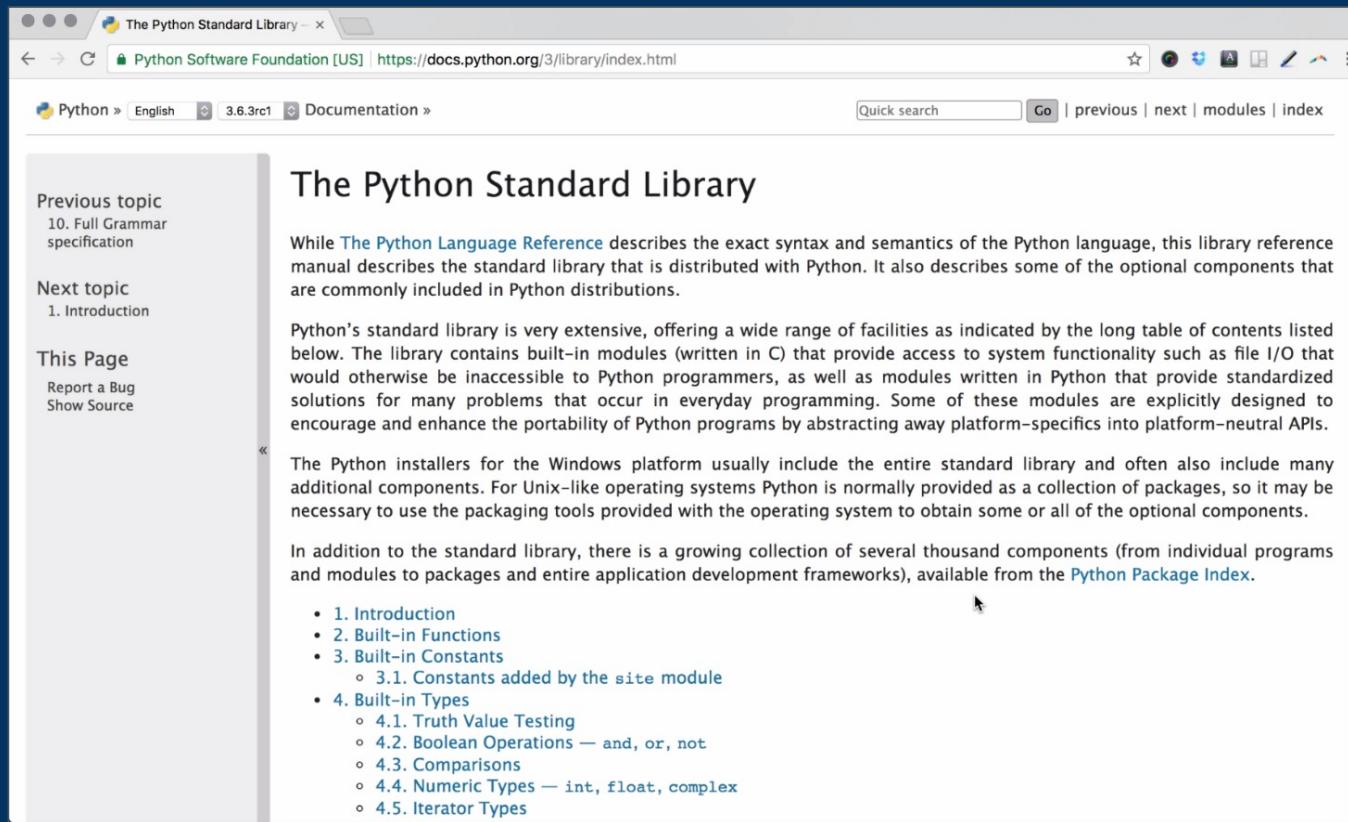
```
os.mkdir('dirname')  
os.rmdir('dirname')  
os.rename('oldname', 'newname')
```

```
# 파일삭제
```

```
os.unlink('filename')
```

[모듈/패키지] 파이썬 내장 라이브러리 (3/3)

- 이미 만들어진 built-in 모듈을 가져와서 사용할 수 있습니다.
- 파이썬 built-in 모듈은 파이썬 공식 홈페이지(<https://docs.python.org/3/library/>)에서 확인할 수 있습니다.



[모듈/패키지] 파이썬 외부 라이브러리 (1/2)

- 파이썬 내장 라이브러리 외에 특수한 기능을 하는 **외부 라이브러리**가 있으며 **3rd-party 라이브러리**라고 부릅니다.
- pip**(Pip Install Package)으로 PyPI(Python Package Index)에서 제공하는 **파이썬 라이브러리를 설치할 수 있습니다.**

1. pip 프로그램 이용

```
$ pip install pandas  
$ pip remove pandas  
$ pip search pandas (v20.10에서 지원 중단)
```

2. 다운로드 후, 설치 스크립트 실행

```
$ wget http://biopython.org/DIST/biopython-1.70.tar.gz  
$ tar zxvf biopython-1.70  
$ cd biopython-1.70  
$ less README.md  
$ python setup.py install
```

[모듈/패키지] 파이썬 외부 라이브러리 (2/2)

3. 직접 만든 모듈 사용하기

- 직접 만든 계산기 모듈을 불러와 사용할 수 있습니다.
- 아래 계산기 모듈은 덧셈, 뺄셈 등 연산에 관련된 함수와 num1, num2와 같은 변수로 구성되어 있습니다.

```
### calculation.py
def add(x, y):
    return x + y

def sub(x, y):
    return x - y

def mul(x, y):
    return x * y

def div(x, y):
    return x / y

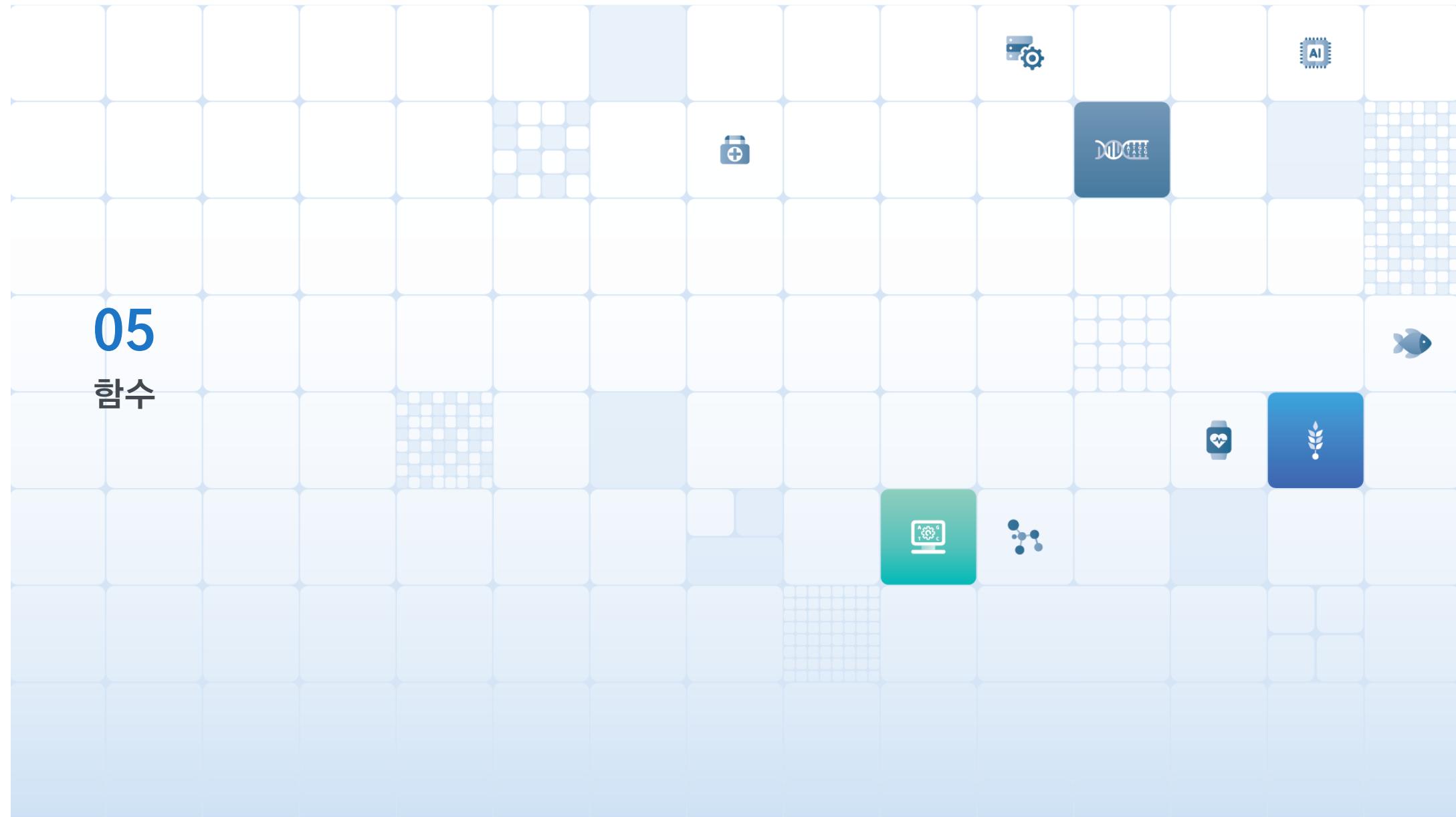
num1 = float(input('Enter your first number: '))
num2 = float(input('Enter your second number: '))
type_ = input('Enter type of operation (+, -, *, or /): ')
```

```
if type_ == '+':
    print(add(num1, num2))
elif type_ == '-':
    print(sub(num1, num2))
elif type_ == '*':
    print(mul(num1, num2))
elif type_ == '/':
    print(div(num1, num2))
else:
    print("Invalid type of operation: '{}'".format(type_))
```

```
import calculation
>>> Enter your first number: 1
>>> Enter your second number: 2
>>> Enter type of operation (+, -, *, or /): -
-1.0
```

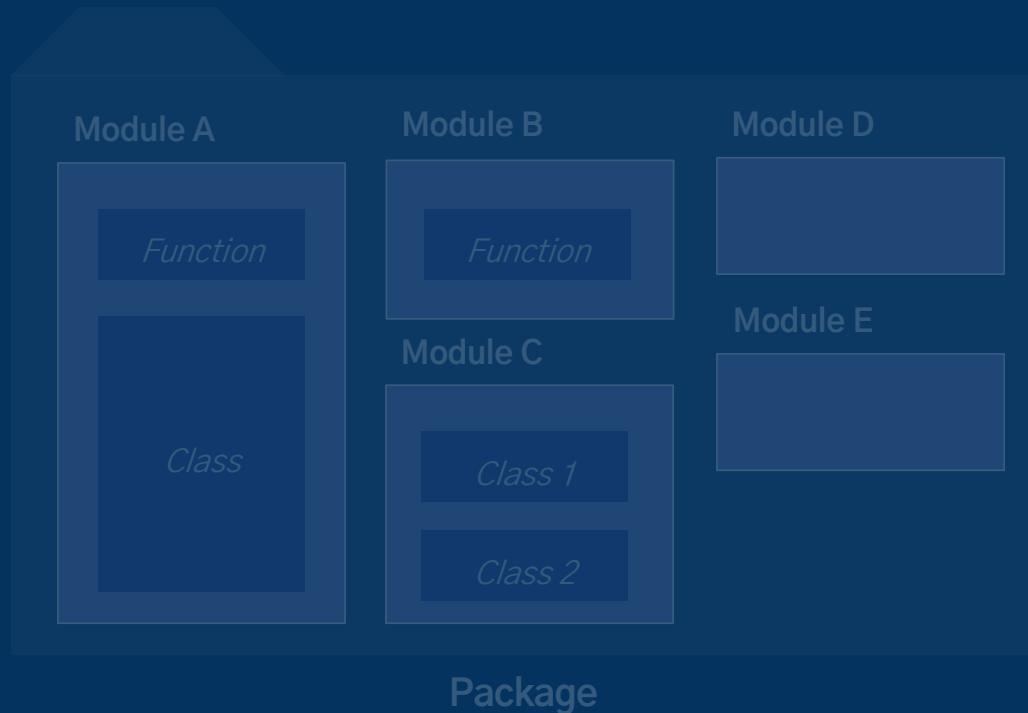
05

함수



모듈과 패키지

- 필요한 패키지를 불러와 사용할 수 있습니다.
- 나만의 함수를 만들어 활용할 수 있습니다.
- 파이썬 스크립트를 작성하여 데이터 입력하고 원하는 형태로 출력할 수 있습니다.



데이터 입력



```
def function(val):  
    return val  
print(function(8))  
  
class Object():  
    def method(self, val):  
        return val  
obj = Object()  
print(obj.method(8))
```

데이터 출력

[함수] 파이썬 함수의 기본 구조 (1/2)

파이썬 함수의 기본 구조

파이썬 함수의 구조는 다음과 같습니다.

```
def 함수명(입력 인수):  
    수행할 문장1  
    ...  
    return 결과값
```

```
# 함수 정의  
def sum(a,b):  
    return a+b  
  
# 함수 실행 및 출력  
result = sum(3, 4)  
print(result)  
=> 7
```

아래와 같은 함수를 만들어봅시다.

```
함수명은 sum이고 입력 인수는 2개,  
결과값은 이 2개의 인수를 더한 값
```

[함수] 파이썬 함수의 기본 구조 (1/2)

예를 들어,

이차방정식 근의 공식 함수를 만든다면

$$23x^2 + 43.2x + 34 = 0$$

위와 같은 이차방정식의 근을 구하는 것은

$$x = \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

근의 공식을 써서, a, b, c 를 입력으로 받아,
근 2개를 출력으로 하는 함수를 만들면,

앞으로, a, b, c 만으로 곧바로 근을 알 수 있다.

```
import cmath

# 함수 정의
def quadratic_equation(a, b, c):
    in_sqrt = cmath.sqrt(b*b - 4*a*c)
    x1 = (-b + in_sqrt) / (2.0*a)
    x2 = (-b - in_sqrt) / (2.0*a)
    return x1, x2

# 함수 출력
print(quadratic_equation(23, 43.2, 34))
>>> ((-0.9391304347+0.7722013312j),
     (-0.9391304347-0.7722013312j))
```

[함수] 파이썬 함수의 기본 구조 (2/2)

결과값은 언제나 하나

다수의 값을 반환하는 함수의 경우,

이를 하나의 변수로 받으면 **튜플**로 받습니다.

반환 값을 여러 변수로 각각 받으려면,
반환 값의 개수와 변수의 개수가 일치해야 합니다.

```
def sum_and_mul(a, b):
    return a+b, a*b

# 반환 값을 하나의 변수로 받는 경우
result = sum_and_mul(3, 4)
result
>>> (7, 12)
```

```
# 반환 값을 여러 변수로 받는 경우
sum, mul = sum_and_mul(3, 4)
sum
>>> 7
mul
>>>12
```

```
# 오류
def add_and_mul(a,b):
    return a+b
    return a*b
```

[함수] 파이썬 함수의 형태 (1/2)

1. 입력값과 결과값이 모두 있을 때

```
def 함수명(입력 인수):  
    수행할 문장1  
    ...  
    return 결과값
```

```
결과값 변수 = 함수명(인수1, 인수2, ...)
```

```
# 함수 정의  
def sum(a,b):  
    result = a+b  
    return result
```

```
# 함수 실행 및 출력  
a = sum(3, 4)  
print(a)  
>>> 7
```

2. 입력값 및 결과값이 모두 없을 때

```
def 함수명():  
    수행할 문장1  
    수행할 문장2  
    ...
```

```
함수명()
```

```
# 함수 정의  
def say():  
    print('hi')
```

```
# 함수 실행  
say()  
>>> hi
```

[함수] 파이썬 함수의 형태 (2/2)

3. 입력값만 있을 때

```
def 함수명(인수1, 인수2, ...):  
    수행할 문장1  
    수행할 문장2  
    ...
```

```
함수명()
```

```
# 함수 정의  
def sum(a,b):  
    print(f'{a}와 {b}의 합은 {a+b}.')
```

```
# 함수 실행  
a = sum(3, 4)  
>>> 3과 4의 합은 7.
```

```
# 함수 출력  
print(a)  
>>> None
```

4. 출력값만 있을 때

```
def 함수명():  
    수행할 문장1  
    ...  
    return 결과값
```

```
결과값 변수 = 함수명()
```

```
# 함수 정의  
def say():  
    return 'hi'
```

```
# 함수 실행  
a = say()
```

```
# 함수 출력  
print(a)  
>>> hi
```

[인수와 변수] 인수의 특징 (1/3)

1. 초기값 설정 (키워드 인수)

입력 인수의 초기값을 미리 설정할 수 있습니다.

입력 인수를 설정하더라도
해당 인수에 다른 변수를 입력할 시에는
입력한 인수로 값이 치환됩니다.

초기값이 설정된 인수는 보통 오른쪽에 배치합니다.
아니라면 아래와 같은 에러가 출력됩니다.

```
SyntaxError: non-default argument follows  
default argument
```

```
# 세 개의 매개변수로 이루어진 함수 정의  
def say_myself(name, age, man=True):  
    print(f'나의 이름은 {name}입니다.')  
    print(f'나이는 {age}입니다.')  
    if man:  
        print('성별은 남자입니다.')  
    else:  
        print('성별은 여자입니다.')
```

```
# 함수 실행 예시 (1)  
say_myself('철수', 30)  
=> 나의 이름은 철수입니다.  
=> 나이는 30입니다.  
=> 성별은 남자입니다.  
# 함수 실행 예시 (2)  
say_myself('영희', 20, False)  
=> 나의 이름은 영희입니다.  
=> 나이는 20입니다.  
=> 성별은 여자입니다.
```

[인수와 변수] 인수의 특징 (2/3)

2. 임의의 입력값일 경우 (*args)

입력값이 여러 개인 경우에는 *args를 이용합니다.

*args는 인수의 개수를 제한하지 않고,
여러 개의 인수를 **튜플 형태**로 함수에 전달 합니다.

또한 *args 와 다른 키워드들의 조합으로 함수를 만들 수 있으며
이에 대한 예제는 오른쪽과 같이 작성할 수 있습니다.

보통 인수는 왼쪽,
여러 개의 입력값을 받는 인수를 **오른쪽에 배치**합니다.

```
# 예시 (1)
def join_lists(seperator, *words):
    return seperator.join(words)

# 함수 실행 예시 (1)
join_lists('-', 'Hello', 'world')
>>> 'Hello-world'

# 함수 실행 예시 (2)
join_lists('-', 'Hello', 'my', 'world')
>>> 'Hello-my-world'
```

[인수와 변수] 인수의 특징 (2/3)

2. 임의의 입력값일 경우 (*args)

입력값이 여러 개인 경우에는 *args를 이용합니다.

*args는 인수의 개수를 제한하지 않고,
여러 개의 인수를 **튜플 형태**로 함수에 전달 합니다.

또한 *args 와 다른 키워드들의 조합으로 함수를 만들 수 있으며
이에 대한 예제는 오른쪽과 같이 작성할 수 있습니다.

보통 인수는 왼쪽,
여러 개의 입력값을 받는 인수를 **오른쪽에 배치**합니다.

```
# 예시 (2)
def sum_mul(choice, *args):
    if choice == 'sum':
        result = 0
        for i in args:
            result = result + i
    elif choice == 'mul':
        result = 1
        for i in args:
            result = result * i
    return result
# choice가 'sum'인 경우
result = sum_mul('sum', 1, 2, 3, 4, 5)
print(result)
>>> 15
# choice가 'mul'인 경우
result = sum_mul('mul', 1, 2, 3, 4, 5)
print(result)
>>> 120
```

[인수와 변수] 인수의 특징 (3/3)

3. 임의 갯수 키워드 인수 목록 (**kwargs)

함수 정의에서 임의의 개수의 키워드 인수에는 **kwargs를 이용합니다.

**kwargs는 dictionary 형태로 “key: value” 쌍으로 구성됩니다.

(참고)

*args는 위치 인수를 튜플 형태

**kwargs는 키워드 인수를 dictionary 형태

기본적인 **kwargs 사용 예시

```
# 함수 정의
def print_kwargs(**kwargs):
    for key, value in kwargs.items():
        print(f'{key}: {value}')
```

함수 호출

```
print_kwargs(name='John', age=25, city='New York')
>>> name: John
>>> age: 25
>>> city: New York
```

[인수와 변수] 인수의 특징 (3/3)

3. 임의 갯수 키워드 인수 목록 (**kwargs)

함수 정의에서 임의의 개수의 키워드 인수에는 **kwargs를 이용합니다.

**kwargs는 dictionary 형태로 “key: value” 쌍으로 구성됩니다.

(참고)

*args는 위치 인수를 튜플 형태

**kwargs는 키워드 인수를 dictionary 형태

*args와 **kwargs를 함께 사용하는 예제

```
# 함수 정의
def example_function(arg1, *args, **kwargs):
    print(f'arg1: {type(arg1)}, {arg1}')
    print(f'args: {type(args)}, {args}')
    print(f'kwargs: {type(kwargs)}, {kwargs}')
```

함수 실행 및 출력

```
example_function(1, 2, 3, a='John', b=25)
>>> arg1: <class 'int'>, 1
>>> args: <class 'tuple'>, (2, 3, 4)
>>> kwargs: <class 'dict'>, {'name': 'John', 'age': 25}
```

[인수와 변수] 변수의 특징 (1/2)

지역 변수와 전역 변수

1. 지역변수 (Local Variable)

지역변수는 [함수 안](#)에서 선언됩니다.

해당 함수의 [코드 블록 내](#)에서만 유효합니다.

함수가 호출될 때 지역 변수가 생성되고,
함수 실행이 완료되면 지역 변수는 메모리에서 제거됩니다.

2. 전역 변수 (Global Variable)

전역 변수는 [모든 함수에서 접근](#)할 수 있는 변수입니다.

함수 외부에서 선언되며, 프로그램 전체에서 유효합니다.

```
def example_function():
    local_variable = 10 # 지역 변수
    print(local_variable)
```

```
example_function() # 함수 호출
print(local_variable) # 오류 발생
```

```
global_variable = 5 # 전역 변수
def example_function():
    local_variable = 10 # 지역 변수
    print("전역 변수:", global_variable)
    print("지역 변수:", local_variable)
```

```
example_function() # 함수 호출
print("전역 변수:", global_variable) # 전역 변수 접근
print("지역 변수:", local_variable) # 오류 발생
```

[인수와 변수] 변수의 특징 (2/2)

1. return을 이용하기

: return으로 함수 안의 변수를 출력한 뒤 이를 치환하는 방식으로 함수 밖의 변수를 바꿀 수 있습니다.

```
# 함수 정의
a = 1
def vartest(a):
    a = a+1
    return a
```

```
# 함수 실행 및 출력
a = vartest(a)
print(a)
>>> 2
```

2. global 명령어 이용하기

: global을 이용하여 함수 밖의 변수를 직접 불러와 읽고 쓸 수 있습니다. (전역변수)

```
# 함수 정의
a = 1
def vartest():
    global a
    a = a+1
```

```
# 함수 실행 및 출력
vartest()
print(a)
>>> 2
```

[팁] 한 줄 함수 lambda

lambda

한 줄로 함수를 생성할 수 있습니다.

lambda는 def를 이용한 함수 정의보다
훨씬 간결하게 사용할 수 있습니다.

오른쪽 예제의 위는 기존의 def를 이용한 함수 정의 방식
아래는 lambda를 이용하여 정의한 방식입니다.

```
# 함수 정의
def sum(a,b):
    return a+b

# 함수 실행 및 출력
sum(3, 4)
>>> 7
```

```
# Lambda 활용한 함수와 그 출력
sum = lambda a, b: a+b
sum(3, 4)
>>> 7
```

[실습] 커멘드 라인에서 파이썬 스크립트를 실행해봅시다.

설명	옵션 종류
현재 디렉토리	pwd
폴더 목록	ls -la
폴더 이동	cd
상위 폴더	cd ..
파일 읽기	cat

리눅스 셸 명령어

cat calculation_practice.py ... 코드 읽기

calculation_practice.py 수정

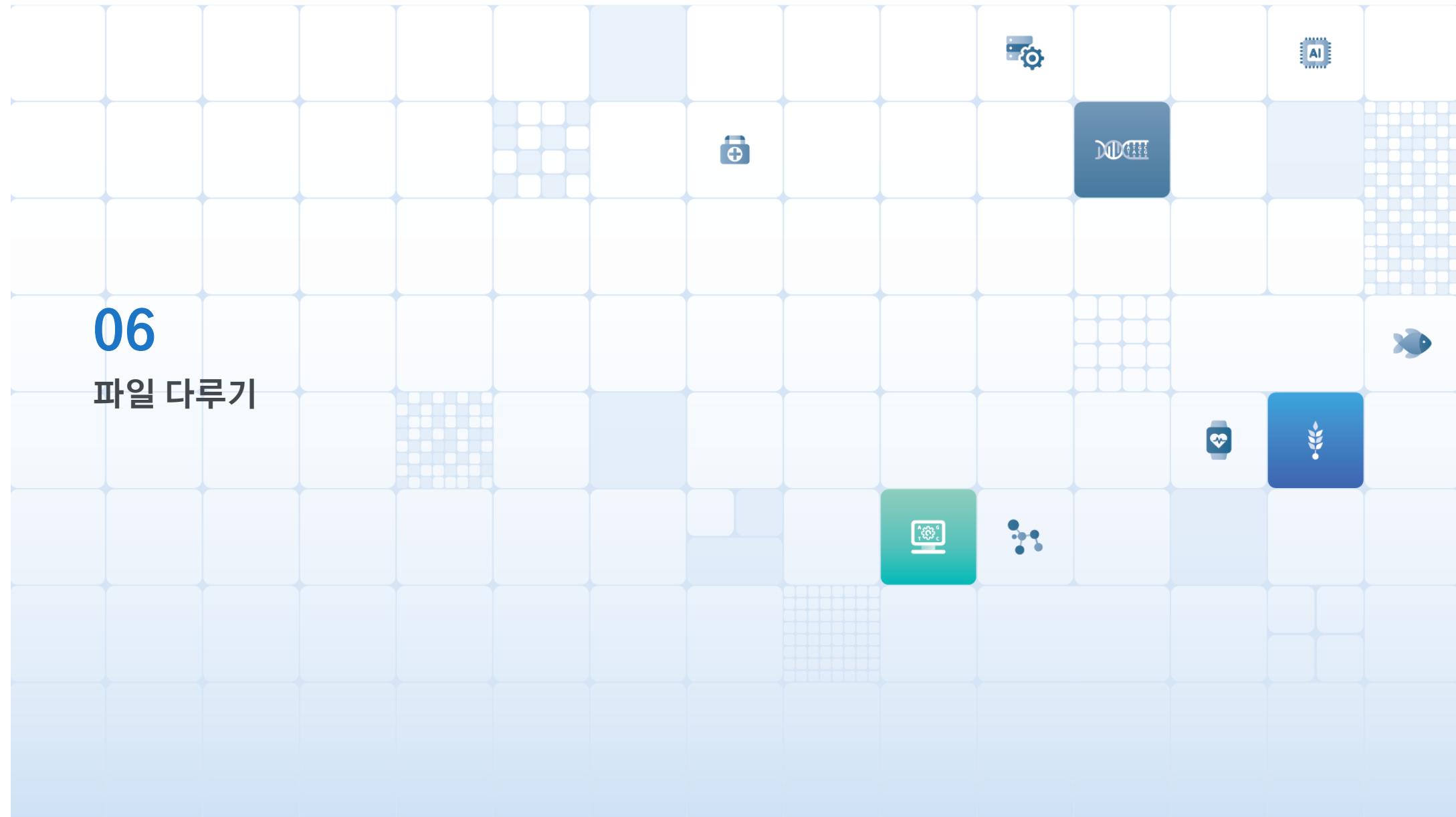
calculation_practice.py 저장

calculation_practice.py 실행

리눅스 셸에서 calculation_practice.py 수정

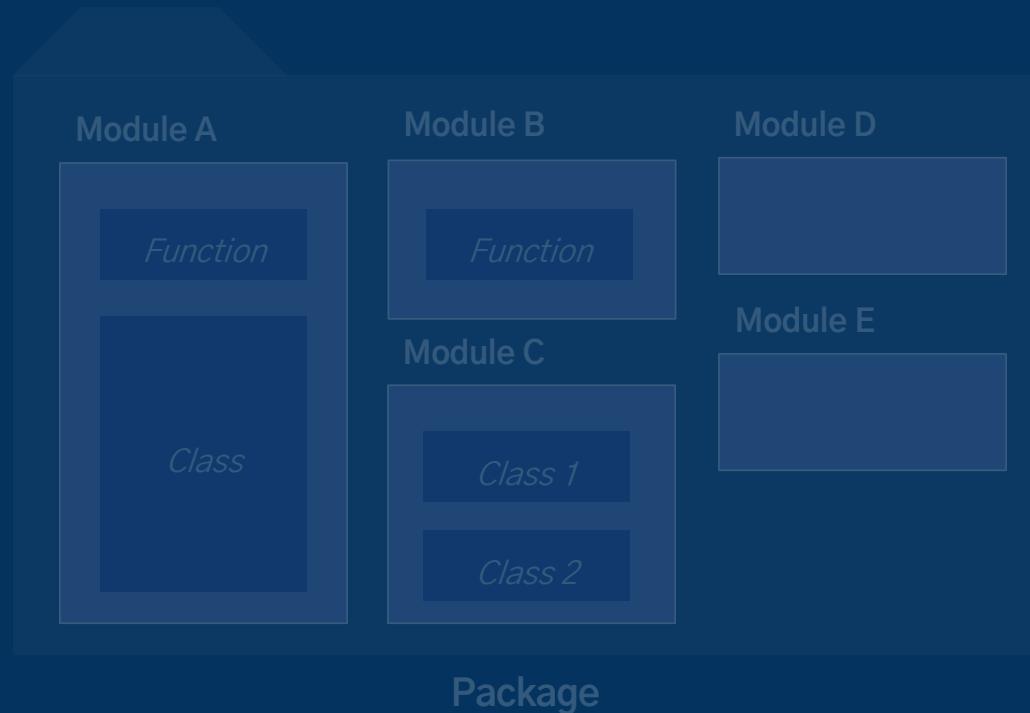
06

파일 다루기



모듈과 패키지

- 필요한 패키지를 불러와 사용할 수 있습니다.
- 나만의 함수를 만들어 활용할 수 있습니다.
- 파이썬 스크립트를 작성하여 데이터 입력하고 원하는 형태로 출력할 수 있습니다.



데이터 입력



```
def function(val):  
    return val  
print(function(8))  
  
class Object():  
    def method(self, val):  
        return val  
obj = Object()  
print(obj.method(8))
```

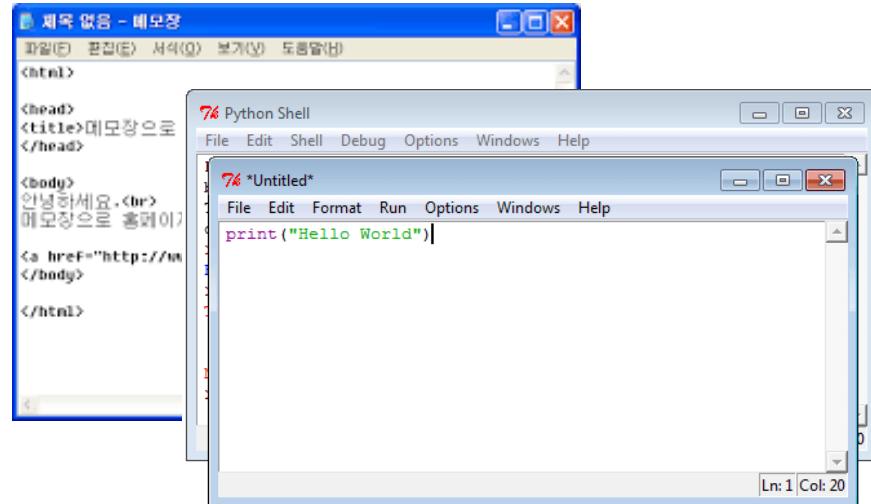


데이터 출력

텍스트 파일과 바이너리 파일

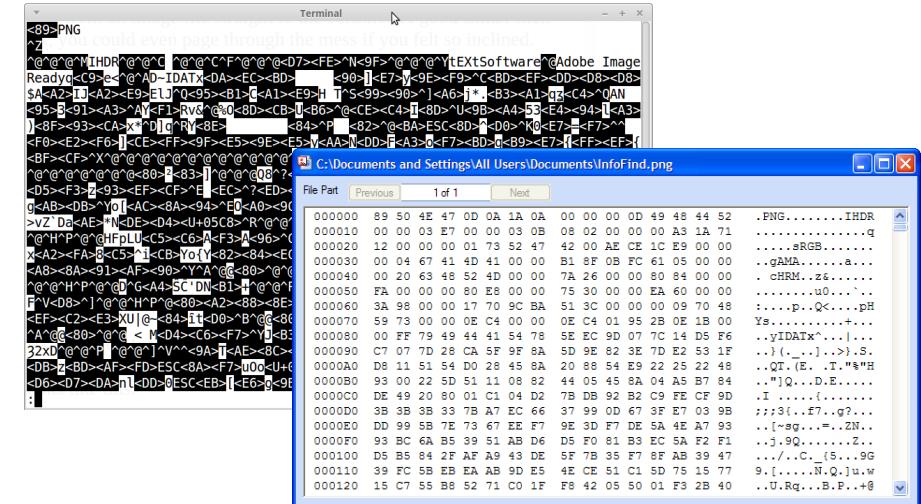
컴퓨터에 있는 모든 파일은 텍스트(text) 파일 혹은 바이너리(binary) 파일

텍스트(text) 파일



글자 정보를 저장 (ASCII, UniCode, UTF-8)

바이너리(binary) 파일



영상 정보 등 다양한 정보를
각각의 방식으로 기계어 저장



[파일 입출력] 파일 읽기 및 쓰기

두 가지 방식의 파일 입출력 방법이 있습니다.

with문을 사용할 경우 파일을 열고 닫음을 자동으로 처리할 수 있습니다.

```
# 파일 읽기
f = open('filename.txt', 'r') # 파일 열기
content = f.read() # 파일 읽기
f.close() # 파일 닫기
print(content) # 읽은 내용 출력
```

```
# with문으로 파일 읽기
>>> with open('filename.txt', 'r') as file:
... content = file.read()
... print(content)
```

```
# 파일 쓰기
>>> f = open('filename.txt', 'w') # 파일 열기
>>> f.write('insert line here') # 파일 쓰기
>>> f.close() # 파일 닫기
```

```
# with문으로 파일 쓰기
>>> with open('filename.txt', 'w') as file:
... file.write('insert line here')
```

[파일 입출력] 파일 쓰기 모드 ‘w’ (1/2)

파이썬에서 텍스트 파일을 읽고 쓰기 위해 먼저 파일을 생성해야 합니다.

파일 객체를 만들므로써 파일을 생성할 수 있습니다.

파일 객체 = open('파일 이름', '파일 열기 모드')

```
>>> f = open('new_file.txt', 'w')  
>>> f.close()
```

디렉토리에 'new_file.txt'라는 파일이 생성되어 있습니다.

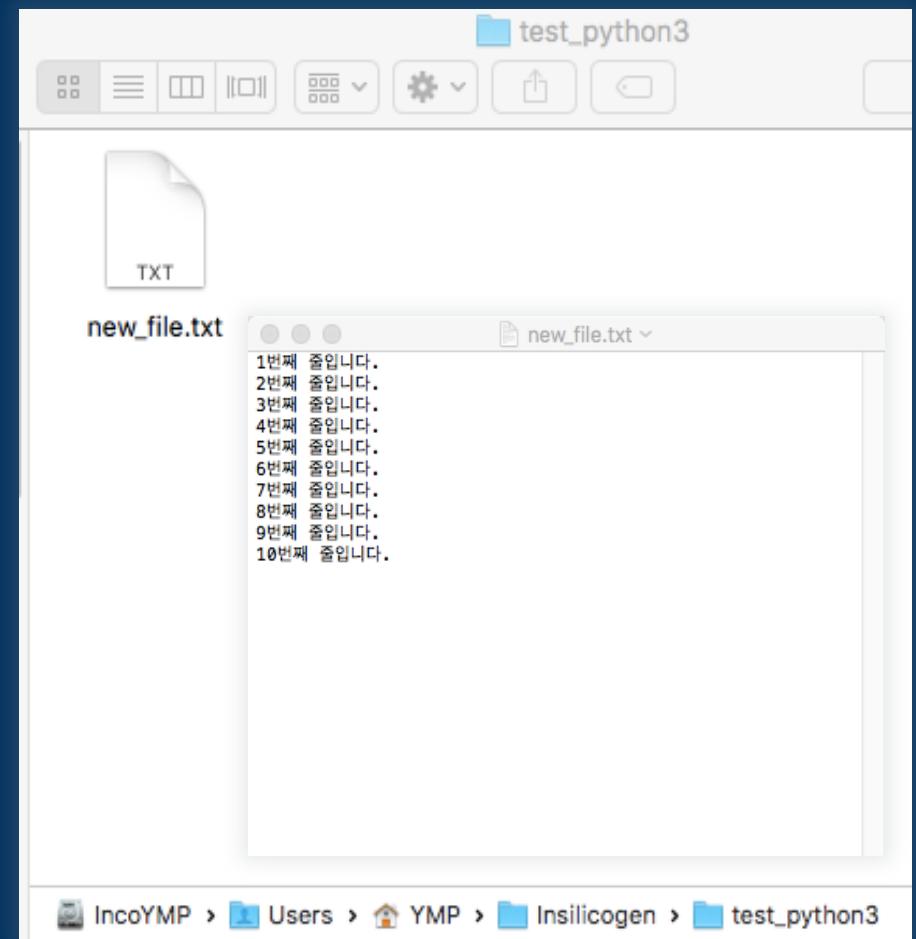
	파일열기 모드	설명
1	r	읽기 모드 (default)
2	w	쓰기 모드
3	a	추가 모드
4	x	쓰기 모드 (같은 파일명 존재시 에러)
5	t	텍스트 모드 (default)
6	b	바이너리 모드
6	+	읽고 쓰기 모드

위 모드를 필요에 따라 조합해 사용한다.
(예, wb: 바이너리 파일 쓰기 모드)

[파일 입출력] 파일 쓰기 모드 ‘w’ (2/2)

파일을 쓰기모드로 열어 출력값을 작성하는 방법입니다.

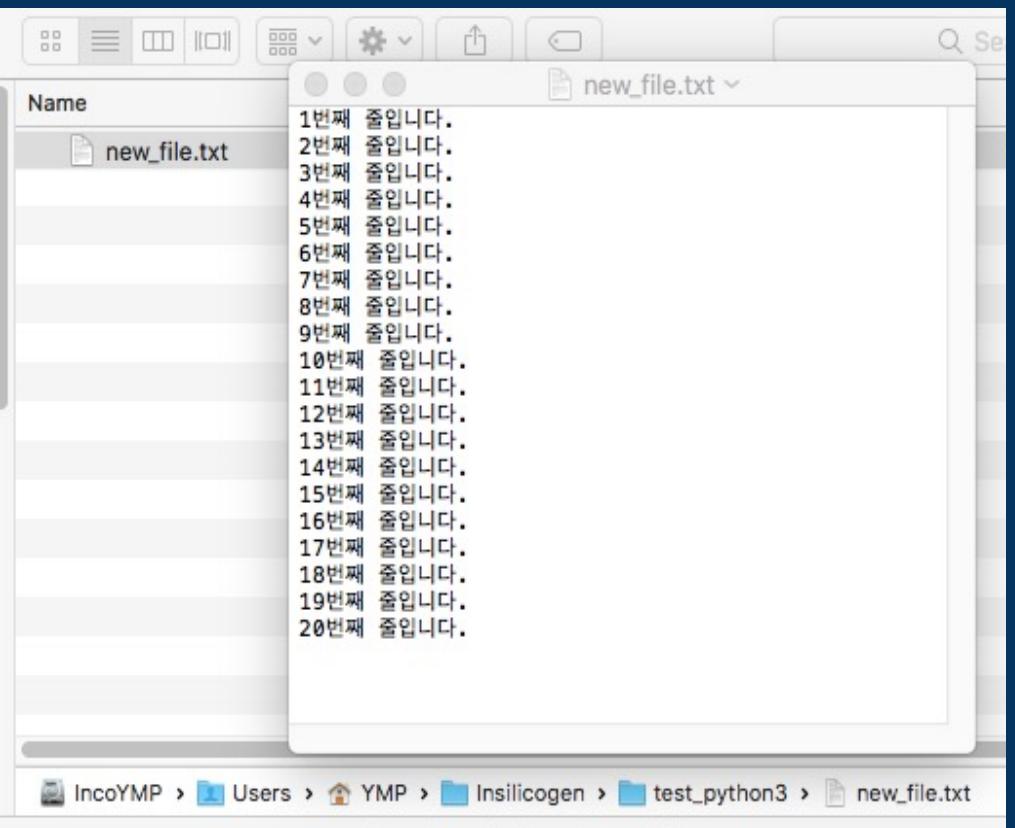
```
>>> filename = 'new_file.txt'  
>>> f = open(filename, 'w')  
>>> for i in range(1, 11):  
... data = f'{i}번째 줄입니다.'  
... f.write(data + '\n')  
...  
...  
>>> f.close()
```



[파일 입출력] 파일 추가 모드 'a'

파일의 추가모드 ('a')로 파일을 열고, 새로운 내용을 파일 끝에 추가할 수 있습니다.

```
>>> f = open(filename, 'a')
>>> for i in range(11, 21):
...     data = f'{i}번째 줄입니다.'
...     f.write(data + '\n')
...
>>> f.close()
```



[파일 입출력] 파일 읽기 모드 'r' (1/2)

1. readline()

readline() 함수를 사용하면 가장 첫 번째 줄을 읽습니다.

```
>>> f = open(filename, 'r')
>>> line = f.readline()
>>> print(line)
1번째 줄입니다.

>>> f.close()
```

모든 라인을 오른쪽과 같이 읽는다.

```
# 모든 라인을 읽을 경우
>>> f = open(filename, 'r')
>>> while True:
...     line = f.readline()
...     if not line:
...         break
...     print(line)
```

1번째 줄입니다.
2번째 줄입니다.
3번째 줄입니다.
4번째 줄입니다.
5번째 줄입니다.
6번째 줄입니다.
7번째 줄입니다.
8번째 줄입니다.
9번째 줄입니다.
10번째 줄입니다.

```
>>> f.close()
```

[파일 입출력] 파일 읽기 모드 'r' (1/2)

1. readline()

파일 객체를 `for`문으로 요소별 반복하면,
줄 단위로 사용할 수 있습니다.

```
# 모든 라인을 읽을 경우
>>> f = open(filename, 'r')
>>> for line in f:
...     print(line)
1번째 줄입니다.
2번째 줄입니다.
3번째 줄입니다.
4번째 줄입니다.
5번째 줄입니다.
6번째 줄입니다.
7번째 줄입니다.
8번째 줄입니다.
9번째 줄입니다.
10번째 줄입니다.
```

```
>>> f.close()
```

[파일 입출력] 파일 읽기 모드 'r' (2/2)

2. read()

read()함수는 파일의 내용 전체를 문자열로 리턴합니다.

```
>>> f = open(filename, 'r')
>>> data = f.read()
>>> print(data)
1번째 줄입니다.
2번째 줄입니다.
3번째 줄입니다.
4번째 줄입니다.
5번째 줄입니다.
6번째 줄입니다.
7번째 줄입니다.
8번째 줄입니다.
9번째 줄입니다.
10번째 줄입니다.

>>> f.close()
```

[표준 입출력] 외부 파일을 읽는 방법 (1/2)

표준 입력

- 같은 디렉토리에 test.txt 파일이 있다고 할 때, 이 파일의 내용을 읽는 방법은 아래와 같습니다.

```
>>> f = open('test.txt')  
>>> for line in f:  
...     print(line)
```

```
import sys  
for line in sys.stdin:  
    print(line)
```

위 파일을 test.py로 저장한 후에,

```
$ python test.py < test.txt
```

[표준 입출력] 외부 파일을 읽는 방법 (2/2)

표준 출력

- 디렉토리에 output.txt 파일을 새로 만들고, 결과를 저 파일에 쓰고자 할 때 사용합니다.

```
>>> f = open('output.txt', 'w')
>>> f.write('Hello world\n')
>>> f.write('My name is Chulsu\n')
>>> f.close()
```

```
import sys
sys.stdout.write('Hello world\n')
sys.stdout.write('My name is Chulsu\n')
```

위 파일을 test.py로 저장한 후에,

```
$ python test.py > output.txt
```

[표준 입출력] 가상 파일 객체

io.StringIO

- 실제 파일은 아니지만, 파일처럼 동작하는 객체로 다양한 용도로 활용 가능합니다.

```
from io import StringIO

handle = StringIO("""\
> test fasta
AGTCAGTC
AGTCCCCC
""")

for line in handle:
    print(line)
```

[팁] 명령줄 인자 파싱 argparse

argparse

- Python의 표준 라이브러리 중 하나로, 명령줄 인자를 파싱하는 데 사용됩니다.
- Python 스크립트에서 명령줄 옵션, 인자 및 서브 커맨드를 정의하고 처리할 수 있습니다.

```
import argparse

# ArgumentParser 객체 생성
parser = argparse.ArgumentParser(description="파일 내용을 출력하는 스크립트")

# 입력 파일 인자 추가 및 인자 파싱
parser.add_argument('input_file', type=str, help='읽을 파일 경로')
args = parser.parse_args()

# 파일 읽기
with open(args.input_file, 'r') as infile:
    content = infile.read()

# 파일 내용 출력
print(f"{args.input_file}의 내용:\n")
print(content)
```

파이썬으로 데이터 분석한다 하면,



NumPy C, Fortran으로 만들어진 고속자료구조 위에

Pandas R dataframe 같은 편리한 인터페이스를 올리고,

SciPy C, Fortran으로 만들어진 과학 함수 모음으로 과학연산하고,

Matplotlib MatLab 같은 차트 모듈으로 가시화하며,

scikit-learn 기계학습 알고리즘 모음으로 예측 알고리즘 만들고,

statsmodels 통계모델 모음으로 통계분석하고,

데이터를 **PyTables** HDF5 지원 자료구조에 저장하고 검색함.

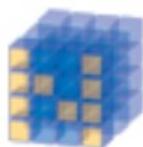
Data Analysis With Python

여기에, **관계형 데이터베이스** Oracle, MySQL, SQLite 나

NoSQL 데이터베이스 MongoDB 를 쓰고,

Django 웹프레임워크로 웹사이트 만들어 서비스

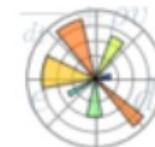
Basic packages



NumPy
Base
N-dimensional
array package



SciPy library
Fundamental library
for scientific
computing



Matplotlib
Comprehensive 2D
Plotting



IPython
Enhanced
Interactive Console



Sympy
Symbolic
mathematics



pandas
Data structures
analysis

Basic packages for data analysis and visualization

For Bioinformatics



pysam

Cytoscape

scanpy

pybedtools

anndata

HTSeq

Basic packages for Bioinformatics

Python 중급

01 객체 지향 프로그래밍

- 클래스
- 메소드
- 생성자
- 상속
- 터미널과 쉘
- 버전 관리
- 코드 편집기
- 파일 형식(.py / .ipynb)
- AI 코딩 어시스턴트

02 개발 환경

03 가상 환경

- 가상 환경
- 패키지 관리 시스템

04 AlphaGenome 및 API 실습

- AlphaGenome 소개
- API
- AlphaGenome API 실습

01

객체 지향 프로그래밍



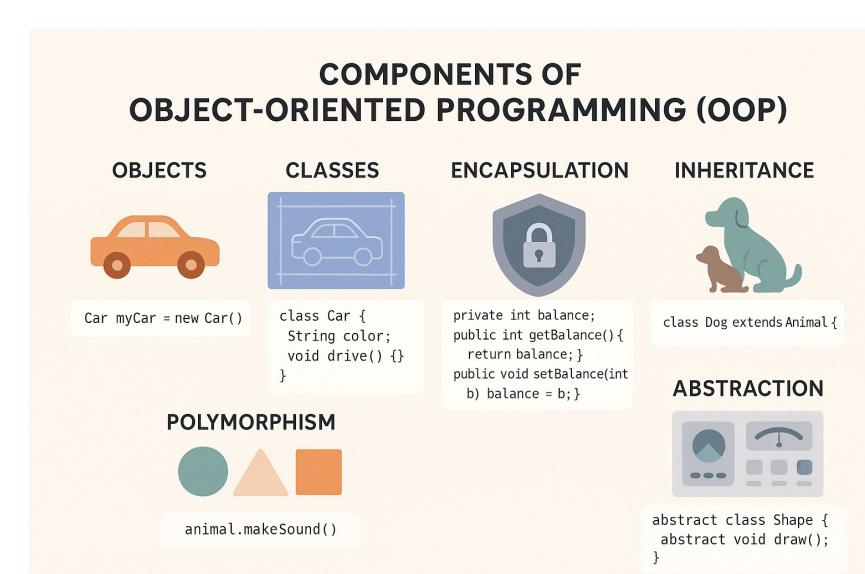
객체 지향 프로그래밍

객체 지향 프로그래밍(OOP)

- 프로그램 구현에 필요한 객체를 파악하고 각각의 객체들의 역할이 무엇인지를 정의하여 객체들 간의 상호작용을 통해 프로그램을 만드는 것
- 프로그램들이 커질수록 **이해하기 쉬운 코드**를 작성하는 것이 중요해짐
- 큰 프로그램을 작은 부분들로** 나눠서 코드를 고치거나 기능을 추가할 때 살펴야 할 코드의 양을 줄일 수 있음



<https://www.codestates.com/blog/content/%EA%B0%9D%EC%B2%B4-%EC%A7%80%ED%96%A5-%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D-%ED%8A%B9%EC%A7%95>



객체 지향 프로그래밍

객체(object)

- 객체 지향 프로그래밍의 가장 기본적 단위이자 시작점
- 기본 전제: 실제 세계는 객체로 구성되며 모든 사건은 객체들 간의 상호작용을 통해 발생
- 책상, 의자, 시계, 전등, 책 등 우리가 주변에서 흔히 볼 수 있는 모든 실재하는 대상
- 유형의 대상 뿐만 아니라 눈에 보이지 않는 논리, 사상, 철학, 개념 등 무형의 대상들도 포함할 수 있음



<https://www.codestates.com/blog/content/%EA%B0%9D%EC%B2%B4-%EC%A7%80%ED%96%A5-%ED%94%84%EB%A1%9C%EA%B7%B8%EB%9E%98%EB%B0%8D-%ED%8A%B9%EC%A7%95>

클래스

클래스(class)

- 클래스는 어떤 object의 속성이나 메소드, 연산 등을 정의
- 클래스는 객체의 청사진 혹은 공장으로 비유할 수 있음
- C언어에는 클래스가 없음
 - 프로그램 작성을 위해 반드시 필요한 요소는 아님
 - 키워드나 문법 요소를 제외한 파이썬의 모든 것은 객체이자 어떤 클래스의 인스턴스



클래스의 필요성

```
balance1 = 0
balance2 = 0
def deposit1(amount): # 장부1
    global balance1
    balance1 += amount
    return balance1
def deposit2(amount): # 장부2
    global result2
    result2 += amount
    return result2
print(deposit(3))
print(deposit2(3))
```

```
class Ledger:
    def __init__(self):
        self.result = 0 # 각 장부의 잔액
    def deposit(self, amount):
        self.result += amount
        return self.result
ledger1 = Ledger() # 계산기1
ledger2 = Ledger() # 계산기2

print(ledger1.deposit(3))
print(ledger1.deposit(4))
print(ledger2.deposit(3))
print(ledger2.deposit(3))
```

```
class Calculator:
    def __init__(self):
        self.result = 0
    def add(self, num):
        self.result += num
        return result
    def sub(self, num): # 뺄셈 추가
        self.result -= num
        return result
    def reset(self): # 리셋 추가
        self.balance = 0
        return result
```

<https://wikidocs.net/28>

...

© 2025 INSILICOGEN, INC. ALL RIGHTS RESERVED. 122

메소드(instance method)

메소드(method)

- 클래스 안에서 정의되며 특정한 객체에서 동작하는 함수
- Self 인자를 첫 매개변수로 받으며 self를 통해 객체의 데이터에 접근하고 변경함
- 각 객체는 자신의 독립적인 데이터를 가지며 그것을 활용하는 함수

```
class BankAccount:  
    def __init__(self, balance):  
        self.balance = balance  
    def deposit(self, amount): # instance method  
        self.balance += amount # modifies THIS object's balance  
        return self.balance  
  
account1 = BankAccount(100)  
account2 = BankAccount(200)  
account1.deposit(50) # only changes account1  
# account1.balance is now 150  
# account2.balance is still 200
```

생성자

__init__ 생성자

- 객체를 생성할 때 자동으로 실행되는 메소드

```
class Dog:  
    def __init__(self, name, age):  
        self.name = name  
        self.age = age  
  
my_dog = Dog("Buddy", 3)
```

Self 인자

- 파이썬에서는 instance 메소드를 실행할 때마다 첫 번째 인자로 instance 자체를 받음
- 하나의 클래스에 대해 여러 객체를 만들 수 있기 때문에 특정 instance를 지칭하는 표현을 할 필요성이 있음

```
class Dog:  
    def __init__(self, name, age):  
        self.name = name # Store data on this instance  
        self.age = age  
  
    def bark(self):  
        print(f"{self.name} says woof!") # Access this instance's data  
  
buddy = Dog("Buddy", 3)  
buddy.bark() # Python automatically passes buddy as self
```

<https://wikidocs.net/28>

클래스 메소드(class method)

클래스 메소드(class method)

- 개별 객체가 아닌 **클래스 자체에서 작동**하는 메소드
- 첫 매개변수로 클래스를 받으며 일반적으로 ‘**cls**’로 지칭함
- **클래스 수준의 속성**이나 **데이터**를 다루고 싶을 때 사용

```
class BankAccount:  
    interest_rate = 0.05 # shared by all accounts  
    def __init__(self, balance):  
        self.balance = balance  
        print(f"__init__ called with balance: {balance}")  
    @classmethod  
    def set_interest_rate(cls, new_rate): # class method  
        cls.interest_rate = new_rate  
    @classmethod  
    def from_euros(cls, euros): # factory method  
        print(f"from_euros called with euros: {euros}")  
        dollars = euros * 1.1  
        print(f"Converting to dollars: {dollars}")  
        return cls(dollars) # this calls __init__
```

```
print("Creating account directly:")  
account1 = BankAccount(100)  
print("\nCreating account from euros:")  
account2 = BankAccount.from_euros(100)  
print(f"\nAccount2 balance: {account2.balance}")  
  
Creating account directly:  
__init__ called with balance: 100  
  
Creating account from euros:  
from_euros called with euros: 100  
Converting to dollars: 110.0  
__init__ called with balance: 110.0  
  
Account2 balance: 110.0
```

정적 메소드(static method)

정적 메소드(static method)

- 개념적으로 클래스가 가지는 능력이기는 하나 **인스턴스나 클래스 데이터에 대해 독립적**일 때 사용
- Validation function 혹은 helper function으로 사용
- Validation function**: 데이터가 특정 형식을 갖추었는지 검증하는 함수
- Helper function**: 다른 함수 내부에서만 쓰이는 좀 더 복잡한 일을 수행하는 것을 **돕는 함수**

Validation function 예시

```
class User:  
    @staticmethod  
    def is_valid_email(email):  
        return '@' in email and '.' in email  
    @staticmethod  
    def is_strong_password(password):  
        return len(password) >= 8 and  
               any(c.isdigit() for c in password) # Usage  
  
if User.is_valid_email("test@example.com"):  
    user = User(email)
```

Helper function 예시

```
class Temperature:  
    def __init__(self, celsius):  
        self.celsius = celsius  
    def get_fahrenheit(self):  
        return self.celsius_to_fahrenheit(self.celsius)  
    @staticmethod  
    def celsius_to_fahrenheit(celsius):  
        result = celsius * 9/5 + 32  
        return result  
  
temp_f = Temperature.celsius_to_fahrenheit(25)
```

상속

상속(inheritance)

- 새로운 클래스를 만들 때 기존의 클래스의 능력을 모두 상속받게 할 수 있음

```
class Animal:  
    def __init__(self, name):  
        self.name = name  
    def speak(self):  
        return "Some sound"  
  
class Dog(Animal):  
    def speak(self):  
        return f"{self.name} says Woof!"  
  
class Cat(Animal):  
    def speak(self):  
        return f"{self.name} says Meow!"
```

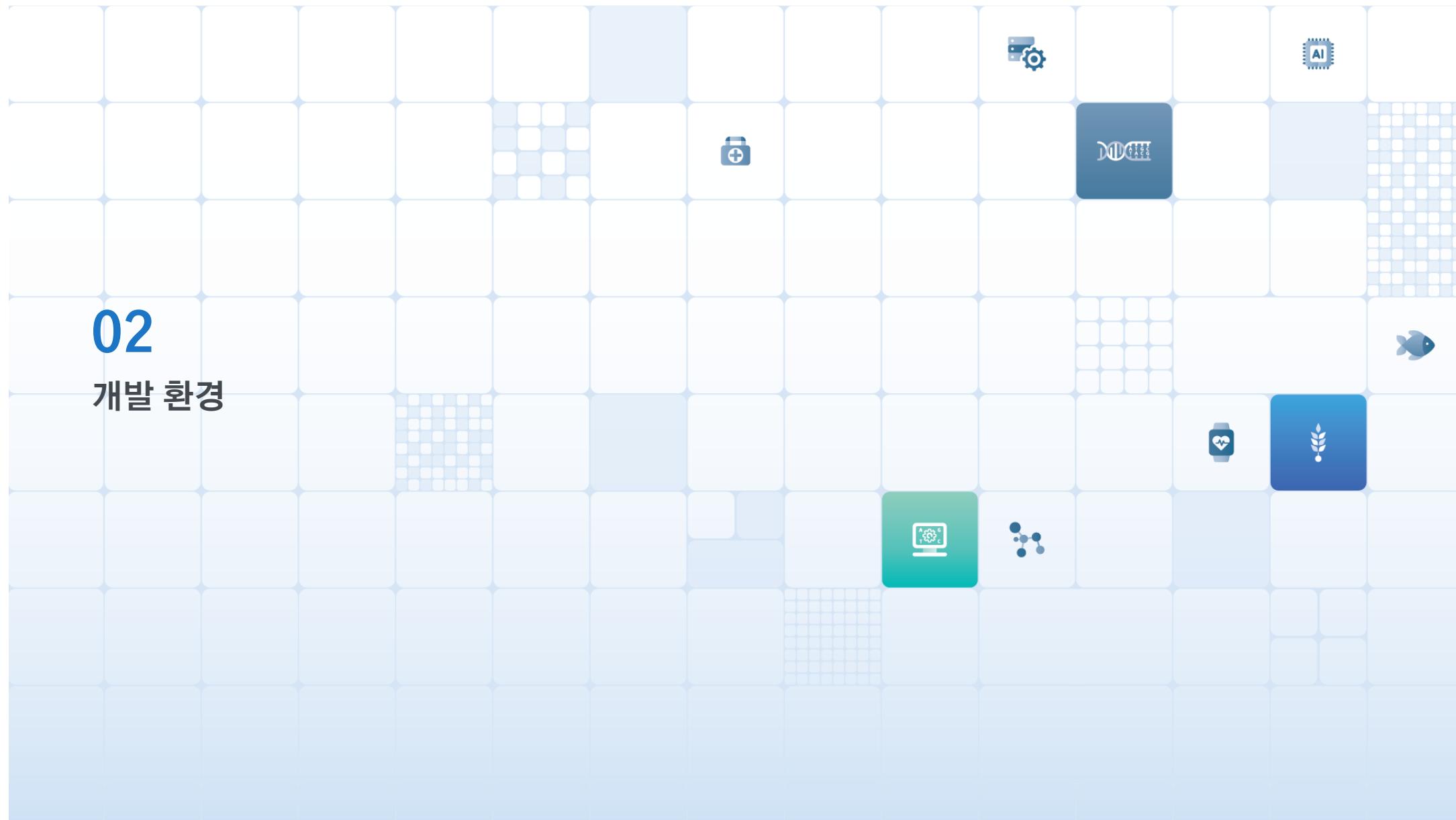
```
# Usage  
dog = Dog("Buddy")  
cat = Cat("Whiskers")  
print(dog.speak()) # Output: Buddy says Woof!  
print(cat.speak()) # Output: Whiskers says Meow!
```

```
class Vehicle:  
    def __init__(self, brand, year):  
        self.brand = brand  
        self.year = year  
    def info(self):  
        return f"{self.year} {self.brand}"  
  
class Car(Vehicle):  
    def __init__(self, brand, year, doors):  
        super().__init__(brand, year)  
        self.doors = doors  
    def description(self):  
        return f"{self.info()} with {self.doors} doors"
```

```
class Motorcycle(Vehicle):  
    def __init__(self, brand, year, engine_cc):  
        super().__init__(brand, year)  
        self.engine_cc = engine_cc  
    def description(self):  
        return f"{self.info()} with {self.engine_cc}cc engine"
```

02

개발 환경



개발 환경

개발 환경

- 개발자가 코드를 작성·실행·관리하기 위해 사용하는 모든 요소의 뮤음
- 개발을 하기 위한 도구(코드 편집기 / 통합 개발 환경 / 런타임)를 비롯하여 개발 형식(.py, .ipynb, 프로젝트 구조)까지
아우르는 말

The screenshot shows the VS Code interface with the following details:

- EXPLORER**: Shows the project structure for "CREATE-REACT-APP".
- EDITOR**: Displays two files: "App.js" and "serviceWorker.js". "App.js" contains React code, and "serviceWorker.js" contains service worker logic.
- TERMINAL**: Shows the output of a build command: "Compiled successfully!" followed by deployment URLs.

vscode (코드 편집기)

The screenshot shows the Cursor application interface with the following details:

- File Explorer**: Shows a file tree with files like ".cursor_rules.mdc", "prd.txt", and "example_prd.txt".
- Terminal**: Shows a terminal window with command history and environment information.
- Chat**: Shows a chat interface with a message about rule structure and a "New chat" button.

Cursor (코드 편집기)

<https://github.com/microsoft/vscode>

python 설치 개요

1단계: Python 다운로드

1. 공식 홈페이지 접속: [python.org/downloads](https://www.python.org/downloads/) 로 들어갑니다.
2. 다운로드: 화면 중앙에 보이는 노란색 버튼 *[Download Python 3.x.x]*를 클릭하여 설치 파일을 받습니다.

2단계: 설치하기 (⚠ 가장 중요!)

다운로드한 파일을 실행하면 설치 창이 뜹니다. 여기서 바로 'Install Now'를 누르지 마시고, 아래 내용을 먼저 확인하세요.

1. 체크박스 체크: 설치 창 하단에 있는 `Add python.exe to PATH` (또는 `Add Python to PATH`) 체크박스를 반드시 체크해주세요.
 - 이것을 체크해야 윈도우의 어느 곳에서든(VS Code 포함) 파이썬을 바로 불러올 수 있습니다.
2. 설치 시작: 체크를 마쳤다면 위쪽의 *[Install Now]*를 클릭하여 설치를 진행합니다.

3단계: 잘 설치됐는지 확인하기

설치가 끝나면 제대로 됐는지 확인해봐야겠죠?

1. 키보드의 `Windows 키 + R` 을 누르고, 입력창에 `cmd` 라고 치고 엔터를 칩니다. (검은색 명령 프롬프트 창이 뜹니다.)
2. 창에 아래 명령어를 입력하고 엔터를 쳐보세요.

DOS

```
python --version
```

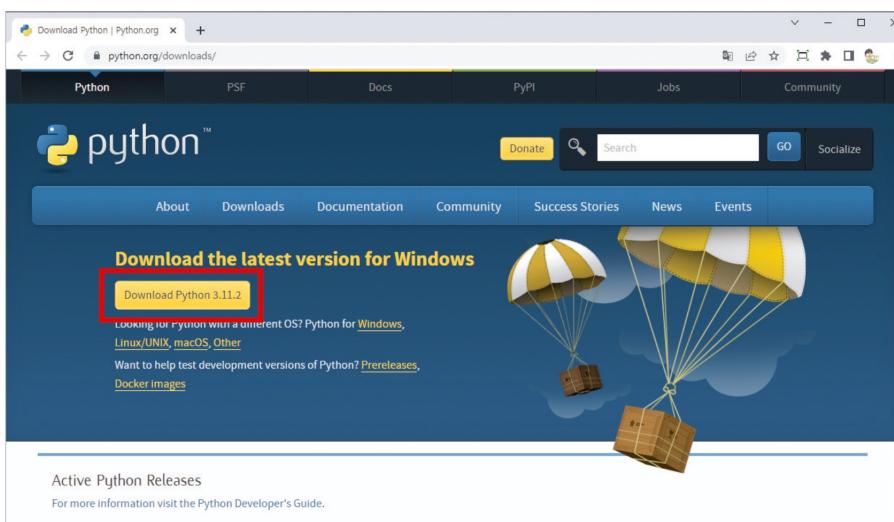
3. `Python 3.12.x` 처럼 버전 숫자가 잘 나오면 성공입니다!

개발 환경

python 설치: 윈도우

윈도우에서 파이썬 설치하기

- 먼저 파이썬 공식 홈페이지의 다운로드 페이지(www.python.org/downloads)에서 윈도우용 파이썬 언어 패키지를 내려받는다. 다음 화면에서 Python 3.x로 시작하는 버전 중 가장 최신의 윈도우 설치 파일을 내려받자(이 글을 작성하는 시점의 최신 버전은 3.11.2이다).



- 설치 파일을 실행한 후 [Install Now]를 클릭하면 설치가 진행된다. 이때 파이썬이 어느 곳에서든지 실행될 수 있도록 [Add python.exe to PATH] 옵션을 반드시 선택해야 한다.

2. 설치 파일을 실행한 후 [Install Now]를 클릭하면 설치가 진행된다. 이때 파이썬이 어느 곳에서든지 실행될 수 있도록 [Add python.exe to PATH] 옵션을 반드시 선택해야 한다.

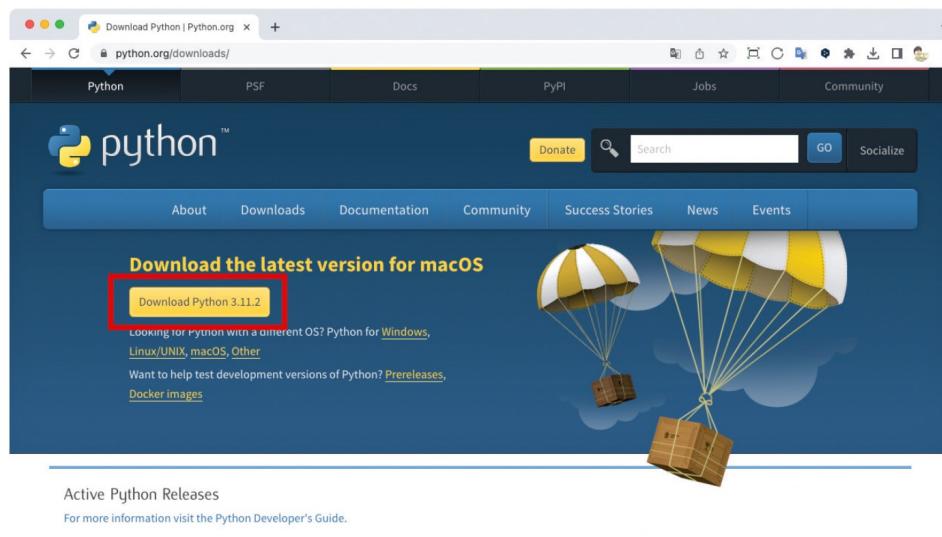


개발 환경

python 설치: macOS

맥에서 파이썬 설치하기

- 파이썬 공식 홈페이지(www.python.org)에서 [Downloads] 메뉴를 클릭하여 맥(Mac)용 파이썬 설치 파일을 내려받은 후 다음 화면에서 [Download Python 3.11.x]를 클릭하면 된다.



- 그런 다음 내려받은 python-3.11.x-macos1.pkg 파일을 실행하여 설치한다.

- 설치가 완료되면 파이썬이 제대로 설치됐는지 확인하기 위해 터미널에서 다음과 같이 명령을 입력해 자신의 맥에 설치된 파이썬 버전을 확인해 보자. 구 버전의 맥에는 파이썬 2.7 버전이 기본으로 설치되어 있기 때문에 python을 입력하면 파이썬 2.7이 실행된다. 맥에서는 항상 'python' 대신 'python3' 명령을 사용하자. 파이썬 버전이 제대로 출력되면 성공적으로 설치한 것이다.

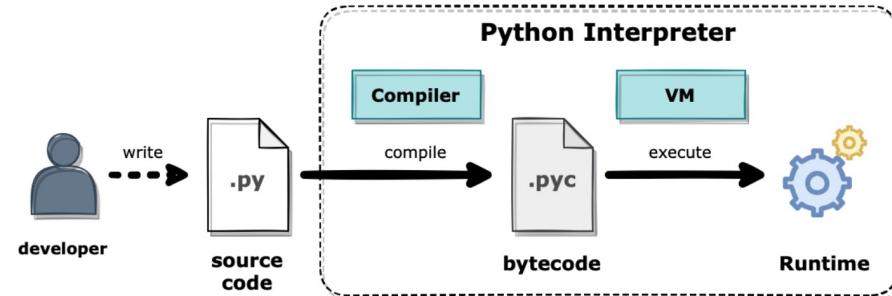
```
pahkey@mymac ~ % python3 -V  
Python 3.11.2
```

```
Last login: Sat Apr 15 10:13:55 on ttys000  
pahkey@mymac ~ % python3 -V  
Python 3.11.2  
pahkey@mymac ~ %
```

개발 환경

.py 파일 (python script)

- 일반적인 Python 코드 파일
- 특징: 한 줄 한 줄 순서대로 실행되는 코드이며 순수 텍스트 기반
- 프로그램, 함수, 모듈 작성에 적합함
- 어플리케이션, 실제 배포하는 프로젝트 등에 용이



.ipynb 파일 (Jupyter Notebook)

- 인터랙티브 노트북 파일(환경)
- 짧은 코드를 빠르게 테스트하고 결과를 확인해볼 때 용이
- 파일 내부가 JSON 구조라 사람이 직접 읽기 어려움
- 결과물이 저장되어 용량이 커질 수 있음
- 코드, 그래프, 표, 문서를 섞어서 만드는 분석 보고서 느낌
- ‘프로그램 파일’로 쓰기에는 구조적으로 부적합

colab jupyter

```
{
  "cells": [
    {
      "cell_type": "code",
      "execution_count": 21,
      "metadata": {
        "collapsed": false
      },
      "outputs": [
        {
          "data": {
            "image/png": "iVBORw0KGgoAAAANSUhEUgAAAAEACAYAAASnvebAAAABHNCSVQICAgfAhkiAAAAAlxSFlz\nnAAALEgAACxIB0t1+/AAAIAjBjRE!\n/HvTg1JC5QbuQAlRggsvkWAMSEQoPfEh\n/n6hEX6u1+N7vE8jvqElp4qcrwgHin2Bykb2hni1ndFFEAERJaijReSiJiAwgQBA8t8!\n\\napl2kpLf2z2e2zGyeJ3u8P8ub37of\\RhrLSi1m1QjGAER5dQpvE2ElpFAMEYlAcMsrRkQ1k8BYR\n\\niUARhxDRCPT18DbGzDDG5b7jNpV77i1jzBfGmM+M48YYxJz2pg1lLeqc9y82wID!npvEKhctbYb\n\\n8Ckw5LdG3i1ilatyeFtvVw7j3puib7w2o7hxODb1I5QeqEz0+z2gYRDF0REKh0u8DbG2AJAH\\nrLWzg\n/F+1iJyjYHVFOu2k3AKDvsY7Tjioi1qfBmuoFu5U296m5w9NGYghB411p7Ao0qTr\\nsce+Ohr+jxxz4\n/fXb6/C1z62SFx2VO2angbGa1cK4x7ct7m824DmgfLDb6fgMcbeq7yci1qe+nywWjtzbjOE/PDOJYRE5ks1slmcsv09PRQDyGi6fM7ffr:\n/Wc7udnTSpbMxxtbW7xRgSuwNmldgnLAUE2EwoPAMEYlAcMsrRkQ1k8BYR\\nAURkBGCRCTwPhGJQApvPANEYlAcMsrRkQ1k8BaJ+\n/Iv+vNeeI1lQKXhLhRvFleurTsQub5zN7Ar\\hnKaobdoSlpIV4vNf4iEST\n\\nY2vNj3Fg2kPkrk0k35mxg6eygvDnnR5LjkuA9Papk2phKJ1P5CP511Mxndt792vz1410buajV\\nRaEelQgbw1Ugckxy\\z52v303m\n\\ppgOX+kHhLRKmjdyc1P+Jea9MvNjD23kj+\\f\\nRKLQPD18Jb\\pYlpqdd3jy2jezf7o+\\vz61Imx2z5tANf1bsqKOV2qb0WvSMBr\n\\nz4tDX10bxC66ty\\vHk3CEHXRXUJH\\nImhueDUJwKvNxUqglq14YegLFTbXaeu4K6fFn41yeLo\n\\nmsRf6GfCoqmsz13H\\n3MKVXPHKbUkq11gYkRv+TPPHBE2Dh6PFpkwyXKPx5Rkne113V32baI\\nPgkz2f5sOkztwIxhMxhWxWgKQa:\n\\6pdutj1lh\\nK3xmvg9ixTY8wY6y/jDHg2OSgjVgkf1rN+hyqqpbLt\\G/u\\W3/M6+q4JeBUo+2wICjnnSYGRKPP9YBj\\wsr1Gj1\n\\nsp1SwRM5QXmWZs\\nKmFehtrc03xrQGfnba8y5b1Wb1jyAN8t1WkqaahR\\nKwCaRc01u1DwGvz\\ngyVfD+R0+23EOO6Fczzn+HN\n\\Tn9PR00tFTg\\/zrCrKb27mHnf3Y+eoS2WErgcioclycqRrljba1H2H2A5\\Ph8/locxlax5MvgFPrtyclya23Ns5b1sZS\n\\nrwCe9yCcUwbo1x52x12V3cf979+\\t1UEf1J6hhrqEc0WDVJgbK+\\d4NsA\\nNhdaoxQxB82z\\nPLZCn94wY3cs\n/Aepg6cquCWU3YqSwN6uBc40xucaYscAfgf7GmH8B58FpFyAh13an1qzI\\nMG7BOP37M9vV\n/yW3/f+Ps+te+mYDlzK2KrcetMyp5qV+QxiJSp3i2v+wHDzh1W2nExnsXKR\\nswlNhlnzDA/0eIC3t7zNtKHTKnTgI1WhvUlEasj+I,
```

JSON 예시

개발 환경

.py 파일 (python script)

```
# 비교할 모델 리스트 (하드코딩)
MODELS = [
    "gpt-4o-mini",
    "gpt-4o-mini-search-preview",
    "biomedgpt",
    "incoagent_251103",
]

# 디렉토리 경로
BASE_DIR = Path(__file__).parent.parent
DATA_DIR = BASE_DIR / "data" / "test_data" / "cleaned_data"
REFERENCE_FILE = "reference.jsonl"

# OpenAI 클라이언트 초기화
client = OpenAI(api_key=os.getenv("OPENAI_API_KEY"))

# 토크나이저
ENC = tiktoken.get_encoding("cl100k_base")

# 캐시
_EMBED_CACHE = {}
_API_CALL_COUNT = 0

# ===== 유ти리티 함수 =====

def chunk_by_tokens(text, chunk_tokens=3500, overlap=0.2):
    """텍스트를 토큰 단위로 청크 분할"""
    ids = ENC.encode(text)
    total_tokens = len(ids)

    if total_tokens <= chunk_tokens:
        return [text]

    step = int(chunk_tokens * (1 - overlap))
    chunks = []

    for i in range(0, total_tokens, step):
        start = i
        end = min(i + step, total_tokens)
        chunk = text[start:end]
        chunks.append(chunk)

    return chunks
```

.ipynb 파일 (Jupyter Notebook)

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns

# Load the sbert_results.csv
df = pd.read_csv('sbert_results_final_with_best.csv')

# Display basic info about the dataset
print("Dataset shape:", df.shape)
print("\nColumn names:")
print(df.columns.tolist())
print("\nFirst few rows:")
df.head()
```

[12]

... Dataset shape: (50, 8)

Column names:

['id', 'gpt-4o-mini-search-preview_final', 'incoagent_251113_final', 'incoagent_251114_final', 'incoagent_251115_final']

First few rows:

	id	gpt-4o-mini-search-preview_final	incoagent_251113_final	incoagent_251114_final	incoagent_251115_final
0	1	0.717005	0.748810	0.767550	0.745702
1	2	0.749156	0.836971	0.804142	0.779636
2	3	0.760027	0.766220	0.774220	0.752655

터미널과 쉘

- **터미널:** 컴퓨터와 사용자간 소통을 위한 **인터페이스**로 GUI (Graphical User Interface)를 사용하지 않고 컴퓨터에서 작업을 수행하고 자동화 할 수 있음
 - **Console, command line**과 혼용하여 부르기도 함
 - **대부분의 IDE 혹은 코드 편집기**는 터미널을 열 수 있는 기능을 포함
- **쉘:** 터미널에 입력되는 명령어들을 parsing 하여 **프로그램을 실행시키는 프로그램** (ex. **bash, zsh, Powershell etc.**)

A screenshot of a macOS terminal window. At the top, it shows three circular icons and the text "⌘1". To the right, it displays the user's name "insilicogen@MacBook-Pro:~". Below this, there is a status bar with the text "Click here to configure status bar". The main area of the terminal shows the following text:
Last login: Wed Oct 29 14:02:11 on ttys011
/Users/insilicogen/.zprofile:7: no such file or directory: /usr/local/bin/brew
[>] []

A screenshot of a Windows Command Prompt window. The title bar reads "Command Prompt". Below it, the window displays the following text:
Microsoft Windows [version 10.0.22623.1095]
(c) Microsoft Corporation. Tous droits réservés.
C:\Users\USER>

개발 환경

쉘(bash, zsh) 명령어 예시

ls

`ls` 는 `list` 의 약자로, 현재 디렉토리에 존재하는 파일과 폴더명을 보여준다. 추가 매개변수에 따라 다른 옵션을 설정할 수 있다.

- `-l` : `long` 의 약자로 파일형태, 사용권한, 하드링크번호, 소유자, 그룹, 파일크기, 시간, 연도, 파일명 순으로 자세한 정보를 표시한다.
- `-a` : `all` 의 약자로 현재 디렉토리의 숨겨진 파일을 포함한 모든 파일을 보여준다.

mkdir

```
1 # 현재 디렉토리에 new_dir 폴더를 생성
2 mkdir new_dir
3
4 # 디렉토리를 중첩해서 만들기 위해선 -p 옵션 사용
5 mkdir -p nested_dir/subdir1/subdir2
```

BASH

cd [path]

`cd` 는 `change directory` 의 약자로, 매개변수로 주어진 경로로 디렉토리를 이동한다.

```
1 # 현재 디렉토리에 있는 workspace 폴더로 이동
2 cd workspace
3
4 # 상위 폴더로 이동
5 cd ..
6
```

rm

`rm` 은 `remove` 의 약자로, 대상 파일 또는 디렉토리를 삭제한다.

```
1 # 현재 디렉토리에 있는 file1.txt 를 삭제
2 rm file1.txt
3
4 # 디렉토리를 삭제할 때는 -r (recursive) 옵션을 사용하며, 내부 디렉토리를 모두 삭제한다.
5 rm -r dir2
```

cp

```
1 # file1.txt 를 dir1 에 복사
2 cp file1.txt dir1/
3
4 # file1.txt 를 현재 디렉토리에 file2.txt 로 복사
5 cp file1.txt file2.txt
```

BASH

mv

`mv` 는 `move` 의 약자로, 현재 디렉토리에 존재하는 파일을 대상 경로에 이동한다.

```
1 # file1.txt 를 dir1 에 이동
2 mv file1.txt dir1/
3
```

BASH

<https://han-joon-hyek.github.io/posts/linux-terminal-command/>

개발 환경

버전 관리 시스템

Git & Github: 가장 보편적으로 묶어서 사용하는 버전 관리 툴

- Git: 버전 관리 시스템
- GitHub: Git으로 관리하는 프로젝트를 웹에도 올릴 수 있게 해주는 서비스

기본적인 Git 명령어

- git clone <repo-url>
- git add <file>
- git commit -m "<commit message>"
- git status
- git log <options>
- git switch (checkout) <branch-name>
- git restore <file>
- git fetch
- git pull
- git push



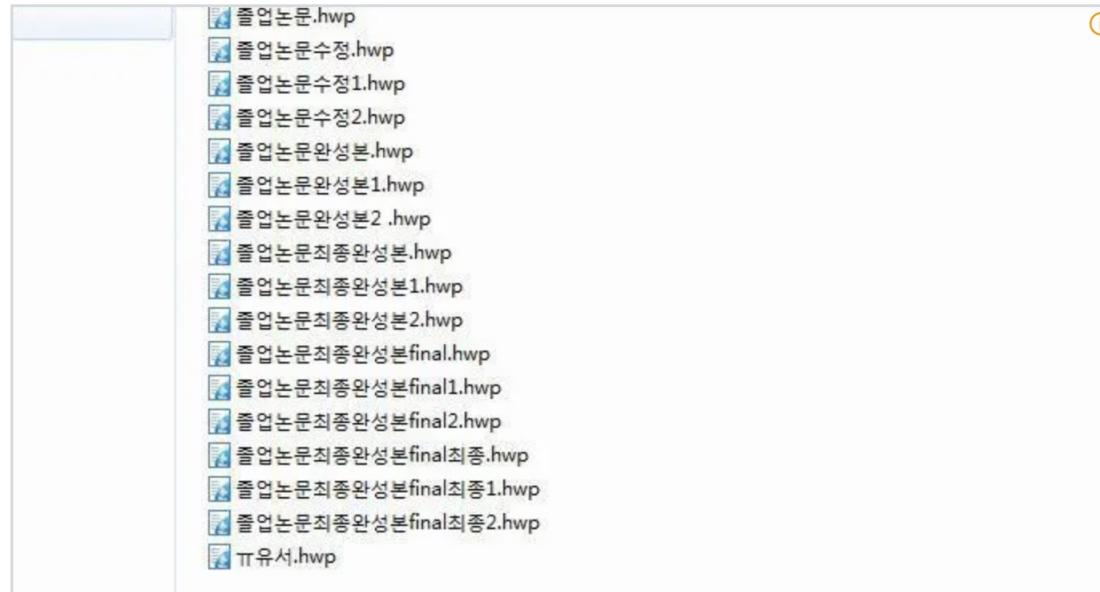
오픈 소스 프로젝트를 학습만 하고 변경은 안할 것이다: 터미널에서 git clone

코드베이스를 수정하거나 오픈소스에 기여하고 싶다: 내 저장소로 fork 이후 git clone

<https://han-joon-hyeok.github.io/posts/linux-terminal-command/>

개발 환경

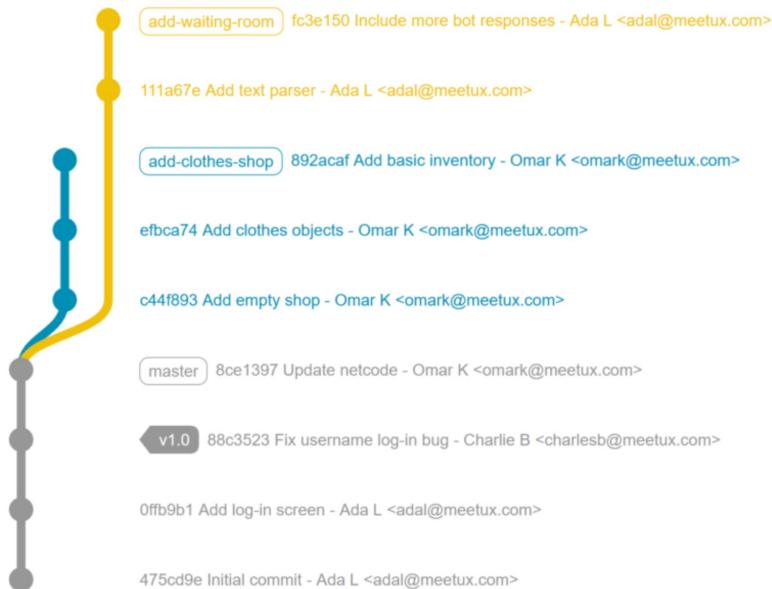
버전관리를 하지 않으면,



버전관리 없이 문서를 작성한 경우

개발 환경

버전관리를 하면,



히스토리 관리, 주요 변경 내용을 한눈에 파악

Changes 7

Showing 7 changed files ▾ with 458 additions and 645 deletions

Expand all Hide whitespace changes Inline Side-by-side

+1 -1 View file @ 9e485fbe

CLAUDE.md

```
... ... @@ -76,7 +76,7 @@ python manage.py collectstatic # Collect static files
76 76 ##### VLLM Server (Required)
77 77 ````bash
78 78 # Start vLLM server with GPT-OS2-20B model (run this first)
79 79 - VLLM_ATTENTION_BACKEND=TRITON_ATTN_VLLM_V1 vllm serve openai/gpt-oss-20b --port 8000
+ VLLM_ATTENTION_BACKEND=TRITON_ATTN_VLLM_V1 vllm serve openai/gpt-oss-20b --port 8000 --gpu-memory-utilization
0.5
80 80
81 81 # In separate terminal: Start FastAPI LLM server
82 82 cd incoagent/llm-server/
...
```

+6 -33 View file @ 9e485fbe

incoagent/llm-server/llm/intent_classifier.py

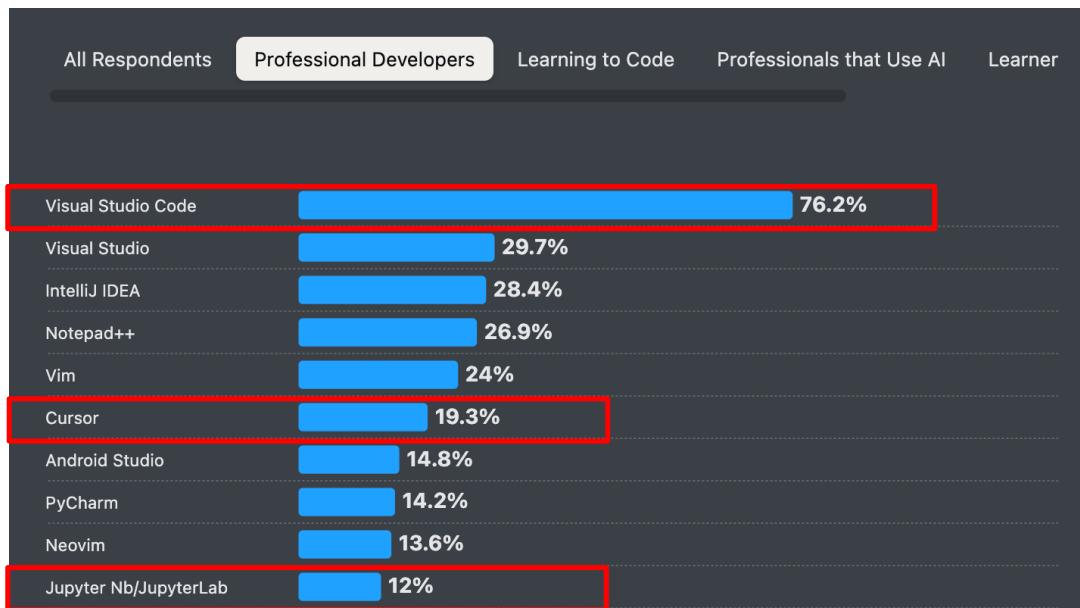
```
... ... @@ -8,8 +8,8 @@ whether a user is asking about accessible resources or general chat.
8 8 import logging
9 9 import json
10 10 from typing import Dict, List, Optional, Tuple
11 11 - from .loader import get_current_model, load_model
12 12 - from .chat_template import apply_chat_template_if_available
11 11 + from .vllm_client import VLLMClient
```

상세한 변경 내용까지 확인 및 추적 가능

개발 환경

코드 편집기

- 코드를 작성하고 읽기 쉽게 도와주는 소프트웨어
- 오류를 찾기 쉽게 해줌
- 파일 구조를 한눈에 볼 수 있음
- VS Code, PyCharm, Sublime Text, Cursor etc.



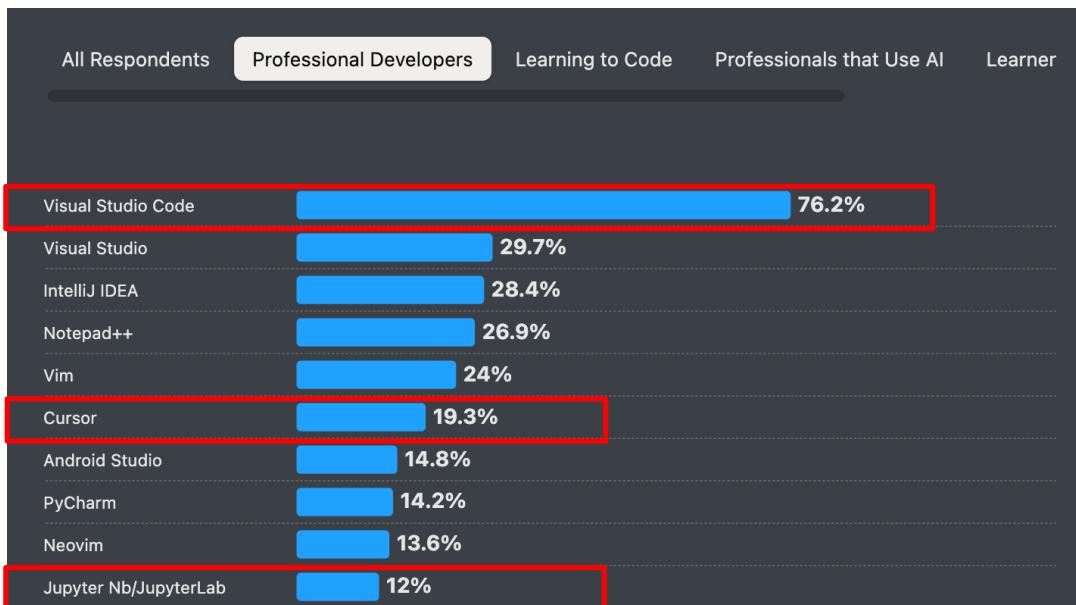
A screenshot of the Visual Studio Code (VS Code) interface. The left sidebar shows the file structure of a TypeScript project with folders like 'src', 'vs', 'base', 'common', and 'actions.ts'. The main editor area displays a portion of the 'arrays.ts' file, which contains code related to array operations. The bottom status bar shows the path 'C:\code\vscode [main □]'.

<https://survey.stackoverflow.co/2025/technology#most-popular-technologies-dev-envs-prof>

개발 환경

코드 편집기

- 코드를 작성하고 읽기 쉽게 도와주는 소프트웨어
- 빠르게 쓰고 고치기 만들며 오류를 찾기 쉽게 해줌
- 파일 구조를 한눈에 볼 수 있음
- VS Code, PyCharm, Sublime Text, Cursor etc.



Google Antigravity

Experience liftoff with the next-generation IDE

Download for MacOS Explore use cases

<https://survey.stackoverflow.co/2025/technology#most-popular-technologies-dev-envs-prof>

개발 환경

vscode 설치

1

Google

vscode 설치

X | ⌨ | 🎤 | 📸 | 🔎

AI 모드 전체 동영상 이미지 쇼핑 짧은 동영상 뉴스 더보기 ▾ 도구 ▾

See detailed insights & Compare multiple related Papers for :
“vscode 설치”

Compare insights ↗

2

Visual Studio Code
<https://code.visualstudio.com> › download :

Download Visual Studio Code - Mac, Linux, Windows

Visual Studio Code is free and available on your favorite platform - Linux, macOS, and Windows.
Download Visual Studio Code to experience a redefined code ...

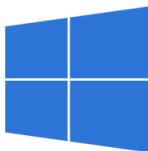
Docs Visual Studio Code FAQ Microsoft software license terms

The screenshot shows a Google search results page. Step 1 highlights the search term "vscode 설치" in the search bar. Step 2 highlights the "Download Visual Studio Code - Mac, Linux, Windows" button on the official Microsoft website. The URL is https://code.visualstudio.com.

vscode 설치

Download Visual Studio Code

Free and built on open source. Integrated Git, debugging and extensions.



↓ Windows

Windows 10, 11

User Installer x64 Arm64

System Installer x64 Arm64

.zip x64 Arm64

CLI x64 Arm64



↓ .deb

Debian, Ubuntu

↓ .rpm

Red Hat, Fedora, SUSE



↓ Mac

macOS 11.0+

.deb x64 Arm32 Arm64

.rpm x64 Arm32 Arm64

.tar.gz x64 Arm32 Arm64

Snap Snap Store

CLI x64 Arm32 Arm64

.zip Intel chip Apple silicon Universal

CLI Intel chip Apple silicon

운영체제에 맞게 선택!

개발 환경

vscode 설치

1) Mac OS

단 한줄로 설명이 가능합니다.

다운받고 압축을 풀고 실행을 하면 바로 실행이 됩니다.

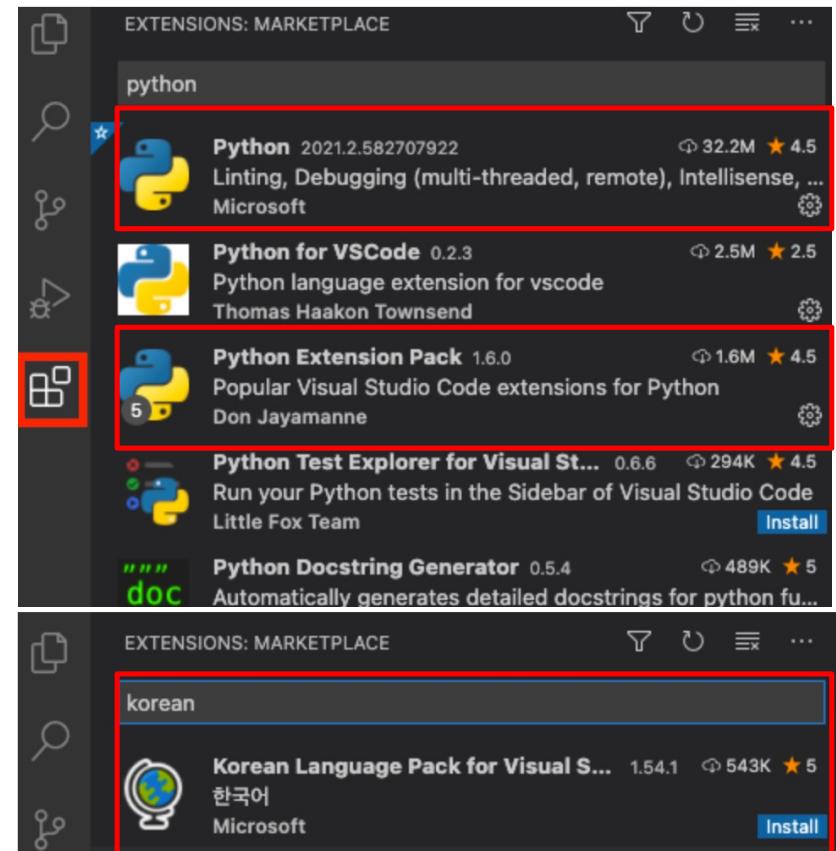
2) Windows OS

마찬가지로 매우 쉽습니다.

다운로드 파일 실행 후, 동의함, 다음, 다음, 다음, 설치, 마침

3. 필수 익스텐션 설치

VS code에서 파이썬을 사용하기 위해서는 몇 가지 EXTENSIONS을 설치해주어야 합니다.



개발 환경

차선책: github codespaces 이용 (python 설치 불필요)

The screenshot shows a Google search results page. At the top, there is a search bar with the query "github codespaces" highlighted by a red box. To the left of the search bar is the Google logo, and above it is a blue numbered box containing the number "1". Below the search bar are various search filters: AI 모드, 전체, 이미지, 동영상, 쇼핑, 뉴스, 짧은 동영상, 더보기 ▾, and 도구 ▾. The main search results area displays a snippet from GitHub's official website. The snippet includes a link to "Compare insights" and a section titled "Codespaces" which is also highlighted by a red box. Below this, there is a brief description of what GitHub Codespaces offers. Further down, there is another section titled "Codespaces features" with a description and a right-pointing arrow.

1

Google

github codespaces

X | ⌨ | 🎤 | 📸 | 🔎

AI 모드 전체 이미지 동영상 쇼핑 뉴스 짧은 동영상 더보기 ▾ 도구 ▾

See detailed insights & Compare multiple related Papers for :
“github codespaces”

Compare insights ↗

2 GitHub
https://github.com › features › codespaces :

Codespaces

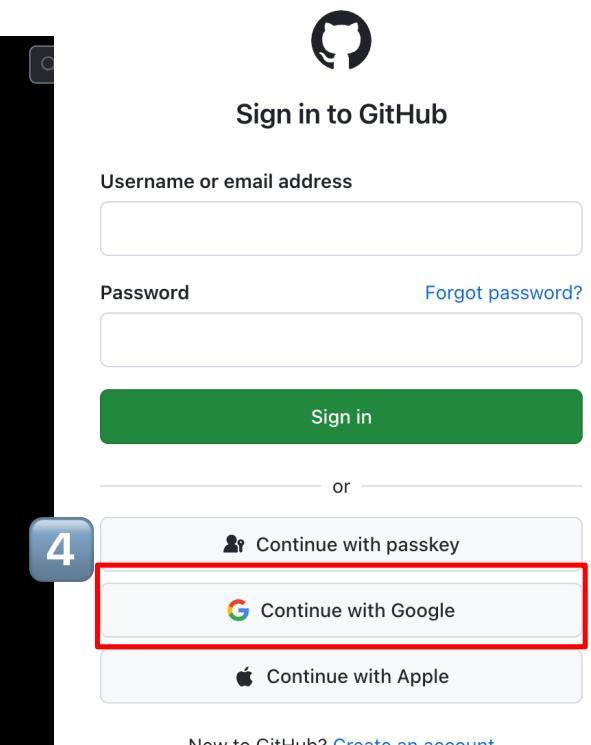
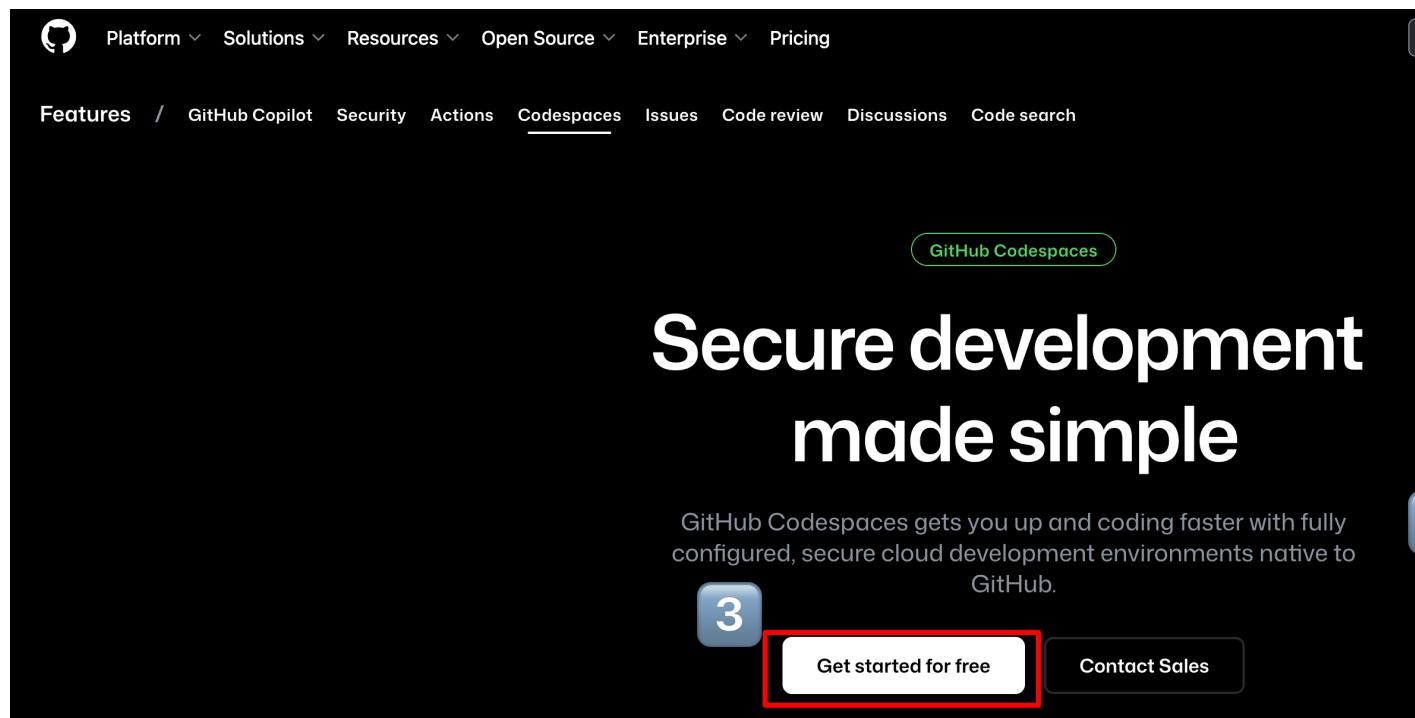
GitHub Codespaces gets you up and coding faster with fully configured, secure cloud development environments native to GitHub.

Codespaces features >

GitHub Codespaces allows you to work remotely on a machine ...

개발 환경

차선책: github codespaces 이용 (python 설치 불필요)



차선책: github codespaces 이용 (python 설치 불필요)

Your instant dev environment
Go from code to commit faster on any project.

[Go to docs](#) [New codespace](#)

Explore quick start templates [See all](#)

Blank By github ⓘ  [Use this template](#)

Start with a blank canvas or import any packages you need.

React By github ⓘ  [Use this template](#)

A popular JavaScript library for building user interfaces based on UI components.

Jupyter Notebook By github ⓘ  [Use this template](#)

JupyterLab is the latest web-based interactive development environment for notebooks, code, and data.

.NET By github ⓘ  [Use this template](#)

A full-stack web application template written in C# leveraging the power of .NET 8.

Getting started with GitHub Codespaces

Learn core concepts
New to Codespaces? [Start here](#). Learn the core concepts and how to get started.

Configure and manage
Learn more about features like [secret management](#) and [port forwarding](#).

Develop locally
Access codespaces from within [Visual Studio Code](#).

개발 환경

Github codespaces란?

- Github에서 제공하는 클라우드 개발 환경
 - 클라우드: 인터넷을 통해 다른 이(github)가 제공하는 컴퓨터 자원 사용
 - Github에서 제공하는 가상 머신 위에서 개발 진행
- 리눅스 개발 환경 제공
- 환경 안에 Python, git 등이 포함
- 코드 스페이스를 사용하면 각자 사용하는 디바이스에 상관 없이 같은 환경에서 작업할 수 있음
- vscode와 거의 동일한 인터페이스 제공



The screenshot shows the GitHub Codespaces interface. At the top, there's a large "Codespaces" logo with the GitHub octocat icon. Below it is a code editor window titled "emotion.rb" with the file content:

```
attr_reader :label
attr_reader :pronounceable_label
# Public: Get the Emoji that this reaction's content represents.
#
# Returns an Emoji.
attr_reader :emoji_character

def initialize(content:, label: nil, pronounceable_label: nil, emoji_character: nil)
  @content = content
  @label = label || @content
  @pronounceable_label = pronounceable_label || @label
  @emoji_character = emoji_character || Emoji.find_by_alias(@content)
  @platform_enum = @pronounceable_label.gsub(" ", "_").upcase
end

Emotion.create(content: "+1", pronounceable_label: "thumbs up")
Emotion.create(content: "-1", pronounceable_label: "thumbs down")
```

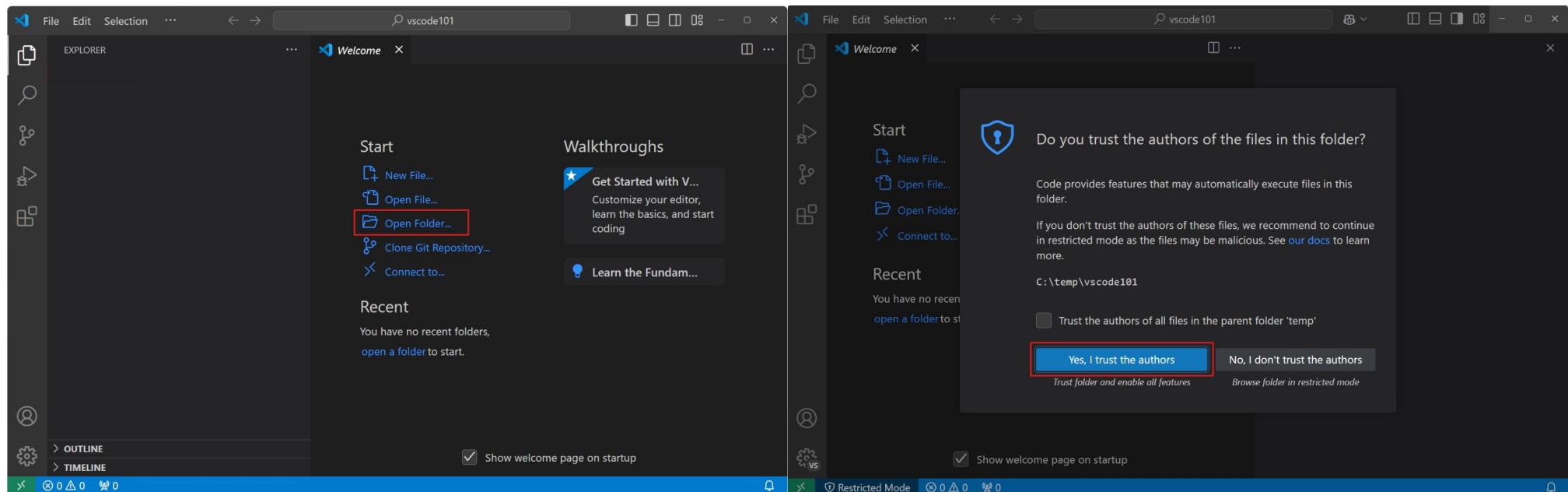
To the left of the code editor is the "EXPLORER" sidebar, which lists the project structure under "GITHUB". The structure includes ".github", ".vscode", "app", "components", "controllers", "helpers", "jobs", "mailers", and "models" directories, along with several shell scripts like "build-devcontainer.sh", "devcontainer.json", "on-create-command.sh", "post-attach-command.sh", and "setup-devcontainer.sh".

<https://github.blog/engineering/infrastructure/githubs-engineering-team-moved-codespaces/>

개발 환경

VS Code: 폴더 열기

- VS Code를 실행하면 Welcome (시작 페이지)가 나타나고 시작하기 위한 여러 기능들이 보입니다.
- **Open Folder (ctrl + o)**를 활용하여 작업 디렉토리를 엽니다.

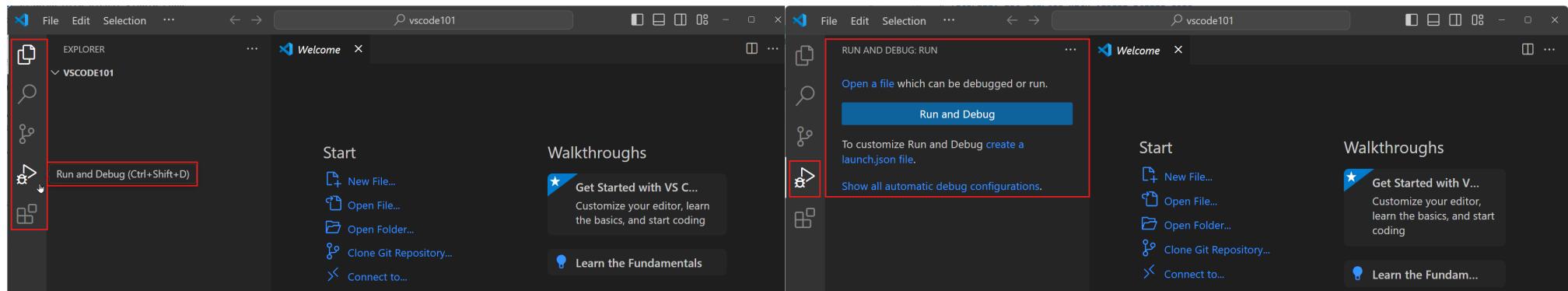


<https://code.visualstudio.com/docs/getstarted/getting-started>

개발 환경

VS Code: 사용자 인터페이스 탐색

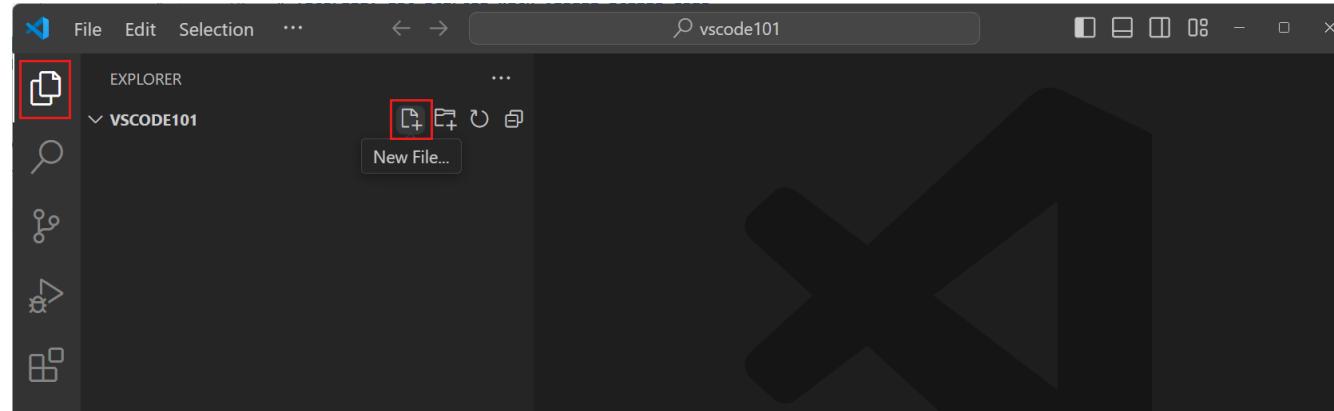
- 왼쪽 활동 표시줄에서 다양한 view를 선택할 수 있음
- Tip: 활동 표시줄의 아이콘에 커서를 올려 놓으면 각 뷰의 이름과 해당 키보드 단축키 표시
- 아이콘을 클릭하면 사이드바가 열리고 해당하는 view에 대한 정보 표시



<https://code.visualstudio.com/docs/getstarted/getting-started>

개발 환경

VS Code: 편집기로 파일 보기 및 편집



- 새 파일 버튼을 클릭하여 작업 공간에 새 파일 생성

A screenshot of the Visual Studio Code interface in dark mode. The top bar shows 'File', 'Edit', 'Selection', and other icons. The main area shows the code editor with the following content:

```
<html>
  <body>
    <h1>Hello, VS Code!</h1>
  </body>
</html>
```

The file name 'index.html' is highlighted in the Explorer sidebar with a red box. The code editor shows the same file name in the title bar.

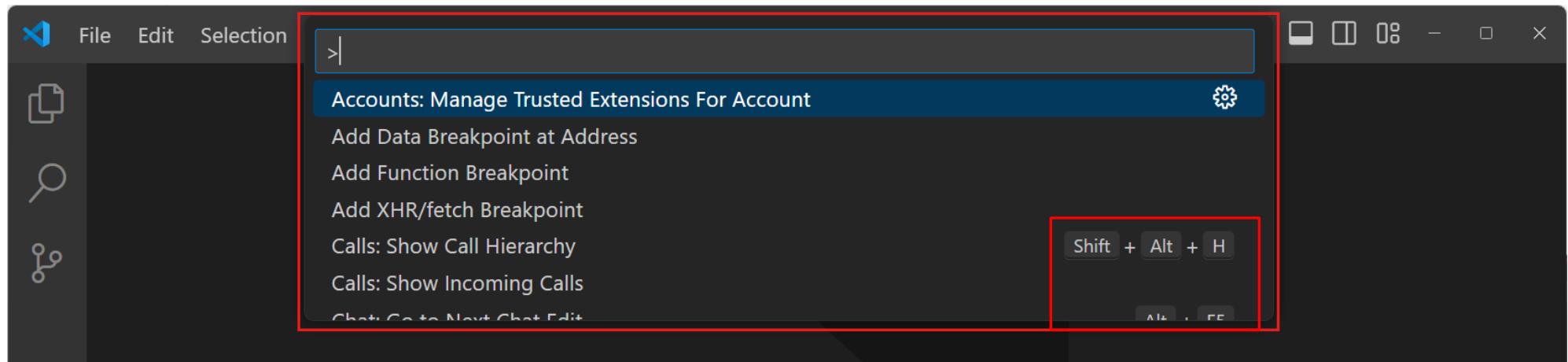
- 파일명 및 확장자 입력 후 Enter키를 누르면 파일이 생성되며 편집 가능

<https://code.visualstudio.com/docs/getstarted/getting-started>

개발 환경

VS Code: 명령 팔레트 사용(**ctrl + shift + p** / ⌘⇧P)

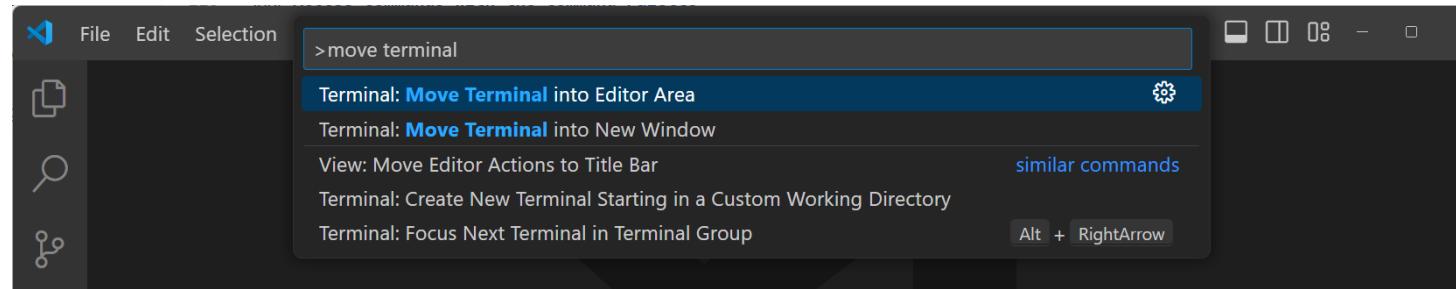
- VS Code 명령 팔레트를 통해 많은 명령을 사용 가능하며 확장 프로그램 설치 시 커스텀 명령도 추가 가능
- Tip: 명령 팔레트에는 기본 키보드 단축키가 표시 되며 단축키를 사용하여 명령 바로 실행 가능



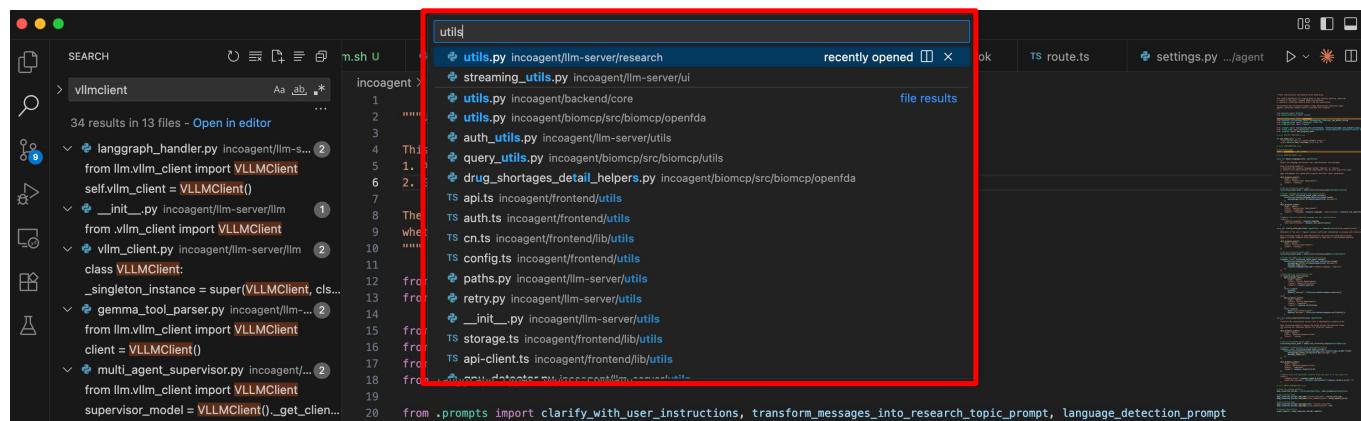
개발 환경

VS Code: 명령 팔레트 사용: (ctrl + shift + p / ⌘⇧P)

- 명령 팔레트는 다양한 작동모드 지원



- '>' 제거 후 입력을 시작하여 작업 공간에서 파일 검색이 가능합니다. (ctrl + P, ⌘P)

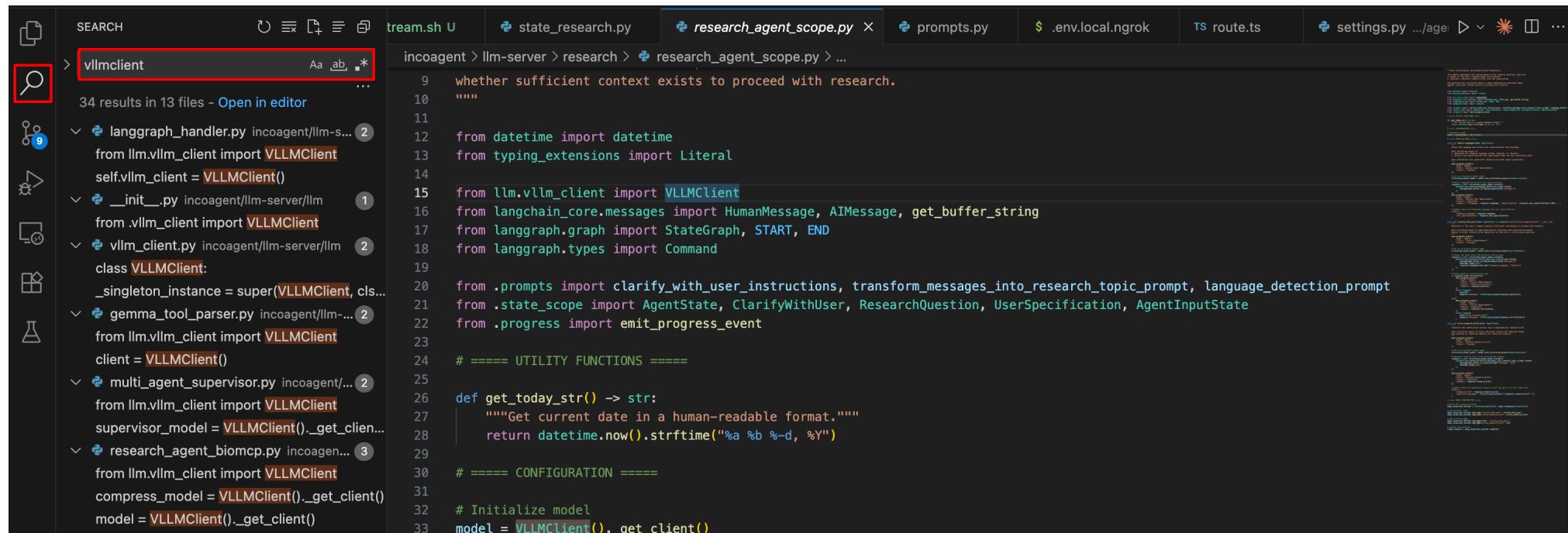


<https://code.visualstudio.com/docs/getstarted/getting-started>

개발 환경

VS Code: 워크스페이스 검색(ctrl + shift + F / ⌘⇧F)

- 워크스페이스 전체 파일에 대해 해당 키워드가 포함된 부분을 검색할 수 있음
- 클릭 시 해당 파일이 열리고 해당 부분을 보여줌



The screenshot shows the VS Code interface with the search bar at the top containing the text "vllmclient". Below the search bar, a sidebar displays a list of files where the keyword was found, with 34 results across 13 files. The main editor area shows a Python file with several occurrences of "VLLMClient" highlighted in orange. The code snippet includes imports from "llm.vllm_client" and "langchain_core.messages", and defines a class "VLLMClient" with methods like "get_today_str" and "initialize_model".

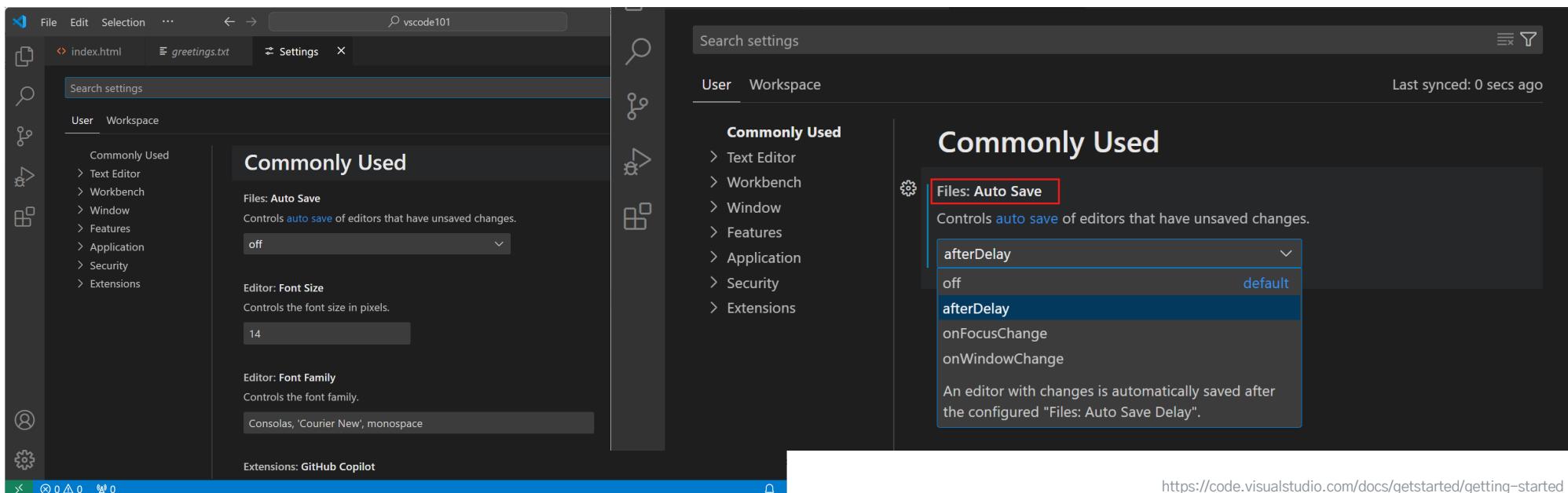
```
SEARCH          tream.sh U state_research.py research_agent_scope.py prompts.py .env.local.ngrok route.ts settings.py .../agei D ...
> vllmclient Aa ab *
34 results in 13 files - Open in editor
langgraph_handler.py incoagent/llm-server/llm-... 2
from llm.vllm_client import VLLMClient
self.vllm_client = VLLMClient()
__init__.py incoagent/llm-server/llm 1
from vllm_client import VLLMClient
vllm_client.py incoagent/llm-server/llm 2
class VLLMClient:
    _singleton_instance = super(VLLMClient, cls...
gemma_tool_parser.py incoagent/llm-... 2
from llm.vllm_client import VLLMClient
client = VLLMClient()
multi_agent_supervisor.py incoagent/llm-... 2
from llm.vllm_client import VLLMClient
supervisor_model = VLLMClient().__get_clien...
research_agent_biomcp.py incoagen... 3
from llm.vllm_client import VLLMClient
compress_model = VLLMClient().__get_client()
model = VLLMClient().__get_client()

9     whether sufficient context exists to proceed with research.
10    """
11
12    from datetime import datetime
13    from typing_extensions import Literal
14
15    from llm.vllm_client import VLLMClient
16    from langchain_core.messages import HumanMessage, AIMessage, get_buffer_string
17    from langgraph.graph import StateGraph, START, END
18    from langgraph.types import Command
19
20    from .prompts import clarify_with_user_instructions, transform_messages_into_research_topic_prompt, language_detection_prompt
21    from .state_scope import AgentState, ClarifyWithUser, ResearchQuestion, UserSpecification, AgentInputState
22    from .progress import emit_progress_event
23
24    # ===== UTILITY FUNCTIONS =====
25
26    def get_today_str() -> str:
27        """Get current date in a human-readable format."""
28        return datetime.now().strftime("%a %b %-d, %Y")
29
30    # ===== CONFIGURATION =====
31
32    # Initialize model
33    model = VLLMClient().__get_client()
```

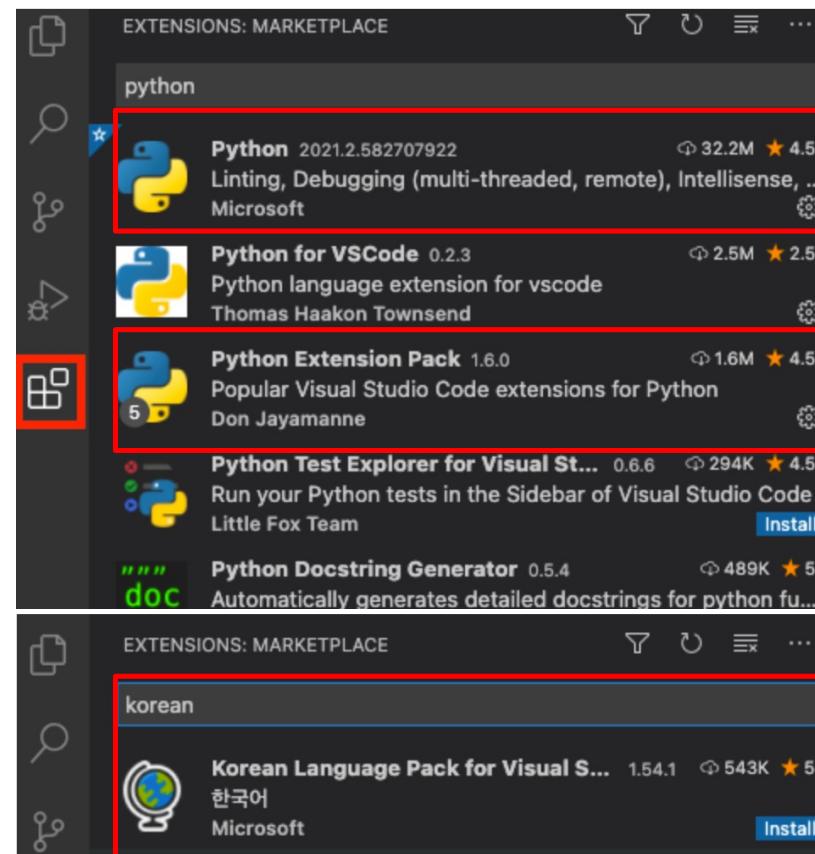
개발 환경

VS Code: 설정 구성

- **Ctrl + , (⌘,)**를 눌러 **설정 편집기**를 열 수 있습니다.
- **Tip:** 검색창을 사용하여 표시되는 설정 목록을 필터링하세요.
- 사용자 설정은 모든 작업 공간에 적용되며 작업 공간 설정은 현재 작업 공간에만 적용됩니다.
- **기본적으로 VS Code는 수정된 파일을 자동으로 저장하지 않습니다.** 이 동작을 변경하려면 '**자동 저장**' 설정을 변경하세요



VS Code: python 기본 extension 설치

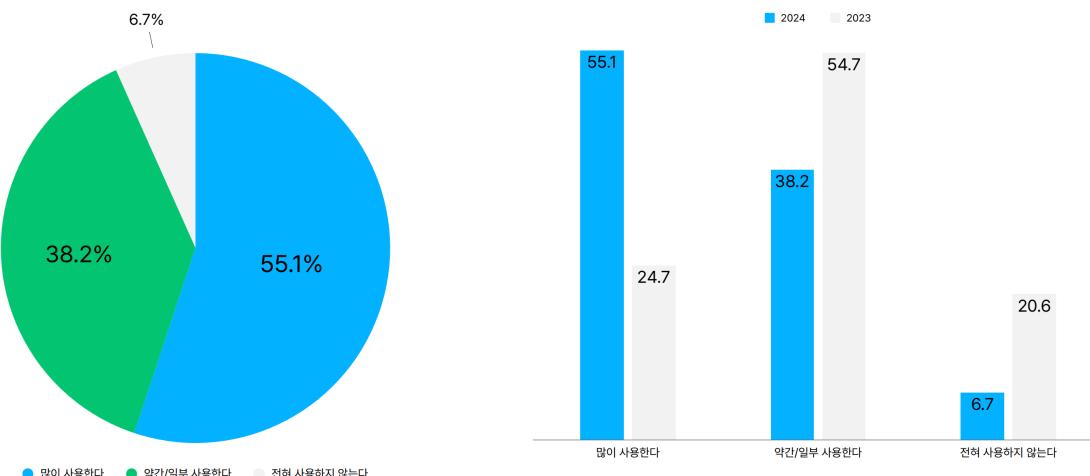


개발 환경

AI coding assistant (agent, tools)

- LLM이 발전하면서 **가장 빠르게 성능이 향상 되는 활용 분야가 코딩**
- 현재 나오고 있는 AI 기반 코딩 어시스턴트는 **기존의 코드 편집기, 터미널에 통합됨**
- 코딩, 버전 관리, 디버깅을 포함한 **개발 과정 전체를 AI와 함께 할 수 있음**
 - AI와 상호작용할 수 있는 **인터페이스** 제공: 채팅, 명령어, agent 등

현재 업무(개발)에 생성형 AI(AI)를 사용 중인가요?



2025 원티드 개발자 리포트



stackoverflow 2025 developer survey

개발 환경

대표적 AI 코딩 툴들

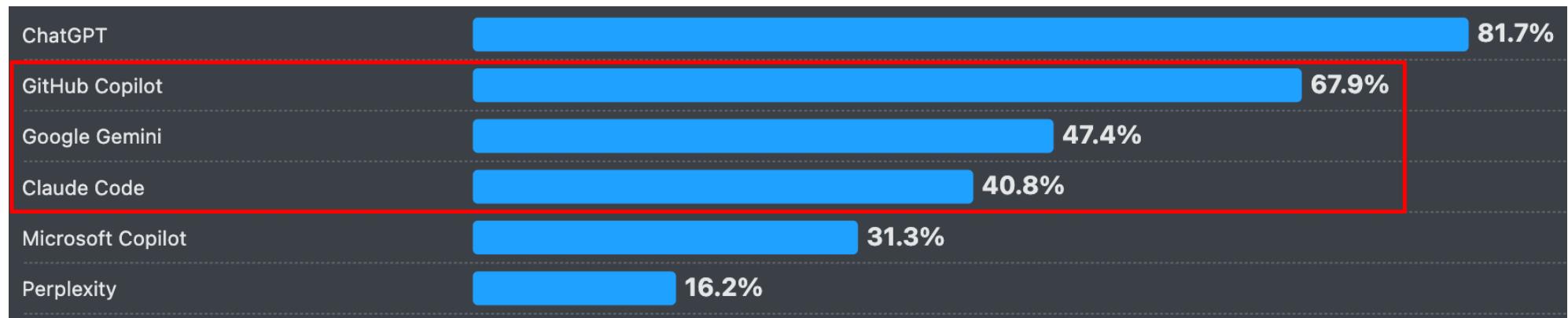


<https://amzn.to/4nolZ2I>

개발 환경

Stackoverflow 2025 설문조사: 사용해본 적 있는 AI 도구 순위

- Q: 개발에서 AI agent를 활용하거나 **개발하는 이들이 사용한 적이 있는 AI 도구**



<https://survey.stackoverflow.co/2025/ai>

개발 환경

Github copilot

- VS code에 통합된 AI (LLM) 기반 코딩 어시스턴트(코드 편집기에서 자연스럽게 사용 가능, CLI 버전도 존재)
- 자연어 프롬프트와 코드 맥락을 기반으로 코드 제안 및 설명하는 기능을 갖춤
- 다른 툴들과 비슷하게 작업 공간을 컨텍스트로 사용 가능

@-mention과 #-mention

• **@-mention**: 채팅 참여자

- @<name>은 채팅 참여자, @멘션으로 채팅 참여자 호출 가능 (채팅 참여자는 해당 분야의 전문가 agent)
- [@vscode](#), [@terminal](#), [@workspace](#) 등의 채팅 참여자 존재
- 예시: [@terminal](#) 현재 디렉토리에서 가장 용량을 많이 잡아먹는 5개의 파일이 어떤 것들이야?

• **#-mention**: 채팅 도구

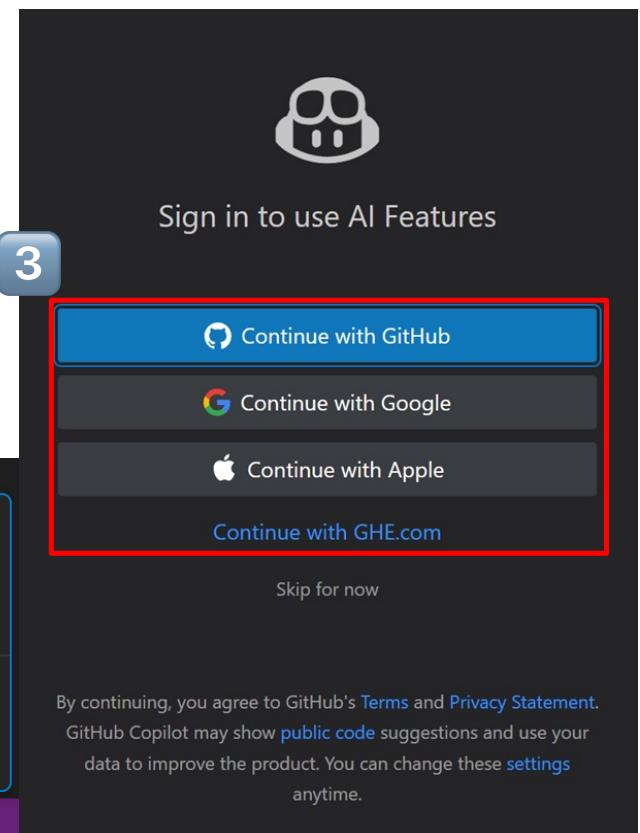
- #<name>은 도구, VS code는 3가지 도구를 지원 ([내장 도구](#), [MCP](#), [확장 도구](#))
- 프롬프트에서 도구 사용 가능
- 예시: [#fetch https://code.visualstudio.com/updates](#)의 내용을 요약해줘
- 예시: [#codebase](#)에서 인증이 어떻게 이루어지는지 설명해줘

https://code.visualstudio.com/docs/copilot/chat/copilot-chat-context#_atmentions
<https://code.visualstudio.com/docs/copilot/chat/chat-tools>

개발 환경

VS Code: Copilot 설정

- VS Code에서 Copilot을 사용하려면 [Github Copilot 구독에 액세스](#) 해야합니다.
 - [VS Code 내에서 바로 Copilot을 설정할 수](#) 있습니다.
- 상태 표시줄에서 Copilot 아이콘 위에 마우스를 올려놓고 AI 기능 사용을 선택하세요
 - 로그인 방법을 선택하고 안내를 따르세요. 아직 Copilot 구독이 없는 경우 Copilot 무료 플랜에 가입됩니다.



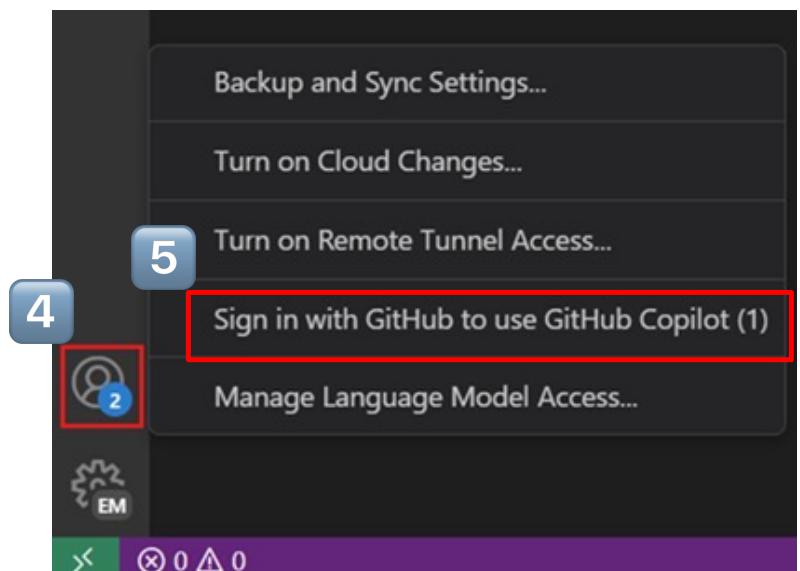
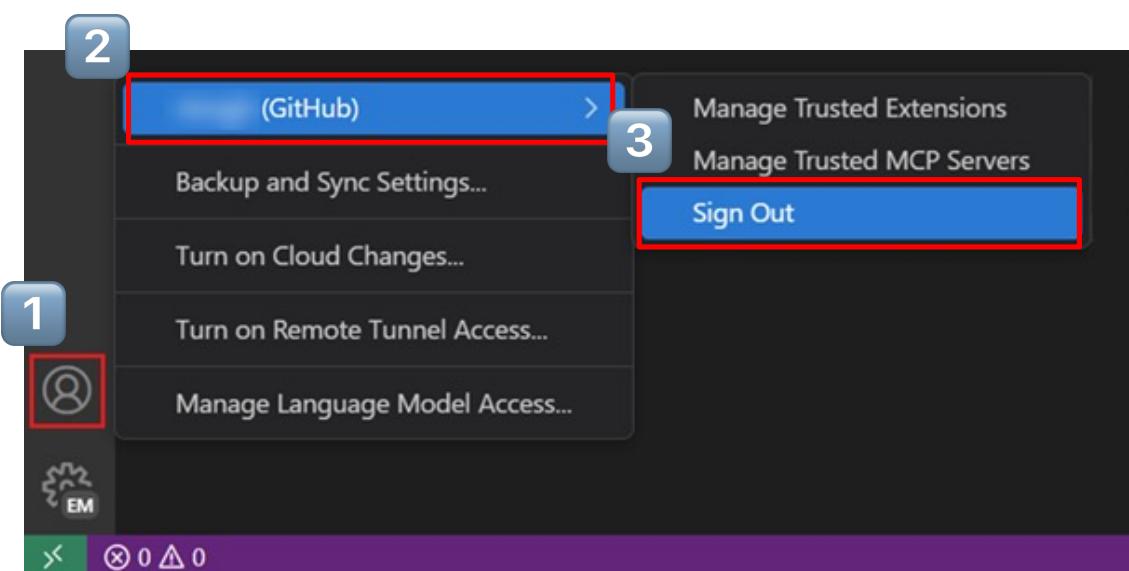
<https://code.visualstudio.com/docs/copilot/setup>

개발 환경

VS Code: Copilot에서 다른 github 계정 사용하기

- Copilot 구독이 다른 Github 계정과 연결된 경우, 다음 단계에 따라 VS Code에서 Github 계정에서 로그아웃하고 다른 계정으로 로그인하세요

 1. 좌하단의 계정 메뉴를 선택한 후, 현재 로그인한 계정에서 로그아웃 선택
 2. 좌하단의 계정 메뉴를 선택한 후 GitHub으로 로그인 선택



<https://code.visualstudio.com/docs/copilot/setup>

개발 환경

Gemini CLI

- **Gemini의 능력을 터미널에서** 바로 사용할 수 있게 해주는 소프트웨어
- 무료 할당량이 상당히 후함 (Gemini 2.5 Flash 및 Pro 사용 가능) **Benefits:**

설치하기

- **nvm install node**로 **npm** 설치 후

Gemini CLI를 설치하고 실행하는 표준 방법은 다음과 같습니다 `npm`.

- **Free tier:** 60 requests/min and 1,000 requests/day
- **Gemini 2.5 Pro** with 1M token context window
- **No API key management** - just sign in with your Google account
- **Automatic updates** to latest models



```
npm install -g @google/gemini-cli
```

Gemini CLI가 설치되면 명령줄에서 Gemini CLI를 실행합니다.

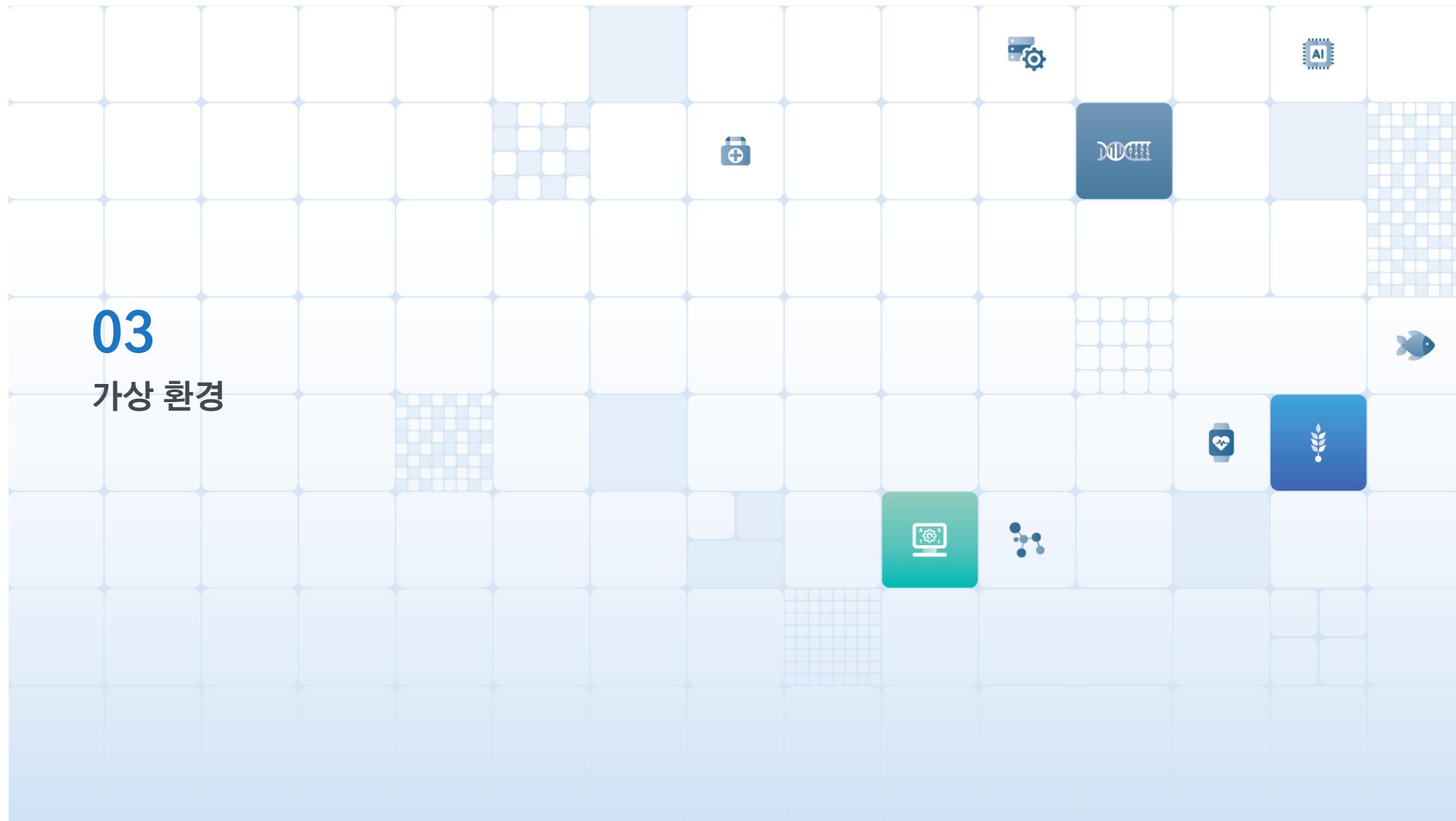


```
gemini
```

<https://github.com/google-gemini/gemini-cli>
<https://geminicli.com/docs/get-started/>

03

가상 환경



가상 환경

가상 환경의 개념

- **프로젝트마다 독립된** 파이썬 공간
- 프로젝트마다 필요한 패키지와 버전을 별도로 설치하고 관리 가능
- 프로젝트별로 다른 파이썬 버전도 사용 가능
- 예시
 - 환경 A는 pandas 2.2
 - 환경 B는 pandas 1.5
- 두 환경은 서로 간섭하지 않음

가상 환경과 패키지 관리가 필요한 이유

- **패키지마다** 다른 패키지에 의존하는 **의존성이 다름**
- 프로젝트마다 요구하는 패키지의 버전이 다름. 가상환경은 이 문제를 해결해주어 **충돌 방지**
- 특정 프로젝트에서 사용된 패키지 버전을 그대로 재현 가능하며 **다른 개발자가 동일한 환경에서 작업할 수 있도록 도움**

가상 환경

참고: 컨테이너(Docker)

- 가상환경은 Python 패키지를 프로젝트별로 분리한다면 **컨테이너는 실행 환경 전체를 분리**
- 가상 환경은** 프로젝트마다 다른 Python 패키지를 사용하나 **OS, 시스템 라이브러리 등은 공유**
- 컨테이너는 소프트웨어 서비스를 실행하는 데 필요한 특정 버전의 프로그래밍 언어 런타임 및 라이브러리와 같은 종속 항목과 애플리케이션 코드를 함께 포함하는 경량 패키지
- 운영체제 수준에서 CPU, 메모리, 스토리지, 네트워크 리소스를 쉽게 공유할 수 있게 해줌
- 배포 환경에서 많이 사용**

항목	가상환경	Docker
분리 범위	Python 패키지	OS + Python + 패키지 전체
사용 목적	개발, 실험	배포, 재현성 강한 실행
생성 속도	매우 빠름	상대적으로 느림
학습 난이도	낮음	중간
유용한 상황	연구 코드 실행	팀 공유, 서버 배포



<https://cloud.google.com/learn/what-are-containers?hl=ko>

가상 환경

파이썬 패키지 관리 시스템

- 가상 환경을 만들 때 **패키지 설치와 의존성 관리**가 필요함
- 다양한 파이썬 패키지 관리 시스템이 존재하며 각각 특성과 장단점 존재

pip

- 파이썬 **기본 패키지 관리자**, 파이썬 설치 시 함께 제공
- **PyPI**(패키지 소프트웨어를 저장하고 검색할 수 있는 저장소)에 있는 패키지 설치

uv

- Pip과 동일한 역할 수행하지만 **속도와 의존성 관리 능력이 더 우수함**
- **패키지 캐시 기능**이 강력하여 반복 설치가 많은 연구 환경에 특히 유용

anaconda (conda)

- **C 기반 패키지 설치가 안정적**이며 **OS 의존성이 큰 바이오, 딥러닝 관련 패키지 이용 시** 유용
- 환경이 무겁고 설치가 느릴 수 있음

<https://wikidocs.net/252578>

가상 환경

패키지 관리 시스템 핵심 요약

구분	pip	uv	conda
패키지 관리	✓	✓	✓
가상환경 생성	✗(venv 필요)	✓(부가기능)	✓
Python 버전 관리	✗	✗	✓
시스템 라이브러리	✗	✗	✓
설치 속도	보통	매우 빠름	느림
안정성	보통	높음	매우 높음
딥러닝(CUDA)	어려움	어려움	안정적

pip

- 단순한 파이썬 프로젝트
- 충돌 위험이 적을 때
- 표준 방식이 필요할 때

uv

- 패키지를 자주 설치·삭제할 때
- 설치 속도가 중요할 때
- pip에서 충돌이 자주 날 때

conda

- PyTorch·CUDA 같은 C 기반 패키지가 필요
- samtools 같은 바이오 도구와 함께 쓸 때
- Windows·macOS·Linux 차이가 클 때

가상 환경

uv로 가상환경 만들고 관리하기

- uv는 패키지 설치 도구로서의 역할이 중심이지만 가상환경을 만들고 관리하는 기능도 제공

설치

```
# 맥OS, 리눅스, WSL  
curl -LsSf https://astral.sh/uv/install.sh | sh
```

```
# 윈도우  
powershell -c "irm https://astral.sh/uv/install.ps1 | iex"
```

가상환경 활성화 하기

Linux/macOS:

```
bash  
  
source .venv/bin/activate
```

☞ 코드 복사

Windows:

```
.venv\Scripts\activate
```

☞ 코드 복사

가상 환경 만들기

```
uv venv .venv
```

패키지 설치하기

```
uv pip install pytests
```

파이썬 환경 설정 및 자동 설치

```
uv venv -p 3.13
```

의존성 추출하기

```
pip freeze > requirements.txt
```

04

AlphaGenome 및 API 실습



AlphaGenome 및 API 실습

유전체 해석의 중요성

- 세포마다 유전 정보를 해석하고 활용하는 방식은 서로 다름
- DNA의 작동 방식을 이해하는 것은 질병의 원인 규명과 신약 개발, 희귀 유전 질환을 진단하는데 핵심적 단서를 제공
- 그러나 인간 유전체의 약 98%는 비암호화(non-coding) 영역으로 구성되며 이 영역에 대해 충분히 밝혀지지 않음
- AlphaGenome은 비암호화 영역을 정밀하게 해석하고 그 안에 숨겨진 신호를 예측함으로써 유전체 연구의 새로운 지평을 여는 것을 목표로 함
 - 비암호화 부위의 DNA 염기 하나하나가 “변했을 때” 어떤 일들이 생길지를 예측하는 모델

AlphaGenome

- 긴 DNA 시퀀스를 입력으로 받아 그 안에서 수 천개에 이르는 분자생물학적 지표를 동시에 예측하는 통합형 인공지능 모델
- 최대 100만 염기쌍(bp)에 이르는 긴 DNA 서열을 분석하면서도 염기 단위의 해상도 유지
- 합성곱 신경망(CNN)이 짧은 염기 패턴을 감지하여 초기 특징 추출, 이어지는 트랜스포머 층이 수십 만 염기쌍 거리의 장거리 상호작용을 반영
- 비암호화 부위에서 발생하는 non-coding variant는 직접적으로 단백질을 바꾸지는 않지만 유전자의 발현 정도를 조절하는 역할을 할 수 있으며 이를 양적 형질 유전자좌(QTL)라고 부르기도 함. AlphaGenome이 주목하는 영역 중 하나

<https://deepmind.google/blog/alphagenome-ai-for-better-understanding-the-genome/>
<https://www.ai-bio.info/alphagenome-review>

AlphaGenome 및 API 실습

AlphaGenome은 서열로부터 다양한 genomic track을 예측

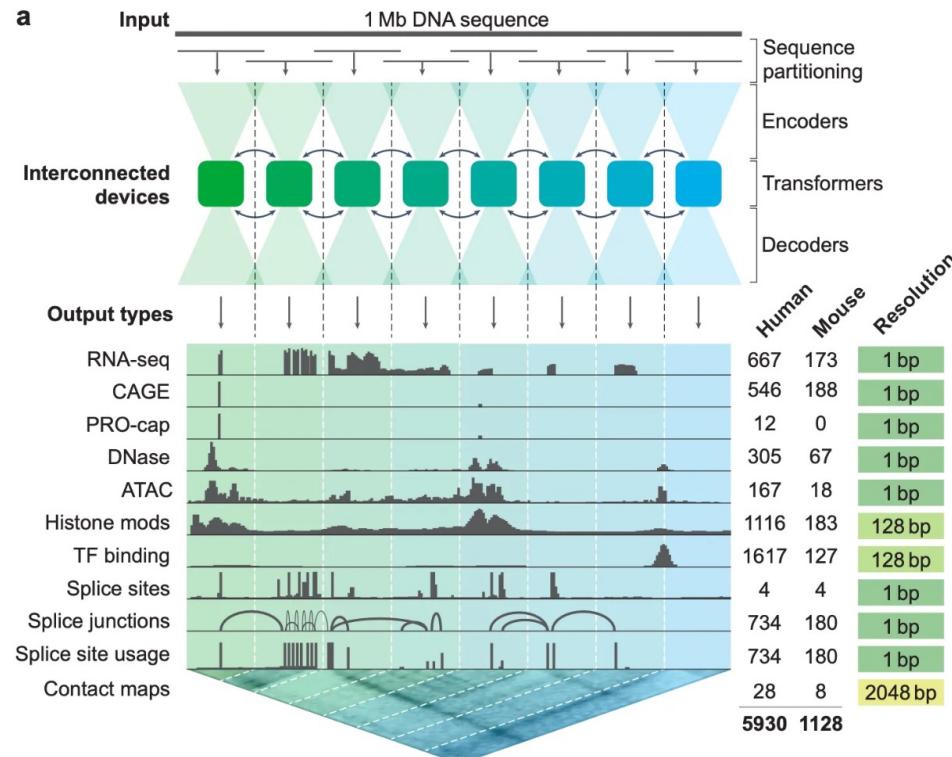


Fig.1a - AlphaGenome 모델의 입력/출력 구조 개요.

d Track prediction: Pre-trained model, fold split

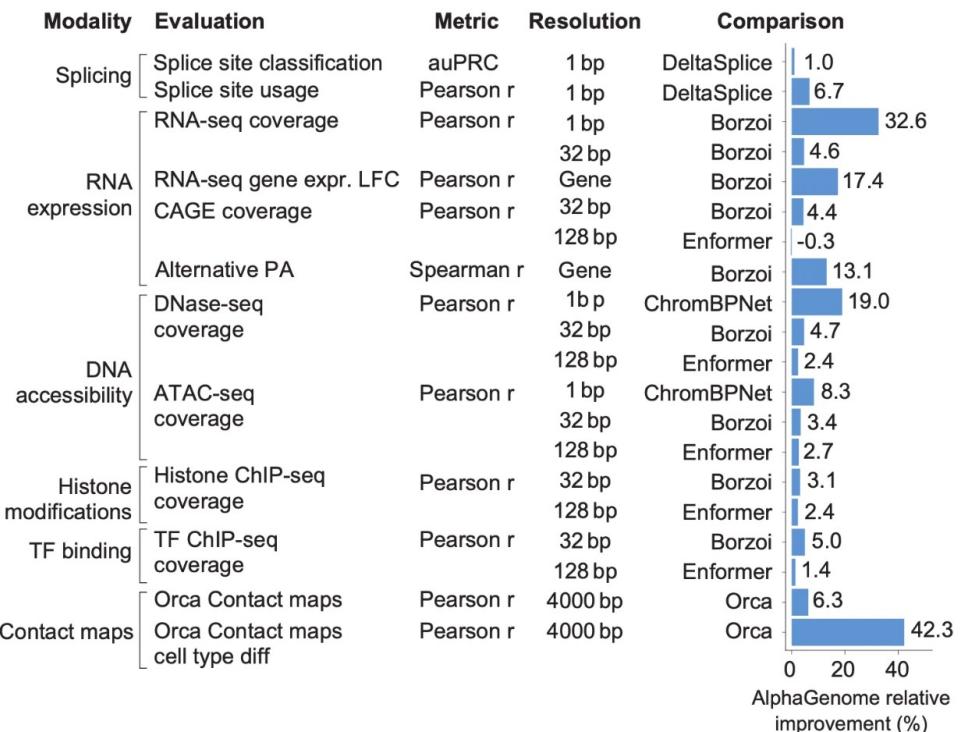


Fig. 1d - Track prediction 성능 평가 metric 및 기존 모델과의 성능 비교

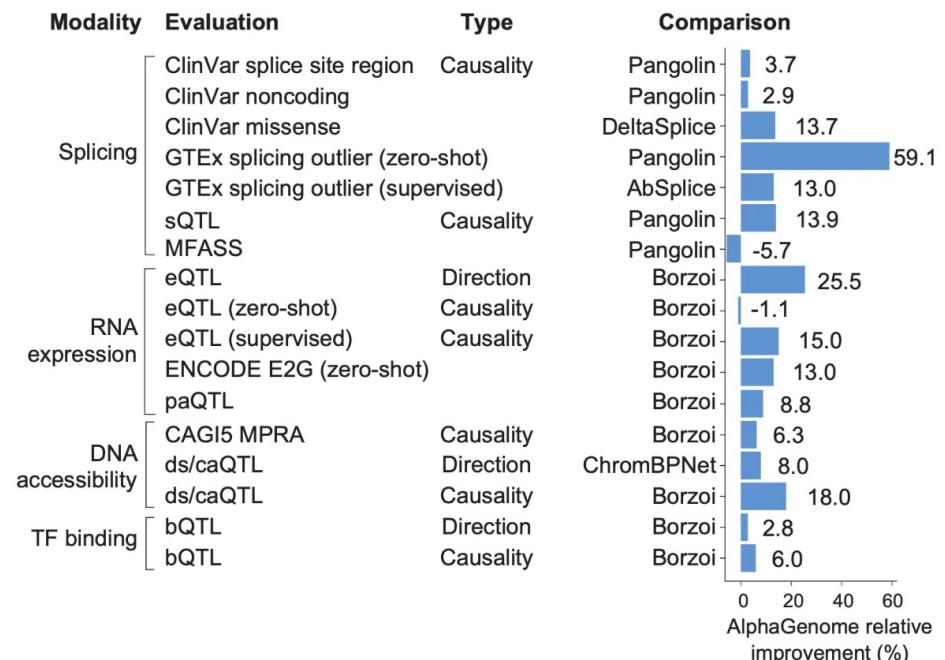
<https://www.biorxiv.org/content/10.1101/2025.06.25.661532v1.full.pdf>
<https://www.ai-bio.info/alphagenome-review>

AlphaGenome 및 API 실습

AlphaGenome은 유전체의 기능적 이상 예측 가능

- 많은 질병은 유전체 상의 잘못된 정보(유전 변이)에 의해 발생
- 서열이 주어졌을 때 해당 유전체의 기능적 활성을 예측하도록 학습한 Alphagenome은 “변이된 서열”이 주어졌을 때 유전체의 기능적 이상을 잘 예측 할 수 있음
- 각 task에 대해 가장 높은 성능을 보이던 **기존 모델 대비 성능 향상**

e Variant effect prediction: Distilled model, all-folds



1.3

Fig. 1e - Variant effect prediction 성능 평가 metric 및 기존 모델과의 성능 비교

<https://www.biorxiv.org/content/10.1101/2025.06.25.661532v1.full.pdf>
<https://www.ai-bio.info/alphagenome-review>

AlphaGenome 및 API 실습

AlphaGenome은 변이에 따른 유전자 발현 변화 예측 가능

- eQTL (Expression quantitative trait loci) 이란 **유전자의 발현 정도에 영향을 미치는 염기서열** 변이체가 포함된 게놈 영역

b Predictions for a known eQTL

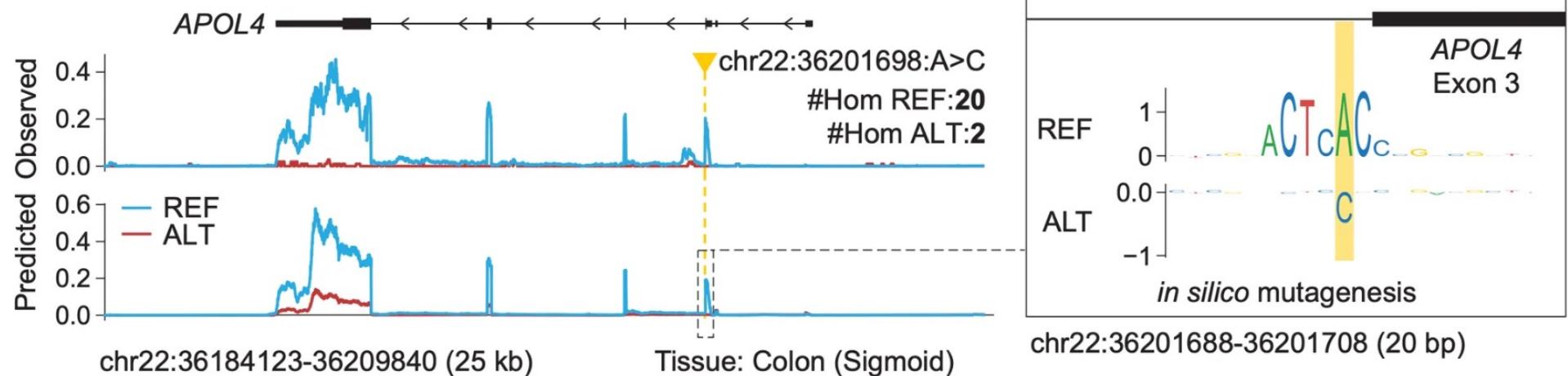


Figure 4b - chr22:36201698:A>C 변이와 연관된 APOL4 유전자 발현량 감소를 예측하는 AlphaGenome.

<https://incodom.kr/eQTL>
<https://www.biorxiv.org/content/10.1101/2025.06.25.661532v1.full.pdf>
<https://www.ai-bio.info/alphagenome-review>

AlphaGenome은 변이에 따른 유전자 발현 변화 예측 가능

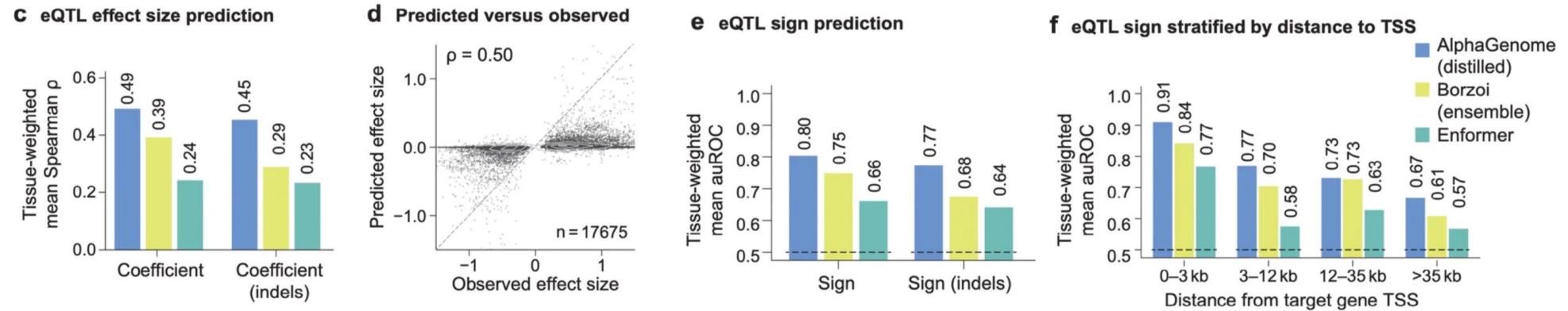


Figure 4c-f - eQTL 효과 관련 예측 성능

<https://www.biorxiv.org/content/10.1101/2025.06.25.661532v1.full.pdf>
<https://www.ai-bio.info/alphagenome-review>

AlphaGenome 및 API 실습

AlphaGenome 유스케이스

- T-세포 급성 림프모구 백혈병(T-ALL)을 가진 환자들에 대한 연구에서 나온 변이들이 있었음
- AlphaGenome은 해당 변이들이 TAL1을 활성화 시켜 MYB DNA binding motif를 생성할 것이라 예측
- 해당 예측은 알려져 있는 질병의 메커니즘과 일치하며 이는 AlphaGenome의 non-coding variant를 질병 유전자에 연결시키는 능력을 보여줌

AlphaGenome의 한계

- 10만 bp 이상 떨어져 있는 조절 인자들의 영향을 예측하는 것은 **아직 잘 하지 못함**
- 세포, 조직 특이적인 패턴을 포착하는 **능력**을 향상 시키는 것이 주요 과제 중 하나
- 개인 유전체를 기반으로 한 **질병 예측이나 임상 진단** 역시 현재로서는 **직접 활용하기 어려움**
- **기초 연구 목적에 초점**을 맞춘 도구로 연구자들을 위한 **비상업적 API 형태로만 제공되고 있음**
- 임상적 활용을 위한 후속 **검증과 안전성 평가 과정은 아직 진행 중**

<https://deepmind.google/blog/alphagenome-ai-for-better-understanding-the-genome/>

AlphaGenome 및 API 실습

API (Application Programming Interface)

- 두 소프트웨어 프로그램이 서로 통신하고 데이터를 주고받을 수 있도록 하는 규칙과 정의의 집합
- API를 통해 사용자는 다른 시스템의 복잡한 내부 구현을 알지 못하고도 해당 시스템의 기능과 데이터 활용 가능

📍 Google Maps Platform

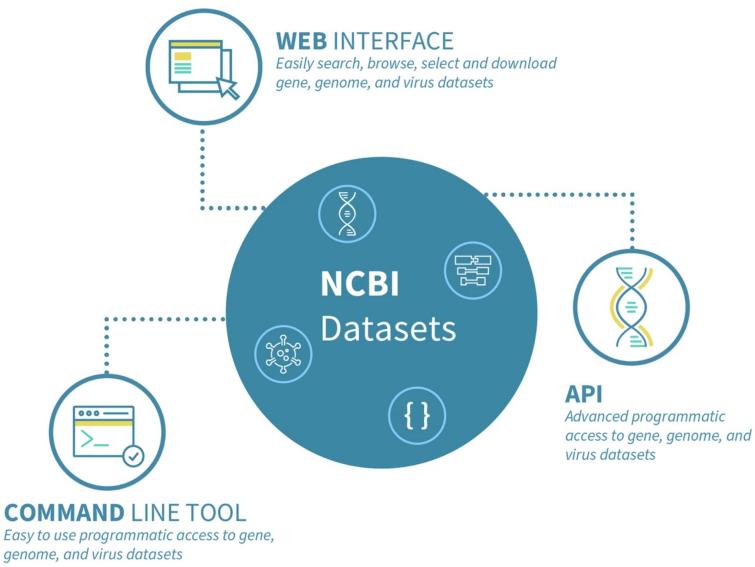
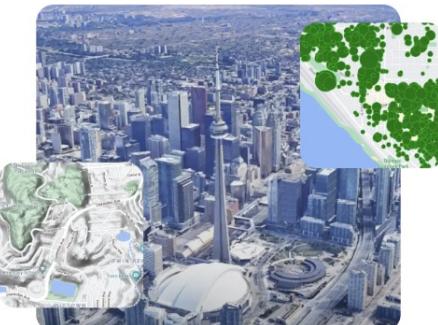
MAPS API

Start building with Maps APIs at no cost

Get free* calls every month based on the complexity of your Maps needs.

Create your account Calculate your costs

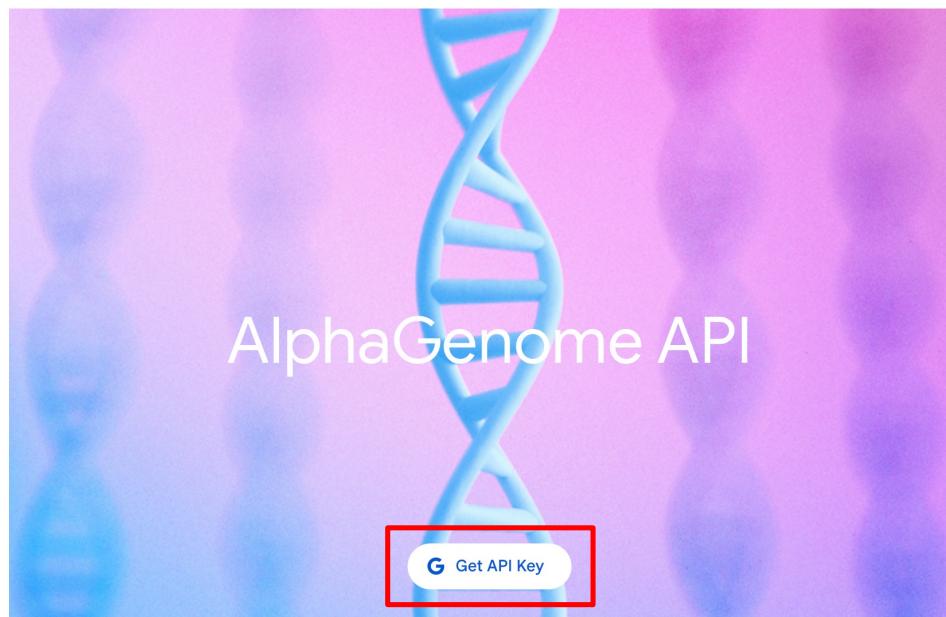
<https://mapsplatform.google.com/lp/maps-apis/>



AlphaGenome 및 API 실습

API Key

- API 키는 API 공급자가 API에 대한 접근을 제어하는 데 사용하는 영문, 숫자 문자열
- API를 통해 통신할 때 신원을 확인하는 수단이며 사용자마다 서로 다른 값을 가짐
- 토큰이라고도 표현
- 보통 공급자가 사용량을 모니터링하기 위해 많이 사용



The image shows the "Access Tokens" section of the Hugging Face Hub profile for "Beomsu Park". A red box highlights the "Access Tokens" heading. Below it, a sub-section titled "User Access Tokens" is also highlighted with a red box. A button "+ Create new token" is visible. To the right, a detailed explanation of access tokens is provided, warning against sharing them. A table lists existing tokens, with one row for "my_token" highlighted with a red box. The table includes columns for Name, Value, Last Refreshed Date, Last Used Date, and Permissions (with a "WRITE" button).

AlphaGenome 및 API 실습

AlphaGenome API key 발급

1

Google alphagenome api

AI 모드 전체 이미지 동영상 쇼핑 뉴스 짧은 동영상 더보기 ▾ 도구 ▾

See detailed insights & Compare multiple related Papers for : "alphagenome api" Compare insights ↗

GitHub https://github.com › google-deepmind › alphagenome ::
[google-deepmind/alphagenome](https://github.com/google-deepmind/alphagenome)
The AlphaGenome API provides access to AlphaGenome, Google DeepMind's unifying model for deciphering the regulatory code within DNA sequences.

2

Google DeepMind https://deepmind.google.com › science › alphagenome ::
[AlphaGenome](https://deepmind.google.com/science/alphagenome)
AlphaGenome – Access Google DeepMind's unifying genomics model for deciphering DNA function.

AlphaGenome 및 API 실습

AlphaGenome API key 발급

AlphaGenome PREVIEW API Community Sign in



AlphaGenome 및 API 실습

AlphaGenome API key 발급

Complete profile

Google account
 goosebumps0704@gmail.com

What is your role?*
ML researcher

How did you hear about us?*
Other

How often do you use APIs?*
Sometimes

- Subscribe to announcements about new AlphaGenome products and features.
- Subscribe to tips and surveys about AlphaGenome.

4

[Continue to Terms of Service](#)

AlphaGenome 및 API 실습

AlphaGenome API key 발급

If applicable, please confirm your:

Organization, university, or other affiliation(s)

As a reminder, AlphaGenome API is not available for: (i) commercial work, even if on behalf of a non-commercial organization; or (ii) commercial organizations. The only exception is journalism which is permitted.

5 I accept the [Google APIs Terms of Service](#) and [AlphaGenome API Additional Terms of Service](#).

6 [Accept and continue](#)

AlphaGenome 및 API 실습

AlphaGenome API key 발급

Account settings

API Key

To use the AlphaGenome API you need to create an API key below.

7

Create API Key

API Key generated

Use your API Key securely.

Your API Key —

8

Copy

9

vscode 혹은 github
codespaces에 붙여넣기

05

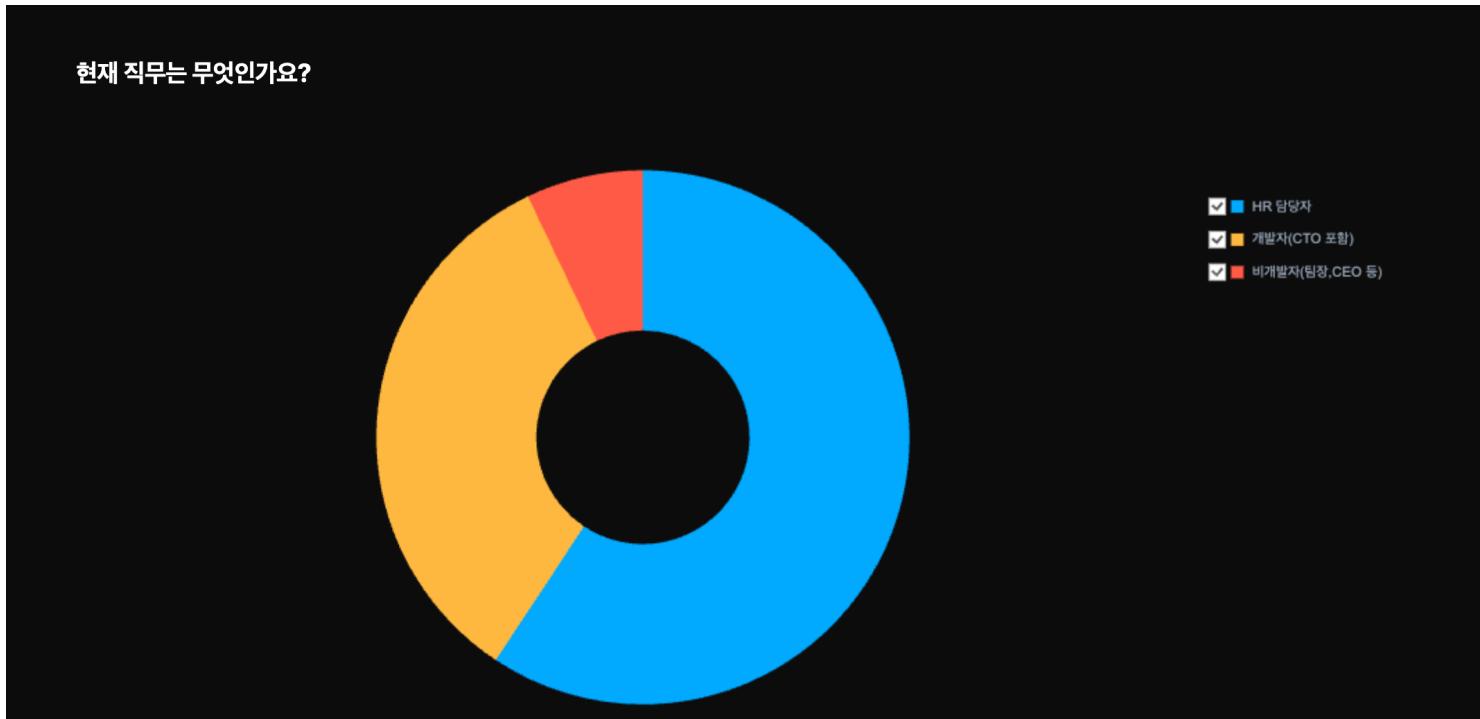
번외: 개발자 및 채용자 설문



2023 Programmers Recruiting Survey

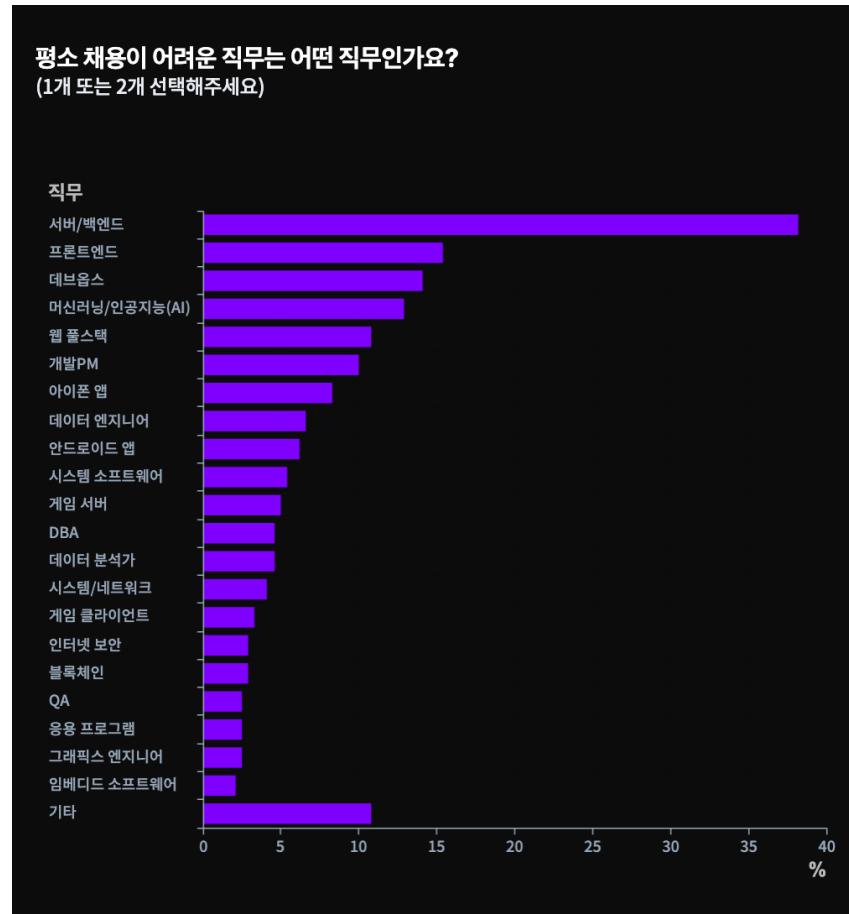
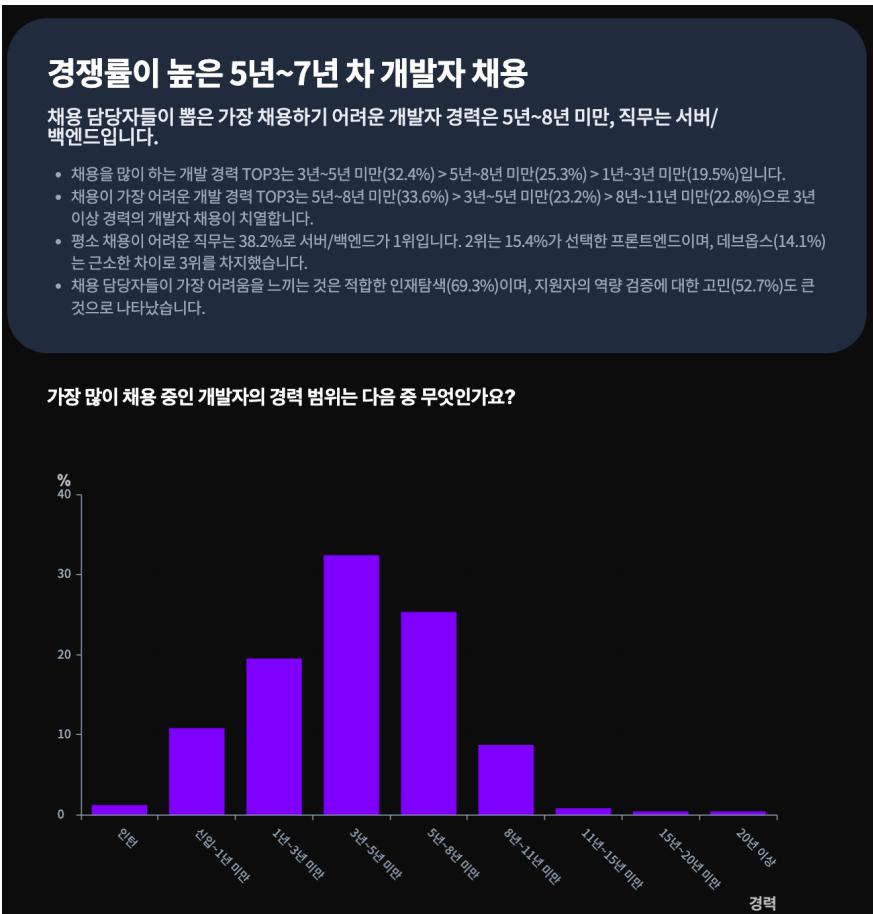
개요

- 조사기간 : 2023-01-03 ~ 2023-01-31
- 143명의 HR담당자, 98명의 개발자 채용에 참여하는 관계자 대상 참여한 온라인 설문조사



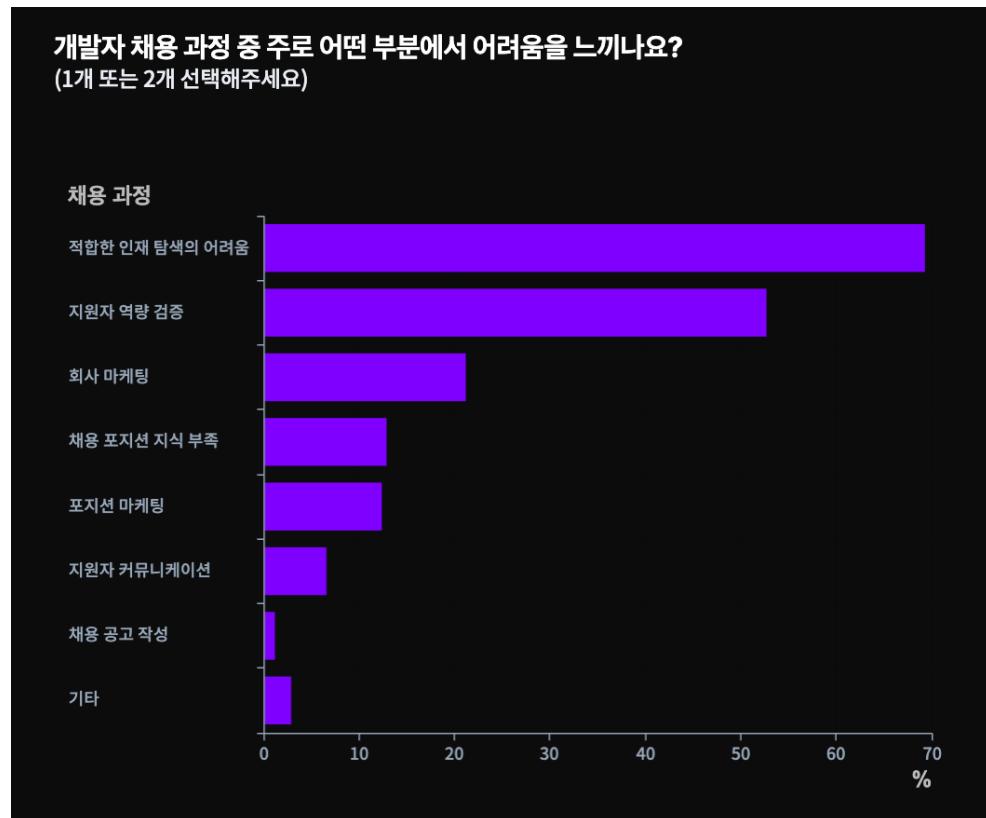
<https://programmers.co.kr/pages/2023-recruiting-survey>

2023 Programmers Recruiting Survey



<https://programmers.co.kr/pages/2023-recruiting-survey>

2023 Programmers Recruiting Survey



채용 담당자들이 가장 어려워 하는 부분:

정확히 회사가 원하는 역량을 가진 인재를 찾는 것

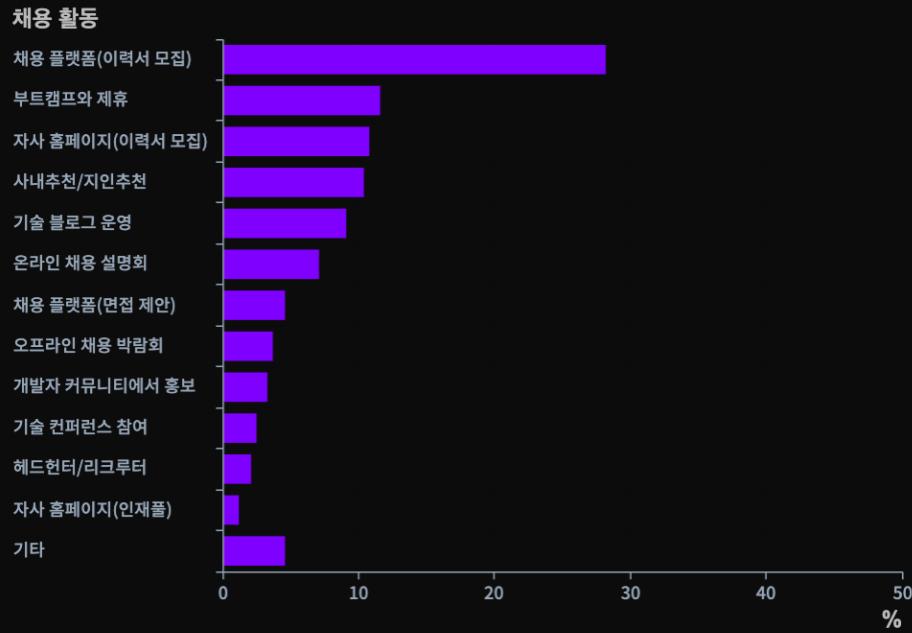
개발에서 기술 생태계가 워낙 폭이 넓기 때문에 회사에서 현재 원하는 기술 스택과 도메인 지식을 갖춘 인재를 찾는 것은 거의 불가능

회사에서 원하는 기술 스택과 도메인 지식에 내가 가진 역량이 정확히 부합하지 않더라도 합격 가능성 존재

바이오, 의료 AI 회사가 많은 추세에 바이오 도메인 지식을 가진 것은 큰 메리트

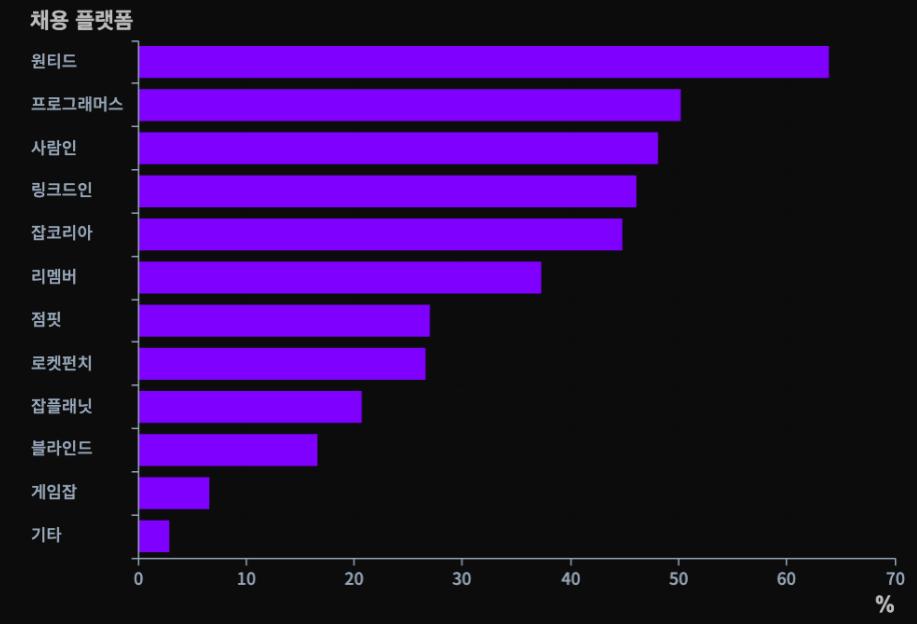
2023 Programmers Recruiting Survey

최종 합격자 기준으로 신입 개발자 채용에 가장 도움이 되는 채널은 무엇인가요?



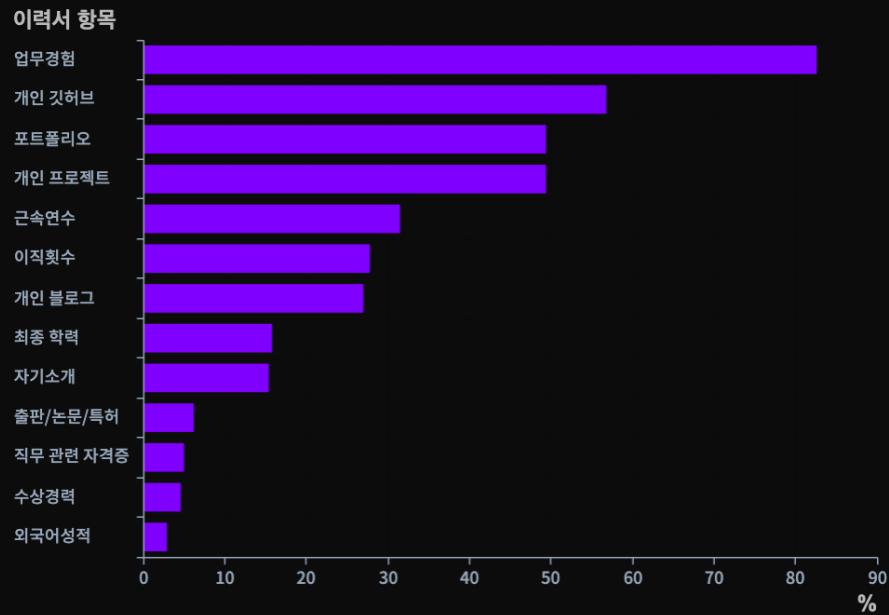
다음 중 이용 중인 채용 플랫폼은 무엇인가요?

*채용 플랫폼 선택한 사람만 응답 (모두 선택해주세요)

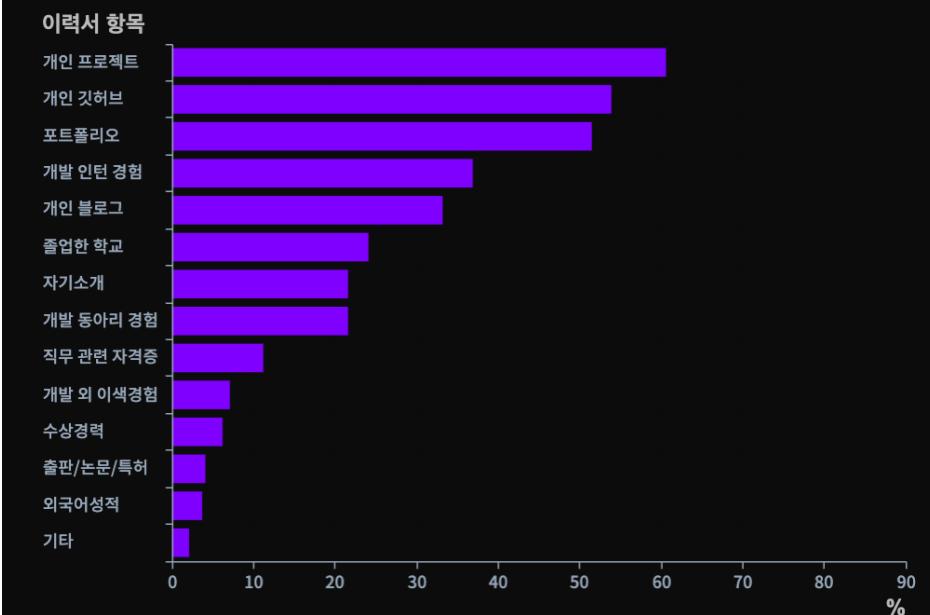


2023 Programmers Recruiting Survey

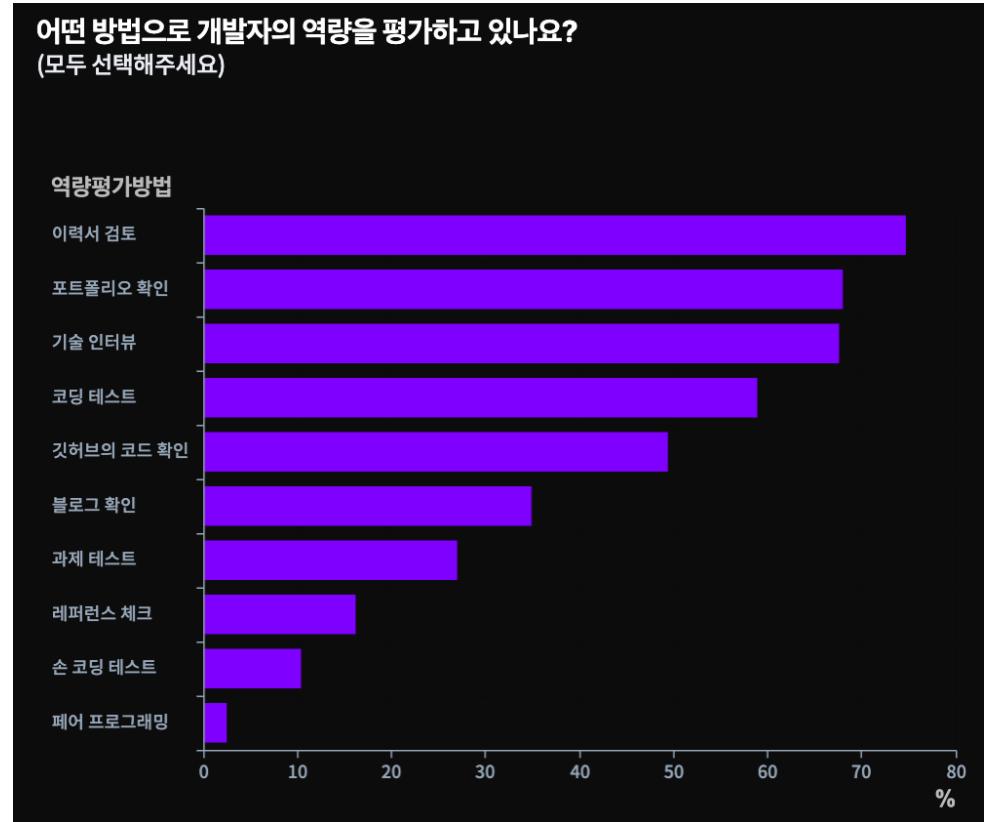
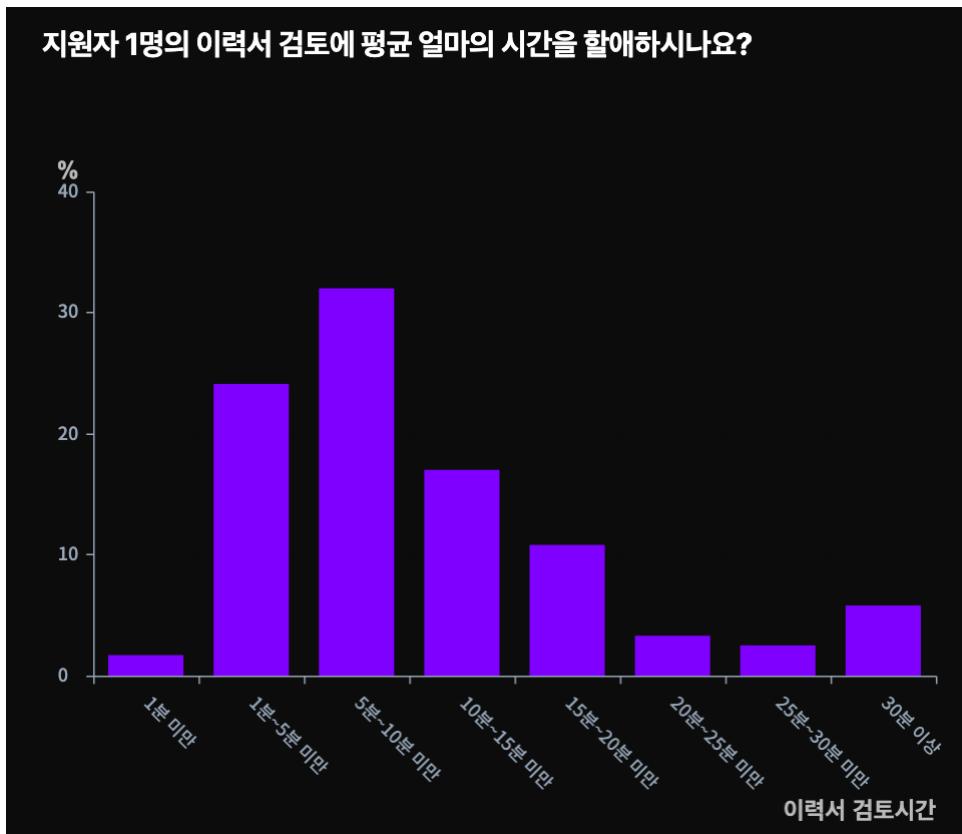
개발자 이력서에서 눈여겨보는 것은 무엇인가요?
(모두 선택해주세요)



신입 개발자 이력서에서 추가적으로 눈여겨보는 것이 있다면, 무엇인가요?
(모두 선택해주세요)



2023 Programmers Recruiting Survey

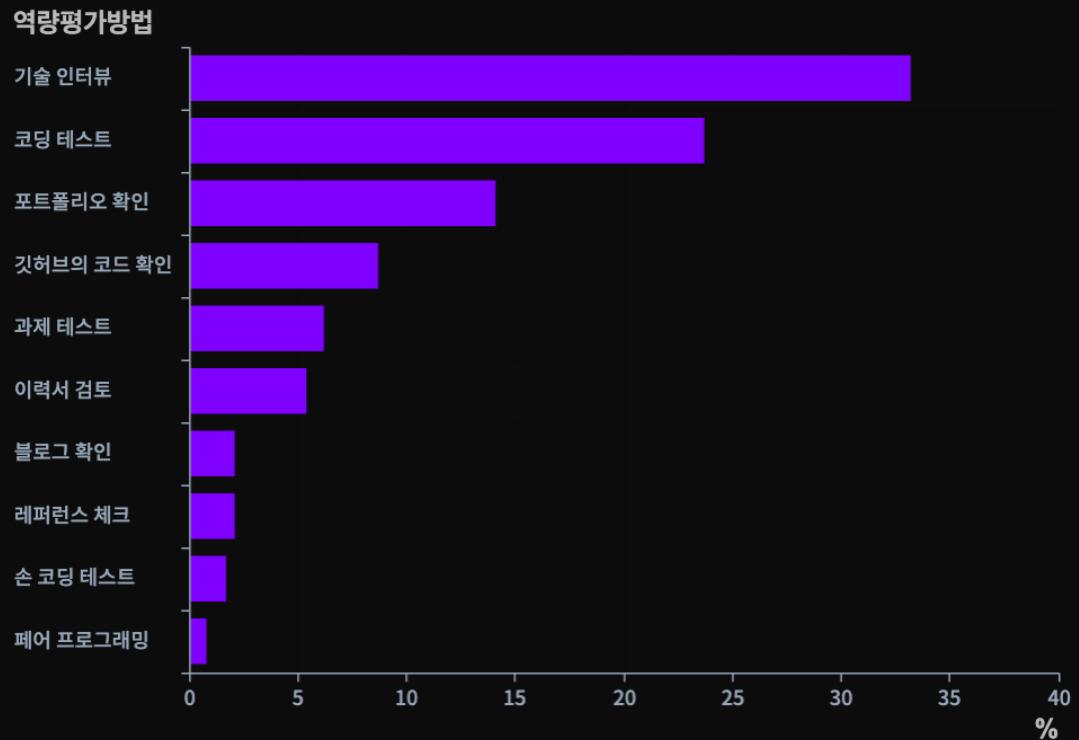


<https://programmers.co.kr/pages/2023-recruiting-survey>

© 2025 INSILICOGEN, INC. ALL RIGHTS RESERVED. 190

2023 Programmers Recruiting Survey

개발자 역량 평가에 가장 도움이 되는 방법은 무엇인가요?



결국 회사 입장에서는 직접 면접을 봐야 어떤 사람이 회사에서 원하는 역량을 갖췄는지 확인할 수 있음.

이력서에서 눈 여겨 보는 요소들이 막상 역량 평가에 가장 도움이 되는 영역은 아님.

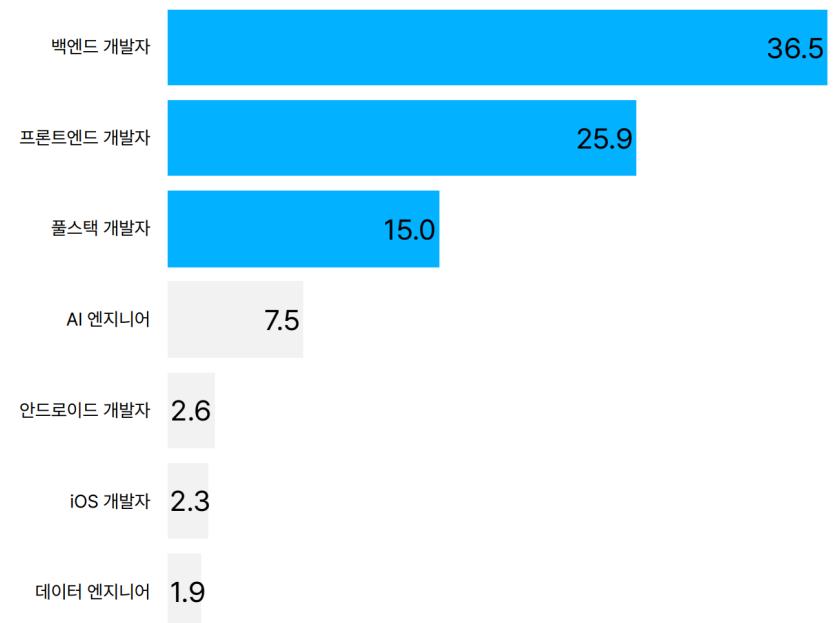
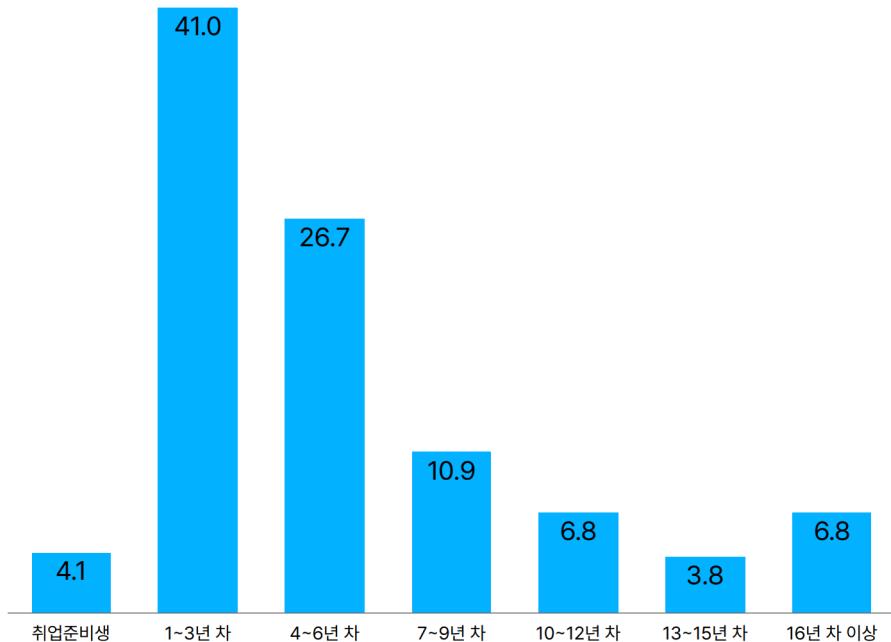
서류 탈락을 면했다면 그 이후부터 이력서의 중요도는 떨어진다고 볼 수 있음

깃허브 코드 확인을 가장 중요하게 생각하는 비중은 매우 낮음.

2025 원티드 개발자 리포트

개요

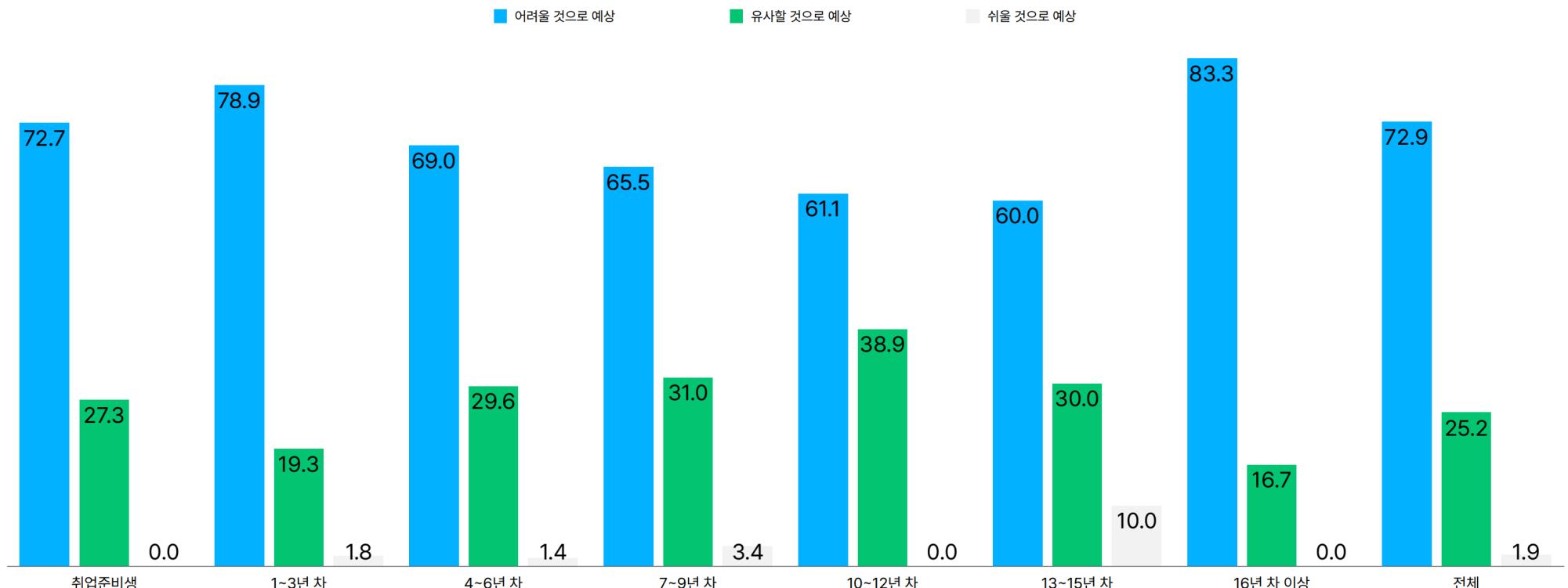
- 조사기간 : 2024-12-02 ~ 2025-01-07
- 300여 명의 온라인 설문 조사 결과와 원티드의 채용 데이터를 바탕으로 제작
- 채용을 많이 하는 개발 경력 TOP3: 3년 ~5년 미만 > 5년 ~ 8년 미만 > 1년 ~ 3년 미만



<https://www.wanted.co.kr/events/2025devreport>

2025 원티드 개발자 리포트

2025년의 취업 / 이직 난이도 예상



<https://www.wanted.co.kr/events/2025devreport>

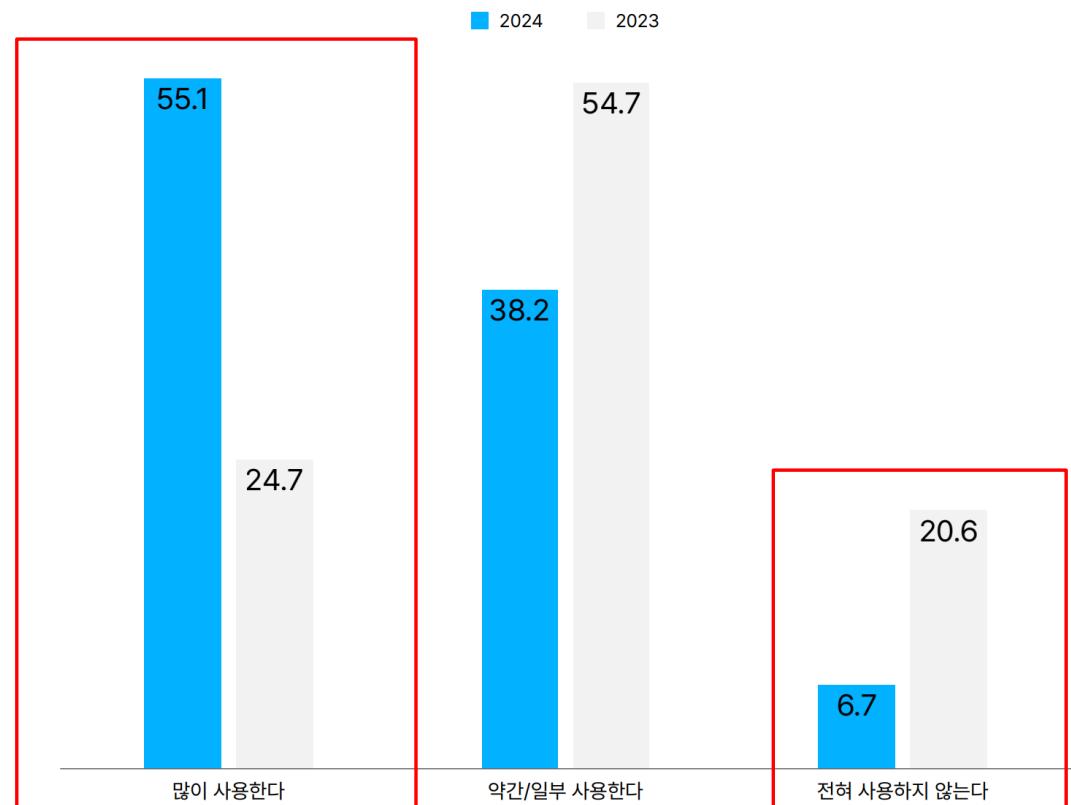
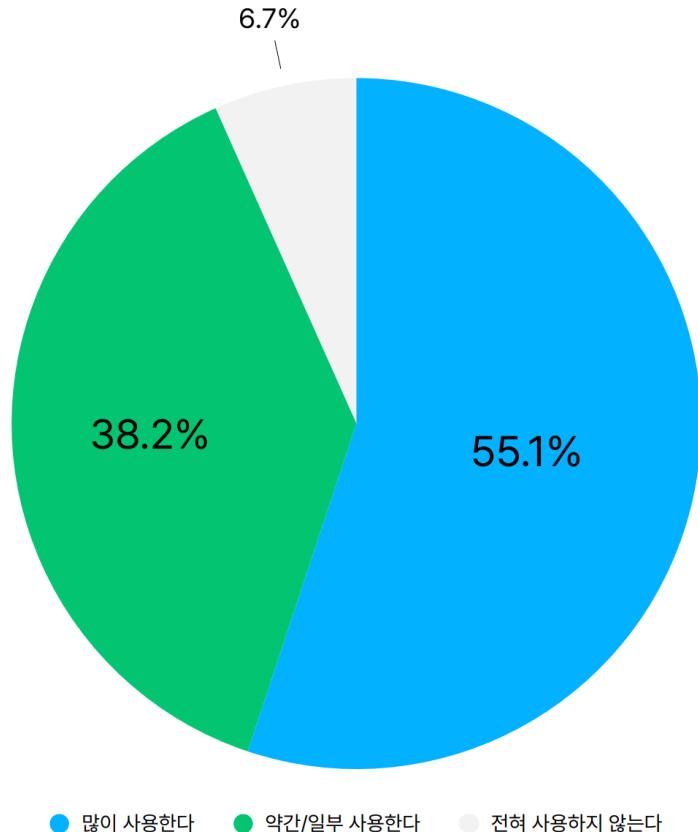
© 2025 INSILICOGEN, INC. ALL RIGHTS RESERVED. 193

구직자가 자신을 어필하는 부분 vs 면접관이 채용 시 보는 부분



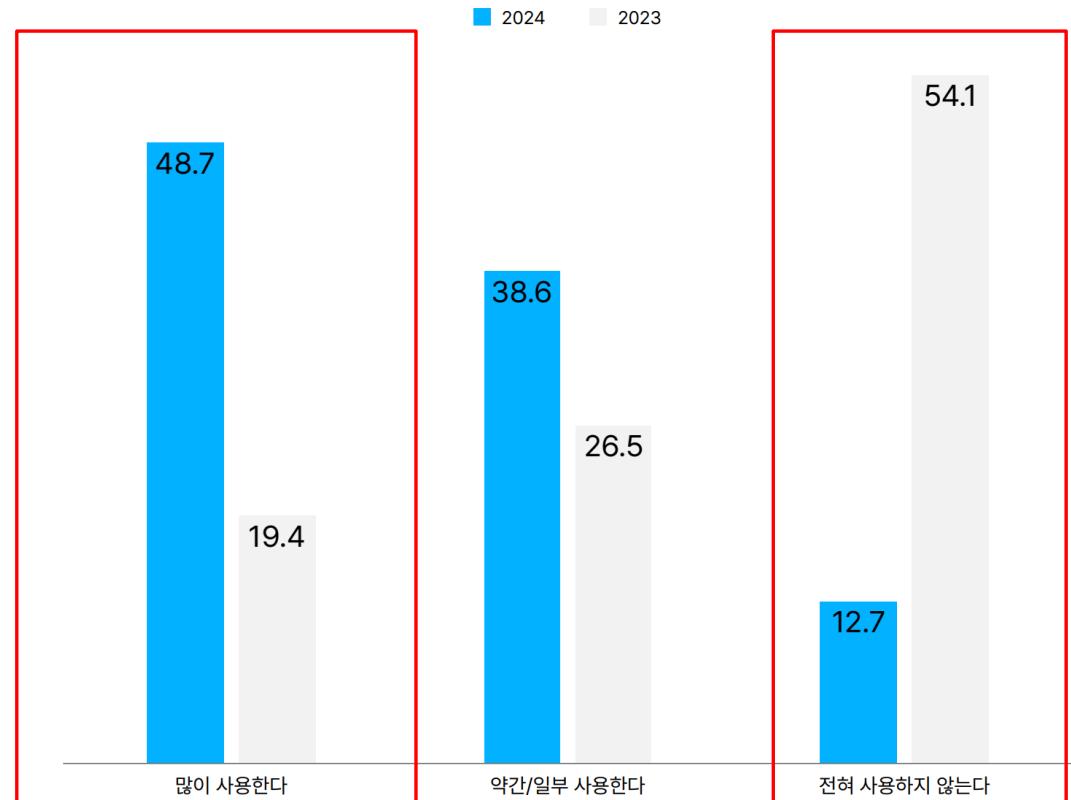
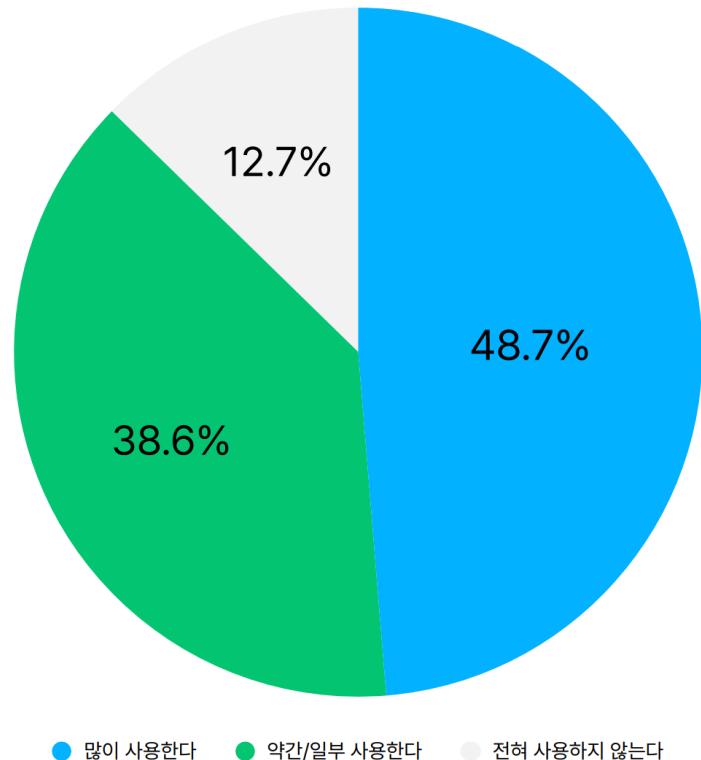
<https://www.wanted.co.kr/events/2025devreport>

Q. 업무에 생성 AI를 활용하고 계신가요?



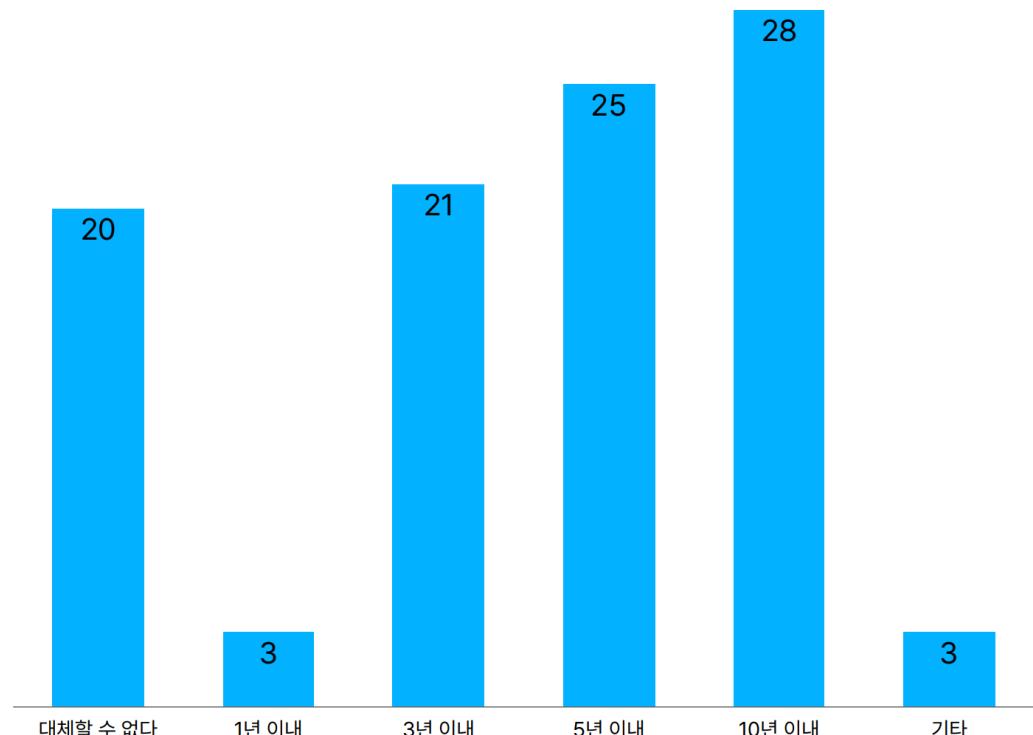
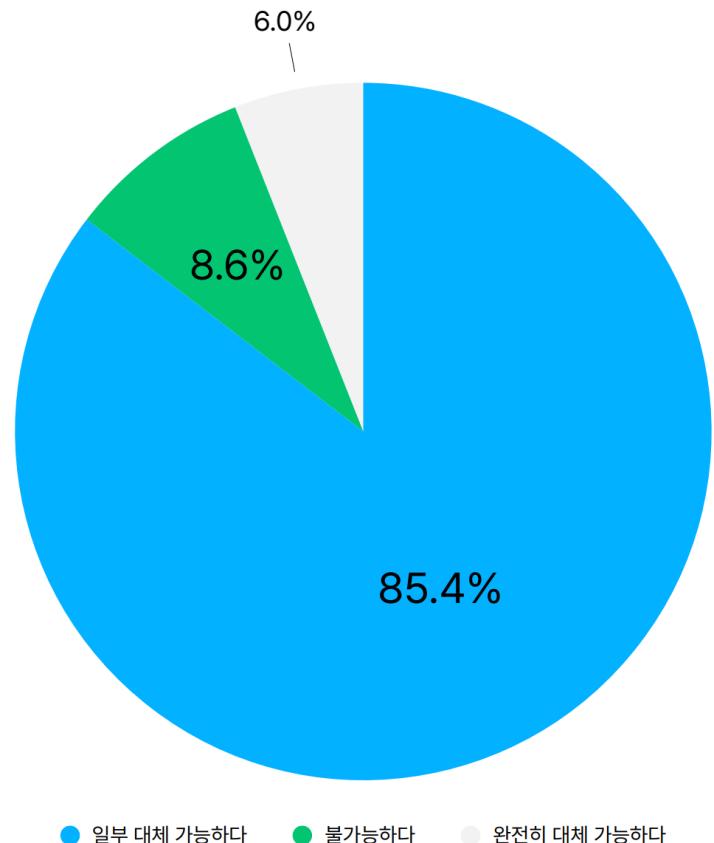
<https://www.wanted.co.kr/events/2025devreport>

Q. 업무 외 개인 프로젝트에 생성 AI를 활용하고 계신가요?



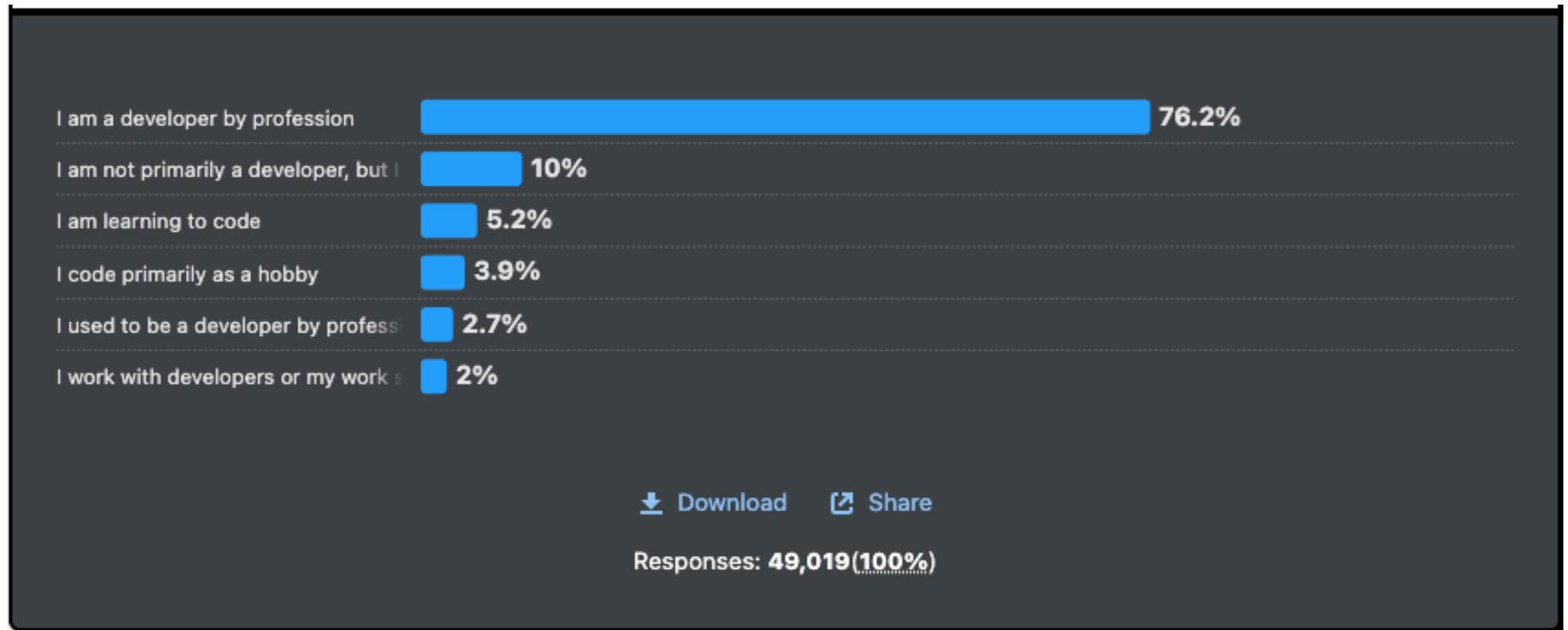
<https://www.wanted.co.kr/events/2025devreport>

Q. 생성 AI가 개발자를 대체할 수 있다고 생각하시나요?



<https://www.wanted.co.kr/events/2025devreport>

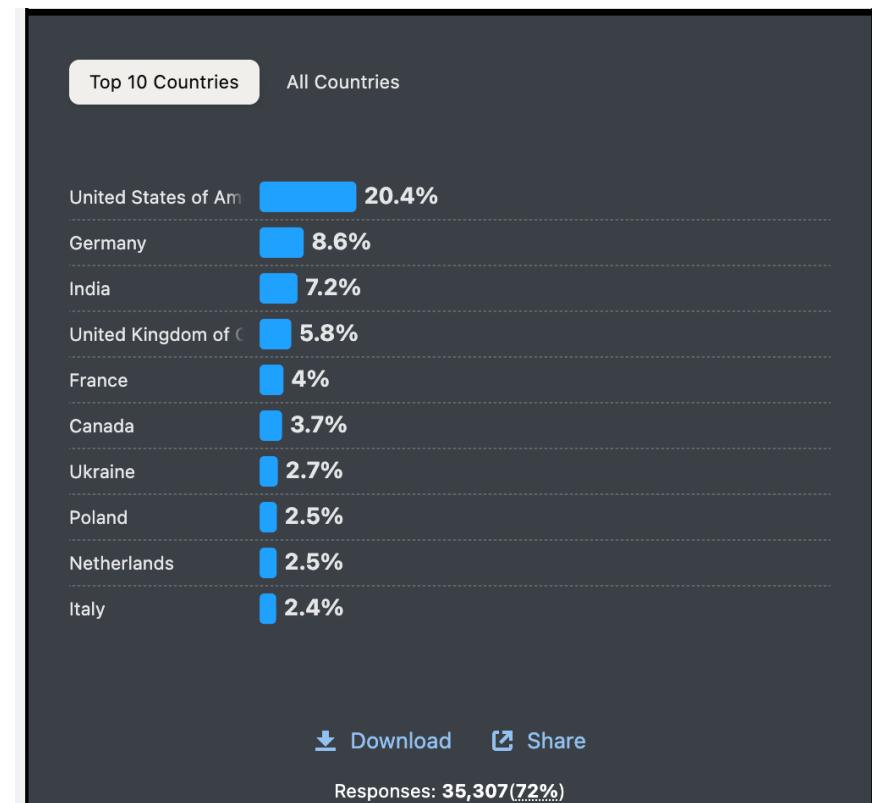
Q. 코딩을 하시나요? 자신을 가장 잘 표현한 문장은?



<https://survey.stackoverflow.co/2025/developers#education-experience-ai-learn-how>

2025 Stack Overflow 설문조사

Q. 나이 및 국적



<https://survey.stackoverflow.co/2025/developers#education-experience-ai-learn-how>

2025 Stack Overflow 설문조사

Q. 코딩을 어떻게 배웠나요? 해당 문항 모두 선택

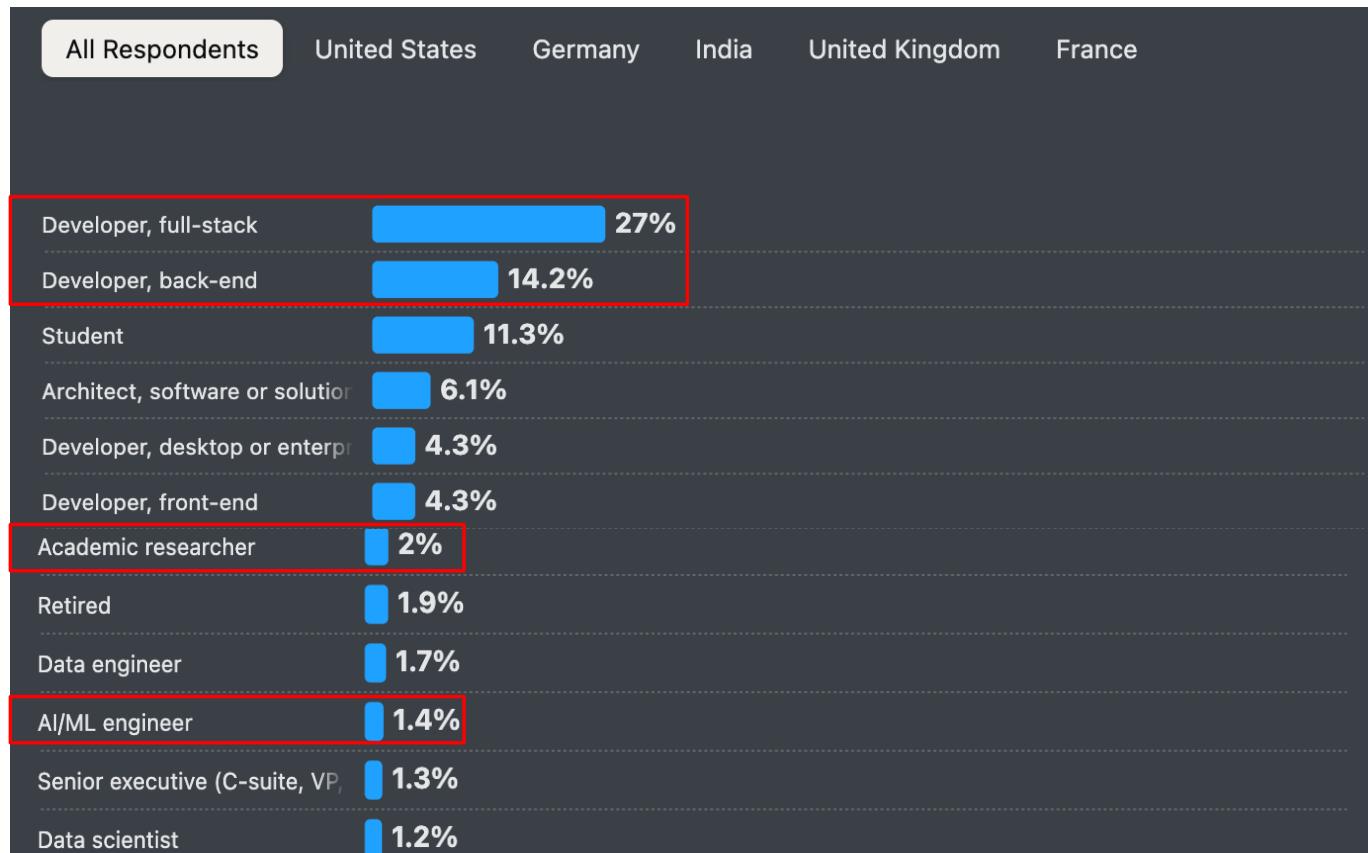


기술 문서 및 다양한 온라인 리소스가 가장 인기 있는 학습 방식

AI를 활용하는 비중도 매우 높음

2025 Stack Overflow 설문조사

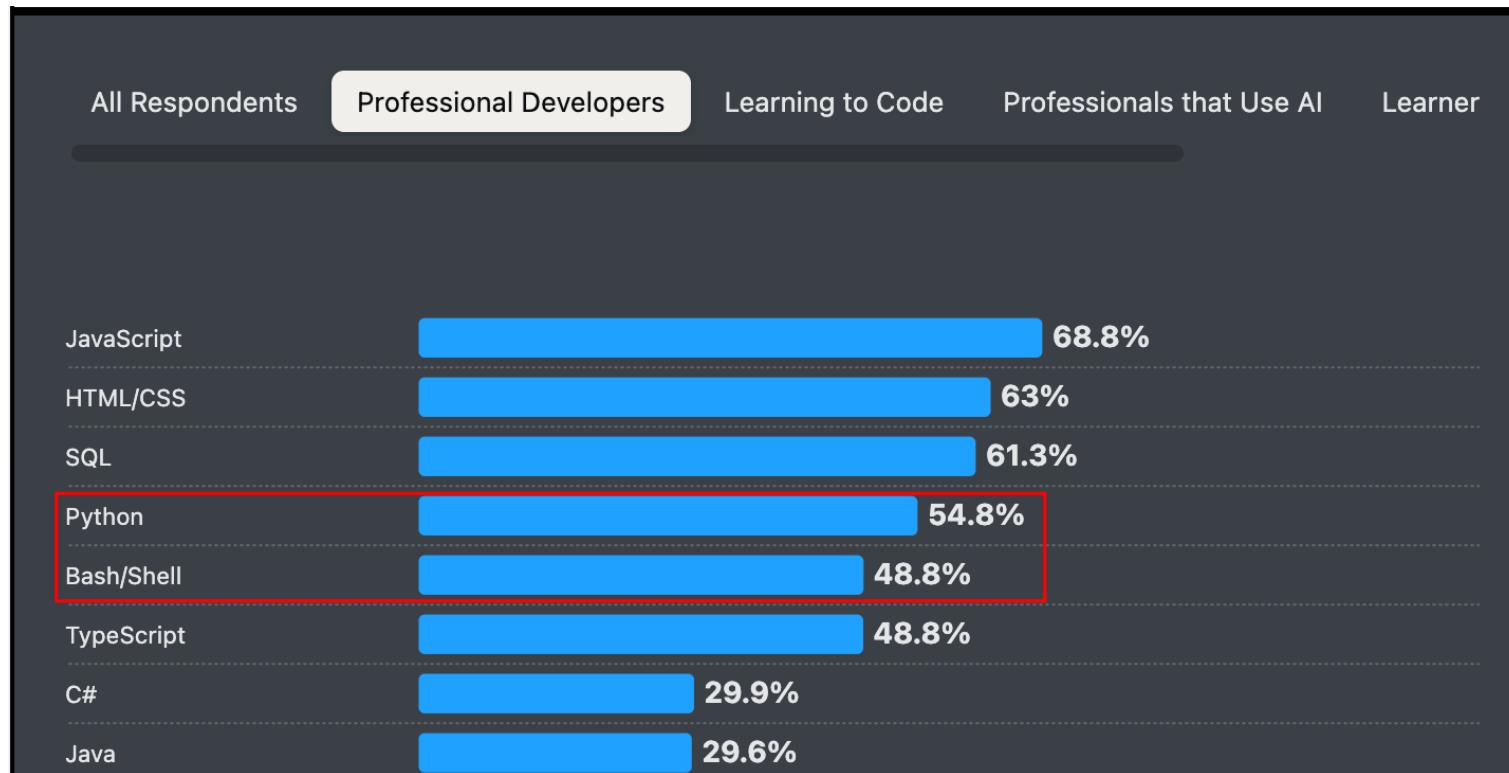
Q. 지난 1년간 가장 많은 시간을 보낸 직무를 가장 잘 표현한 문항은?



<https://survey.stackoverflow.co/2025/developers#education-experience-ai-learn-how>

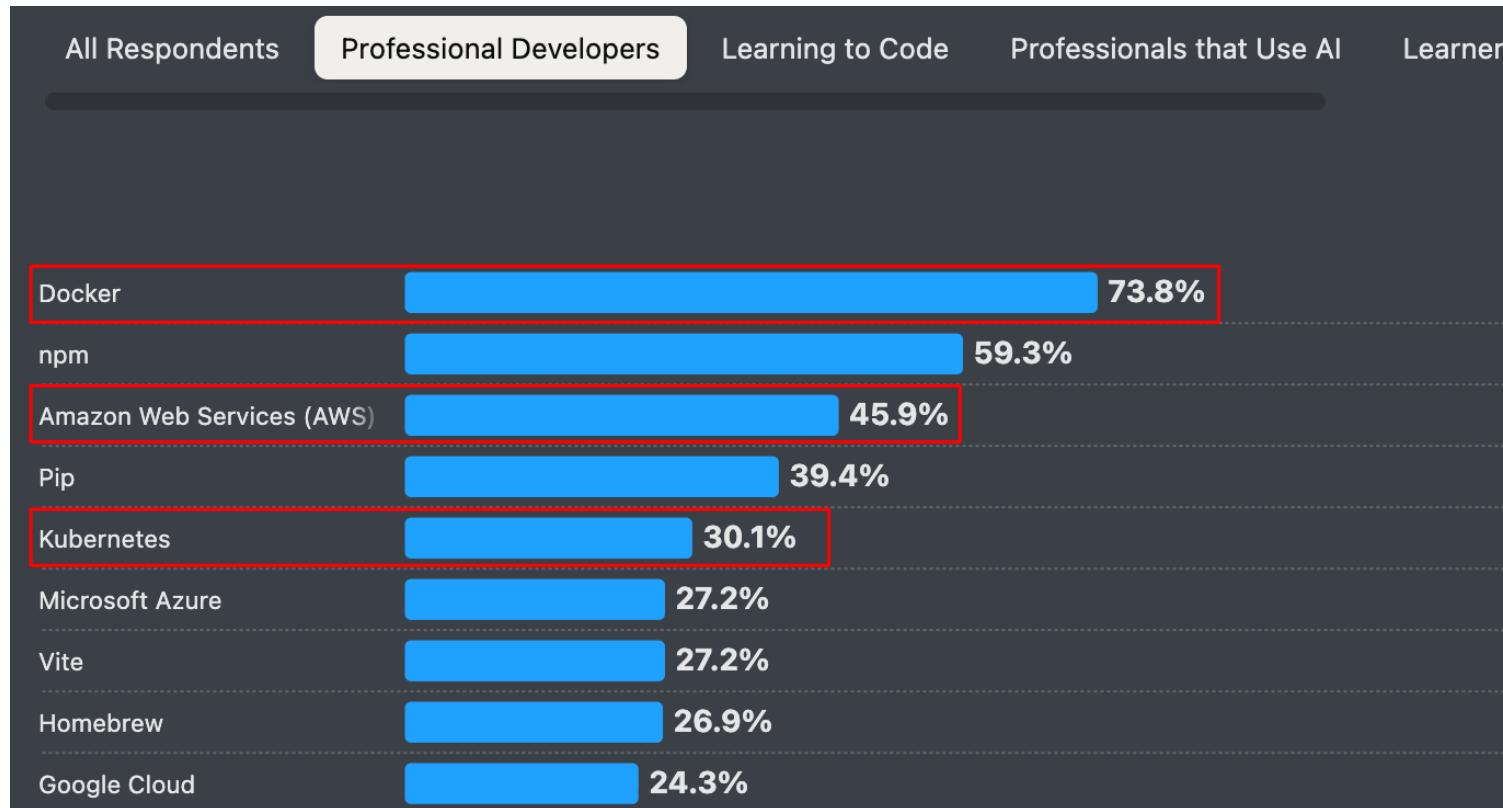
2025 Stack Overflow 설문조사

Q. 지난 1년간 많이 써온 그리고 앞으로 1년간 쓰고 싶은 프로그래밍 언어를 모두 선택



<https://survey.stackoverflow.co/2025/developers#education-experience-ai-learn-how>

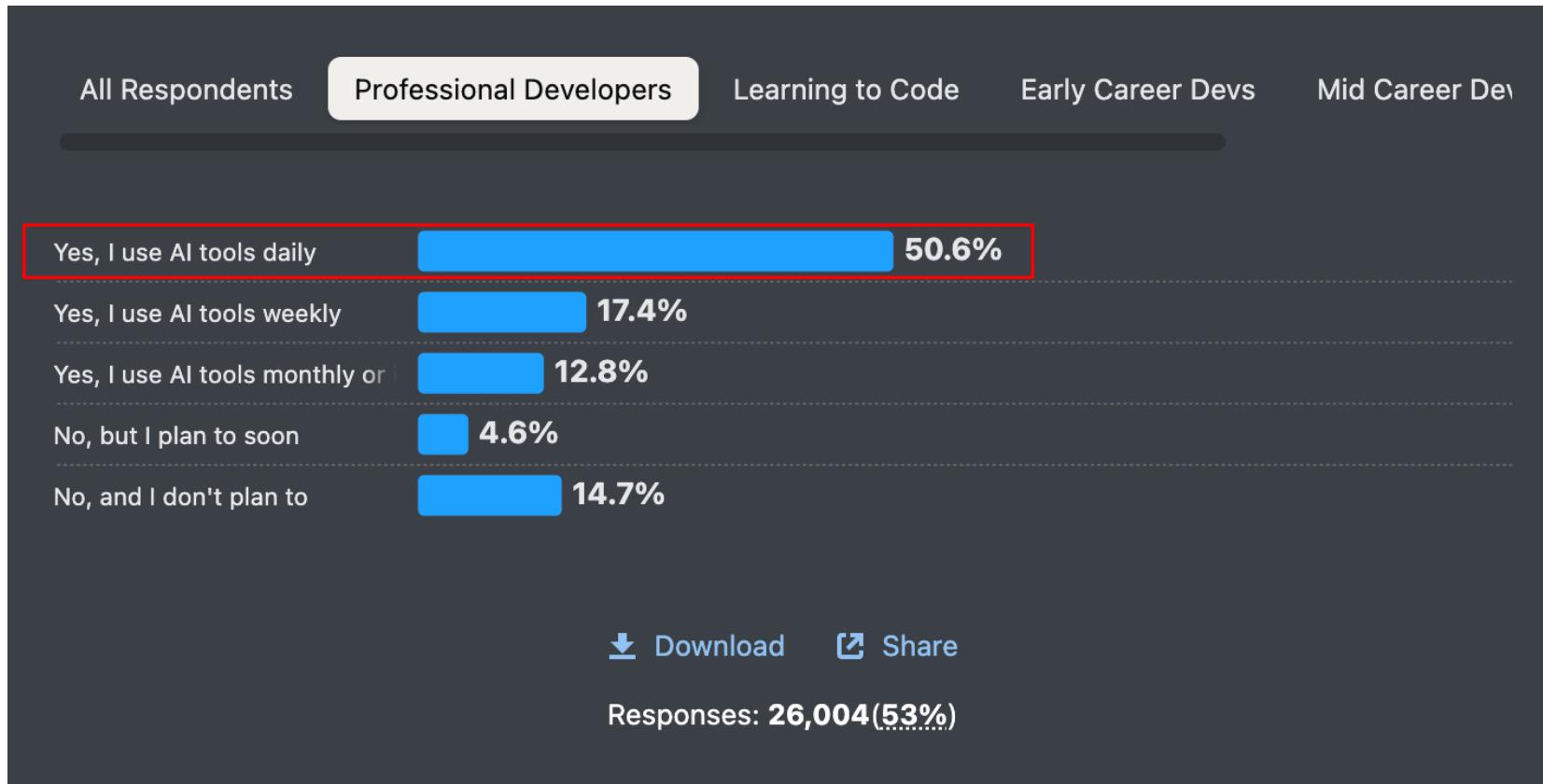
Q. 지난 1년간 많이 써온 그리고 앞으로 1년간 쓰고 싶은 클라우드 관련 플랫폼, 도구, 솔루션



<https://survey.stackoverflow.co/2025/developers#education-experience-ai-learn-how>

2025 Stack Overflow 설문조사

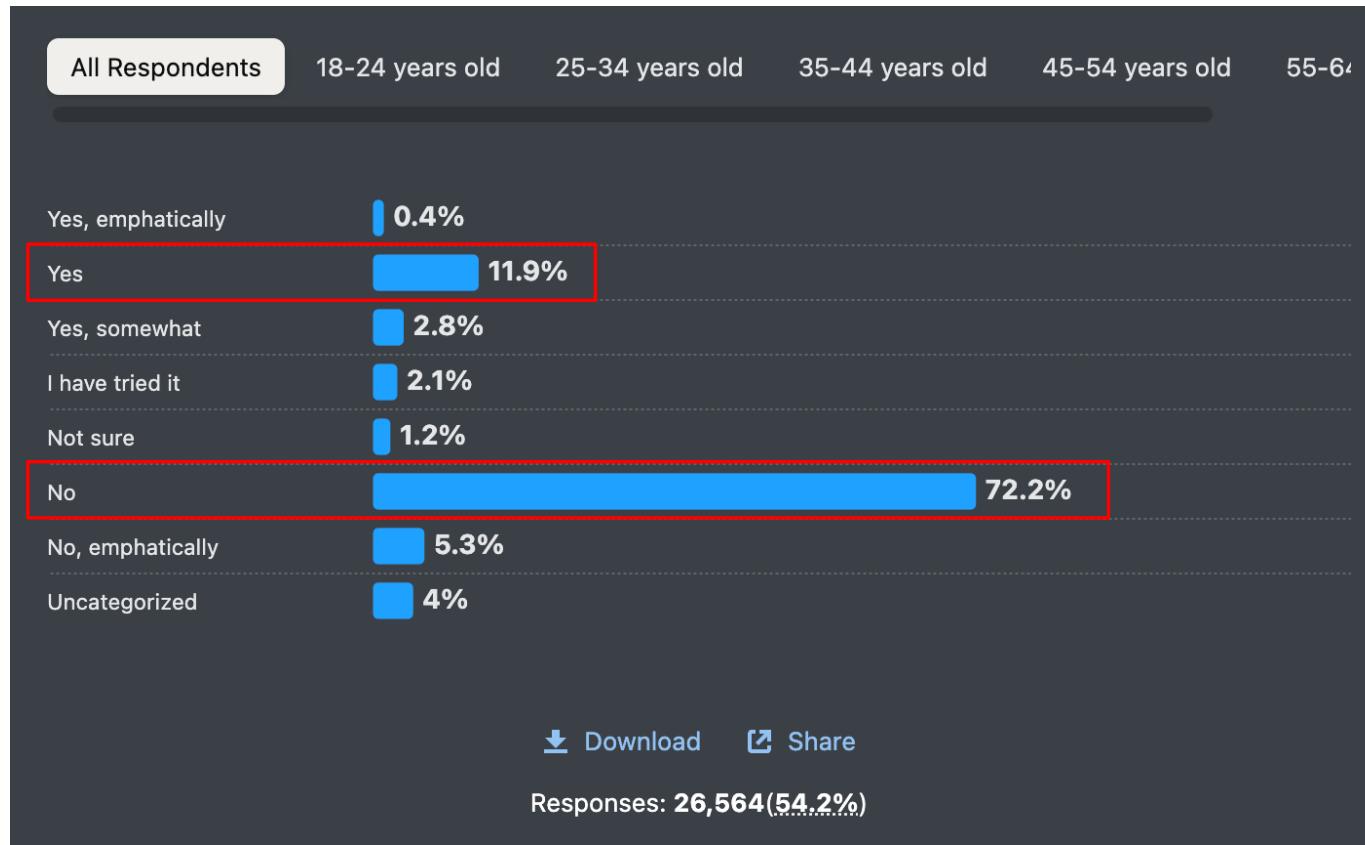
Q. 현재 개발 과정에서 AI를 사용중인가요?



<https://survey.stackoverflow.co/2025/developers#education-experience-ai-learn-how>

2025 Stack Overflow 설문조사

Q. 바이브 코딩을 하고 있나요?



<https://survey.stackoverflow.co/2025/developers#education-experience-ai-learn-how>

AI 엔지니어 로드맵

AI 엔지니어가 하는 일

- <https://roadmap.sh/ai-engineer>
- <https://www.youtube.com/watch?v=YcZFRsjbwW4>
- <https://chatgpt.com/share/69197d6e-70c8-8013-9f48-99634dcccd66a>

AI 엔지니어가 하는 일

데이터 수집 및 전처리	모델 설계 및 학습	모델 배포 및 API화	성능 최적화 및 운영
<ul style="list-style-type: none">• 사내 데이터 웨어하우스 (BigQuery, Snowflake 등)에서 모델 학습에 필요한 데이터를 가져옵니다.• 모델 학습에 필요한 데이터를 수집하고, 결측치 제거, 정규화, 토큰화 등 전처리 과정을 거칩니다.• 데이터 파이프라인 자동화를 위해 Airflow나 Prefect를 활용하고, 피처 엔지니어링을 위해 Feast, Tecton 같은 Feature Store를 사용합니다.	<ul style="list-style-type: none">• 문제에 맞는 머신러닝 또는 딥러닝 모델 (예: 분류기, 회귀모델, CNN, RNN, Transformer 등)을 설계하고 학습시킵니다.• PyTorch, TensorFlow, scikit-learn 같은 프레임워크를 활용합니다.• 하이퍼파라미터 튜닝, 교차 검증, 오버피팅 방지 등의 기법도 함께 고려합니다.	<ul style="list-style-type: none">• 학습된 모델을 실제 서비스에서 사용할 수 있도록 API로 감싸 배포합니다.• Flask, FastAPI 같은 프레임워크를 이용해 AI 모델을 서버와 연동합니다.• 모델 서빙 플랫폼(예: NVIDIA Triton(유지), Ray Serve)을 사용하기도 합니다.	<ul style="list-style-type: none">• 모델 추론 속도 개선, 경량화(양자화, 프루닝), 캐싱 등을 통해 성능을 최적화합니다.• 실시간 로그 분석, 모니터링, A/B 테스트 등을 통해 모델의 실서비스 성능을 점검합니다.• MLOps 도구(예: MLflow, Airflow, DVC, Kubeflow)를 통해 모델의 학습-배포 주기를 자동화합니다.

https://sprint.codeit.kr/blog/ai-engineer-job-roadmap?utm_source=chatgpt.com

2025 원티드 개발자 리포트

개발자에게 필요한 역량

하드 스킬(기술력) 강화

- 최신 프로그래밍 언어 및 기술 트렌드 학습: 지속적으로 변화하는 기술 환경에서 최신 프로그래밍 언어, 프레임워크, 도구를 학습
- 복잡한 문제 해결 능력 향상: 알고리즘과 자료 구조를 깊이 있게 학습하고, 문제 해결 능력을 키우기 위해 코딩 테스트 및 알고리즘 문제를 꾸준히 연습
- 실제 프로젝트 경험 쌓기: 오픈 소스 프로젝트에 참여하거나 개인 프로젝트를 진행하며 실전 경험을 쌓기

소프트 스킬(협업 능력) 강화

- 효과적인 커뮤니케이션 스킬 익히기: 개발 과정에서 팀원들과 의견을 나누고, 자신의 생각을 논리적으로 전달하는 연습하기
- 팀워크 및 협업 능력 개발: 팀 내에서 주도적으로 코드 리뷰 세션을 제안하거나, 테크 세션을 통해 팀워크를 강화
- 갈등 해결 능력 키우기: 개발 과정에서 팀원들 간의 의견이 갈릴 때, 효과적으로 중재하고 합리적인 결정을 내리는 능력 키우기

AI 활용 역량 강화

- AI 개발 도구와 자동화 프로세스 익히기: 깃허브 코파일럿 (GitHub Copilot), ChatGPT 등 AI 기반 코드 작성 도구를 활용
- AI를 통한 코드 품질 개선: 단순히 AI가 제공하는 코드에 의존하지 않고, AI가 생성한 코드를 평가하고 개선

이제 기술력과 소프트 스킬 중 하나만을 선택하는 것은 더 이상 유효하지 않습니다. 이 세 가지 역량을 조화롭게 발전시키는 것이 현대 개발자가 갖춰야 할 가장 중요한 능력입니다.

겁먹지 말고 도전합시다

이력서에 있어서는 무엇보다도 '완벽한 이력서란 없다.'라는 말을 먼저 해드리고 싶습니다.

멘토링을 하거나 이력서 첨삭을 하다 보면 지원자분이 지원할 때가 되었는데도 완벽한 이력서가 될 때까지 지원을 미루고 미루다 좋은 기회를 많이 놓치는 경우를 많이 봅니다. 적당히 준비가 되었다면 일단 지원해보고 시장의 평가를 받아보세요. 미루고 미루다 보면 더 간절한 사람에게 그 기회가 돌아갑니다.

Thank you!

Contact Info.

OFFICE	경기도 용인시 기흥구 흥덕1로 13 흥덕IT밸리 타워 A동 2901~2904, 2906호
EMAIL	info@insilicogen.com
TEL	031 278 0061
FAX	031 278 0062

설문 조사

고생 많으셨습니다! QR로 접속하셔서 설문조사 및 피드백 부탁드립니다

