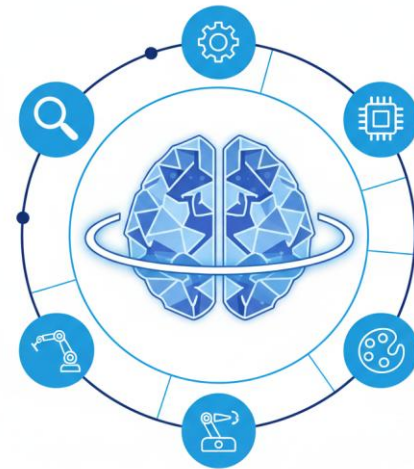


# 파이썬 객체와 모듈

함수/ 클래스/ 모듈



# 함수(function)

+ 파이썬의 함수에 대해 알려줘



## 1 함수란?

함수는 특정 작업을 수행하는 코드 묶음이에요.

👉 한 번 만들어 두면

- 여러 번 사용할 수 있고
- 코드가 깔끔해지고
- 수정도 쉬워져요

예: "두 수를 더하는 기능", "이름을 출력하는 기능" 같은 것들



# 함수(function)

- 사용자 정의 함수의 형태

```
def 함수 이름():  
    함수의 내용
```

return 반환값이 없는 함수

```
def 함수 이름(매개 변수):  
    함수의 내용  
    return 반환값
```

return 반환값이 있는 함수



# 함수(function)

- return이 없는 함수

```
def greet():  
    print("Hello, World!")
```

함수의 정의

```
def greet2(name):  
    print(f"Hello, {name}!")
```

```
greet() # Hello, World!  
greet2("Elsa") # Hello, Elsa!  
greet2("명제") # Hello, 명제!
```

함수의 호출



# 함수(function)

- return이 있는 함수

```
def message():  
    return "Good Luck!"  
  
def square(x):  
    return x * x  
  
def add(a, b):  
    return a + b  
  
print("5의 제곱:", square(5)) # 5의 제곱: 25  
result = add(3, 5)  
print("3 + 5 =", result) # 3 + 5 = 8
```



# 함수(function)

- 리스트를 매개변수로 전달하여 합계, 평균 계산하기

```
def calc_sum(numbers):  
    total = 0  
    for num in numbers:  
        total += num  
    return total  
  
def average(numbers):  
    total = calc_sum(numbers)  
    return total / len(numbers)  
  
num_list = [1, 2, 3, 4, 5]  
print("리스트의 합:", calc_sum(num_list)) # 리스트의 합: 15  
print("리스트의 평균:", average(num_list)) # 리스트의 평균: 3.0
```



# 함수(function)

- **전역 변수(global variable)의 유효 범위**

전역 변수는 메인 함수 영역에 선언하여 사용하고, 영향 범위가 전체로 미친다. 프로그램이 종료되면 메모리에서 소멸한다.

- **지역 변수(local variable)의 유효 범위**

지역변수는 함수나 명령문(조건, 반복)의 블록 안에서 생성되며 블록{ }을 벗어나면 메모리에서 소멸한다.



# 함수(function)

- 변수의 유효 범위

```
def get_price():  
    price = 1000 * quantity #price - 지역 변수  
    print(f"{quantity}개에 {price}원 입니다.")  
  
# 전역 변수  
quantity = 2 # 수량  
get_price() # 호출(사용)  
  
# 변수 출력  
# print(price) # 소멸된 변수임(오류 발생)
```





# 함수(function)

- 정적 변수의 유효 범위

지역 변수에 **global** 키워드를 붙이면 정적 변수가 되어 값을 유지하고, 프로그램이 종료되면 메모리에서 소멸한다.

```
def click():  
    global x #지역 변수가 전역변수로 됨  
    # x = 0 #지역 변수  
    x = x + 1  
  
    print(f"x = {x}")  
  
x = 0 #전역 변수  
  
# 함수 호출  
click()  
click()  
click()
```

```
x = 1  
x = 2  
x = 3
```



# 함수(function)

- 기본 매개변수

매개변수를 초기화하여 선언하고 함수 호출시 매개변수를 생략하면 기본 값으로 출력된다.

```
def 함수 이름(변수1, 변수2=1):  
    코드블럭
```

```
# 버스 요금 계산 함수  
def take_bus(fare=1200):  
    print(f"버스 요금은 {fare}원입니다.")  
  
take_bus() # 버스 요금은 1200원입니다.  
take_bus(1500) # 버스 요금은 1500원입니다.  
  
# 거듭제곱 계산 함수  
def power(base, exponent=2):  
    return base ** exponent  
  
print("3의 제곱:", power(3)) # 3의 제곱: 9  
print("2의 세제곱:", power(2, 3)) # 2의 세제곱: 8
```



# 함수(function)

## ❖ 내장 함수(Built in Function)

함수	설명	사용 예
sum(iterable)	리스트나 튜플의 모든 요소의 합을 반환	sum([1, 2, 3]) 6 sum((1.2.3)) 6
max(iterable)	리스트나 튜플의 최대값을 반환	max([1, 2, 3]) 3 max((1.2.3)) 3
round(n, digit)	숫자를 입력받아 반올림하여 돌려줌	round(4.6) 5 round(4.4) 4
eval(expression)	문자열 표현식을 숫자로 변환	eval('1+2') 3
list(s)	반복가능한 문자열을 입력받아 리스트로 반환	list("python") ['p', 'y', 't', 'h', 'o', 'n']



# 함수(function)

## ❖ 내장 함수(Built in Function)

```
a = [1, 2, 3, 4]
b = (1, 2, 3, 4)
```

```
print(sum(a)) # 합계
print(sum(b))
```

```
print(max(a)) # 최대값
print(max(b))
```

```
print(min(a)) # 최소값
print(min(b))
```

```
# 반올림
```

```
print(round(2.74)) #3
print(round(2.14)) #2
```

```
# 소수 자리수
```

```
x = 706.351
print(round(x, 1)) # 소수첫째자리, 706.4
print(round(x, 0)) # 706.0
print(round(x)) # 정수 706
print(int(round(x, -1))) # 일의 자리 710
```



# AI 사용 – 코파일럿(copilot)

## ❖ 생성형 AI 코파일럿

### ■ 챗GPT 프롬프트 입력

+ ai 코파일럿에 대해 설명해 줘




좋은 질문이에요 👍

**\*\*AI 코파일럿(AI Copilot)\*\***은 말 그대로 *사람을 대신하는 AI*가 아니라, **사람 옆에서 함께 일해주는 똑똑한 조수**라고 보면 딱 맞아요.

#### **AI 코파일럿이란?**

AI 코파일럿은 **인공지능이 사용자의 작업을 보조**해 주는 시스템이에요.

사용자의 **의도·맥락·데이터**를 이해해서 제안, 자동완성, 분석, 요약 같은 걸 해줍니다.

 비유하자면

비행기 조종사 = 사람

코파일럿 = AI (옆에서 체크하고, 추천하고, 자동 처리)



# AI 사용 – 코파일럿(copilot)

## ❖ 생성형 AI 코파일럿

### ■ 챗GPT 프롬프트 입력

+ GitHub Copilot을 사용하는 방법을 알려줘



#### ✓ GitHub 계정

- <https://github.com> ↗
- 로그인만 하면 OK

#### ✓ 지원되는 에디터

가장 많이 쓰는 건:

- VS Code ★ (추천)
- JetBrains (PyCharm, IntelliJ 등)
- Visual Studio

👉 여기서는 VS Code 기준으로 설명할게요.

#### ✓ GitHub Copilot 구독

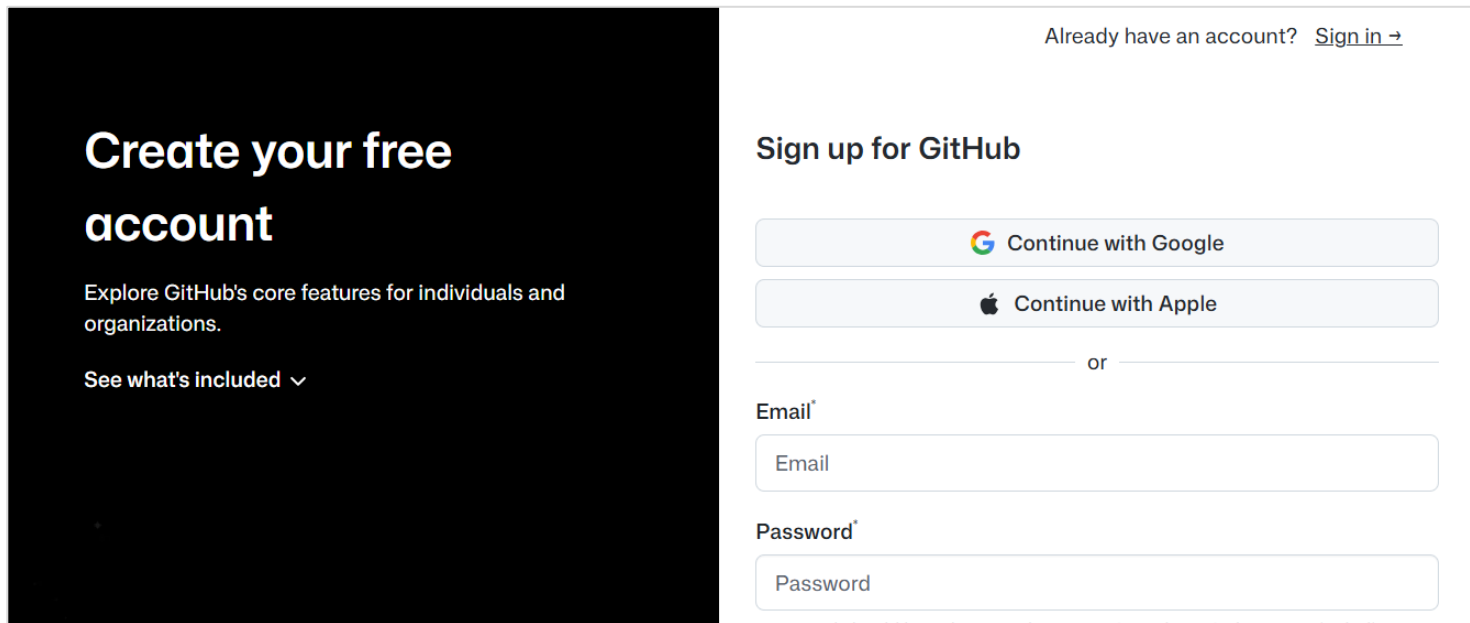
- 개인: 유료 (무료 체험 있음)
- 학생/교사: 무료
- 기업: 별도 플랜



# AI 사용 – 코파일럿(copilot)

## ❖ 깃허브 계정 만들기

<https://github.com/> > 회원가입(sign up)



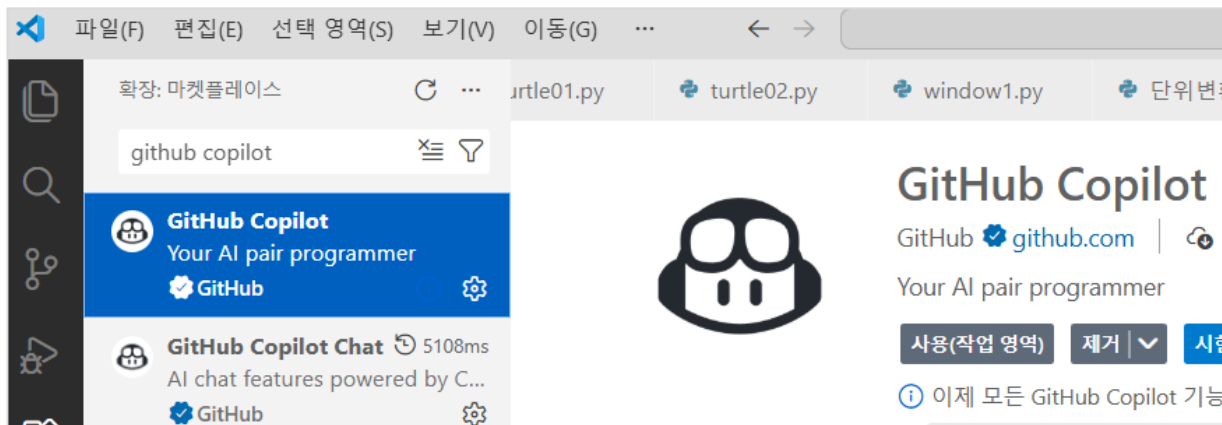
The screenshot shows the GitHub sign-up page. On the left, a black box contains the text "Create your free account" in white, followed by "Explore GitHub's core features for individuals and organizations." and "See what's included ▾". On the right, the page is white with the text "Already have an account? [Sign in →](#)" at the top. Below this is the heading "Sign up for GitHub". There are two buttons: "Continue with Google" (with the Google logo) and "Continue with Apple" (with the Apple logo). Below these is a horizontal line with the word "or" in the center. Underneath is the "Email\*" label followed by an input field containing the placeholder text "Email". Below that is the "Password\*" label followed by an input field containing the placeholder text "Password".



# AI 사용 – 코파일럿(copilot)

## ❖ VS Code에서 GitHub Copilot 사용하기

확장 > github copilot 검색 > 설치





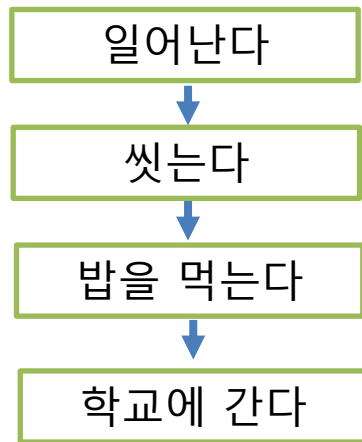
# 클래스와 객체

## ■ 객체(Object)란?

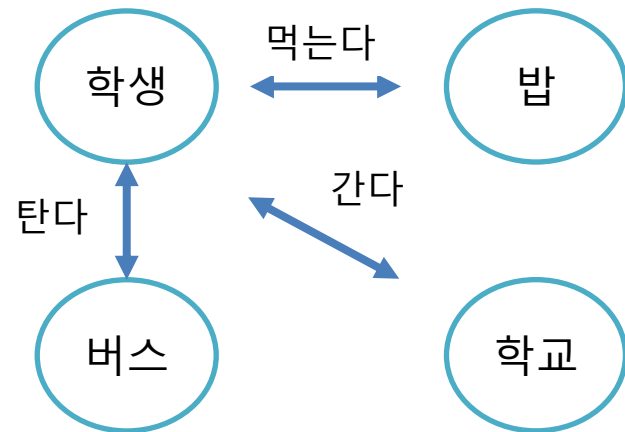
- "의사나 행위가 미치는 대상" -> 사전적 의미
- 구체적, 추상적 데이터 단위 (구체적- 책상, 추상적-회사)

## ■ 객체지향 프로그래밍(Objected Oriented Programming)

- 객체를 기반으로 하는 프로그래밍
- 먼저 객체를 만들고 객체 사이에 일어나는 일을 구현함.



<절차지향 -C언어>



<객체지향 - Python>



# 클래스와 객체

## ■ 클래스란?

- 객체 지향 프로그래밍(OOP)의 기본 개념 중 하나로, 관련된 속성과 메서드를 하나의 단위로 묶는 틀입니다.
- 클래스를 사용하면 코드의 재사용성과 유지보수성을 높일 수 있습니다.
- 클래스는 설계도와 같으며, 이를 기반으로 여러 객체(인스턴스)를 생성할 수 있습니다.

## ■ 클래스의 속성과 기능

- 객체의 특성(property), 속성(attribute) -> **멤버 변수**
- 객체가 하는 기능 -> **멤버 함수**



# 클래스와 객체

## ■ 클래스 정의 및 사용

```
class 클래스 이름 :  
    def __init__(self):  
        멤버변수  
  
    def 함수이름(self):  
        return
```

객체(인스턴스) = 클래스( )

객체.멤버변수  
객체.멤버 메서드

객체를 생성한 후 점(.) 연산자를 사용하여 멤버변수나 메서드에 접근함



# 클래스와 객체

## ■ 생성자(constructor)

- 객체가 생성될때 자동으로 호출되는 특별한 메서드
- 생성자는 `__init__`(매개변수)의 형태로 작성하고, 리턴값이 없다
- 클래스 내의 모든 함수(메서드)의 매개변수에 **self**를 넣어줌

Bike
color gears
ride() get_info()

속성(Attributes)

메서드(Methord)



# 클래스와 객체

## ▪ Bike 클래스 정의 및 사용

```
# 클래스의 생성자(Constructor)
class Bike:
    # 생성자 메서드 - 객체가 생성될 때 자동으로 호출되는 특별한 메서드
    def __init__(self, color, gears):
        self.color = color # 자전거 색상
        self.gears = gears # 자전거 기어 수

    def ride(self):
        print(f"{self.color} 자전거가 {self.gears}단 기어로 달린다.")

    def get_info(self):
        return f"자전거 색상: {self.color}, 기어 수: {self.gears}"

# 클래스 사용 예시
if __name__ == "__main__":
    my_bike = Bike("blue", 21)
    print(my_bike.get_info())
    my_bike.ride()
```



# 클래스와 객체

## ▪ 쇼핑물 장바구니 구현

```
# 장바구니 관리 클래스 - cart.py
class Cart:
    # 초기화 메서드
    def __init__(self, user_id):
        self.user_id = user_id
        self.items = [] # 장바구니 아이템 리스트

    # 아이템 추가 메서드
    def add_item(self, item):
        self.items.append(item)
        return f"아이템 '{item}'이(가) 장바구니에 추가되었습니다."

    # 아이템 제거 메서드
    def remove_item(self, item):
        if item in self.items:
            self.items.remove(item)
            return f"아이템 '{item}'이(가) 장바구니에서 제거되었습니다."
        else:
            return f"아이템 '{item}'이(가) 장바구니에 없습니다."
```



# 클래스와 객체

## ▪ 쇼핑물 장바구니 구현

```
# 장바구니 조회 메서드
def view_cart(self):
    if not self.items:
        return "장바구니가 비어 있습니다."
    return f"장바구니 아이템: {' '.join(self.items)}"

# 장바구니 사용 예시
if __name__ == "__main__":
    cart = Cart(user_id="user123")
    print(cart.add_item("노트북"))
    print(cart.add_item("마우스"))
    print(cart.view_cart())
    print(cart.remove_item("마우스"))
    print(cart.view_cart())
```



# 모듈(Module)과 패키지

- 모듈(module)

모듈은 Python 파일 하나를 의미하며, 다른 Python 프로그램에서 모듈의 함수, 클래스 등을 재사용할 수 있다.

- 사용자 모듈 정의

```
# my_lib/todos.py
# 요리하다
def cook(name):
    return f"{name} 요리를 합니다."

# 청소하다
def clean(place):
    return f"{place}를 청소합니다."
```





# 모듈(Module)과 패키지

- 사용자 정의 모듈

```
# my_lib/bike.py
class Bike:
    # 생성자 메서드 - 객체가 생성될 때 자동으로 호출되는 특별한 메서드
    def __init__(self, color, gears):
        self.color = color # 자전거 색상
        self.gears = gears # 자전거 기어 수

    def ride(self):
        print(f"{self.color} 자전거가 {self.gears}단 기어로 달린다.")

    def get_info(self):
        return f"자전거 색상: {self.color}, 기어 수: {self.gears}"
```



# 모듈(Module)과 패키지

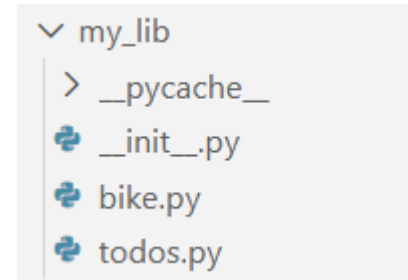
## ❖ 패키지

여러 모듈을 묶은 디렉터리 구조로 파이썬의 패키지로 인정받기 위해서는 `__init__.py`를 필수로 생성해야 한다.

`__init__.py`는 코드가 없는 빈 파일이다.

### ▪ 패키지 사용하기

- **import** 패키지 이름
- **from** 패키지 이름 **import** 모듈 이름
- **from** 패키지이름.모듈이름 **import** 함수, 클래스



# 모듈(Module)과 패키지

## ❖ 모듈 사용하기

**from** 패키지이름. 모듈 이름 **import** 함수, 클래스

```
# main.py
# my_lib 패키지에서 모듈 불러오기
from my_lib.todos import cook, clean

print("요리 시작!")
print(cook("파스타"))
print("청소 시작!")
print(clean("욕실"))

from my_lib.bike import Bike
# Bike 클래스 사용
my_bike = Bike("red", 18)
print(my_bike.get_info())
my_bike.ride()
```



# 모듈(Module)과 패키지

- 주요 모듈 및 패키지

모듈	설명
math	수학 계산과 관련된 모듈
datetime	날짜 및 시간과 관련된 모듈
time	시간과 관련된 모듈
random	난수를 발생시키는 모듈
os	운영 체제(OS) 자원 제어 관련 모듈



# math 모듈

## ● math 모듈

기능	함수의 사용
올림	<code>math.ceil(2.54)</code> # 3
내림	<code>math.floor(2.54)</code> # 2
제곱근	<code>math.sqrt(16)</code> # 4.0
원주율	<code>math.pi</code> # 3.1415



# math 모듈

## ● math 모듈

```
import math

# 올림
print(math.ceil(2.54))

# 반올림 - math 모듈이 아님
print(round(2.54))

# 내림(버림)
print(math.floor(2.54))

# 제곱근 - 실수로 반환
print(math.sqrt(2))
print(math.sqrt(25))
```

```
3
3
2
1.4142135623730951
5.0
3.141592653589793
원의 넓이: 50.27
```



# datetime 모듈

## ● datetime 모듈

```
import datetime

# datetime.datetime - 날짜와 시간을 사용
now = datetime.datetime.today() #오늘 날짜
print(now) # 2025-05-18 11:16:33.750807
# 년, 월, 일 출력
print(now.year)
print(now.month)
print(now.day)

# 현재 날짜 표기
print(f"{now.year}. {now.month}. {now.day}.")

# 시, 분, 초 출력
print(now.hour)
print(now.minute)
print(now.second)

# 현재 시간 표기
print(f"{now.hour} : {now.minute} : {now.second}")
```

```
2025-08-01 10:10:24.530677
2025
8
1
2025. 8. 1.
10
10
24
10 : 10 : 24
2025-08-15
2025-08-01
```



# datetime 모듈

## ● datetime 모듈

```
# 특정한 날짜 설정
# datetime.date - 날짜만 사용 가능
the_day = datetime.date(2025, 8, 15)
print(the_day)
```

```
today = datetime.date.today()
print(today)
```

```
# DDay 계산
print("광복절까지 몇일 남았나(DDay)?")
```

```
remain_day = the_day - today
print(f"광복절까지 {remain_day.days}일 남았습니다.")
```

```
2025-08-15
2025-08-01
광복절까지 몇일 남았나(DDay)?
광복절까지 14일 남았습니다.
```





# Random 모듈

+ 파이썬의 random 모듈을 사용하는 방법에 대해서 알려줘



## 1 random 모듈 불러오기

```
python  
  
import random
```

📄 코드 복사

## 2 기본 난수 생성

### ◆ 0 이상 1 미만의 실수

```
python  
  
random.random()
```

📄 코드 복사

👉 예: 0.3748293

### ◆ 특정 범위의 정수

```
python  
  
random.randint(1, 10)
```

📄 코드 복사

👉 1 이상 10 이하 정수 하나



# Random 모듈

- random 모듈

```
# 난수 시드 설정 - 동일한 시드를 사용하면 동일한 난수 시퀀스 생성
random.seed(42)
print(f"시드 설정 후 난수: {random.randint(1, 10)}")
random.seed(42)
print(f"시드 설정 후 난수: {random.randint(1, 10)}")
random.seed() # 시드 초기화
print(f"시드 초기화 후 난수: {random.randint(1, 10)}")

# 동전 던지기 시뮬레이션
# '앞면' 또는 '뒷면' 중 하나를 무작위로 선택
# 방법1
coin = random.randint(0, 1)
if coin == 0:
    print("동전 던지기 결과: 앞면")
else:
    print("동전 던지기 결과: 뒷면")

# 방법2
coin = random.choice(['앞면', '뒷면'])
print(f"동전 던지기 결과: {coin}")
```



# Random 모듈

```
# 주사위 굴리기 시뮬레이션
dice_roll = random.randint(1, 6)
print(f"주사위 굴리기 결과: {dice_roll}")

# 로또 번호 생성 (1부터 45 사이의 숫자 중 6개 선택)
lotto_numbers = random.sample(range(1, 46), k=6)
print(f"생성된 로또 번호: {sorted(lotto_numbers)}")

# 리스트에서 무작위로 요소 선택
fruits = ['사과', '바나나', '딸기', '감', '사과']
selected_fruit = random.choice(fruits)
print(f"선택된 과일: {selected_fruit}")

# 리스트에서 무작위로 여러 요소 선택 (중복 허용)
selected_fruits = random.choices(fruits, k=2)
print(f"선택된 과일들 (중복 허용): {selected_fruits}")

# 리스트에서 무작위로 여러 요소 선택 (중복 비허용)
selected_fruits = random.sample(fruits, k=2)
print(f"선택된 과일들 (중복 비허용): {selected_fruits}")
```



# Random 모듈

- 숫자 맞추기 게임
  - 챗GPT 프롬프트 입력

파이썬으로 숫자 맞추기 게임을 만들 거야.  
1~100까지 무작위 숫자를 생성하고,  
내가 숫자를 입력하면 무작위 숫자와 비교해서 숫자가 큰지 작은지 알려줘  
숫자를 맞추면 몇 회 만에 맞췄는지 알려주고 게임을 종료해

+



# Random 모듈

- 숫자를 추측해서 맞추는 게임

```
import random

# 1~100 사이의 무작위 숫자 생성
answer = random.randint(1, 100)
# print(answer) # 디버깅용 출력, 실제 게임에서는 제거할 것

count = 0 # 시도 횟수

print("숫자 맞추기 게임!")
print("1부터 100 사이의 숫자를 맞춰보세요.")

while True:
    guess = int(input("숫자를 입력하세요: "))
    count += 1

    if guess > answer:
        print("더 작은 숫자입니다.")
    elif guess < answer:
        print("더 큰 숫자입니다.")
    else:
        print(f"정답입니다! {count}회 만에 맞췄어요.")
        break
```



# pyinstaller 모듈

- **pyinstaller 모듈**

파이썬으로 만든 프로그램을 "실행 파일(.exe)"로 묶어주는 도구이다.  
즉, 파이썬이 설치되지 않은 PC에서도 실행 가능하게 만들어준다.

- **pyinstaller 모듈 설치**  
**pip install pyinstaller**

- **터미널에 명령어 입력**

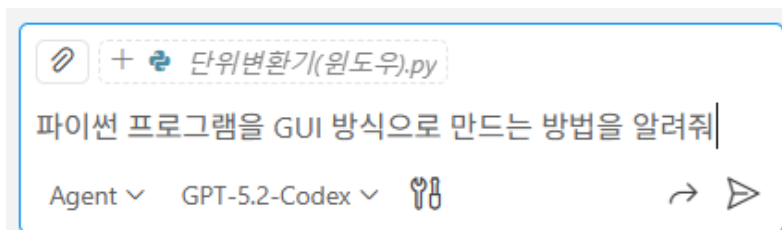
**C:\Wpyworks/module>pyinstaller --onefile guess\_number.py**

같은 경로에 dist 폴더 생성됨



# 윈도우 프로그래밍

## ■ 코파일럿 프롬프트 입력



- GUI 라이브러리 선택: 표준 `tkinter` (가장 간단), `PyQt/PySide` (기능 풍부), `wxPython` 등
- 기본 흐름: 창 생성 → 위젯 배치 → 이벤트 연결 → 실행

예) `tkinter` 최소 구조

- `Tk()` 로 창 생성
- `Label`, `Entry`, `Button` 등 위젯 추가
- `command` 로 버튼 클릭 이벤트 연결
- `mainloop()` 로 실행



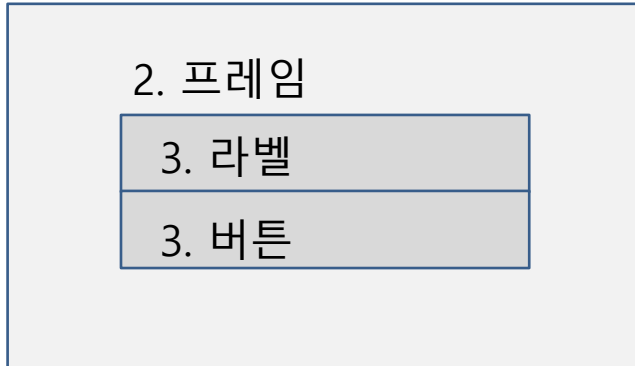
# 윈도우 프로그래밍

- GUI(Graphical User Interface)란?

그래픽 사용자 인터페이스를 줄여서 GUI라고 한다. GUI는 '화면'에 표시된 메뉴나 버튼으로 사용자와 상호 작용을 하는 간단한 프로그램이다.

**tkinter** 라이브러리를 사용한다. -> **import tkinter**

## 1. Tk 루트



개체이름	클래스
루트	Tk()
프레임	Frame
레이블	Label
입력상자	Entry
버튼	Button
출력상자	Text





# 윈도우 프로그래밍

## ■ 클릭 이벤트 – 버튼을 눌렀을때 문자 출력

```
from tkinter import *

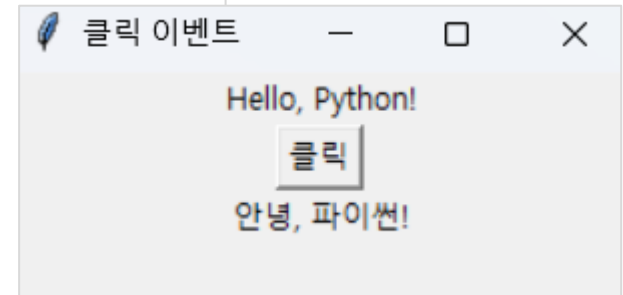
# 버튼 클릭시 호출되는 함수
def click():
    output.config(text="안녕, 파이썬!")

root = Tk() # 윈도우 창 생성
root.title("클릭 이벤트") # 창 제목 설정
root.geometry("250x100+200+100") # 가로x세로+x좌표+y좌표

# 라벨과 버튼 추가
# pack(): 위젯을 창에 추가하는 메서드
Label(root, text="Hello, Python!").pack()
Button(root, text="클릭", command=click).pack()

# 클릭 후 출력 라벨
output = Label(root, text="")
output.pack()

# 이벤트 루프 시작
root.mainloop()
```



# 윈도우 프로그래밍

- Button(버튼) - command

Button(frame, text="확인", command=**click**).pack()

※ click에 괄호를 하면 함수 생성시점에서 작동하고, 괄호를 생략하면  
클릭이 발생한 때 작동함

1. 콘솔에 출력

```
def click():  
    print("안녕~ 파이썬!")
```

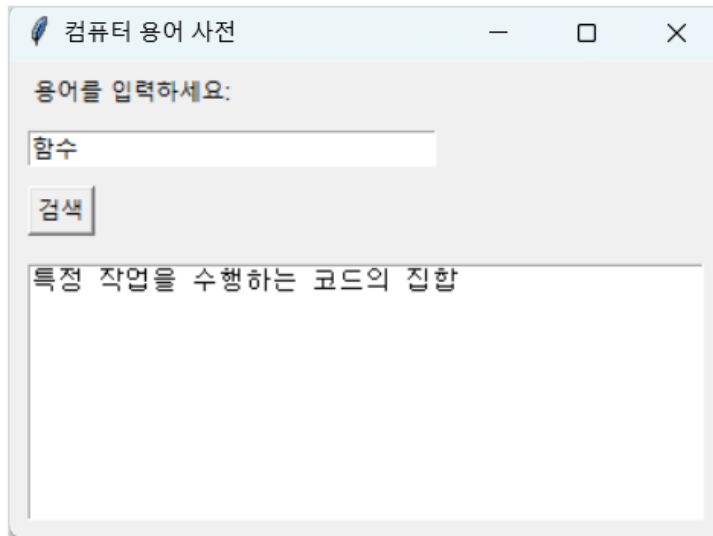
2. 레이블에 출력

```
def click():  
    output.config(text="안녕, 파이썬!")
```



# 컴퓨터 용어사전 앱

- 컴퓨터 용어 사전 – dictionary 자료구조 이용



## App 설명

- 용어를 미리 정의한다. – 딕셔너리 자료 구조
- 단어를 입력하고 검색 버튼을 누르면, 텍스트 상자에 뜻이 출력된다.
- 검색된 단어가 없는 경우 '사전에 없는 용어입니다.'고 출력된다.



# 컴퓨터 용어사전 앱

## ● 컴퓨터 용어사전.py

```
from tkinter import *
dict = {
    "알고리즘": "문제를 해결하기 위한 절차나 방법",
    "함수": "특정 작업을 수행하는 코드의 집합",
    "변수": "데이터를 저장하는 메모리 공간의 이름",
    "클래스": "객체 지향 프로그래밍에서 객체를 생성하기 위한 틀",
}

def search():
    # 입력된 단어로 사전 검색
    try:
        word = entry.get().strip() # 입력창에서 단어 가져오기
        if not word: # 단어가 비어있으면
            output.insert(END, "용어를 입력하세요.")
            return
        meaning = dict[word] # 사전에서 뜻 찾기
        output.delete(1.0, END) # 1번째 줄, 0번째 문자 = 첫 번째 위치
        output.insert(END, meaning) # 끝 위치 (텍스트의 마지막)
    except KeyError:
        output.delete(1.0, END)
        output.insert(END, "사전에 없는 용어입니다.")
```



# 컴퓨터 용어사전 앱

- 컴퓨터 용어사전.py

```
root = Tk()
root.title("컴퓨터 용어 사전")

Label(root, text="용어를 입력하세요:") \
    .grid(row=0, column=0, sticky=W, padx=10, pady=5) # 라벨 추가

entry = Entry(root, width=30)
entry.grid(row=1, column=0, sticky=W, padx=10, pady=5) # 입력창 추가

Button(root, text="검색", command=search) \
    .grid(row=2, column=0, sticky=W, padx=10, pady=5) # 버튼 추가

output = Text(root, width=50, height=10)
output.grid(row=3, column=0, sticky=W, padx=10, pady=10) # 출력창 추가

root.mainloop()
```



# 컴퓨터 용어사전 앱

## ● 코파일럿 채팅 입력

🔗 + 📄 컴퓨터용어사전.py

컴퓨터용어사전.py에 용어 등록 기능을 만들어서 컴퓨터용어사전2.0.py로 만들어줘

Agent ▾ GPT-5.2-Codex ▾ 🛠️ → ➤

컴퓨터 용어 사전 2.0

용어를 입력하세요:

이진수

검색

사전에 없는 용어입니다.

새 용어 등록

용어

이진수

뜻

0과 1로 이루어진 수의 체계

등록

컴퓨터 용어 사전 2.0

용어를 입력하세요:

변수

검색

데이터를 저장하는 메모리 공간의 이름

새 용어 등록

용어

변수

뜻

데이터를 저장하는 메모리 공간의 이름으로 변경할 수 있다.

등록

💡 뒀어쓰기

? 이미 있는 용어입니다. 뜻을 뒀어쓰까요?

예(Y) 아니요(N)



# 컴퓨터 용어사전 앱

- 실행 파일로 배포하기 (.py -> .exe)

- 모듈 설치

pip install pyinstaller

- 터미널 - 명령어

C:\Wpyworks/컴퓨터사전>pyinstaller --onefile  
--windowed 컴퓨터용어사전2.0.py

