

Python

Department of Electrical Engineering, Incheon National University
Cheolmin Jeong

2024. 8. 13.

강사 소개



■ 정철민 (Cheolmin Jeong)

- ~2023: 인천대학교 전기공학과 공학석사
- 2023~현재: 인천대학교 전기공학과 박사과정 재학 중
- 무인지능 시스템제어 연구실 소속
(지도교수: 강창묵 교수, <https://uniconlab.wixsite.com/main>)



- 관심분야: 최적제어, 선형제어, 예측제어, 고장진단, 자율주행, SLAM
- 적용플랫폼: 자율주행차량, 모바일로봇, 드론, 4족보행로봇 등
- Projects:

- 자율주행 전동휠체어 핵심 기술 개발
- 순찰로봇의 도심지 적용을 위한 실외 자율주행 기술 개발
- 인천공항 폐기물 이송 로봇 개발
- UAV 기체 상태 이상 진단 검증 및 연계 모듈 개발
- 다목적 개인이동형 서비스 로봇 플랫폼 기술개발 및 실증
- 자율주행 전동휠체어 의료기기인증 및 표준안을 위한 기능 구현
- ...



Contents

- 1) Python이란?
- 2) 개발 환경 설정
- 3) 기초 문법
- 4) 제어 구조
- 5) 자료 구조
- 6) 함수
- 7) 클래스

Python

1) Python이란?



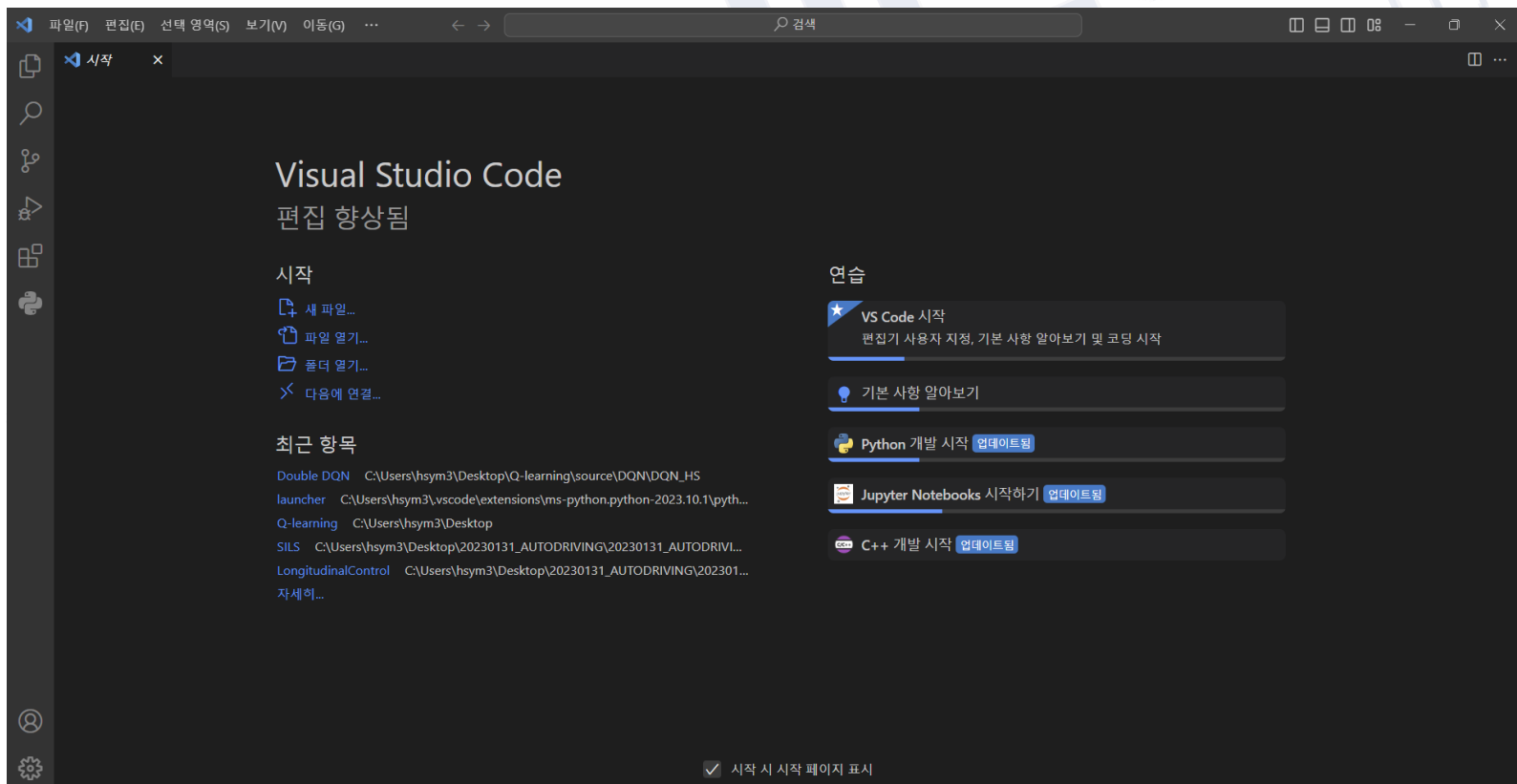
- Python(파이썬)은 코드의 가독성을 높이고, 작업을 보다 간단하게 수행할 수 있도록 1991년 ‘귀도 반 로섬(Guido van Rossum)’에 의해 개발된 프로그래밍 언어
- Python은 interpreter(인터프리터) 언어로, 코드를 작성한 후 별도의 컴파일 과정 없이 바로 실행이 가능하여 개발 과정을 더 빠르고 유연하게 함
- 오픈 소스 언어로서 수많은 라이브러리와 프레임워크가 개발되어 다양한 작업을 쉽고 효율적으로 처리 가능

Python

2) 개발 환경 설정

■ 개발 환경(IDE) : Visual Studio Code (VS Code)

- VS Code는 마이크로소프트에서 개발한 경량화 된 텍스트 에디터이며,
확장성, 디버깅 툴, Git 통합 등의 다양한 기능을 바탕으로 원활한 개발 환경 구축 가능.

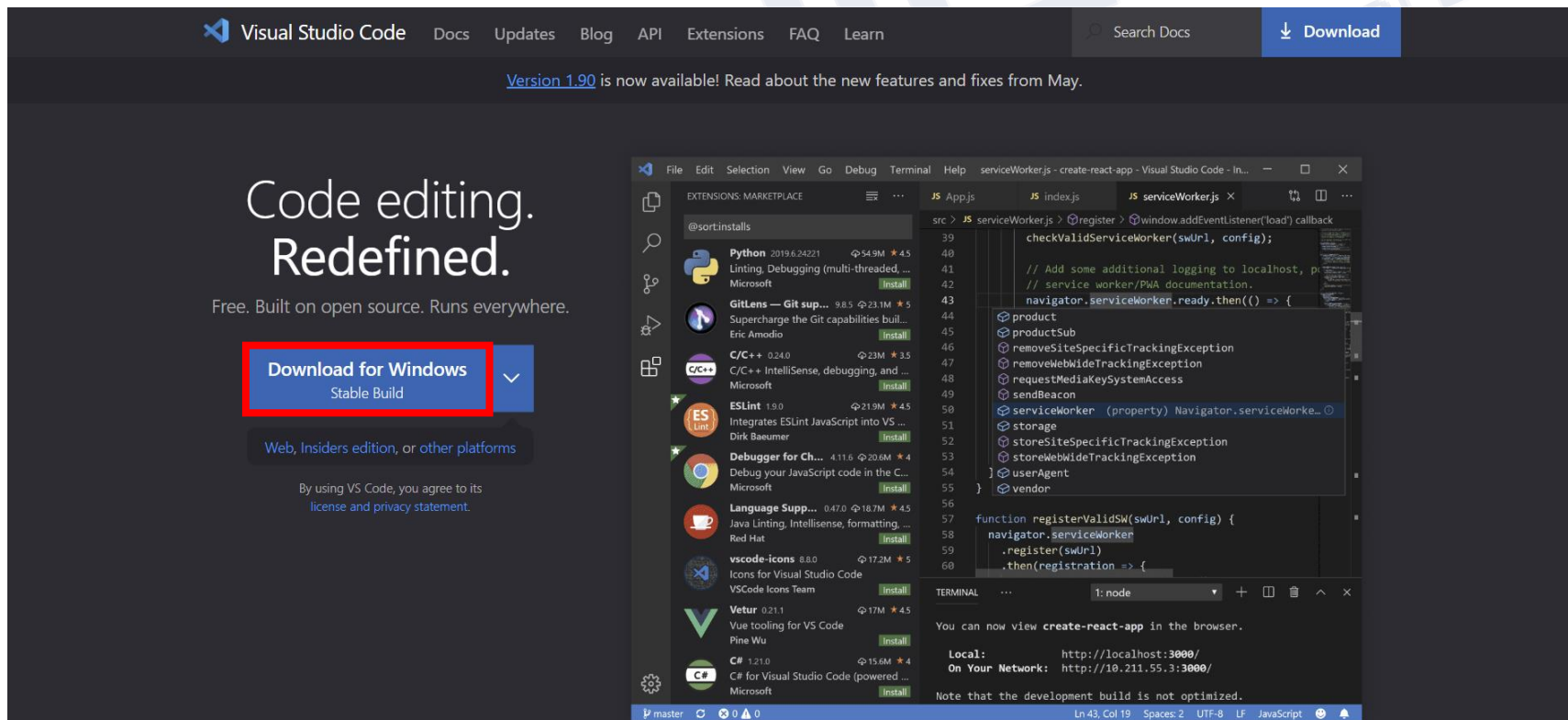


Python

2) 개발 환경 설정

■ 개발 환경(IDE) 설치 : Visual Studio Code (VS Code) [1/2]

- VS Code download link : <https://code.visualstudio.com/>
- Download link에서 **Download for Windows**를 클릭하여 설치 파일을 다운로드

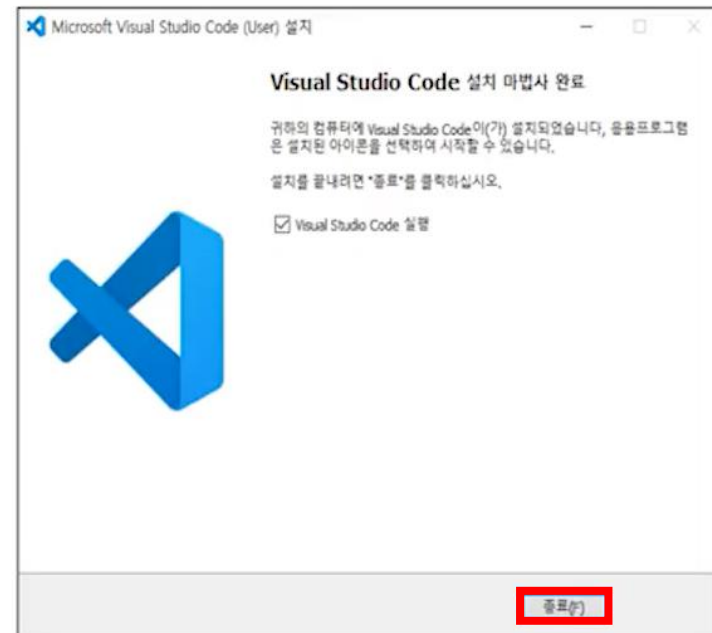
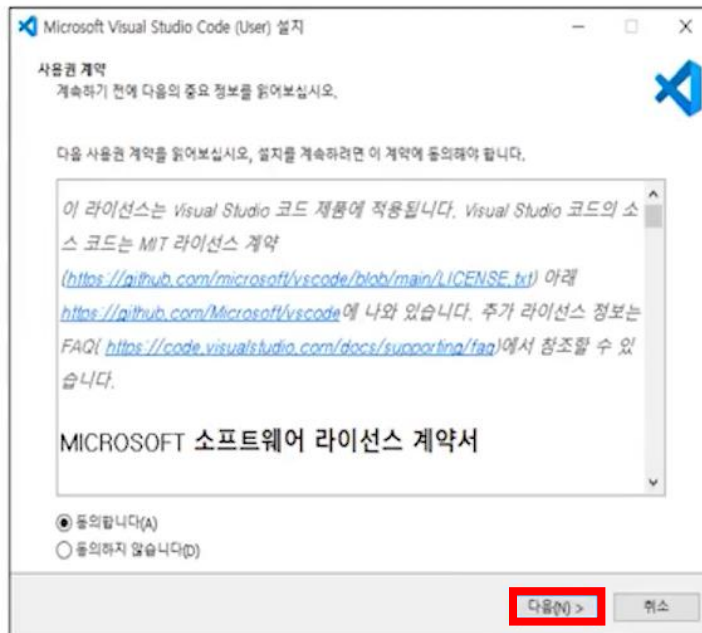


Python

2) 개발 환경 설정

■ 개발 환경(IDE) 설치 : Visual Studio Code (VS Code) [2/2]


- VS Code 설치 파일(VSCodeUserSetup-x64-1.xx.x.exe)을 실행
- 사용권 계약의 “동의합니다” 선택 후, 다음 버튼을 클릭하여 설치를 진행
- 설치 파일에서 제공하는 기본 설정으로 설치를 완료

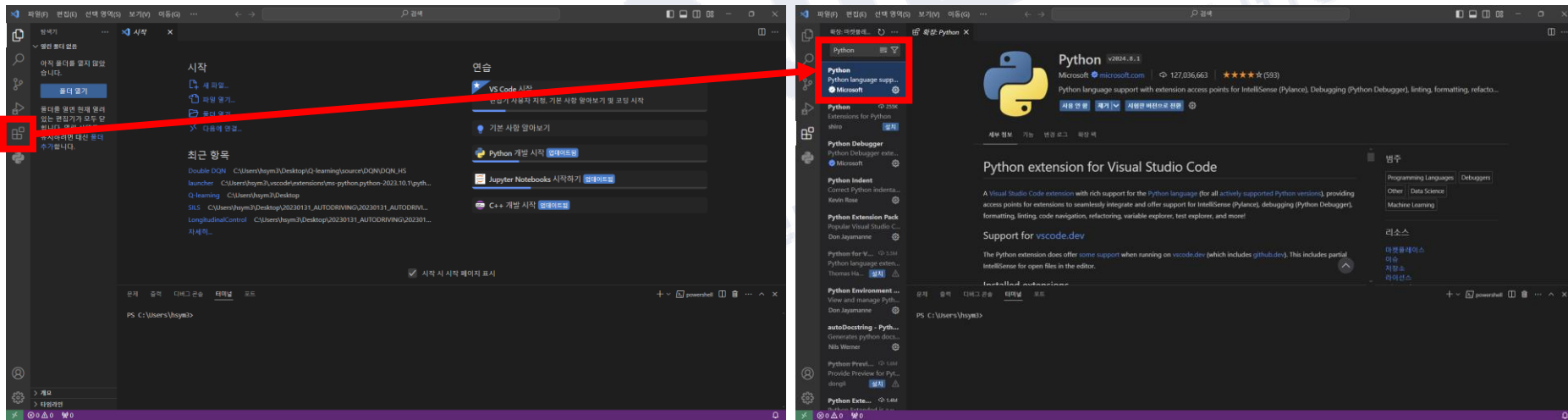


Python

2) 개발 환경 설정

■ VS Code 설정 : Python 기능 설치

- VS Code를 실행한 뒤, 왼쪽 아이콘에서 Extensions  를 클릭
- Extensions 창에서 Python을 검색 후, “install”을 클릭하여 VS Code의 Python 기능을 설치



- VS Code 개발 환경 내에 Python 언어를 인지할 수 있는 기능을 추가하는 과정
- 실제로 Python 언어를 설치한 것은 아님에 유의

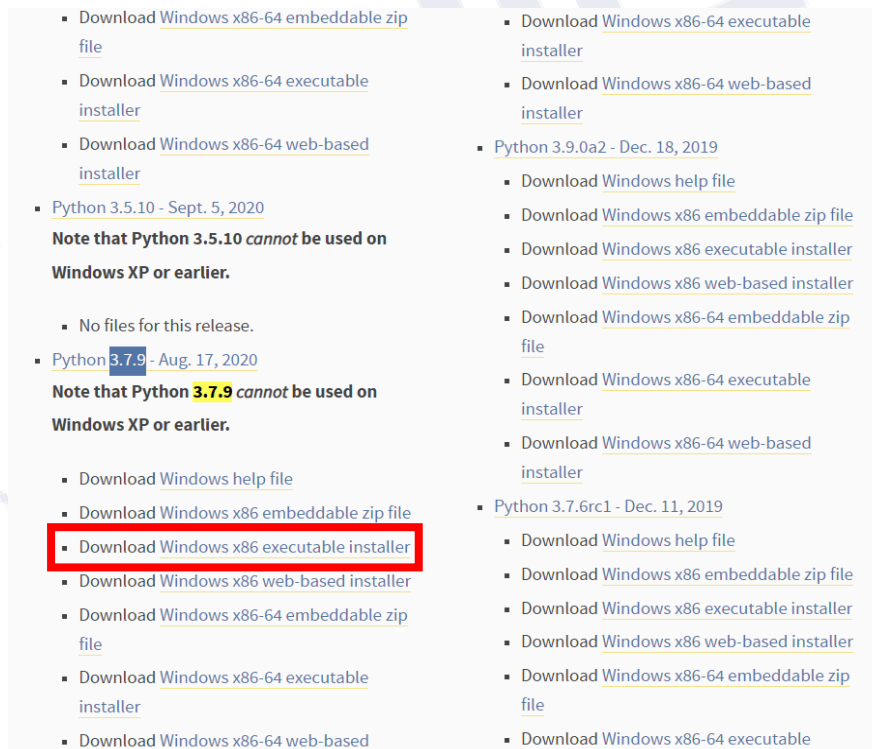
Python

2) 개발 환경 설정

▪ Python 설치 [1/3]

- Python download link : <https://www.python.org/downloads/windows/>

- Download link에서 파이썬 3.7.9(버전 무관) - Windows x86-64 executable installer를 다운로드



The screenshot shows the Python.org download page for Windows. It lists download links for various Python versions and architectures. The link for 'Download Windows x86 executable installer' under the Python 3.7.9 section is highlighted with a red box.

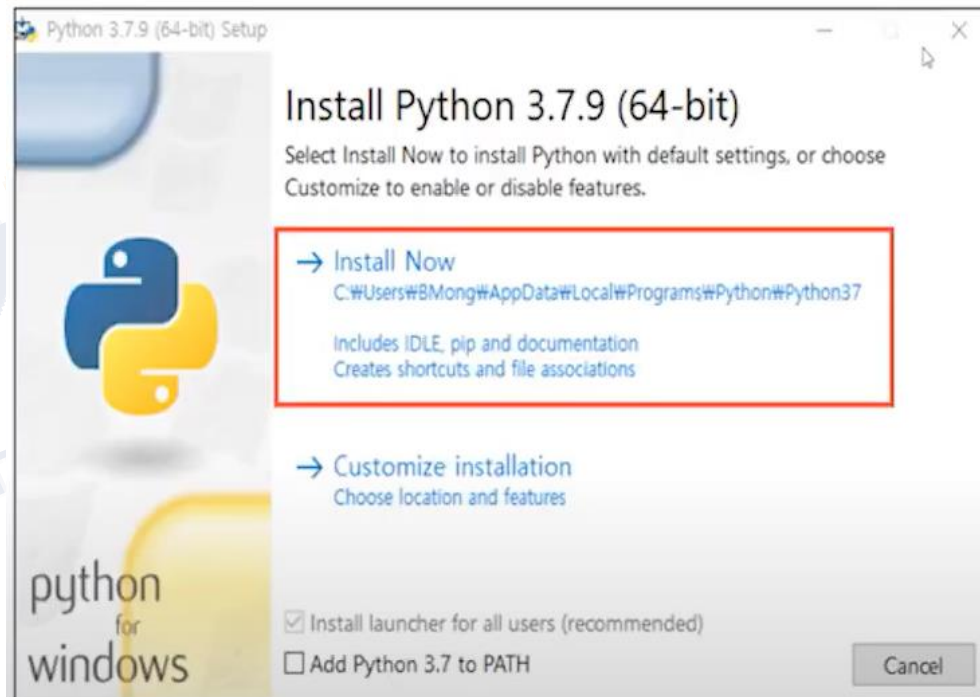
- Download [Windows x86-64 embeddable zip file](#)
- Download [Windows x86-64 executable installer](#)
- Download [Windows x86-64 web-based installer](#)
- Python 3.5.10 - Sept. 5, 2020
 - Note that Python 3.5.10 cannot be used on Windows XP or earlier.**
 - No files for this release.
- Python 3.7.9 - Aug. 17, 2020
 - Note that Python 3.7.9 cannot be used on Windows XP or earlier.**
 - Download [Windows help file](#)
 - Download [Windows x86 embeddable zip file](#)
 - Download [Windows x86 executable installer](#)**
 - Download [Windows x86 web-based installer](#)
 - Download [Windows x86-64 embeddable zip file](#)
 - Download [Windows x86-64 executable installer](#)
 - Download [Windows x86-64 web-based](#)
- Download [Windows x86-64 executable installer](#)
- Download [Windows x86-64 web-based installer](#)
- Python 3.9.0a2 - Dec. 18, 2019
 - Download [Windows help file](#)
 - Download [Windows x86 embeddable zip file](#)
 - Download [Windows x86 executable installer](#)
 - Download [Windows x86 web-based installer](#)
 - Download [Windows x86-64 embeddable zip file](#)
 - Download [Windows x86-64 executable installer](#)
 - Download [Windows x86-64 web-based installer](#)
- Python 3.7.6rc1 - Dec. 11, 2019
 - Download [Windows help file](#)
 - Download [Windows x86 embeddable zip file](#)
 - Download [Windows x86 executable installer](#)
 - Download [Windows x86 web-based installer](#)
 - Download [Windows x86-64 embeddable zip file](#)
 - Download [Windows x86-64 executable](#)

Python

2) 개발 환경 설정

▪ Python 설치 [2/3]

- 설치 파일을 실행하여 아래 그림과 같이 **Install launcher for all users (recommended)**를 체크
- **Install Now**를 클릭하여 설치를 진행

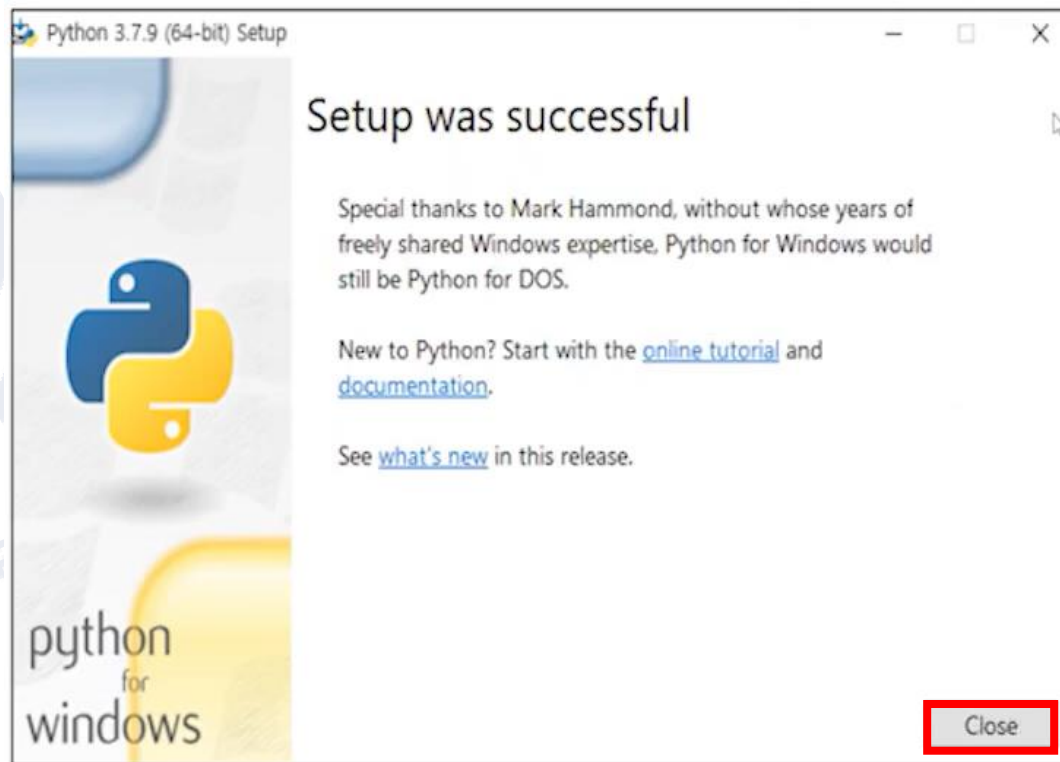


Python

2) 개발 환경 설정

- Python 설치 [3/3]


- 아래 그림과 같은 화면이 나오면 Python 설치 완료

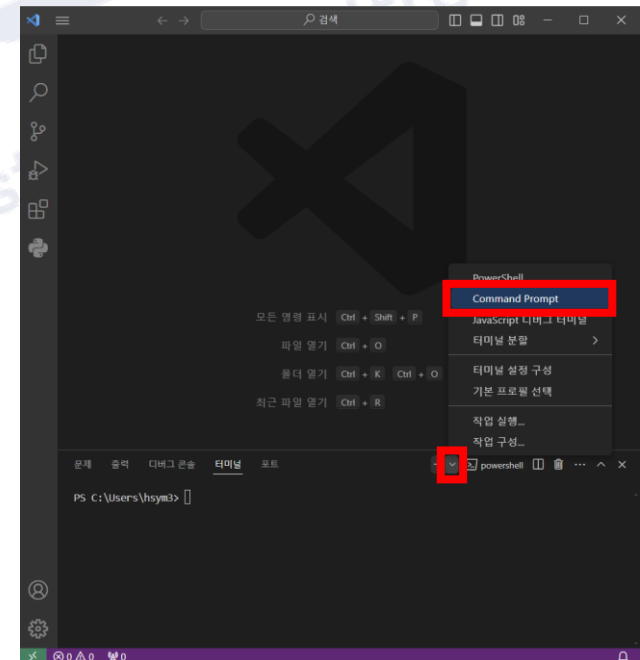
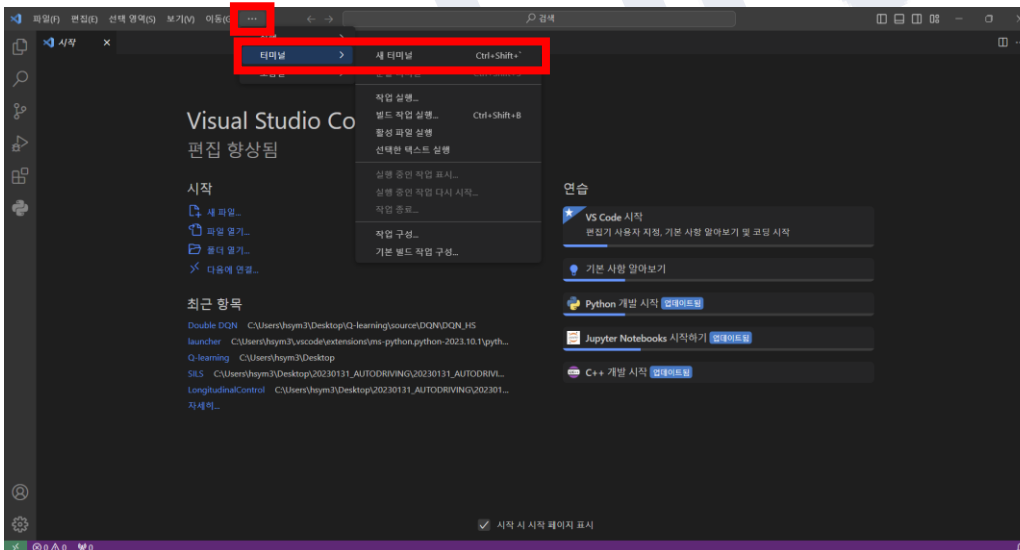


Python

2) 개발 환경 설정

Visual Studio Code 설정 [1/2]

- Python 확인 및 패키지 설치를 위하여 VS Code에서 **Terminal**을 실행 (단축키 : Ctrl + Shift + `)
- 생성된 Terminal에서  토글을 클릭하여 **명령 프롬프트(Command Prompt)**를 선택

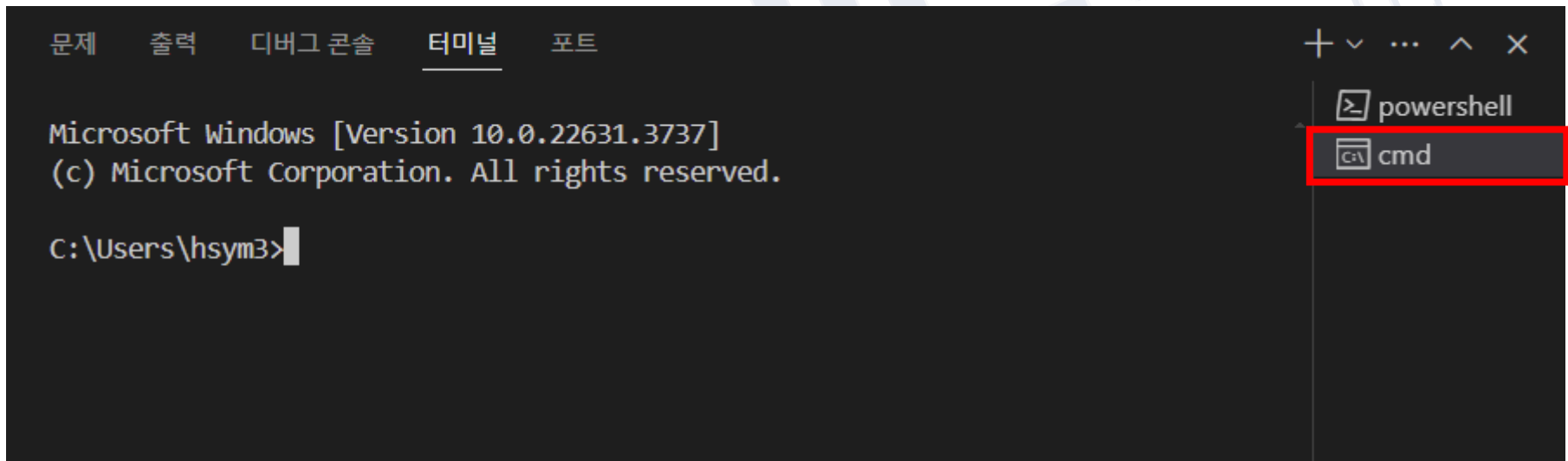


Python

2) 개발 환경 설정

- Visual Studio Code 설정 [2/2]

- Window에서 사용하는 cmd(command)창을 동일하게 사용 가능



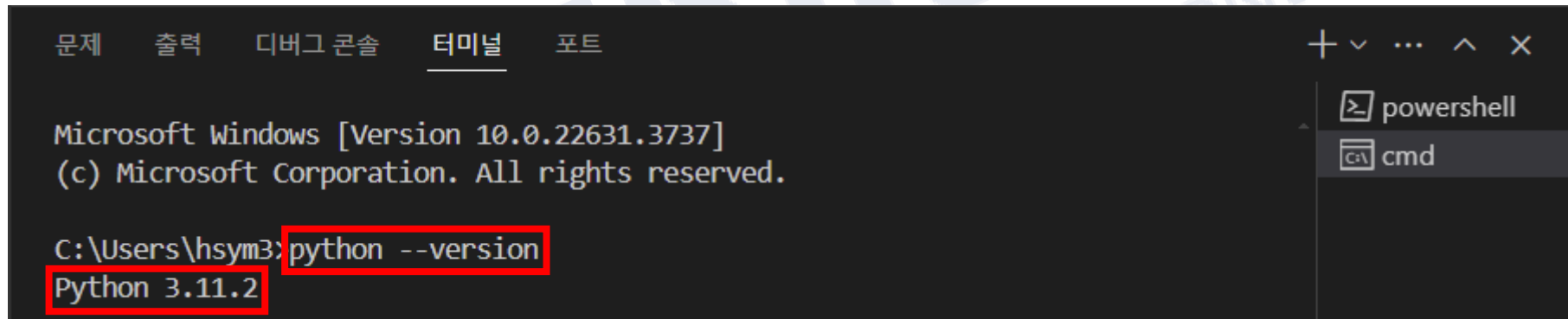
Python

2) 개발 환경 설정

- Python 설치 확인

- VS Code의 명령 프롬프트에서 아래 command를 입력하여 Python 설치를 확인

\$ python --version



The screenshot shows a Windows terminal window with the following content:

```
문제  출력  디버그 콘솔  터미널  포트

Microsoft Windows [Version 10.0.22631.3737]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hsym3>python --version
Python 3.11.2
```

The command prompt and the output 'Python 3.11.2' are highlighted with red boxes. On the right side of the terminal window, there is a taskbar with 'powershell' and 'cmd' icons.

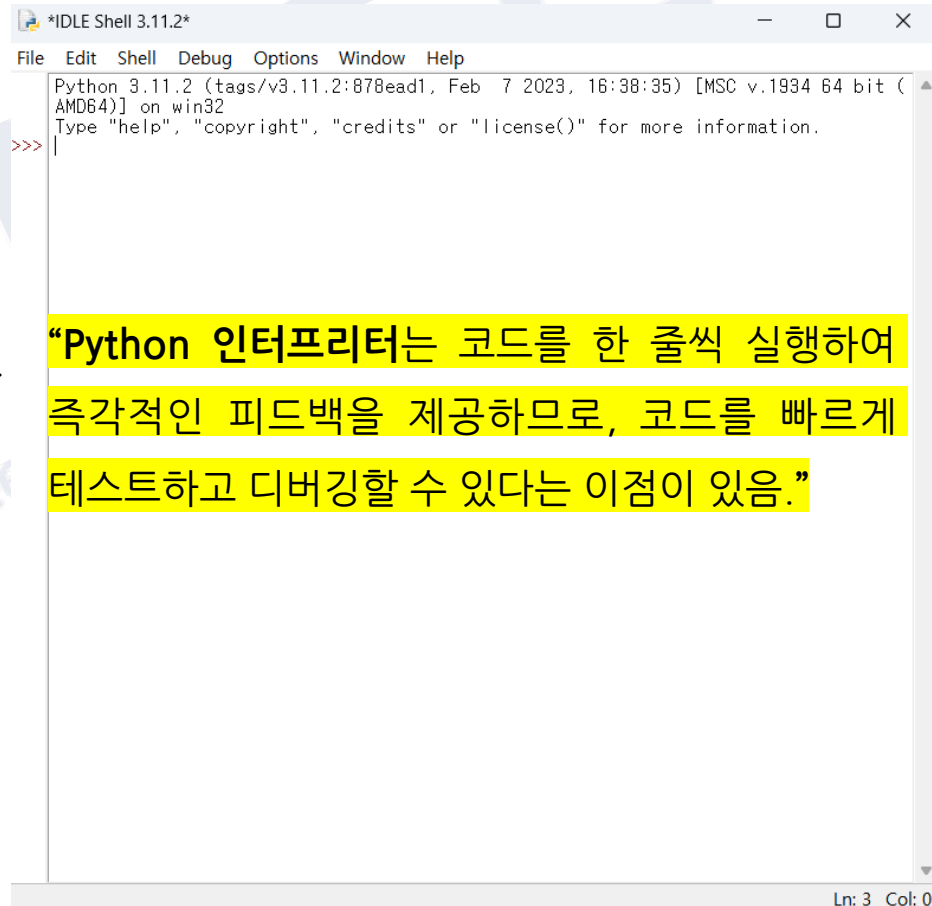
- Python 3.7.9가 확인되면 올바르게 설치된 것

Python

3) 기초 문법 - Python 인터프리터 사용법 (1)

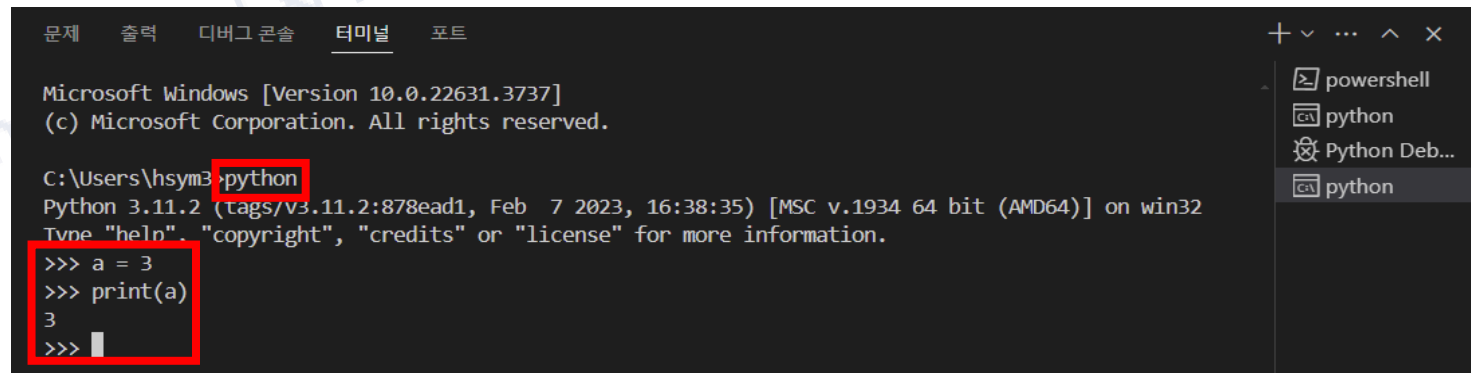
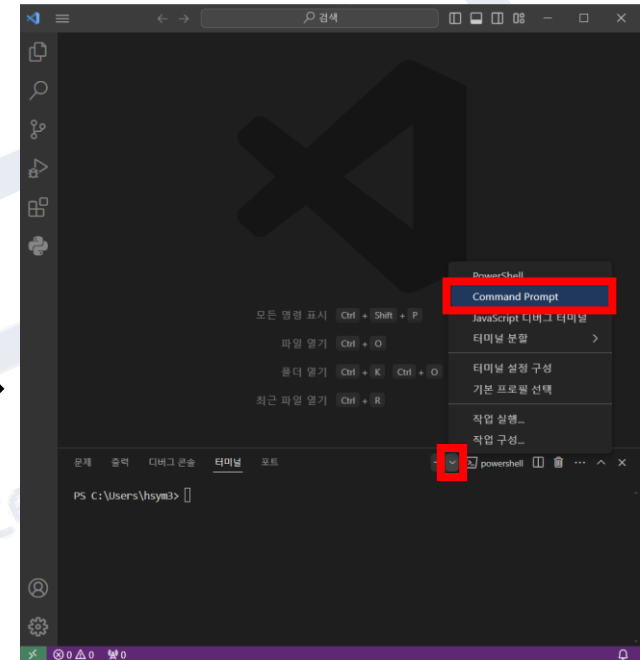
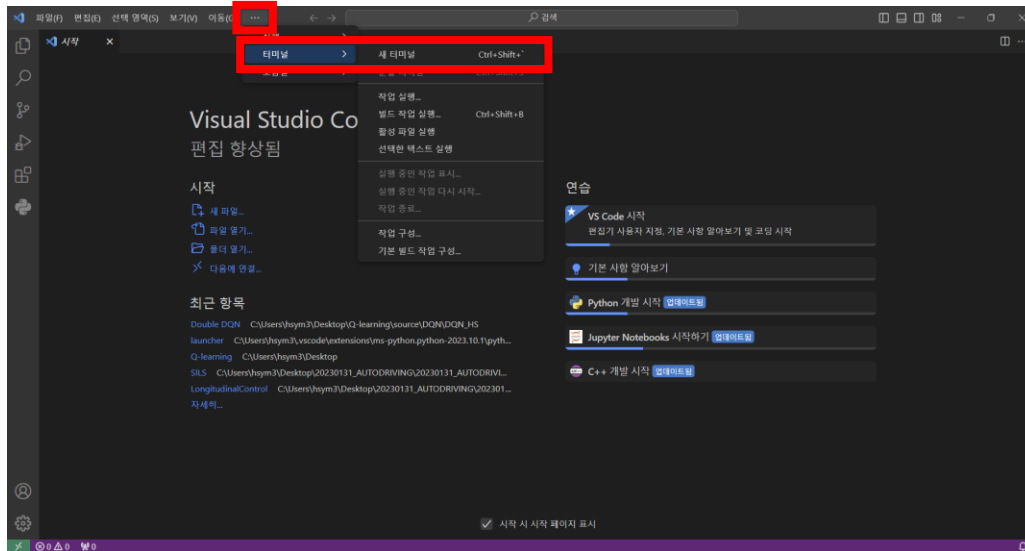


- 윈도우 왼쪽 아래 검색 명령창에 “IDLE”을 검색한 뒤,
왼쪽 그림과 같이 검색된 Python 앱을 클릭하여 실행.



Python

3) 기초 문법 - Python 인터프리터 사용법 (2)

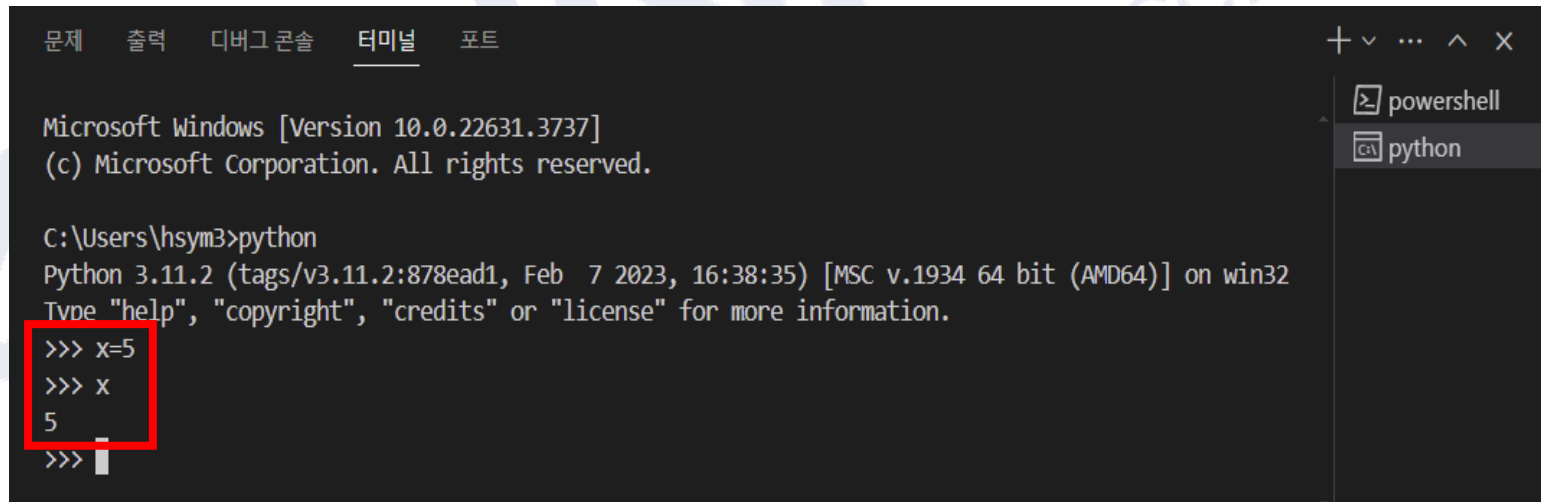


Python

3) 기초 문법 - 변수

- 변수: 변수는 데이터를 저장하는 컨테이너로서, 프로그램 내에서 데이터를 다루기 위해 사용됨.

C 또는 C++ 언어와는 달리 Python에서는 변수를 선언할 때 특별한 명령어나 자료형을 지정할 필요가 없으며, 변수에 값을 할당하는 순간 해당 변수가 생성됨.



```
Microsoft Windows [Version 10.0.22631.3737]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hsym3>python
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> x=5
>>> x
5
>>>
```

Python

3) 기초 문법 - 자료형

- 자료형(타입, Type) : 프로그래밍 언어에서 변수 또는 값이 가질 수 있는 데이터의 종류를 의미
- 자료형 → 정수형(Integer), 부동 소수점 수(Float), 문자열(String), 불리언(Boolean), 리스트(List), 튜플(Tuple), 딕셔너리(Dictionary) 등의 다양한 자료형이 존재

```
문제   출력   디버그 콘솔   터미널   포트
^
(Ctrl+Shift+M)
Microsoft Windows [Version 10.0.22631.3737]
(c) Microsoft Corporation. All rights reserved.

C:\Users\hsym3>python
Python 3.11.2 (tags/v3.11.2:878ead1, Feb  7 2023, 16:38:35) [MSC v.1934 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license" for more information.
>>> a=3
>>> type(a)
<class 'int'>
>>> a=[1,2,3]
>>> type(a)
<class 'list'>
>>> a=3.14159
>>> type(a)
<class 'float'>
>>> █
```

Python

3) 기초 문법 - 변수와 자료형

- 정수형(Integer)
 - ex) $x = 10$ → 정수 10을 변수 x에 할당.
- 부동 소수점 수(Float)
 - ex) $y = 3.14$ → 실수 3.14를 변수 y에 할당.
- 문자열(String)
 - ex) `name = "Alice"` → 작은 따옴표(' ') 또는 큰 따옴표(" ")를 통해 문자열 구성.
- 불리언(Boolean)
 - 조건문에서 주로 사용되며, 'True' 또는 'False'만을 가지는 자료형.

→ 리스트(List), 튜플(Tuple), 딕셔너리(Dictionary)은 뒤에서 설명 예정.

Python

3) 기초 문법 - 변수와 자료형 예시

- 정수형(Integer) 예시

```
>>> a=3
>>> type(a)
<class 'int'>
>>>
```

- 부동 소수점 수(Float) 예시

```
>>> a=3.14159
>>> type(a)
<class 'float'>
>>>
```

- 문자열(String) 예시

```
>>> a='my name is "Hwasu Lee"'
>>> type(a)
<class 'str'>
>>>
```

- 불리언(Boolean) 예시

```
>>> a="Hwasu"
>>> if a=="Hwasu": print("True")
...
True
>>>
```

Python

3) 기초 문법 - 기본 연산자

1. 산술 연산자

- ``+`` : 덧셈
- ``-`` : 뺄셈
- ``*`` : 곱셈
- ``/`` : 나눗셈 (결과는 항상 부동 소수점 수)
- ``//`` : 정수 나눗셈 (결과는 정수, 소수점 이하 버림)
- ``%`` : 나머지
- ``**`` : 거듭제곱

2. 비교 연산자

- ``==`` : 같다
- ``!=`` : 같지 않다
- ``>`` : 크다
- ``<`` : 작다
- ``>=`` : 크거나 같다
- ``<=`` : 작거나 같다

3. 논리 연산자

- ``and`` : 논리적 AND
- ``or`` : 논리적 OR
- ``not`` : 논리적 NOT

4. 할당 연산자

- ``=`` : 값을 변수에 할당
- ``+=`` : 변수에 값을 더한 후 결과를 다시 변수에 할당
- ``-=`` : 변수에서 값을 뺀 후 결과를 다시 변수에 할당
- ``*=`` : 변수에 값을 곱한 후 결과를 다시 변수에 할당
- ``/=`` : 변수를 값으로 나눈 후 결과를 다시 변수에 할당
- ``%=`` : 변수를 값으로 나눈 나머지를 다시 변수에 할당
- ``//=`` : 변수를 값으로 나눈 몫을 다시 변수에 할당
- ``**=`` : 변수를 값으로 거듭제곱한 결과를 다시 변수에 할당

Python

3) 기초 문법 - 주석

■ 주석

- 주석은 코드 내에서 해설이나 설명을 제공하는 부분으로, 코드 실행 시 무시됨.
- 주석은 ‘#’ 기호 뒤에 작성되며, 코드의 가독성을 높이고 다른 개발자들이 코드를 이해하는 데 도움을 줌.
- 주석을 여러 줄 작성할 경우, 보통 문자열(““ 주석 ”” or “ 주석 ””)을 사용하여 작성되지만 일반적으로는 단일 줄 주석(#)이 주로 사용됨.

```

19 class AStarPlanner:
51     def planning(self, sx, sy, gx, gy):
86         # show graph
87         if snow_animation: # pragma: no cover
88             plt.plot(self.calc_grid_position(current.x, self.min_x),
89                     self.calc_grid_position(current.y, self.min_y),
90                     # for stopping simulation with the esc key.
91                     plt.gcf().canvas.mpl_connect('key_release_event',
92                                                  lambda event: [exit(
93                                                      0) if event.key == 'escape'
94                                                      ]))
95             if len(closed_set.keys()) % 10 == 0:
96                 plt.pause(0.001)
97
98             if current.x == goal_node.x and current.y == goal_node.y:
99                 print("Find goal")
100                 goal_node.parent_index = current.parent_index
101                 goal_node.cost = current.cost
102                 break
103
104         # Remove the item from the open set
105         del open_set[c_id]
  
```

```

C: > Users > hsym3 > Desktop > SLAM & Navigation source code > PythonRobotics-master > PythonRobotics-
19 class AStarPlanner:
51     def planning(self, sx, sy, gx, gy):
52         """
53         A star path search
54
55         input:
56             s_x: start x position [m]
57             s_y: start y position [m]
58             gx: goal x position [m]
59             gy: goal y position [m]
60
61         output:
62             rx: x position list of the final path
63             ry: y position list of the final path
64         """
65
66         start_node = self.Node(self.calc_xy_index(sx, self.min_x),
67                               self.calc_xy_index(sy, self.min_y), 0.0, -1)
68         goal_node = self.Node(self.calc_xy_index(gx, self.min_x),
69                               self.calc_xy_index(gy, self.min_y), 0.0, -1)
70
  
```

Python

3) 기초 문법 - 입출력 함수

▪ input()

- input() 함수는 사용자로부터 입력을 받는 데 사용.
- 입력 받은 데이터는 항상 문자열 형태로 반환되며, 다른 데이터 타입이 필요한 경우 변환 과정 필요.

```
name = input("Enter your name: ")  
print("Hello, " + name + "!")
```

- name이라는 변수에 사용자로부터 문자열 타입의 데이터를 직접 입력 받아 출력하는 구조.

```
age = input("Enter your age: ")  
age = int(age) # 문자열을 정수로 변환  
print("You are", age, "years old.")
```

- age라는 변수에 사용자로부터 문자열 타입의 데이터를 직접 입력 받은 뒤,
int() 타입의 데이터 형태로 변환하여 출력하는 구조.

Python

3) 기초 문법 - 입출력 함수

▪ print()

- print() 함수는 화면에 출력을 표시하는 데 사용되는 함수.
- 기본적인 사용법은 매우 간단하나, 다양한 옵션을 통해 출력 형식을 보다 세밀하게 제어 가능.

```
print("Hello, World!")
```

- 가장 기본적인 형태의 print() 구조.
- 출력으로 "Hello, World!" 문자열을 화면에 출력.

```
print("Hello", "World", 123)  
# 출력: Hello World 123
```

- print() 함수는 여러 인수를 쉼표로 구분하여 출력이 가능하며, 각 인수는 공백으로 구분되어 출력됨.
- 출력으로 "Hello World 123" 문자열을 화면에 출력.

```
print("Hello", "World", sep=", ")  
# 출력: Hello, World
```

- print() 함수는 'sep'이라는 키워드 인수를 통해 여러 인수 사이에 출력할 문자열을 지정 가능.
- 출력으로 "Hello, World" 문자열을 화면에 출력.

Python

4) 제어 구조 - 반복문

- **반복문** : 프로그래밍에서 특정 코드 블록을 조건에 따라 여러 번 실행할 수 있게 하는 구조.
반복문을 통해 같은 작업을 반복해서 수행하는 코드를 간결하게 작성할 수 있으며,
데이터 처리, 자동화 작업, 반복적인 태스크 처리 등 다양한 곳에서 활용 가능.
- 대표적으로는 'for'문을 통한 반복문과, 'while'문을 통한 반복문이 있음.
- **'for' 반복문** : 'for' 반복문은 주어진 시퀀스(리스트, 튜플, 딕셔너리, 문자열 등)의 각 요소에
대해 코드 블록을 반복 실행하거나, 정해진 횟수만큼 반복하는 경우 사용.
- **'while' 반복문** : 'while' 반복문은 주어진 조건이 참(True)인 동안 반복해서 코드를 실행.

Python

4) 제어 구조 - 반복문

- 'for' 반복문 : 'for' 반복문은 주어진 시퀀스(리스트, 튜플, 딕셔너리, 문자열 등)의 각 요소에 대해 코드 블록을 반복 실행하거나, 정해진 횟수만큼 반복하는 경우 사용.

```
for 변수 in 시퀀스:  
    실행할 코드
```

들여쓰기 ('Tab 키' or 'Spacebar 키 4번')

- 예제)

```
fruits = ["apple", "banana", "cherry"]  
for fruit in fruits:  
    print(fruit)
```

```
# 0부터 4까지 출력  
for i in range(5):  
    print(i)
```

range(n) → 0 부터 n-1까지 반복

range(1, n+1) → 1부터 n까지 반복

Python

4) 제어 구조 - 반복문

- 'while' 반복문 : 'while' 반복문은 주어진 조건이 참(True)인 동안 반복해서 코드를 실행.

```
while 조건:  
    실행할 코드
```

- 예제)

```
x = 5  
while x > 0:  
    print(x)  
    x -= 1 # x 값을 1씩 감소
```

```
while True:  
    user_input = input("종료하려면 'exit'을 입력하세요: ")  
    if user_input.lower() == 'exit':  
        print("프로그램을 종료합니다.")  
        break  
    else:  
        print(f"입력한 메시지: {user_input}")
```

Python

4) 제어 구조 - 조건문

- **조건문** : 프로그래밍에서 **코드의 실행 흐름을 제어하기 위한 구조.**

조건문을 통해 특정 조건이 참(True)일 때와 거짓(False)일 때, 다른 코드를 실행 가능.

조건문은 프로그램의 논리적 흐름을 제어하는 데 필수적.

- 대표적으로 'if', 'elif', 'else'문을 통해 조건문을 구성하며, 하나의 조건문을 형성할 때 'if', 'elif', 'else'문을 모두 사용할 필요는 없음.

이러한 조건문은 크게 '**조건식**'과 '**블록**'으로 구성됨

- **조건식** : 조건식은 참(True) 또는 거짓(False)으로 평가될 수 있는 표현식을 의미.

비교 연산자('==', '!=', '>', '<', '>=', '<=')나 논리 연산자('and', 'or', 'not')를 사용하여 작성.

- **블록** : 조건이 참(True)일 때 실행될 코드를 의미하며, 들여쓰기를 사용하여 블록을 구분.

Python

4) 제어 구조 - 조건문

- 단일 if문

```
if 조건식:  
    실행할 코드
```

```
x = 10  
if x > 5:  
    print("x is greater than 5")
```

- if-elif-else문

```
if 조건식1:  
    실행할 코드 (조건식1이 참일 때)  
elif 조건식2:  
    실행할 코드 (조건식2가 참일 때)  
else:  
    실행할 코드 (모든 조건이 거짓일 때)
```

```
x = 7  
if x > 10:  
    print("x is greater than 10")  
elif x > 5:  
    print("x is greater than 5 but less than or equal to 10")  
else:  
    print("x is 5 or less")
```

- if-else문

```
if 조건식:  
    실행할 코드 (조건이 참일 때)  
else:  
    실행할 코드 (조건이 거짓일 때)
```

```
x = 3  
if x > 5:  
    print("x is greater than 5")  
else:  
    print("x is 5 or less")
```

Python

실 습

- 반복문(for 반복문 / while 반복문) 및 조건문(if / elif / else) 실습

→ 첫 번째 실습 : “ for 반복문과 input 함수를 통해 1부터 입력 받은 정수까지 모든 수의 합을 구하는 코드 작성 ”

(이 때, 입력 받은 수가 자연수가 아니라면, “ Wrong Number ”를 출력하도록 코드 구성)

→ 두 번째 실습 : “ while 반복문과 input 함수를 통해 1부터 입력 받은 정수까지 모든 수의 곱을 구하는 코드 작성 ”

(이 때, 입력 받은 수가 자연수가 아니라면, “ Wrong Number ”를 출력하도록 코드 구성)

→ 세 번째 실습 : “ 조건문과 input 함수를 통해 입력으로 받은 자연수가 홀수인지 짝수인지 구하는 코드 작성 ”

(이 때, 입력 받은 수가 자연수가 아니라면, “ Wrong Number ”를 출력하도록 코드 구성)

Python

실 습

→ 첫 번째 실습 : “for 반복문과 input 함수를 통해 1부터 입력 받은 정수까지 모든 수의 합을 구하는 코드 작성 ”

```
n = int(input("Enter a positive integer: "))
if n < 1:
    print("Wrong Number")
else:
    total = 0
    for i in range(1, n + 1):
        total += i
    print(f"The sum of numbers from 1 to {n} is: {total}")
```

Python

실 습

→ 두 번째 실습 : “ while 반복문과 input 함수를 통해 1부터 입력 받은 정수까지 모든 수의 곱을 구하는 코드 작성 ”

```
n = int(input("Enter a positive integer: "))
if n < 1:
    print("Wrong Number")
else:
    product = 1
    i = 1
    while i <= n:
        product *= i
        i += 1
    print(f"The product of numbers from 1 to {n} is: {product}")
```


Python

실습

→ 세 번째 실습 : “ 조건문과 input 함수를 통해 입력으로 받은 자연수가 홀수인지 짝수인지 구하는 코드 작성 ”

```
n = int(input("Enter a natural number: "))
if n <= 0:
    print("Wrong Number")
elif n % 2 == 0:
    print(f"{n} is an even number.")
else:
    print(f"{n} is an odd number.")
```

Python

5) 자료 구조 (리스트 / 튜플 / 딕셔너리)

▪ 리스트(List)

- 리스트는 순서가 있는 시퀀스로, 다양한 데이터 타입의 요소를 포함할 수 있음
- 요소(원소)의 추가(append), 삭제(pop 또는 remove), 수정이 자유로움
- 대괄호 '[]' 또는 'list()'를 통해 생성하며, 인덱싱과 슬라이싱으로 요소에 접근 가능
- 인덱싱은 특정 리스트(ex. arr = [1, 2, 3])를 구성하는 원소들의 인덱스를 통해 접근하는 방식

이 때, 리스트의 첫 번째 원소가 가지는 인덱스는 1이 아닌 "0"임을 유의!!

ex) arr[0] = 1, arr[1] = 2, arr[2] = 3

- 슬라이싱은 인덱싱과 유사하게 리스트의 인덱스를 통해 접근하는 방식.

ex) new_arr = arr[0:2] → print(new_arr) → [1, 2]

ex) new_arr = arr[:3] → print(new_arr) → [1, 2, 3]

Python

5) 자료 구조 (리스트 / 튜플 / 딕셔너리)

리스트(List) 주요 내장 함수

- `append(x)` : 리스트의 끝에 항목 'x'를 추가.
- `insert(i, x)` : 리스트의 i번째 인덱스에 항목 'x'를 추가.
- `pop(i)` : 리스트의 i번째 인덱스에 항목을 제거하고, 그 값을 반환.
만약 i가 지정되지 않으면, 리스트의 마지막 항목을 제거하고 반환.
- `remove(x)` : 리스트에서 값 'x'를 찾아 제거.
- `index(x)` : 리스트에서 값 'x'가 위치한 인덱스를 반환.
- `sort(key=None, reverse=False)` : 리스트를 오름차순으로 정렬.
- `reverse()` : 리스트의 항목 순서를 거꾸로 뒤집음.
- `clear()` : 리스트의 모든 항목을 제거.

Python

5) 자료 구조 (리스트 / 튜플 / 딕셔너리)

리스트(List) 예제 코드

```
>>> arr = []  
>>> type(arr)  
<class 'list'>  
>>> arr.append('Hwasu')  
>>> arr.append('Munkyu')  
>>> print(arr)  
['Hwasu', 'Munkyu']  
>>>
```

```
>>> print(arr[0])  
Hwasu  
>>> print(arr[1])  
Munkyu  
>>> arr[0]='Chang Mook'  
>>> print(arr)  
['Chang Mook', 'Munkyu']  
>>> name = arr.pop()  
>>> print(name)  
Munkyu  
>>> print(arr)  
['Chang Mook']  
>>>
```

```
>>> arr = [1,2,3,4,5,6,7,8,9]  
>>> print(arr)  
[1, 2, 3, 4, 5, 6, 7, 8, 9]  
>>> arr2 = arr[:6]  
>>> print(arr2)  
[1, 2, 3, 4, 5, 6]  
>>> arr3 = arr[2:8]  
>>> print(arr3)  
[3, 4, 5, 6, 7, 8]  
>>>
```

arr = ['Hwasu', 'Munkyu']
0번 인덱스 1번 인덱스

Python에서 인덱스(index)는
0번부터 시작!

arr = [1, 2, 3, 4, 5, 6, 7]
arr[:7] = ??

슬라이싱을 통해 arr 리스트의
0번 인덱스 값부터 6번 인덱스
값까지 반환!
ex) [1, 2, 3, 4, 5, 6, 7]

Python

5) 자료 구조 (리스트 / 튜플 / 딕셔너리)

- 리스트 컴프리헨션(List Comprehension)
 - 리스트 컴프리헨션은 파이썬에서 리스트를 생성하는 간결하고 효율적인 방법.
 - 전통적인 for문을 통한 반복문을 사용하는 것보다 더 짧고 가독성 좋은 코드를 작성 가능.
 - 리스트 컴프리헨션을 사용하면 기존의 리스트나 다른 반복 가능한(iterable) 객체를 기반으로 새로운 리스트를 만들 수 있음.

```
[expression for item in iterable if condition]
```

- 'expression'은 각 항목에 대해 실행할 표현식을 의미.
- 'item'은 반복 과정에서 사용되는 변수를 의미.
- 'iterable'은 반복 가능한 객체(예: 리스트, 문자열, range 등)를 의미.
- 'condition'은 선택 사항으로, 각 항목에 대해 참(True)인 경우에만 expression을 적용.

Python

5) 자료 구조 (리스트 / 튜플 / 딕셔너리)

- 리스트 컴프리헨션(List Comprehension) 예제

```
numbers = [1, 2, 3, 4, 5]
doubled = [x * 2 for x in numbers]
print(doubled) # 출력: [2, 4, 6, 8, 10]
```

- 기존 리스트의 각 항목에 2를 곱하여 새로운 리스트 생성

```
numbers = [1, 2, 3, 4, 5, 6]
even_numbers = [x for x in numbers if x % 2 == 0]
print(even_numbers) # 출력: [2, 4, 6]
```

- 조건문을 사용하여 짝수만 필터링

```
numbers = [1, 2, 3, 4, 5]
result = [x * 2 if x % 2 == 0 else x * 3 for x in numbers]
print(result) # 출력: [3, 4, 9, 8, 15]
```

- 조건식에 if-else를 사용하여 값을 변환 (값이 짝수인 경우 2배, 아닌 경우 3배)

- 리스트 컴프리헨션을 사용하면
코드가 간결하고 가독성 있게
구성되며, 보다 빠른 속도로
실행된다는 이점이 있음.

Python

5) 자료 구조 (리스트 / 튜플 / 딕셔너리)

▪ 튜플(Tuple)

- 튜플은 순서가 있는 시퀀스이지만, 한 번 생성된 후에는 수정 불가 (불변 자료형)
- 소괄호 '()' 또는 'tuple()'을 통해 생성하거나, 괄호 없이 요소를 나열하여 생성 가능
- 주로 데이터가 변경될 위험이 없어야 하는 상황에서 유용하며,
함수에서 여러 값을 한 번에 반환할 때 주로 사용됨

튜플(Tuple) 예제 코드

```
dimensions = (1920, 1080)
print(dimensions[0]) # 1920 출력
# dimensions[0] = 2560 # 에러 발생, 튜플은 수정할 수 없음
```

Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: 'tuple' object does not support item assignment
>>> █

Python

5) 자료 구조 (리스트 / 튜플 / 딕셔너리)

▪ 딕셔너리(Dictionary)

- 딕셔너리는 'key(키):value(값)' 쌍으로 데이터를 저장하는 자료 구조
- 중괄호 '{ }' 또는 'dict()'를 통해 생성하며, 'key'를 통해 빠르게 데이터에 접근 가능
- 딕셔너리의 각 'key' 값은 고유해야 하며, 중복 불가
- 리스트(List)와 동일하게 'key'와 'value' 값을 추가, 삭제, 수정할 수 있음

딕셔너리(Dictionary) 예제 코드

```
student_grades = {'Alice': 90, 'Bob': 85, 'Eve': 88}
student_grades['Alice'] = 95 # Alice의 점수를 수정
print(student_grades['Alice']) # 95 출력
```

key value



Python

실 습

▪ 자료구조(리스트 / 튜플 / 딕셔너리) 실습

→ 첫 번째 실습 : “ 리스트 내의 정수 중 가장 큰 수를 출력하는 코드 작성 ”

(단, 파이썬 내장 함수인 **max()** 함수 사용 불가)

→ 두 번째 실습 : “ 리스트 컴프리헨션과 `input()` 함수를 이용하여 입력 받은 수까지의 자연수 중,
3의 배수 또는 5의 배수인 숫자의 제곱을 리스트로 생성하는 코드 작성 ”

→ 세 번째 실습 : “ 크기가 서로 다른 자연수로 구성된 리스트를 오름차순으로 정렬하는 코드 작성 ”

(단, 파이썬 내장 함수인 **sort()** 함수 사용 불가)

Python

실 습

→ 첫 번째 실습 : “ 리스트 내의 정수 중 가장 큰 수를 출력하는 코드 작성 ”

→ 주의사항 : Python 내장 함수인 `max()` 함수를 사용하지 않고 코드를 작성

```
# 예제 리스트
example_list = [23, 1, 45, 34, 75, 54, 24]

if not example_list: # 리스트가 비어있는 경우 처리
    maximum_value = None
else:
    maximum_value = example_list[0] # 리스트의 첫 번째 요소를 최대값으로 초기 설정
    for number in example_list:
        if number > maximum_value:
            maximum_value = number

print("가장 큰 수는:", maximum_value)
```

Python

실 습

→ 두 번째 실습 : “리스트 컴프리헨션과 input() 함수를 이용하여 입력 받은 수까지의 자연수 중, 3의 배수 또는 5의 배수인 숫자의 제곱을 리스트로 생성하는 코드 작성”

```
# 사용자가 입력한 숫자까지의 범위에서 3의 배수 또는 5의 배수인 숫자의 제곱을 리스트로 생성
n = int(input("Enter a positive integer: "))

multiples_of_3_or_5_squares = [x**2 for x in range(1, n + 1) if x % 3 == 0 or x % 5 == 0]

print(multiples_of_3_or_5_squares)
```

Python

실 습

→ 세 번째 실습 : “ 크기가 서로 다른 자연수로 구성된 리스트를 오름차순으로 정렬하는 코드 작성 ”

→ 주의사항 : 파이썬 내장 함수인 `sort()` 함수를 사용하지 않고 코드를 작성

```
# 정렬할 리스트
numbers = [64, 25, 12, 22, 11]

# 선택 정렬 알고리즘
n = len(numbers)
for i in range(n):
    # 현재 위치에서 최소값의 인덱스를 찾음
    min_idx = i
    for j in range(i+1, n):
        if numbers[j] < numbers[min_idx]:
            min_idx = j
    # 현재 위치와 최소값의 위치를 교환
    numbers[i], numbers[min_idx] = numbers[min_idx], numbers[i]

print("Sorted list:", numbers)
```

Python

6) 함수

▪ 함수(Function)

- 코드를 조직화하고 재사용하기 위해 사용
- 복잡한 프로그램을 관리하기 쉽도록 부분으로 나눌 수 있으며, 코드의 중복을 줄이고 프로그램의 가독성을 향상시킬 수 있음
- 'def' 키워드를 통해 정의하며, return이 없는 경우 함수는 'None' 값을 반환
- 함수의 입력이 되는 매개변수는 반드시 있어야 하는 것은 아님

```
def 함수명(매개변수):  
    # 수행할 코드  
    return 반환값
```

함수 예제 코드

```
def greet(name):  
    return f"Hello, {name}!"  
  
message = greet("Alice")  
print(message) # Hello, Alice!
```

```
def describe_pet(animal_type, pet_name):  
    print(f"I have a {animal_type} and its name is {pet_name}.")  
  
describe_pet("hamster", "Harry")  
describe_pet(pet_name="Daisy", animal_type="dog")
```

Python

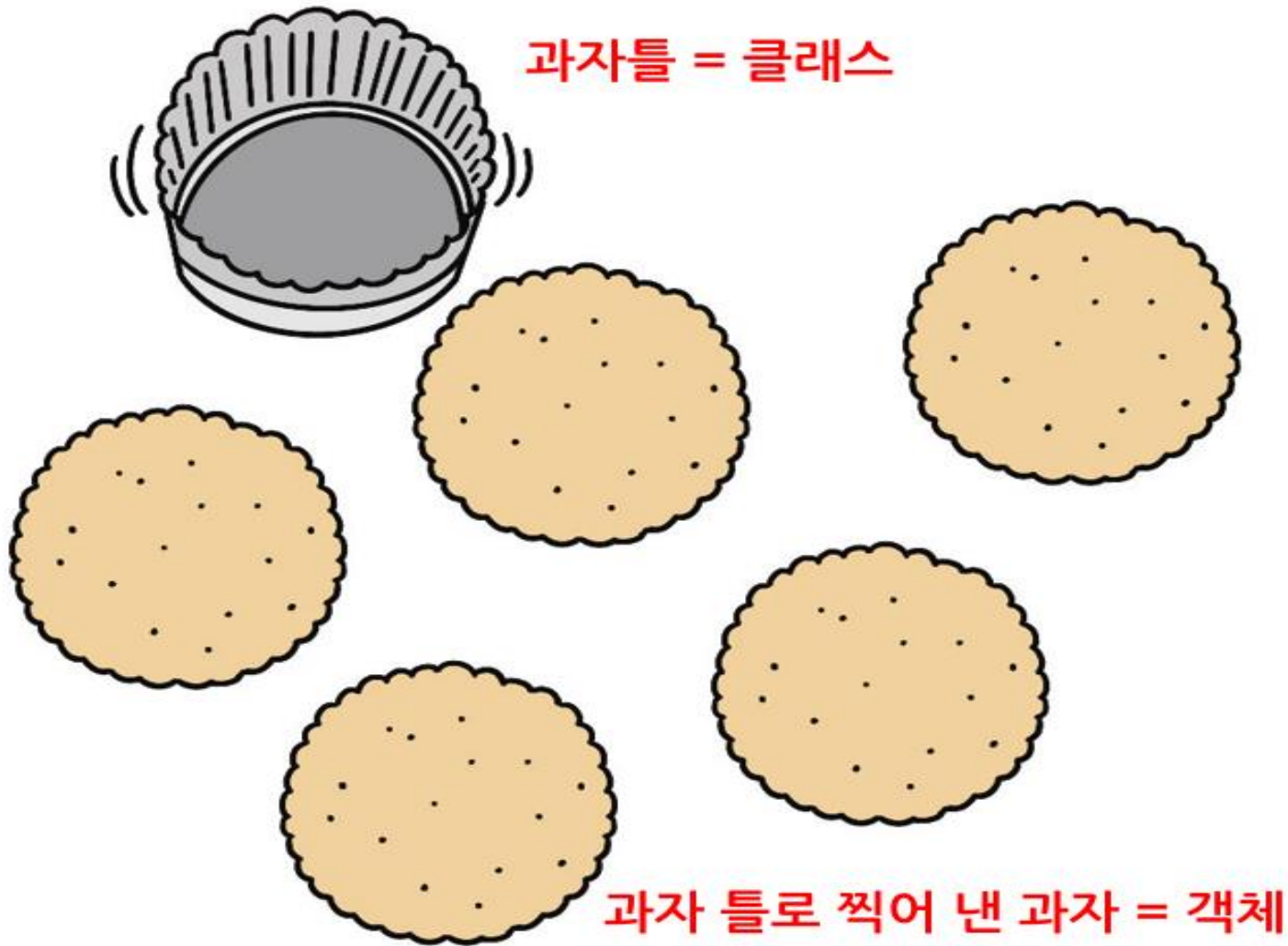
7) 클래스

- 클래스(Class)

- 클래스는 데이터와 함수를 하나로 묶어, 새로운 타입을 정의
- 데이터 구조를 만들고, 해당 구조에 관련된 특정 기능들을 조직화 및 모듈화 가능
- 클래스는 객체를 생성하는 “틀” 또는 “모델”과 같은 역할을 수행
- 클래스의 주요 구성 요소는 ‘속성(Attribute)’과 ‘메서드(Method)’로 구성됨
 - 속성(Attribute) : 클래스에 속한 데이터를 의미하며, 변수로 표현됨
 - 메서드(Method) : 클래스의 객체가 수행할 수 있는 함수를 의미

Python

7) 클래스



Python

7) 클래스

```
class Car:
```

```
    def __init__(self, make, model, year):
```

```
        self.make = make
```

```
        self.model = model
```

```
        self.year = year
```

} 인스턴스 속성

```
    def display_info(self):
```

```
        print(f"Car: {self.year} {self.make} {self.model}")
```

} 메서드

```
# 객체 생성 및 사용
```

```
my_car = Car("Toyota", "Corolla", 2021)
```

```
my_car.display_info()
```


Python

7) 클래스

- 클래스 - 상속(inheritance)

- 상속은 기존 클래스의 (인스턴스) 속성과 메서드를 물려받아 새로운 클래스를 생성하는 것을 의미
- 이를 통해 코드의 재사용성을 높이고, 코드의 관리를 용이하게 할 수 있음

→ 상속의 주요 이점 : 코드 중복 감소, 프로그램의 구조화와 확장성 증대, 유지보수가 용이

```
class Car:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year

    def display_info(self):
        print(f"Car: {self.year} {self.make} {self.model}")

# 객체 생성 및 사용
my_car = Car("Toyota", "Corolla", 2021)
my_car.display_info()
```

앞서 작성하였던 기존 클래스

Python

7) 클래스

- 클래스 - 상속(inheritance)

기존에 작성하였던 클래스
(부모 클래스)

부모 클래스를 상속 받는 클래스
(자식 클래스)

```
class Car:
    def __init__(self, make, model, year):
        self.make = make
        self.model = model
        self.year = year

    def display_info(self):
        print(f"Car: {self.year} {self.make} {self.model}")

class ElectricCar(Car): # Car 클래스를 상속받는 ElectricCar 클래스
    def __init__(self, make, model, year, battery_size):
        super().__init__(make, model, year) # 부모 클래스의 생성자 호출
        self.battery_size = battery_size

    def display_battery(self):
        print(f"This car has a {self.battery_size}-kWh battery.")

# 상속받은 클래스의 객체 생성 및 사용
my_tesla = ElectricCar("Tesla", "Model S", 2019, 75)
my_tesla.display_info()
my_tesla.display_battery()
```

Python

실 습

- 함수(Function) 및 클래스(Class) 실습

→ 첫 번째 실습 : “ 리스트를 입력으로 받아, 해당 리스트의 평균값을 출력으로 도출하는 함수 만들기 ”

→ 두 번째 실습 : “ 클래스를 통해 덧셈과 뺄셈이 가능한 계산기 만들기 ”

→ 세 번째 실습 : “ 상속(inheritance)을 통해 사칙연산이 가능한 계산기 만들기 ”

(반드시 두 번째 실습에서 작성한 클래스를 상속받을 것)

Python

실 습

- 첫 번째 실습 : “ 리스트를 입력으로 받아, 해당 리스트의 평균값을 출력으로 도출하는 함수 만들기 ”
- 주의사항 : Python 내장 함수인 `sum()` 함수를 사용하지 않고 코드를 작성

```
# 숫자 리스트의 평균을 계산하는 함수를 정의합니다.  
def calculate_average(numbers):  
    total = 0  
    count = 0  
    for number in numbers:  
        total += number  
        count += 1  
    if count == 0:  
        return 0  
    else:  
        return total / count  
  
# 숫자 리스트를 생성합니다.  
number_list = [10, 20, 30, 40, 50]  
  
# 함수를 호출하여 평균을 계산하고 출력합니다.  
average = calculate_average(number_list)  
print("The average of the list is:", average)
```

Python

실습

→ 두 번째 실습 : “ 덧셈과 뺄셈이 가능한 계산기 만들기 ”

```
class SimpleCalculator:
    def __init__(self):
        self.result = 0

    def add(self, a, b):
        """두 수를 더하고 결과를 반환합니다."""
        self.result = a + b
        return self.result

    def subtract(self, a, b):
        """첫 번째 수에서 두 번째 수를 빼고 결과를 반환합니다."""
        self.result = a - b
        return self.result

# 계산기 객체 생성
calc = SimpleCalculator()

# 덧셈 예시
print("10 + 5 =", calc.add(10, 5)) # 15

# 뺄셈 예시
print("10 - 5 =", calc.subtract(10, 5)) # 5
```

Python

실습

→ 세 번째 실습 : “ 상속(inheritance)을 통해 사칙연산이 가능한 계산기 만들기 ”

→ 주의사항 : 반드시 두 번째 실습에서 작성한 클래스를 상속받을 것

```
class SimpleCalculator:
    def __init__(self):
        self.result = 0

    def add(self, a, b):
        """두 수를 더하고 결과를 반환합니다."""
        self.result = a + b
        return self.result

    def subtract(self, a, b):
        """첫 번째 수에서 두 번째 수를 빼고 결과를 반환합니다."""
        self.result = a - b
        return self.result
```

SimpleCalculator 클래스를 상속받는 ExtendedCalculator 클래스 정의

```
class ExtendedCalculator(SimpleCalculator):
    def multiply(self, a, b):
        """두 수를 곱하고 결과를 반환합니다."""
        self.result = a * b
        return self.result
```

```
    def divide(self, a, b):
        """첫 번째 수를 두 번째 수로 나누고 결과를 반환합니다. 0으로 나누는 것을 방지합니다."""
        if b == 0:
            return "Error! Division by zero."
        else:
            self.result = a / b
            return self.result
```

확장된 계산기 객체 생성

```
calc = ExtendedCalculator()
```

사칙연산 예시

```
print("10 + 5 =", calc.add(10, 5))      # 15
print("10 - 5 =", calc.subtract(10, 5)) # 5
print("10 * 5 =", calc.multiply(10, 5)) # 50
print("10 / 5 =", calc.divide(10, 5))   # 2.0
print("10 / 0 =", calc.divide(10, 0))   # Error! Division by zero.
```

QnA

E-mail : cmin87394@gmail.com

Mobile : 010-3357-8739

Thank you.

