

✓ 1차시: 프로그래밍 기초 (1)

파이썬 기본 문법

✓ 1. print() 활용

- print() 함수로 화면에 메시지나 변수값을 출력합니다.

```
print("Hello, Python!")
print("나만의 문장을 출력해 보세요!")
```

```
🔄 Hello, Python!
    나만의 문장을 출력해 보세요!
```

실습 1:

print()를 사용해 본인의 이름과 좋아하는 과목을 출력해 보세요.

[+ 코드](#)[+ 텍스트](#)

코딩을 시작하거나 AI로 코드를 [생성](#)하세요.

✓ 2. 기본 연산

- 사칙연산: +, -, *, /
- 몫: //, 나머지: %, 거듭제곱: **

```
print("3 + 5 =", 3 + 5)
print("10 - 4 =", 10 - 4)
print("6 * 7 =", 6 * 7)
print("8 / 3 =", 8 / 3)
print("8 // 3 =", 8 // 3)
print("8 % 3 =", 8 % 3)
print("2 ** 5 =", 2 ** 5)
```

```
🔄 3 + 5 = 8
    10 - 4 = 6
    6 * 7 = 42
    8 / 3 = 2.6666666666666665
    8 // 3 = 2
    8 % 3 = 2
    2 ** 5 = 32
```

-실습 2:

15와 4의 나눗셈 결과(몫, 나머지)를 print()로 출력해 보세요.

코딩을 시작하거나 AI로 코드를 [생성](#)하세요.

```
🔄 3 3
```

✓ 3. 변수 사용

프로그래밍에서 변수는 데이터를 저장하고 참조하기 위해 사용되며, 변수 이름과 데이터 타입이 중요합니다

- 변수는 = 기호로 값을 저장합니다.
- 변수 이름은 문자, 숫자, 언더스코어(_) 사용 가능하며 숫자로 시작하면 안 됩니다.

파이썬에서 사용할 수 있는 자료형 중 많이 사용하는 것은 다음과 같다:

- bool: 참(True) 또는 거짓(False)
- int: 정수형 숫자 (예: 1, -3)
- float: 실수
- complex: 복소수 (예: 2+3j)
- str: 문자열 (예: 'hello', '3')
- tuple: 변경 불가능한 순서 있는 데이터 묶음 (예: (1,2,3))
- list: 변경 가능한 순서 있는 데이터 묶음 (예: [1,2,3])
- dict: 키-값(key-value) 쌍으로 구성된 데이터 구조 (예: {'a': 1})

type 명령을 통해 변수나 값의 자료형을 알아볼 수 있다.

```
# 예제: 사각형 넓이 계산
x = 10
y = 3
sum_xy = x + y
print("x + y =", sum_xy)
```

```
🔄 x + y = 13
```

```
# 예제: 사각형 넓이 계산
width = 5
height = 2
area = width * height
print("사각형의 넓이 =", area)
print(f"사각형의 넓이 = {area}")
```

```
🔄 사각형의 넓이 = 10
    사각형의 넓이 = 10
```

```
# 예제: 안녕지옥
text = "안녕"
print(text*10)
space = " "
print(text+space+text)
```

안녕안녕안녕안녕안녕안녕안녕안녕안녕안녕
안녕 안녕

```
# 문자와 숫자는 더할 수 없습니다
text = "안녕"
num = 5
print(text+num)
```

```
-----
TypeError                                Traceback (most recent call last)
<ipython-input-7-ab69e69d00a9> in <cell line: 0>()
      2 text = "안녕"
      3 num = 5
----> 4 print(text+num)

TypeError: can only concatenate str (not "int") to str
```

실습 3:

a = 7, b = 2일 때, a를 b로 나눈 몫과 나머지를 출력하는 코드를 작성해 보세요.

코딩을 시작하거나 시로 코드를 [생성](#)하세요.

3 1

4. 조건문

1. if 문

- 특정 조건이 참(True)일 때만 코드 블록을 실행하도록 하는 제어문입니다.

2. else 추가하기

- if 조건이 거짓(False)일 때 대체 실행 블록을 정의합니다.

3. elif 로 여러 가지 경우 나누기

- 여러 조건식을 차례로 검사할 때 사용합니다.

```
# 예제: 숫자가 10보다 큰지 확인하기
x = 12
if x > 10:
    print("x는 10보다 큼니다.") # 실행됨
```

x는 10보다 큼니다.

```
# 예제: 짝수/홀수 판별
n = 7
if n % 2 == 0:
    print("짝수입니다.")
else:
    print("홀수입니다.")
```

홀수입니다.

실습 5:

if 문으로, 어떤 학생의 점수가 90점 이상이면 A, 90점 미만 80점 이상이면 B, 80점 미만 70점 이하라면 C, 그 아래의 점수는 F로 성적을 출력하는 코드를 작성하시오.

예를 들어, 점수가 85점이라면, 점수 85점은 B입니다 라는 출력이 나오도록 한다.

코딩을 시작하거나 시로 코드를 [생성](#)하세요.

점수 85점은 B입니다.

4.1 연산자

1. 비교 연산자

- ==, !=, <, >, <=, >=

2. 논리 연산자

- and, or, not


```
x = 5
y = 8

# and: 둘 다 참일 때
if x > 0 and y > 0:
    print("x와 y 모두 양수입니다.")

# or: 둘 중 하나만 참이어도
if x == 5 or y == 5:
    print("x 또는 y 중 하나는 5입니다.")

# not: 부정
```


```
if not x < 0:
    print("x는 음수가 아닙니다.")
```

 x와 y 모두 양수입니다.
x 또는 y 중 하나는 5입니다.
x는 음수가 아닙니다.

5. for 문

- for 문은 반복을 수행할 때 사용합니다.
- range(n) 은 0부터 n-1까지의 정수 시퀀스를 생성합니다.

```
for i in range(5):
    print("현재 i의 값:", i)
```

 현재 i의 값: 0
현재 i의 값: 1
현재 i의 값: 2
현재 i의 값: 3
현재 i의 값: 4

```
# 예제: 1부터 10까지 짝수만 출력하기
for num in range(1, 11):
    if num % 2 == 0:
        print(num)
```

 2
4
6
8
10

실습 5:

for 문을 사용해 1부터 20까지 숫자의 합을 계산하고 출력해 보세요.

코딩을 시작하거나 시로 코드를 [선택](#)하세요.

 20

실습 6:

for 문을 사용해 2단부터 9단까지의 구구단을 출력하는 프로그램을 작성하세요.


코딩을 시작하거나 시로 코드를 [선택](#)하세요.

 숨겨진 출력 표시


6. 리스트 (List)

- 여러 값을 순서대로 저장하는 자료형입니다.
- 대괄호 [] 를 사용합니다.


```
fruits = ["사과", "바나나", "체리"]
print("과일 리스트:", fruits)
print("첫 번째 과일:", fruits[0])
```

 과일 리스트: ['사과', '바나나', '체리']
첫 번째 과일: 사과

```
# for 문과 함께 사용
for fruit in fruits:
    print("과일:", fruit)
```

 과일: 사과
과일: 바나나
과일: 체리

```
# 인덱스와 요소를 함께 출력
for index, word in enumerate(['첫번째', '두번째', '세번째']):
    print(index, word)
```

 0 첫번째
1 두번째
2 세번째

실습 7:

- numbers = [1, 2, 3, 4, 5] 리스트를 만들고, for 문을 사용해 각 요소의 제곱을 출력해 보세요.

코딩을 시작하거나 시로 코드를 [선택](#)하세요.

 1
4
9
16
25


7. 행렬 (Matrix)

- 리스트 안에 리스트를 넣어 2차원 배열처럼 사용할 수 있습니다.

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

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

```
# 행렬의 특정 원소 접근  
print("2행 3열 원소:", matrix[1][2])
```

 `2행 3열 원소: 6`

실습 8:

- `matrix`의 대각선 원소들의 합을 계산해 보세요.

코딩을 시작하거나 AI로 코드를 생성하세요.

 `15`

✓ 심화 연습문제 (도전)

최솟값과 최댓값 구하기:

`[20, 10, 35, 30, 7]`에서 최솟값과 최댓값을 각각 구분해서 출력하시오.

코딩을 시작하거나 AI로 코드를 생성하세요.

 `7 35`