

# 파이썬 3.14 교육 자료

---

## Python 3.14 Educational Materials

### 교육 목표

- 파이썬 기본 문법 완전 정복
- 실무에서 사용할 수 있는 실용적 예제 학습
- 체계적이고 단계별 학습 진행

### 학습 내용

- 변수 (Variables)
- 연산자 (Operators)
- 제어문 (Control Statements)
- 클래스와 상속 (Classes & Inheritance)
- 리스트 (Lists)

## 학습 목차

---

### 1. 변수 (Variables)

- 기본 변수 타입
- 컬렉션 타입 변수
- 변수 스코프와 네이밍

### 2. 연산자 (Operators)

- 산술 연산자
- 비교 및 논리 연산자
- 비트 연산자 및 기타

### 3. 제어문 (Control Statements)

- 조건문
- 반복문
- 고급 제어문

## 학습 목차 (계속)

---

### 4. 클래스와 상속 (Classes & Inheritance)

- 기본 클래스와 상속

### 5. 리스트 (Lists)

- 기본 리스트 연산
- 고급 리스트 연산
- 실용적인 리스트 예제

총 13개의 예제 파일로 구성

# 1. 변수 (Variables)

---

## 기본 변수 타입

```
# 숫자형 변수
age = 25           # 정수 (Integer)
height = 175.5     # 실수 (Float)
temperature = -10.3 # 음수 실수

# 문자열 변수
name = "김철수"    # 한글 문자열
english_name = "John Doe" # 영문 문자열

# 불린 변수
is_student = True  # 참 (True)
is_working = False # 거짓 (False)
```

**\*\*핵심 포인트\*\*:** 파이썬은 동적 타입 언어로 변수 선언 시 타입을 명시하지 않습니다.

# 1. 변수 (Variables) - 컬렉션 타입

---

## 리스트, 튜플, 딕셔너리, 집합

```
# 리스트 (List) - 순서가 있고 변경 가능
fruits = ["사과", "바나나", "오렌지", "포도"]
```

```
# 튜플 (Tuple) - 순서가 있고 변경 불가능
coordinates = (10, 20)
person_info = ("김철수", 25, "서울")
```

```
# 딕셔너리 (Dictionary) - 키-값 쌍
student = {
    "이름": "이영희",
    "나이": 22,
    "전공": "컴퓨터공학"
}
```

```
# 집합 (Set) - 중복 제거, 순서 없음
unique_numbers = {1, 2, 3, 4, 5, 5, 4} # {1, 2, 3, 4, 5}
```

## 2. 연산자 (Operators) - 산술 연산자

---

### 기본 산술 연산

```
a = 10
b = 3

print(f"덧셈: {a} + {b} = {a + b}")      # 13
print(f"뺄셈: {a} - {b} = {a - b}")      # 7
print(f"곱셈: {a} * {b} = {a * b}")      # 30
print(f"나눗셈: {a} / {b} = {a / b}")     # 3.333...
print(f"나머지: {a} % {b} = {a % b}")    # 1
print(f"거듭제곱: {a} ** {b} = {a ** b}") # 1000
print(f"정수 나눗셈: {a} // {b} = {a // b}") # 3
```

### 복합 할당 연산자

```
x = 10
x += 5 # x = x + 5
x -= 3 # x = x - 3
x *= 2 # x = x * 2
```

## 2. 연산자 (Operators) - 비교 및 논리 연산자

---

### 비교 연산자

```
a = 10
b = 5

print(f"a == b: {a == b}") # False
print(f"a != b: {a != b}") # True
print(f"a > b: {a > b}") # True
print(f"a < b: {a < b}") # False
print(f"a >= b: {a >= b}") # True
print(f"a <= b: {a <= b}") # False
```

### 논리 연산자

```
age = 25
has_license = True

# AND 연산자
can_drive = age >= 18 and has_license # True

# OR 연산자
can_vote = age >= 18 or has_license # True

# NOT 연산자
is_minor = not (age >= 18) # False
```

### 3. 제어문 (Control Statements) - 조건문

---

#### if, elif, else 문

```
score = 85

if score >= 90:
    grade = "A"
elif score >= 80:
    grade = "B"
elif score >= 70:
    grade = "C"
elif score >= 60:
    grade = "D"
else:
    grade = "F"

print(f"점수 {score}점의 등급은 {grade}입니다.")
```

#### 중첩된 조건문

```
age = 25
has_license = True

if age >= 18:
    print("성인입니다.")
    if has_license:
        print("면허가 있어서 운전할 수 있습니다.")
    else:
        print("면허가 없어서 운전할 수 없습니다.")
else:
    print("미성년자입니다.")
```



### 3. 제어문 (Control Statements) - 반복문

---

#### for문

```
# 기본 for문
fruits = ["사과", "바나나", "오렌지", "포도"]
for fruit in fruits:
    print(f"과일: {fruit}")

# range()를 사용한 for문
for i in range(1, 6):
    print(f"숫자: {i}")

# enumerate()를 사용한 for문
subjects = ["수학", "영어", "과학", "사회"]
for index, subject in enumerate(subjects, 1):
    print(f"{index}. {subject}")
```

#### while문

```
count = 1
while count <= 5:
    print(f"카운트: {count}")
    count += 1
```

### 3. 제어문 (Control Statements) - 고급 제어문

---

#### break와 continue

```
# break 예제 - 5에서 중단
for i in range(1, 11):
    if i == 5:
        break
    print(f"숫자: {i}")

# continue 예제 - 홀수만 출력
for i in range(1, 11):
    if i % 2 == 0:
        continue
    print(f"홀수: {i}")
```

#### 리스트 컴프리헨션

```
# 기본 리스트 컴프리헨션
squares = [x**2 for x in range(1, 6)] # [1, 4, 9, 16, 25]

# 조건이 있는 리스트 컴프리헨션
even_squares = [x**2 for x in range(1, 11) if x % 2 == 0]

# 문자열 리스트 컴프리헨션
words = ["hello", "world", "python"]
upper_words = [word.upper() for word in words]
```

## 4. 클래스와 상속 (Classes & Inheritance)

---

### 기본 클래스 정의

```
class Person:
    # 클래스 변수
    species = "Homo sapiens"

    def __init__(self, name, age, gender):
        # 인스턴스 변수
        self.name = name
        self.age = age
        self.gender = gender

    def introduce(self):
        return f"안녕하세요, 저는 {self.name}이고 {self.age}세 {self.gender}입니다."

    def have_birthday(self):
        self.age += 1
        print(f"{self.name}의 생일! 이제 {self.age}세가 되었습니다.")
```

### 클래스 사용

```
person1 = Person("김철수", 25, "남성")
print(person1.introduce())
person1.have_birthday()
```

## 4. 클래스와 상속 (Classes & Inheritance) - 상속

---

### 상속 예제

```
class Student(Person):
    def __init__(self, name, age, gender, student_id, major):
        # 부모 클래스 생성자 호출
        super().__init__(name, age, gender)

        # 학생만의 속성
        self.student_id = student_id
        self.major = major
        self.grades = []

    def add_grade(self, subject, score):
        self.grades.append({"subject": subject, "score": score})

    def introduce(self):
        # 메서드 오버라이딩
        base_intro = super().introduce()
        return f"{base_intro} 저는 {self.major} 전공 학생입니다."
```

### 상속 사용

```
student1 = Student("박민수", 20, "남성", "2023001", "컴퓨터공학")
student1.add_grade("프로그래밍", 95)
print(student1.introduce())
```

## 5. 리스트 (Lists) - 기본 연산

### 리스트 생성과 접근

```
# 리스트 생성
fruits = ["사과", "바나나", "오렌지", "포도"]
numbers = [1, 2, 3, 4, 5]

# 인덱싱
print(fruits[0])    # "사과" - 첫 번째 요소
print(fruits[-1])   # "포도" - 마지막 요소

# 슬라이싱
print(fruits[1:3])   # ["바나나", "오렌지"] - 1번부터 2번까지
print(fruits[:2])    # ["사과", "바나나"] - 처음부터 1번까지
print(fruits[2:])    # ["오렌지", "포도"] - 2번부터 끝까지
```

### 리스트 요소 추가/삭제

```
shopping_list = ["우유", "빵"]

# 요소 추가
shopping_list.append("계란")    # 맨 뒤에 추가
shopping_list.insert(1, "치즈") # 1번 위치에 삽입
shopping_list.extend(["사과", "바나나"]) # 여러 요소 한번에 추가

# 요소 삭제
shopping_list.remove("치즈")    # 특정 값 삭제
popped_item = shopping_list.pop() # 마지막 요소 삭제하고 반환
del shopping_list[0]           # 0번 위치 요소 삭제
```

## 5. 리스트 (Lists) - 고급 연산

---

### 리스트 정렬과 검색

```
scores = [85, 92, 78, 96, 88, 85, 90]

# 정렬
scores.sort()           # 오름차순 정렬 (원본 변경)
scores.sort(reverse=True) # 내림차순 정렬

sorted_scores = sorted(scores) # 정렬된 새 리스트 생성

# 검색
print(scores.count(85))      # 85가 몇 개 있는가?
print(scores.index(92))     # 92의 인덱스는?
print(90 in scores)         # 90이 리스트에 있는가?
```

### 리스트 복사

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

# 얕은 복사
shallow_copy = original.copy() # 또는 original[:]

# 깊은 복사
import copy
deep_copy = copy.deepcopy(original)
```

## 5. 리스트 (Lists) - 실용 예제

---

### 학생 성적 관리 시스템

```
class StudentManager:
    def __init__(self):
        self.students = []

    def add_student(self, name, student_id, grades):
        student = {
            "name": name,
            "id": student_id,
            "grades": grades,
            "average": sum(grades) / len(grades) if grades else 0
        }
        self.students.append(student)

    def get_class_average(self):
        if not self.students:
            return 0
        total_average = sum(student["average"] for student in self.students)
        return total_average / len(self.students)

    def get_top_students(self, n=3):
        sorted_students = sorted(self.students,
                                  key=lambda x: x["average"], reverse=True)
        return sorted_students[:n]
```

## 5. 리스트 (Lists) - 실용 예제 (계속)

### 쇼핑몰 장바구니 시스템

```
class ShoppingCart:
    def __init__(self):
        self.items = []

    def add_item(self, name, price, quantity=1):
        # 이미 있는 상품인지 확인
        for item in self.items:
            if item["name"] == name:
                item["quantity"] += quantity
                return

        # 새로운 상품 추가
        item = {"name": name, "price": price, "quantity": quantity}
        self.items.append(item)

    def get_total_price(self):
        total = 0
        for item in self.items:
            total += item["price"] * item["quantity"]
        return total

    def apply_discount(self, discount_rate):
        total = self.get_total_price()
        discount_amount = total * discount_rate
        return total - discount_amount
```



# 실습 예제 - 종합 프로젝트

## 도서관 도서 관리 시스템

```
class LibraryManager:
    def __init__(self):
        self.books = []
        self.borrowed_books = []

    def add_book(self, title, author, isbn, copies=1):
        book = {
            "title": title,
            "author": author,
            "isbn": isbn,
            "copies": copies,
            "available": copies
        }
        self.books.append(book)

    def search_books(self, keyword):
        results = []
        keyword_lower = keyword.lower()

        for book in self.books:
            if (keyword_lower in book["title"].lower() or
                keyword_lower in book["author"].lower()):
                results.append(book)

        return results

    def borrow_book(self, isbn, borrower_name):
        for book in self.books:
            if book["isbn"] == isbn and book["available"] > 0:
                book["available"] -= 1
                borrowed_record = {
                    "isbn": isbn,
                    "title": book["title"],
                    "borrower": borrower_name
                }
                self.borrowed_books.append(borrowed_record)
                return True
        return False
```

# 학습 정리 및 다음 단계

---

## 학습 완료 내용

- ✓ 변수: 기본 타입, 컬렉션 타입, 스코프와 네이밍
- ✓ 연산자: 산술, 비교/논리, 비트 연산자
- ✓ 제어문: 조건문, 반복문, 고급 제어문
- ✓ 클래스와 상속: 기본 클래스, 상속, 메서드 오버라이딩
- ✓ 리스트: 기본/고급 연산, 실용 예제

## 다음 학습 단계

- 함수 (Functions): 매개변수, 반환값, 람다 함수
- 예외 처리 (Exception Handling): try-except-finally
- 파일 입출력 (File I/O): 파일 읽기/쓰기
- 모듈과 패키지 (Modules & Packages): 코드 재사용
- 데이터베이스 연동: SQLite, MySQL

## 실습 권장사항

1. 각 예제 코드를 직접 실행해보기
2. 코드를 수정하며 다양한 시나리오 테스트
3. 실무 프로젝트에 적용해보기

감사합니다! 🐍✨

---

## 파이썬 3.14 교육 자료

### 교육 자료 특징

- 📖 체계적인 단계별 학습
- 💻 실무 중심의 실용적 예제
- 🔍 상세한 한글 주석과 설명
- 🎯 파이썬 3.14 최신 기능 반영

### 연락처

- 교육 자료 문의 및 개선 제안 환영
- 지속적인 업데이트와 보완 예정

Happy Coding! 🚀