제6장 6-1 구문 오류와 예외 마무리

▶ 4가지 키워드로 정리하는 핵심 포인트

1. (　　)는 프로그램의 문법적인 오류로 프로그램이 실행조차 되지 않게 만드는 오류입니다.
2. (　　)는 프로그램 실행 중에 발생하는 오류입니다. try catch 구문 등으로 처리할 수 있습니다. 반대로 구문 오류는 실행 자체가 안 되므로 try catch 구문으로 처리할 수 없습니다.
3. (　　)는 조건문 등을 사용해 예외를 처리하는 기본적인 방법입니다.
4. (　　)은 예외 처리에 특화된 구문입니다.

정답 : 1. 구문 오류, 2. 예외(런타임 에러), 3. 기본 예외 처리
4. try except 구문

▶ 확인문제

1. 구문 오류(Synatax Error)와 예외(Exception)의 차이를 설명해 보세요.

()

정답

구문 오류 : 프로그램이 실행되기도 전에 발생하는 오류. 해결하지 않으면 프로그램 자체가 실행되지 않음.
예외 : 프로그램 실행 중에 발생하는 오류. 프로그램이 일단 실행되고 해당 지점에서 오류를 발생.

2. 다음 코드의 빈칸은 조건문을 사용한 코드, try except 구문을 사용한 코드로 채워서 예외가 발생하지 않게 만들어 주세요.

numbers = [52, 273, 32, 103, 90, 10, 275]

```
print("# (1) 요소 내부에 있는 값 찾기")
print("- {}는 {} 위치에 있습니다.".format(52, numbers.index(52)))
print()

print("# (2) 요소 내부에 없는 값 찾기")
number = 10000
():
        print("- {}는 {} 위치에 있습니다.".format(52, numbers.index(52)))
():
        print("- 리스트 내부에 없는 값입니다.")
print()

print("--- 정상적으로 종료되었습니다. ---")
```

정답

try 또는 if number in numbers
except : 또는 else

3. 다음 중 구문 오류 발생이 예상되면 '구문 오류'에, 예외 발생이 예상되면 '예외'에 ○ 표시를 한 후, 예상되는 에러명도 적어 보세요.

1. output = 10 + "개" -> 구문 오류 ( ), 예외 ( )
2. int("안녕하세요") -> 구문 오류 ( ), 예외 ( )
3. cursor.close() -> 구문 오류 ( ), 예외 ( )
4. [1, 2, 3, 4, 5][10] -> 구문 오류 ( ), 예외 ( )

정답

1. 예외 (○) -> ValueError
2. 예외 (○) -> ValueError
3. 구문 오류 (○) -> SyntaxError
4. 예외 (○) -> IndexError