제2장 데이터 (02-2 문자 데이터 처리)

▶ 6가지 키워드로 정리하는 핵심 포인트

1. (문자 데이터 연결 연산자(+))는 서로 다른 문자 데이터를 연결하는 처리를 합니다.

2. (문자 데이터 반복 연결 연산자(*))는 지정된 횟수만큼 문자 데이터를 반복 연결하는 처리를 합니다.

3. 파이썬 (len) 명령어를 사용하면 문자 데이터의 크기를 알 수 있습니다.

4. 문자 데이터를 구성하는 각각의 문자에 숫자가 부여되는데, 이 숫자를 (인덱스)라고 합니다.

5. (슬라이싱)이란 범위를 지정해서 새로운 문자 데이터를 만드는 방법입니다.

6. (인덱싱)이란 문자 데이터에서 특정한 위치에 존재하는 문자를 선택하는 방법입니다.

정답 : 1. 문자 데이터 연결 연산자(+),
2. 문자 데이터 반복 연결 연산자(*),
3. len, 4. 인덱스, 5. 슬라이싱, 6. 인덱싱

▶ 확인 문제

1. 파이썬을 기준으로 할 때, 다음 중 올바른 문장은 무엇인가요?

① 문자 데이터의 길이란 공백(Space)을 제거한 문자의 개수를 의미합니다.

② 이스케이프 문자는 역슬래시와 기호로 이루어졌으므로, 문자 데이터 길이

를 계산할 때 2로 계산합니다.

③ 문자 데이터에 저장된 각 문자는 인덱스라는 숫자로 관리하는데, 인덱스는 1부터 시작할 수도 있습니다.

④ 슬라이싱은 범위를 지정해서 여러 개의 문자를 선택할 때 사용하고, 인덱싱은 1개의 문자를 선택할 때 사용합니다.

⑤ 슬라이싱과 인덱싱을 할 때 문자 데이터의 길이를 넘어선 인덱스 번호를 사용하면 오류가 발생합니다.

⑥ 슬라이싱을 할 때 인덱스 번호를 생략할 수 있듯이, 인덱싱을 할 때도 인덱스 번호를 생략할 수 있습니다.

정답 : ④ 슬라이싱은 범위를 지정해서 여러 개의 문자를 선택할 때 사용하고, 인덱싱은 1개의 문자를 선택할 때 사용합니다.

해설 : ① 문자 데이터의 길이를 계산할 때 하나의 공백을 1로 계산합니다.

② 문자 데이터의 길이를 계산할 때 이스케이프 문자는 역슬래시와 기호를 하나로 생각해서 1로 계산합니다.

③ 문자 데이터의 인덱스는 0부터 시작하고, 다른 번호로 시작할 수 없습니다.

⑤ 슬라이싱을 할 때 문자 데이터의 길이를 넘어선 인덱스 번호를 사용하면 자동으로 마지막 문자까지 처리합니다.

⑥ 인덱싱을 할 때는 반드시 인덱스 번호를 알맞게 적어 줘야 합니다.

2. 다음 소스 코드를 보고 실행 결과를 맞혀 보세요.

```
print("3" + " + " + "2" + " = " + "5")
```

```
print("=" * 10)
print("5" + "분" + "" + "23" + "초")
```

실행 결과

```
3 + 2 = 5
==========
5분23초
```

해설 : 없음

3. 다음 소스 코드를 보고 실행 결과를 맞혀 보세요.

```
print("0123456789"[3])
print("0123456789"[3:3])
print("0123456789"[2:3])
```

실행 결과

```
3
2
```

해설 : 없음

4. 다음 소스 코드를 보고 실행 결과를 맞혀 보세요.

```
print("혼자 공부하는 프로그래밍")
print(len("혼자 공부하는 프로그래밍"))
print("혼자 공부하는 프로그래밍"[:2])
print("혼자 공부하는 프로그래밍"[3:7])
print("혼자 공부하는 프로그래밍"[8:])
```

실행 결과

혼자 공부하는 프로그래밍
13
혼자
공부하는
프로그래밍

해설 : 없음

5. 다음 소스 코드를 보고 실행 결과를 맞혀 보세요.

```
print("C Major Scale")
print("-" * len("C Major Scale"))
print(
        "ABCDEFG"[2]
        + "ABCDEFG"[3]
        + "ABCDEFG"[4]
        + "ABCDEFG"[5]
        + "ABCDEFG"[6]
        + "ABCDEFG"[0]
        + "ABCDEFG"[1]
)
```

실행 결과

```
C Major Scale
-------------
CDEFGAB
```

해설 : 없음