KG 아이티뱅크

파이썬

PYTHON

문자열 변수

문자열 변수(String)

❖ **따옴표로 묶어 표현된 값 : 문자열**

1) 파이썬에서는 **쌍따옴표와 따옴표를 구분하지 않음**
   - ➢ **단, 짝은 엄격하게 맞춰 사용할 것을 요구함**
   - ➢ 왼쪽기준 **바깥쪽이 가장 가까운 쪽과 매칭**되는 구조
2) 파이썬에서는 문자와 문자열을 구별하지 않음
   - ➢ 문자 : 문자가 하나 들어있는 값
   - ➢ 문자열 : 문자가 여러 개 들어있는 값
3) 문자열에는 문자, 숫자, 특수문자, 띄어쓰기, 엔터 등이 포함됨
4) 파이썬에는 이러한 문자열 값을 만드는 문법으로 두가지를 제공
   - ➢ " " : 따옴표 1회. 한줄로 작성된 문자열
   - ➢ """ """ : 따옴표 3회. 여러줄로 작성된 문자열

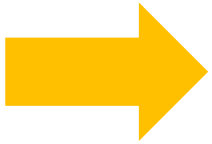
문자열 변수(String)

❖ 따옴표 1회 사용 : 한줄로 작성된 문자열 생성

➢ 한줄로 이루어진 값 작성에 유리

❖ 코드

```
print(“APPLE”)
print(“APPLE
APPLE”)
```



❖ 결과

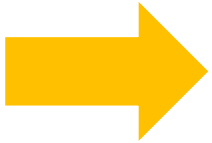
```
APPLE
Syntax Error
```

❖ 따옴표 3회 사용 : 여러줄로 작성된 문자열 생성

➢ 좀 더 직관적인 형태로 여러줄의 문자열 작성 가능

❖ 코드

```
print(“””APPLE”””)
print(“””APPLE
APPLE”””)
```



❖ 결과

```
APPLE
APPLE
APPLE
```

문자열 변수(String)

❖ **문자열에 대해서 두가지 연산이 가능함**

1) + : 문자열을 하나로 합치기

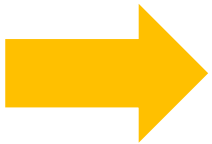
- ➢ 두 문자열을 합쳐 새로운 문자열로 만들어내는 것
- ➢ 값을 이용해 새로 만들어진 독립적인 문자열
- ➢ 변수, 값 등 조합을 가리지는 않으나 문자열끼리만 가능

2) * : 해당 문자열을 정수만큼 반복

1) 정수를 횟수로 인식하여 반복되는 횟수를 지정
2) 1 + 1 + 1 => 1 * 3과 같은 논리
3) 변수, 값 등 조합을 가리지 않으나 정수와 문자열끼리만 가능

❖ 코드

```
print("APPLE" + "TOWN")
print("AS" * 3)
```



❖ 결과

```
APPLETOWN
ASASAS
```

문자열 변수(String)

❖ **문자열에는 특별한 용도로 사용되도록 설정된 문자가 있음**

➢ **이스케이프 문자(Escape Sequence)**

- **역슬래시와 조합**하여 **지정한 기능으로 동작**하도록 설정됨
- 종류는 많지만 그 중 자주 쓰이며 **필수인 문자 5종**이 존재
- 파이썬 기준으로 **융통성을 위하여 사용**되는 경우가 많음

종류	기능
\n	줄바꾸기
\t	탭문자
\”	쌍따옴표 문자
\’	작은 따옴표 문자
\\	역슬래시 문자

문자열 변수(String)

<파일이름 : printF1.py>

실습문제1. 다음처럼 출력되는 코드를 작성해보세요.

결과

```
“QUICK” Brown “Fox”
====================
JUMP ‘over’
===========
“the \LAZY\ DOG”.
==================
```

문자열 변수(String)

<파일이름 : printF2.py>

실습문제2. 다음처럼 출력되는 코드를 한줄로 작성하세요.

결과

```
파이썬은 “쉽습니다.”
Python is ‘SIMPLE’

“다양한 방법을 시도하세요.”
```

문자열 변수(String)

<파일이름 : printF3.py>

실습문제3. 다음처럼 출력되는 코드를 한번 작성해보세요.

결과 (3을 입력했을 때 기준)

```
-- 값의 입력구간 --
큐브 크기 입력 :

-- 값의 출력구간 --
    □ □ □
    □ □ □
    □ □ □
```

문자열 인덱싱

문자열 인덱싱

❖ 인덱싱(indexing)

- 인덱스(색인)을 찾아보는 것
- 알아보기 쉽게 나열된 목록에서 찾아보는 것
- 지정한 문자의 위치를 확인해 사용하기 위한 방법

❖ 구조 word = “Python !” word[5] (or word[-3])

문자	P	y	t	h	o	n		!	문자
번호	0	1	2	3	4	5	6	7	번호
번호	-8	-7	-6	-5	-4	-3	-2	-1	번호

❖ 사용

❖ 코드

```
word = “Python !”
print( word[2], word[-3] )
```

❖ 결과

```
t n
```

문자열 인덱싱

<파일이름 : indexF1.py>

실습예제1. 아래의 문자열 변수만 이용해 출력해보세요.

변수

```
word="HLEO! WRD"
```

결과

```
HELLO WORLD
```

문자열 인덱싱

❖ 인덱스를 이용한 슬라이싱

- 슬라이싱(slicing) : 잘라내기
- 문자열에서 필요한 부분만 잘라 새로운 문자열 생성
- 원본은 건드리지 않고 원본에서 값을 복사하여 처리

❖ 구조 **word = “Python !”** **word[3:6]**

문자	P	y	t	h	o	n		!	문자
번호	0	1	2	3	4	5	6	7	번호
번호	-8	-7	-6	-5	-4	-3	-2	-1	번호

❖ 사용

❖ 코드

```
word = “Python !”
print( word[3:6] )
print( word[-5:-2] )
```

❖ 결과

```
hon
hon
```

문자열 인덱싱

❖ **문법 : 변수 또는 값[시작인덱스 : 끝인덱스 : 증감값]**

문자	P	y	t	h	o	n		!	문자
번호	0	1	2	3	4	5	6	7	번호
번호	-8	-7	-6	-5	-4	-3	-2	-1	번호

- 인덱스는 정방향/역방향 혼용이 가능
- 인덱스의 구간 설정은 시작이상 ~ 끝미만 방식
- 필요에 따라서는 증감값을 부여해 방향과 규칙성 설정이 가능
- 증감값은 구간에 맞춰 설정해줘야 정상작동

❖ **예시**

- word[3:6] : 3번이상 6번미만 구간의 문자열 잘라내기
- word[-3:-7:-1] : -3번이상 -7번미만 구간의 문자열 거꾸로 잘라내기
- **word[3:6:-2] : 증감값이 맞지 않아 작동하지 않는 문법**

문자열 인덱싱

<파일이름 : indexF2.py >

실습예제2. 시/구/동을 입력 받아서 슬라이싱 해보세요.

결과
출력예시는 부산시 수영구 민락동 기준

```
--값의 입력구간--
주소 입력 :

--값의 출력구간--
시 : 부산시
구 : 수영구
동 : 민락동
```

문자열 인덱싱

<파일이름 : indexF3.py >

실습예제3. 아래의 변수와 값을 이용해 값을 출력하세요.

변수

```
word=“Pyon Simple”
값1 : th
값2 : is
```

결과

```
Python is Simple
```

문자열 인덱싱

< 파일이름 : indexF4.py >

실습예제4. 준비된 변수를 이용해 아래와 같이 출력하세요.

변수

```
word="국민은행 123-444555 100만원 입금"
```

결과

```
은행 : 국민은행
계좌 : 123-444555
금액 : 100만원
```

문자열 메서드

문자열 메서드

❖ 문자열 메서드

- 메서드(Method) : 일정한 계획에 따른 방법
- 문자열을 좀 더 응용하기 쉽게 해주는 전용내장함수

❖ 종류

- count : 지정한 값의 개수 세기
- index : 지정한 값의 인덱스 확인
- upper / lower : 대문자 / 소문자로 변경
- strip : 지정한 문자열 제거
- replace : 지정한 문자열을 교체하여 새로운 문자열 생성
- split : 지정한 값을 기준으로 문자열 쪼개기
- format : 문자열을 양식으로 보고 값을 넣기