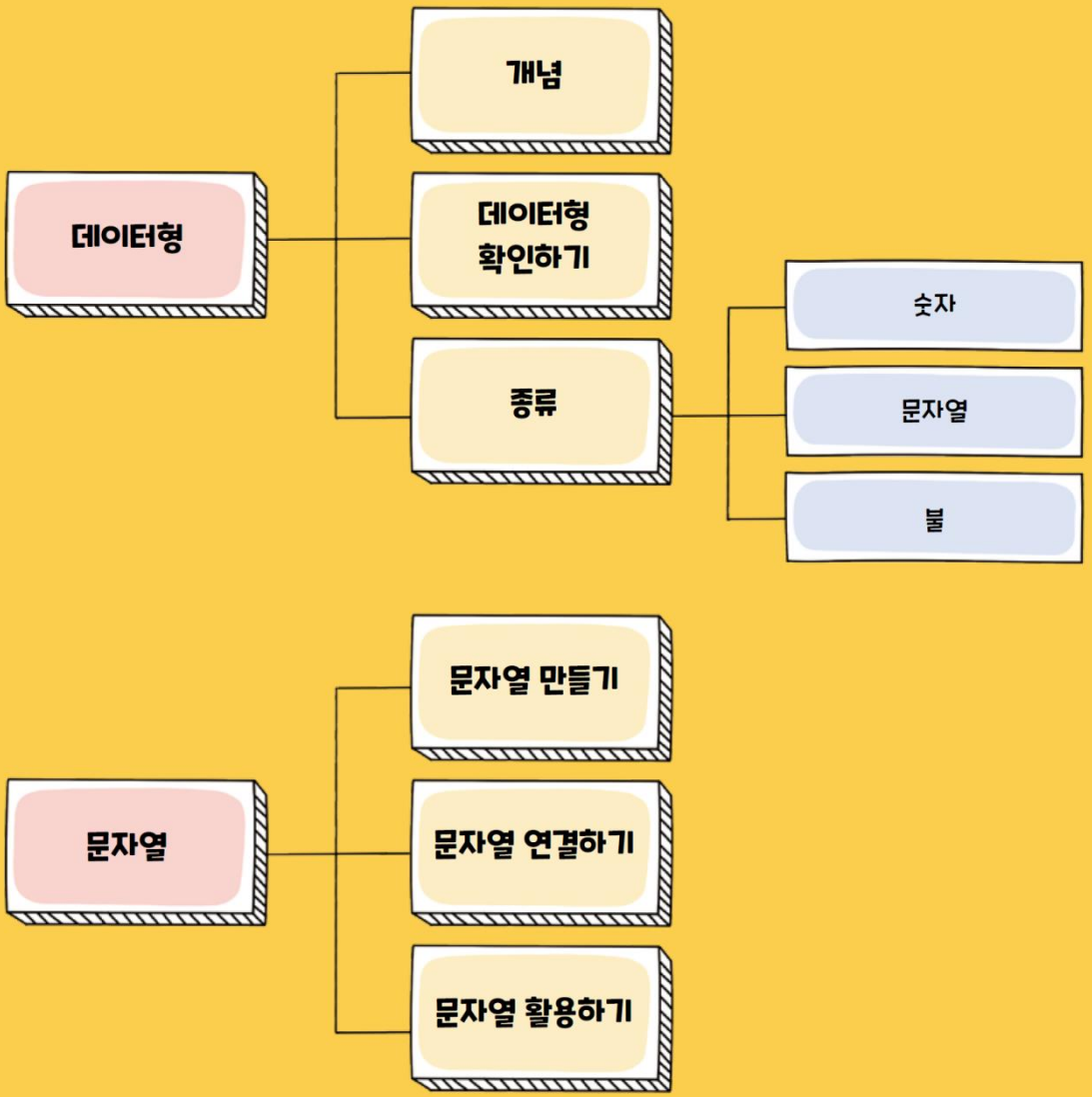
파이썬 프로그래밍

한빛아카데미
Hanbit Academy, Inc.

Chapter 04
데이터형과 문자열

목차

Preview



학습목표

- 파이썬의 데이터 형식을 이해합니다.
- 데이터형을 활용한 응용 프로그램을 작성합니다.
- 문자열에 대해 이해하고, 응용 방법을 학습합니다.
- 문자열 함수의 종류와 그 활용법을 익힙니다.

Section 01
데이터형이란?

데이터형의 개념

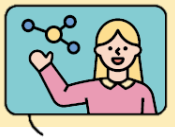
- 데이터형(Data Type)
  - 변수나 상수의 종류를 의미함
  - 그릇의 용도에 따라 국그릇, 밥그릇이 있듯이 변수의 종류도 다양함

- 4가지 기본 데이터형

```
var1 = 100
var2 = 3.14
var3 = "파이썬"
var4 = True
```

그림 4-1 변수의 종류

데이터형 확인하기



- type() 함수
  - 변수(그릇)의 종류 확인하기

```
>>> var1 = 100
>>> type(var1)
<class 'int'>
```

  - int: 정수형 / float: 실수형 / str: 문자형 / bool: 불형

```
>>> var2 = 3.14
>>> type(var2)
<class 'float'>
```

```
>>> var3 = "파이썬"
>>> type(var3)
<class 'str'>
```

```
>>> var4 = True
>>> type(var4)
<class 'bool'>
```

데이터형 확인하기

- 들어있는 데이터에 따라 변하는 변수의 데이터형
  - 현재 정수형인 var1에 문자열 "Hello" 넣고 type()으로 확인해보기

```
>>> type(var1)
<class 'int'>    <--- 정수형
>>> var1 = "Hello"
>>> type(var1)
<class 'str'>    <--- 문자열형
```

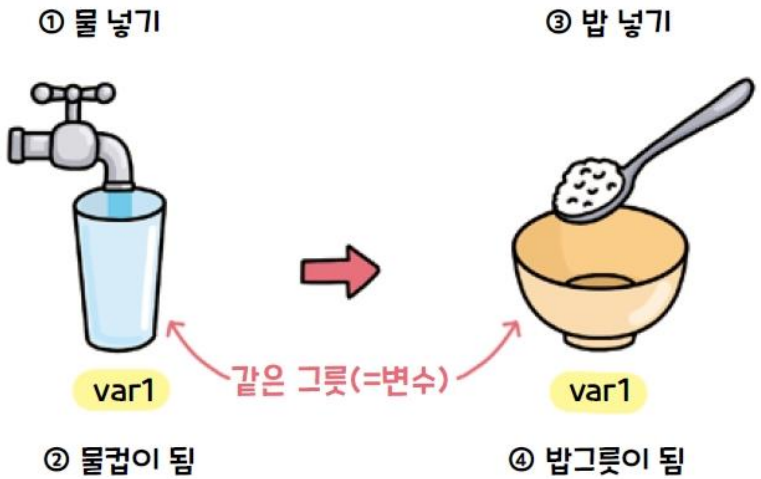
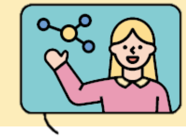


그림 4-2 들어있는 데이터에 따라 변하는 변수의 데이터형

Section 02
데이터형의 종류

숫자 데이터 형식



정수형

- 소수점이 없는 수
- 10, -300, 0 등

```
>>> var1 = 123
>>> type(var1)
<class 'int'>
```

- int의 크기에 제한이 없음

```
>>> var1 = 100**100
>>> print(var1)
10000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
00000000000000000000000000000000000000000000000000000000000000
000000000000000
```

숫자 데이터 형식

- 실수형
  - 소수점이 있는 수
  - 3.14, -8.8 등

```
>>> var2 = 3.14
>>> type(var2)
<class 'float'>
```

숫자 데이터 형식

- 정수와 정수의 연산은 정수

```
>>> var1 = 100
>>> var2 = 200
>>> res = var1 + var2
>>> print(res)
300
>>> type (res)
<class 'int'>
```

- 실수와 실수의 연산은 실수

```
>>> var1 = 100.0
>>> var2 = 200.0
>>> res = var1 + var2
>>> print(res)
300.0
>>> type(res)
<class 'float'>
```

숫자 데이터 형식

- 정수와 실수의 연산은 실수

```
>>> var1 = 100
>>> var2 = 200.0
>>> res = var1 + var2
>>> print(res)
300.0
>>> type(res)
<class 'float'>
```

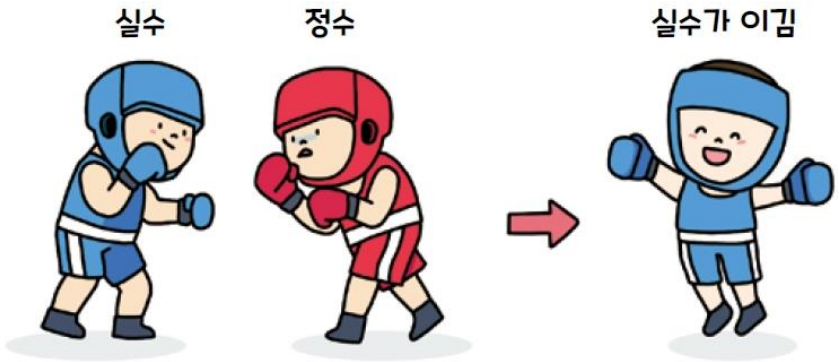


그림 4-3 정수와 실수의 연산 결과

- 정수와 정수여도 나눗셈을 하면 실수가 나옴

```
>>> var1 = 100
>>> var2 = 200
>>> res = var1 / var2
>>> print(res)
0.5
>>> type(res)
<class 'float'>
```

문자열 데이터 형식

- 문자열(String)
  - 문자열은 String의 약자로 str로 표현함
  - 글자들의 집합
  - "파이썬", "python", 123' 등
  - 문자열은 양쪽을 큰따옴표("")나 작은따옴표(' ')로 감싸야 함
  - 중간에 띄어쓰기가 있어도 상관 없음

```
>>> var1 = "난생처음 파이썬"
>>> print(var1)
난생처음 파이썬
>>> type(var1)
<class 'str'>
```

불 데이터 형식

- 불(Bool) 형
  - 참(True)이나 거짓(False)만 저장할 수 있는 데이터 형식
  - 논리형이라고도 함

```
>>> var1 = (100 > 10)
>>> print(var1)
True
```

```
>>> var2 = (100 <= 20)
>>> print(var2)
False
```

그림 4-4 True, False로만 구분되는 불형

불 데이터 형식

확인문제

1. 다음 중 잘못된 것을 고르시오.

① 파이썬에서 정수형은 int로 표현한다.
② 파이썬에서 정수의 크기에는 제한이 있다.
③ 정수와 정수를 덧셈하면 정수가 된다.
④ 정수와 정수를 나눗셈하면 정수가 된다.

2. 다음 중 내용이 잘못된 것을 모두 고르시오.

① 문자열은 작은따옴표로 묶을 수 있다.
② 문자열은 큰따옴표로 묶을 수 있다.
③ 문자열 중간에 띄어쓰기가 올 수 없다.
④ 정수와 문자열을 더하면 문자열이 된다.

3. 다음 중 결과가 다른 것 하나를 고르시오.

① `var1 = (30 > 300)`
② `var1 = (300 <= 300)`
③ `var1 = (30 <= 300)`
④ `var1 = (300 > 30)`

정답

Click!

Section 03
문자열 알아보기

문자열 만들기

- 문자열의 형태
  - 큰따옴표 또는 작은따옴표로 묶어서 표현
  - 문자열은 0개의 글자부터 여러 개의 글자까지 모두 문자열로 취급함

```
>>> var1 = "난생처음 파이썬"
>>> var2 = '난생처음 파이썬'
```

```
>>> var3 = "난"
>>> var4 = '난'
```

```
>>> var1 = ""
>>> var2 = ''
```

문자열 만들기

- 문자열의 형태
  - 여러 줄의 문자열을 표현하기
    - 큰따옴표나 작은따옴표를 3개 연속 사용해서 묶어줌

```
>>> var1 = """난생처음
파이썬을
열공 중입니다."""
>>> print(var1)
난생처음
파이썬을
열공 중입니다.
```

이스케이프 문자 활용하기

- 큰따옴표를 연속으로 사용한 것과 같은 효과 내기

```
>>> var1 = """난생
처음"""
>>> print(var1)
난생
처음
```

```
>>> var2 = "난생\n처음"
>>> print(var2)
난생
처음
```

- 이스케이프(escape) 문자
  - 서식 문자라고도 부름
  - 이스케이프 문자는 앞에 \(백슬래시)를 붙여 주어야 함

표 4-1 이스케이프 문자의 종류

이스케이프 문자	역할	설명
\n	새로운 줄로 이동	Enter 를 누른 효과
\t	다음 탭으로 이동	Tab 을 누른 효과
\b	뒤로 한 칸 이동	Backspace 를 누른 효과
\‘	‘를 출력	
\“	“를 출력	
\\	\를 출력	

이스케이프 문자 활용하기

이스케이프 문자 연습하기

[코드 4-1]

```
print("\n줄바꿈\n연습 ")
print("\t탭키\t연습")
print("어떤 글자를 \"강조\"하는 효과1")
print("어떤 글자를 \'강조\'하는 효과2")
print("\\\\ 백슬래시 2개 출력")
```

[실행결과]

```
줄바꿈
연습
        탭키    연습
어떤 글자를 "강조"하는 효과1
어떤 글자를 '강조'하는 효과2
\\ 백슬래시 2개 출력
```

이스케이프 문자 활용하기

이스케이프 문자 연습하기

- \를 1개만 사용하여 출력할 수도 있음
- 하지만 \ 바로 뒤에 큰따옴표, 작은따옴표와 같이 이스케이프 문자로 취급될 수 있는 문자가 와서는 안 됨

[코드 4-2]

```
print("\난생처음")
print("\\난생처음")
print("\ ")
print("\")
```

[실행결과]

```
\난생처음
\난생처음
\
오류 발생
```

문자열 연결하기

문자열 연결

- 더하기(+) 연산자 사용

```
>>> var1 = "난생" + "처음" + "파이썬"
>>> print(var1)
난생처음파이썬
```

- 더하기 연산자를 사용하면 띄어쓰기 없이 문자열이 연결됨
- 여러 행에 이어서 연결해도 됨

```
>>> var1 = "난생"
>>> var1 = var1 + "처음"
>>> var1 += "파이썬"
>>> print(var1)
난생처음파이썬
```

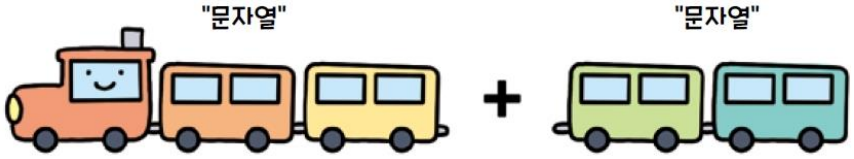


그림 4-5 문자열을 연결하는 방법

문자열 연결하기

- 문자열 연결 시 주의사항
  - 문자열과 숫자는 데이터형이 다르기 때문에 더할 수 없음
  - 문자열끼리의 뺄셈, 곱셈, 나눗셈도 모두 오류 발생함

```
var1 = "난생" - "처음"
var2 = "난생" * "처음"
var3 = "난생" / "처음"
```

  - 문자열에 숫자를 곱하는 것은 가능함

```
>>> var1 = "난생" * 3
난생난생난생
```

문자열 연결하기

확인문제

1. 다음 중 내용이 잘못된 것을 고르시오.

① 문자열은 큰따옴표 또는 작은따옴표로 묶을 수 있다.
② 문자열의 길이는 한 글자나 여러 글자 모두 허용된다.
③ 0개 글자는 문자열이 될 수 없다.
④ 문자열 중간에 따옴표를 표현할 수 있다.

2. 다음은 문자열의 연산과 관련된 내용이다. 잘못된 것을 고르시오.

① 문자열과 문자열을 더하면 문자열이 연결된다.
② 문자열에서 문자열을 빼면 오류가 발생한다.
③ 문자열에서 숫자를 빼면 오류가 발생한다.
④ 문자열에 숫자를 곱하면 오류가 발생한다.

정답

Click!

문자열과 관련된 유용한 함수

- len()
  - 문자열의 길이를 파악할 때 사용함

```
>>> var1 = "난생처음! Python"
>>> len(var1)
12
```

  - 한글, 기호, 영문, 공백, 숫자까지 모두 글자로 취급함

문자열과 관련된 유용한 함수

- 두 문자열을 입력받고 두 문자열의 길이 차이 체크하기

[코드 4-3]

```
var1 = input("첫 번째 문자열 ==>")
var2 = input("두 번째 문자열 ==>")

len1 = len(var1)
len2 = len(var2)

diff = len1 - len2

print("두 문자열의 길이 차이는", diff, "입니다.")
```

[실행결과]

```
첫 번째 문자열 ==>난생처음 파이썬
두 번째 문자열 ==>First Python
두 문자열의 길이 차이는 -4 입니다.
```

사용자 입력

문자열과 관련된 유용한 함수

- upper(), lower()
  - upper(): 영문 소문자를 대문자로 변환함
    - 문자열.upper() 형식으로 사용함
  - lower(): 영문 대문자를 소문자로 변환함
    - 문자열.lower() 형식으로 사용함

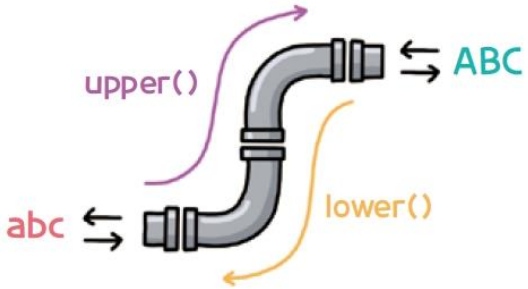


그림 4-6 대소문자 변환 함수

  - 영문을 제외한 한글, 숫자, 기호 등은 upper( ), lower( ) 함수의 영향을 받지 않음

```
>>> ss = 'First Python을 밤 12시까지 열공 중!'
>>> var1 = ss.upper()
>>> print(var1)
FIRST PYTHON을 밤 12시까지 열공 중!
>>> var2 = ss.lower( )
>>> print(var2)
first python을 밤 12시까지 열공 중!
```

문자열과 관련된 유용한 함수

- isupper(), islower()
  - isupper(): 문자열이 모두 대문자이면 True를 반환함
  - islower(): 문자열이 모두 소문자이면 True를 반환함

```
>>> ss = "first python"
>>> ss.isupper( )
False
>>> ss.islower( )
True
```

문자열과 관련된 유용한 함수

- count()
  - 문자열에서 어떤 글자가 몇 번 등장하는지 확인함

```
>>> ss = "난생처음 파이썬을 처음으로 학습 중입니다. 파이썬은
처음이지만 재미있네요. ^^"
>>> ss.count("처음")
3
>>> ss.count("Python")
0
```

문자열과 관련된 유용한 함수

- find()
  - 어떤 글자가 문자열의 몇 번째에 위치하는지 찾음
  - 문자열의 위치는 0번째부터 시작함

```
>>> ss = "난생처음 Python"
>>> ss.find("난생")
0
>>> ss.find("P")
5
```

그림 4-7 문자열의 순서

문자열과 관련된 유용한 함수

- find()
  - 똑같은 문자가 여러 개 나올 때 위치 찾기
  - find("찾을 단어", 시작위치)

```
>>> ss = "난생처음을 공부하는게 처음이네요"
>>> ss.find("처음")
2
>>> ss.find("처음")
2
>>> ss.find("처음", 4)
12
```

문자열 위치에 접근하기

문자의 위치값

```
>>> ss = "Python"
>>> len(ss)
6
```

그림 4-8 문자열의 순번 개념

- 각 문자는 '문자열[번호]' 형식으로 한 글자씩 접근 가능함
  - 문자열의 길이보다 큰 위치에 접근하면 인덱스 오류 발생함

```
>>> print(ss[5])
n
>>> print(ss[6])
오류 발생
```

문자열 위치에 접근하기

확인문제

1. 다음 설명에 해당하는 함수를 고르시오.

문자열을 대문자로 변경하는 함수는 [ ]이고, 소문자로 변경하는 함수는 [ ]이다.

① upper() - lower()
② lower() - upper()
③ isupper() - lower()
④ isupper() - islower()

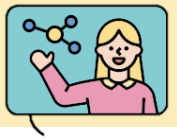
2. 다음 빈칸에 들어갈 단어를 채우시오.

문자열의 길이가 5라면, 위치번호는 [ ]번부터 [ ]번까지 총 5개가 할당된다.

정답

Click!

문자열 위치에 접근하기



하나 더 알기 **리스트 맛보기**

리스트(List)는 파이썬에서 가장 중요한 데이터 형식입니다. 7장에서 자세히 알아보겠지만, 지금 미리 간단히 살펴보겠습니다.
리스트와 문자열은 상당히 비슷한 점이 많은데, 리스트는 여러 개의 값을 하나로 묶어 놓은 꾸러미라고 생각할 수 있습니다. 다음은 3개의 값 10, 20, 30을 하나의 리스트로 묶은 것입니다.

```
myList = [ 10, 20, 30 ]
```

다음 [그림 4-9]와 같이 표현할 수 있습니다.

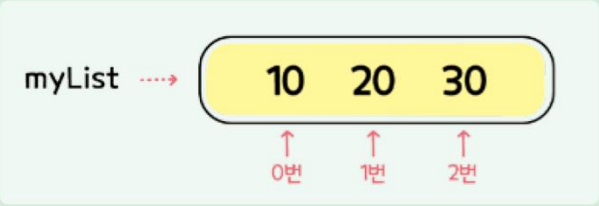


그림 4-9 **리스트 개념**

우선 myList의 개수는 문자열과 마찬가지로 len() 함수를 사용하여 구합니다.

```
len(myList)
```

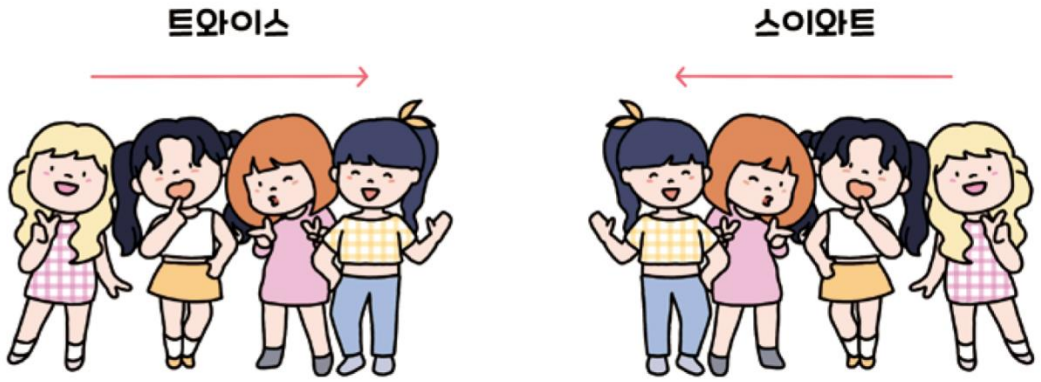
myList의 개수가 3개이므로 3이 반환됩니다. 그리고 각 값에 접근하기 위해서는 myList[위치]를 사용합니다. 위치도 문자열과 마찬가지로 0번째부터 시작합니다. 만약 20을 출력하기 위해서는 다음 코드와 같이 사용합니다.

```
print(myList[1])
```

지금은 이 정도로 살펴보고 7장에서 리스트에 대하여 자세히 공부하겠습니다.

LAB 문자열을 거꾸로 출력하는 프로그램

문자열 '트와이스'를 거꾸로 뒤집어서 출력하는 프로그램을 만들어 봅시다.



실행 결과

```
원본 문자열 ==> 트와이스
반대 문자열 ==> 스이와트
```

LAB 문자열을 거꾸로 출력하는 프로그램

1. lab04-01.py 파일을 만들고, ss 변수에 문자열 '트와이스'를 저장하기

```
ss = "트와이스"
print("원본 문자열 ==>", ss)
```

2. 반대 방향으로 출력될 문자열을 표시하기

- 마지막에 end=''를 사용하여 다음 행으로 넘어가서 출력되는 것을 방지함

```
print("반대 문자열 ==> ", end='')
```

3. 마지막 글자인 ss[3]부터 출력하기

- 글자의 길이가 4이므로 마지막 글자는 ss[3]이 됨. 3번째 글자부터 0번째 글자까지 차례대로 출력하면 글자가 거꾸로 출력되는 효과를 보여줌

```
print(ss[3], end='')
print(ss[2], end='')
print(ss[1], end='')
print(ss[0], end='')
```

4. <Ctrl>+<S>를 눌러서 변경된 내용을 저장하고, <F5>를 눌러 실행 결과 확인하기

LAB 대소문자 변환 프로그램

대문자는 소문자로, 소문자는 대문자로 변환하는 프로그램을 만들어 봅시다.

실행 결과

```
원본 문자열 ==>Python
반대 문자열 ==>pYTHON
```

지금은 배운 내용만으로 프로그램을 작성해야 해서 상당히 비효율적일 수밖에 없습니다. 6장까지 학습하고 나면 지금의 코드를 훨씬 효율적으로 작성할 수 있습니다. 우선은 대소문자 변환에만 초점을 맞춰 실습합니다.

LAB 대소문자 변환 프로그램

1. lab04-02.py 파일을 만들고, ss 변수에 문자열 "Python"을 저장하기
   - 대소문자를 변경한 문자를 저장할 빈 문자열 변수 ss2도 선언하기

```
ss = "Python"
print("원본 문자열 ==>", ss)
ss2 = ""
```

2. 첫 번째 글자인 ss[0]은 대문자이므로, lower() 함수를 사용해서 소문자로 변경하기

```
ss2 += ss[0].lower( )
```

3. 두 번째부터 여섯 번째 글자는 소문자이므로, upper( ) 함수를 사용해서 대문자로 변경한 후 ss2에 연결하기

```
ss2 += ss[1].upper( )
ss2 += ss[2].upper( )
ss2 += ss[3].upper( )
ss2 += ss[4].upper( )
ss2 += ss[5].upper( )
```

LAB 대소문자 변환 프로그램

4. 결국 ss2에는 대문자인 제일 앞 글자는 소문자로, 나머지 소문자는 대문자로 변경해서 저장됨

```
print("변환 문자열 ==>", end='')
print(ss2)
```

5. <Ctrl>+<S>를 눌러서 변경된 내용을 저장하고, <F5>를 눌러 실행 결과 확인하기

[실전 예제]
모험을 떠나는 거북이

실전 예제 모험을 떠나는 거북이

[문제]

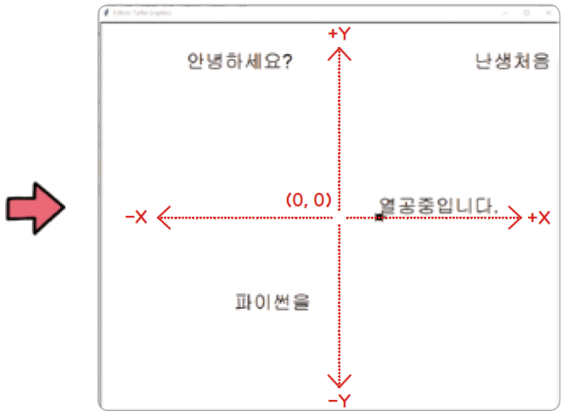
거북이는 화면의 가운데서 출발한다.

사용자가 원하는 X위치와 Y위치, 그리고 거북이가 쓸 문자열을 입력하면 거북이가 해당 위치를 찾아가 글자를 쓰는 프로그램을 작성해 보자.

실행 결과

```
X 이동량 ==> -300
Y 이동량 ==> 300
쓰고 싶은 글자 ==> 안녕하세요?
X 이동량 ==> 300
Y 이동량 ==> 300
쓰고 싶은 글자 ==> 난생처음
X 이동량 ==> -200
Y 이동량 ==> -200
쓰고 싶은 글자 ==> 파이썬을
X 이동량 ==> 100
Y 이동량 ==> 0
쓰고 싶은 글자 ==> 열공중입니다.
```

사용자 입력



실전 예제 모험을 떠나는 거북이

[해결]

```
import turtle

turtle.shape("turtle")
turtle.penup( )

while True :
    x = int(input("X위치 ==>"))
    y = int(input("Y위치 ==>"))
    text= input("쓰고 싶은 글자 ==>")

    turtle.goto(x,y)
    turtle.write(text, font=("Arial", 30))

turtle.done( )
```

Thank you!

한빛아카데미 Hanbit Academy, Inc.