

데 이 터 분 석 도 구

데이터 분석을 위한 파이썬

파이썬 소개

01

파이썬이란?

파이썬은 1991년 귀도 반 로섬(Guido van Rossum)이라는 프로그래머에 의해 개발된 언어로, 가독성이 높고 쉬운 문법 덕분에 다른 프로그래밍 언어보다 빠른 습득이 가능하다는 특징이 있다. 그 덕에 프로그래밍을 전공하지 않은 비전공자 중심으로 인기를 얻어 데이터 분석과 모델링을 다루는 통계학부터 딥러닝과 인공지능을 활용하는 의학에까지 다양한 분야에 두루 활용되고 있다.

검색량을 기준으로 프로그래밍 언어 선호도를 조사하는 TIOBE index에서 2021년 2월 기준 파이썬은 선호하는 프로그래밍 언어 3위(10.86%)를 차지했으며, 오라일리 미디어가 온라인 학습 플랫폼의 학습 과정 및 사용자 선호도를 분석해 발표한 프로그래밍 언어 순위에서는 파이썬이 Java와 C++을 제치고 가장 높은 사용률을 보였습니다.

01

파이썬 특징

스크립트 언어(Script Language)

- 스크립트 언어로 컴파일 과정 없이 인터프리터에 의해 실행 결과를 바로 확인하고 수정하며 코드를 작성할 수 있다.

동적 타입 언어

- 변수의 자료형을 지정하지 않고 선언하는 것만으로 값을 지정할 수 있다.

플랫폼 독립적 언어

- 대부분의 운영체제에서 동작하는 언어이기 때문에 어떤 환경에서도 활용이 가능하다.

01

컴파일 언어? 스크립트 언어?

컴파일 언어는 소스 코드를 컴파일 한 후 기계어를 CPU/메모리를 통해 읽어 실행하는 방식으로 동작한다.

컴파일을 하기 때문에 규모가 큰 프로그램은 시간이 오래 걸릴 수 있지만, 컴파일 후에는 기계어를 통하여 프로그램을 실행하기 때문에 실행 시간은 빠르다.

C++, Java가 대표적인 컴파일 언어이다.

스크립트 언어는 컴파일을 하지 않고 인터프리터로 소스 코드를 한 줄씩 읽어 바로 실행하는 방식으로 동작한다.

컴파일을 하지 않고 바로 실행한다는 특징을 가지고 있지만 소스 코드를 읽으며 실행하기 때문에 프로그램의 실행 시간이 컴파일 언어에 비해 느리다.

Python이 대표적인 스크립트 언어이다.

01

파이썬 장점

사용이 쉽다

- 파이썬의 문법 자체가 쉽고 간결하다. 사람의 사고 체계와 매우 닮아있어서 비전공자도 배우기 쉬운 언어이다.

개발 속도가 빠르다

- 쉽고 간결하기 때문에 높은 생산성을 보여준다. 다른 언어들과 비교를 하면 더 적은 코드로 많은 작업을 수행 할 수 있으며 복잡한 구문으로 인한 오류를 줄일 수 있다.

높은 확장성과 이식성을 가지고 있다.

- 다른 언어나 라이브러리에 쉽게 접근하여 연동할 수 있다. 이러한 특징으로 인하여 모든 코드를 일일이 작성할 필요가 없다.

01

파이썬 단점

느린 속도

- 컴파일 언어가 아닌 스크립트 언어이기 때문에 컴파일 언어에 비해 상대적으로 느리다.

모바일 개발

- 모바일 개발용 언어가 아니기 때문에 개발이 가능한 하지만 모바일 개발용 언어를 이용하는 편이 좋다.

런타임 에러

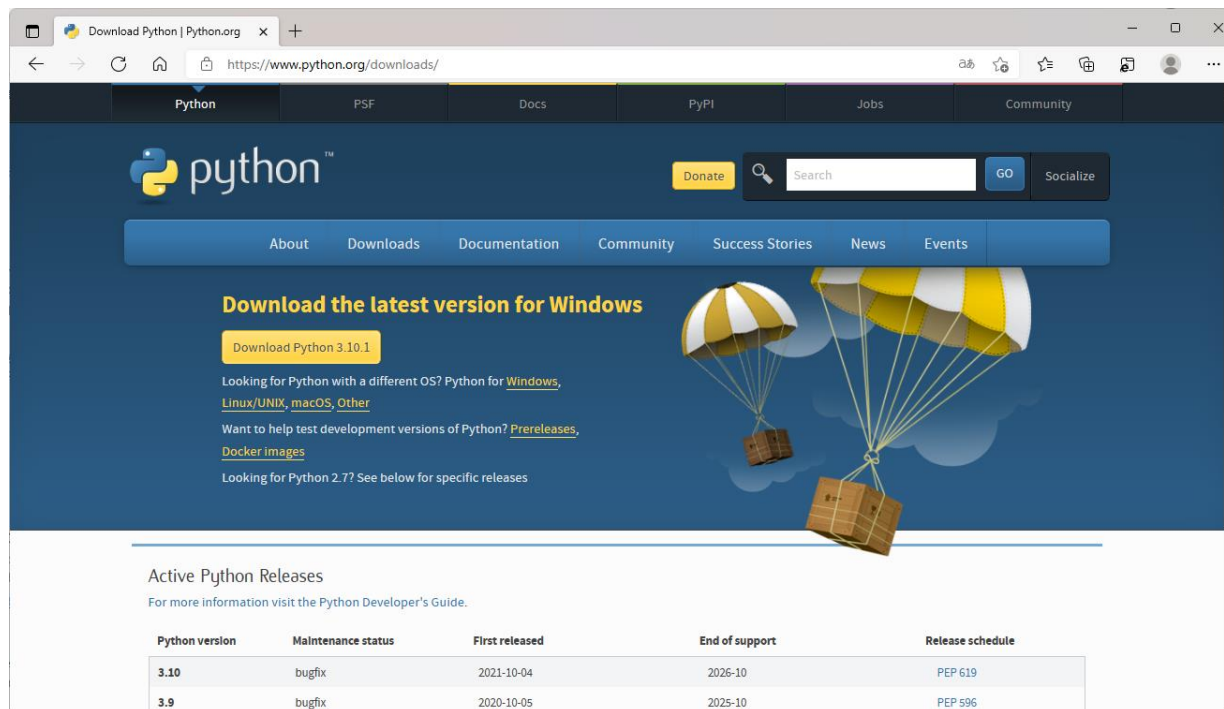
- 다른 언어들과 달리 실행할 때마다 컴파일을 진행한다. 이런 특성이 안 좋은 성능을 불러오고 많은 테스트를 요구하게 된다.

01

파이썬 설치

<https://www.python.org/> 접속

[Downloads] -> [Python 3.10.1] 클릭하여 저장



01

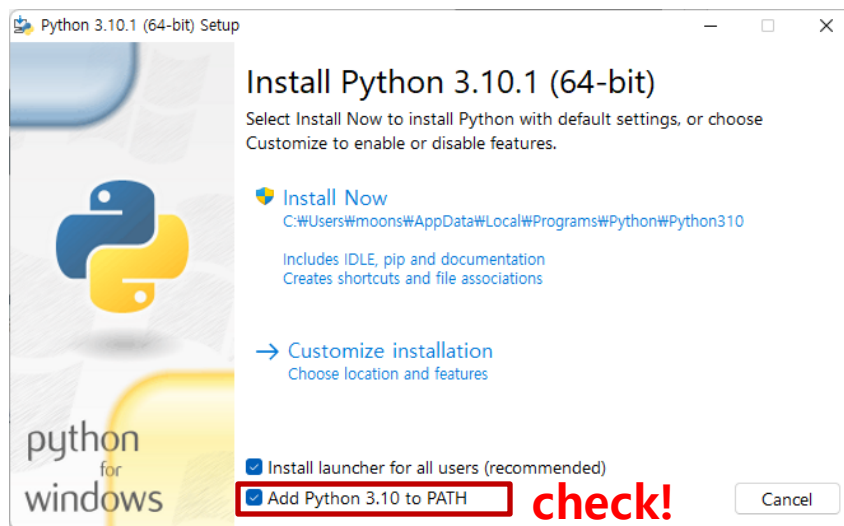
파이썬 설치

다운 받은 python3.10.1.exe 파일을 실행

Add Python 3.10 to PATH 체크(필수!)

[Install Now] 버튼 클릭

설치 진행

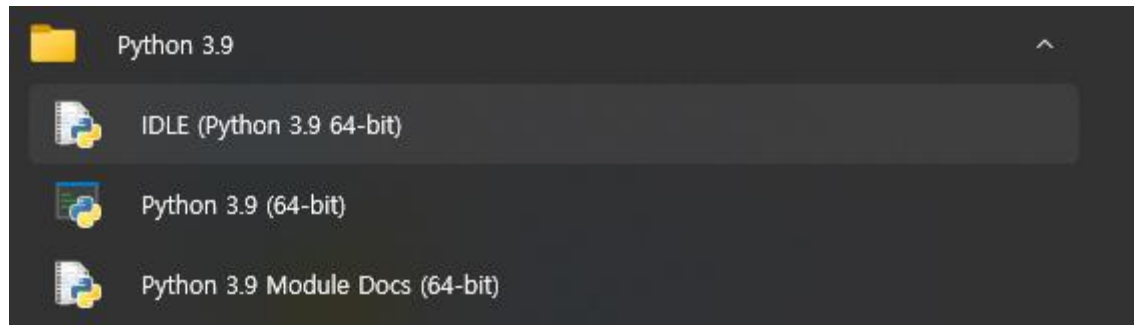


01

파이썬 실행

윈도우 [시작] 버튼

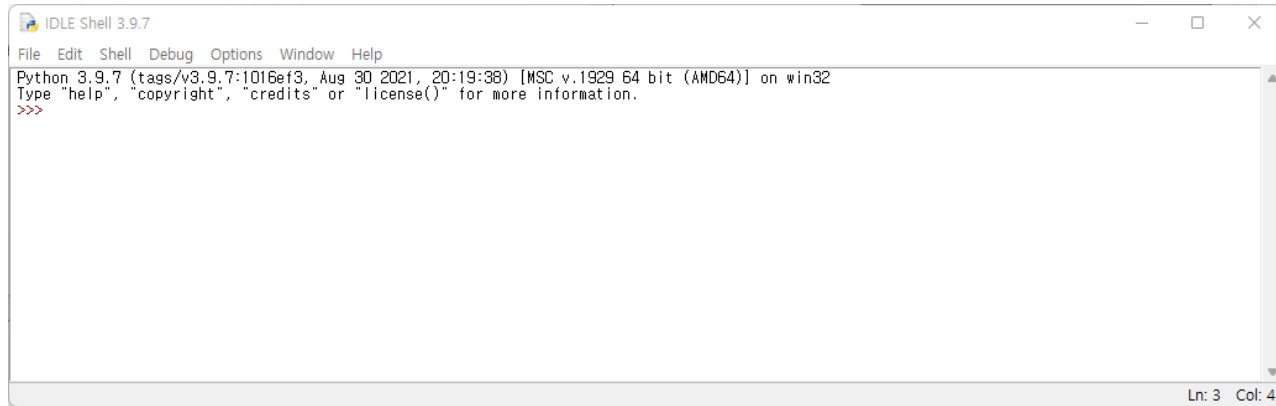
[모든 프로그램] -> [python 3.10] -> [IDLE (python 3.10 64bit)] 클릭



01

파이썬 실행

ILDE이 실행되면 아래 그림과 같이 대화형 모드의 파이썬 셸이 나타난다.

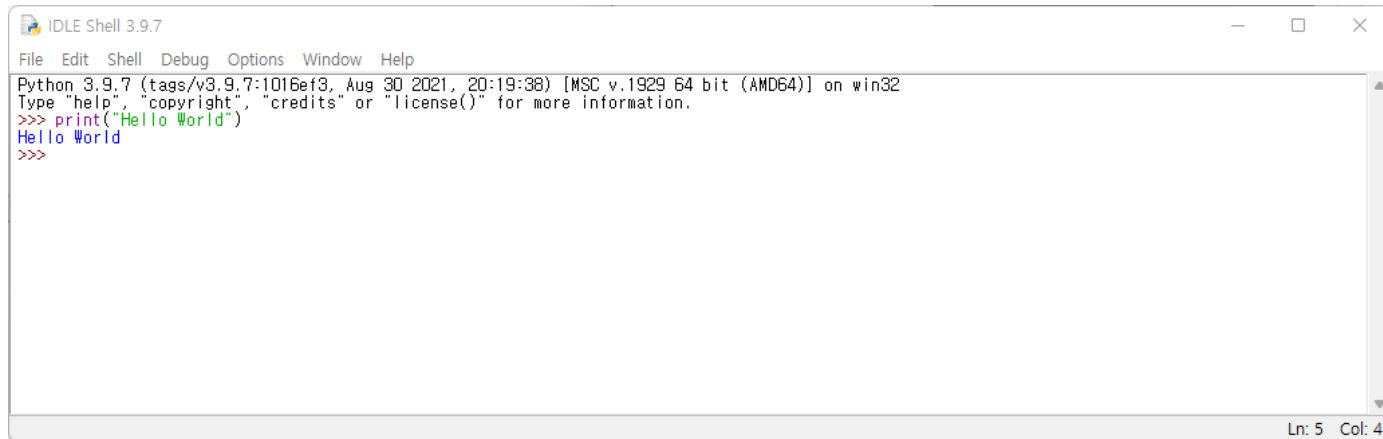


01

파이썬 코드 입력

`print("Hello World")`를 입력한 뒤 [Enter]을 클릭한다.

Hello World 가 출력되는 것을 확인 할 수 있다.



```
IDLE Shell 3.9.7
File Edit Shell Debug Options Window Help
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello World")
Hello World
>>>
```

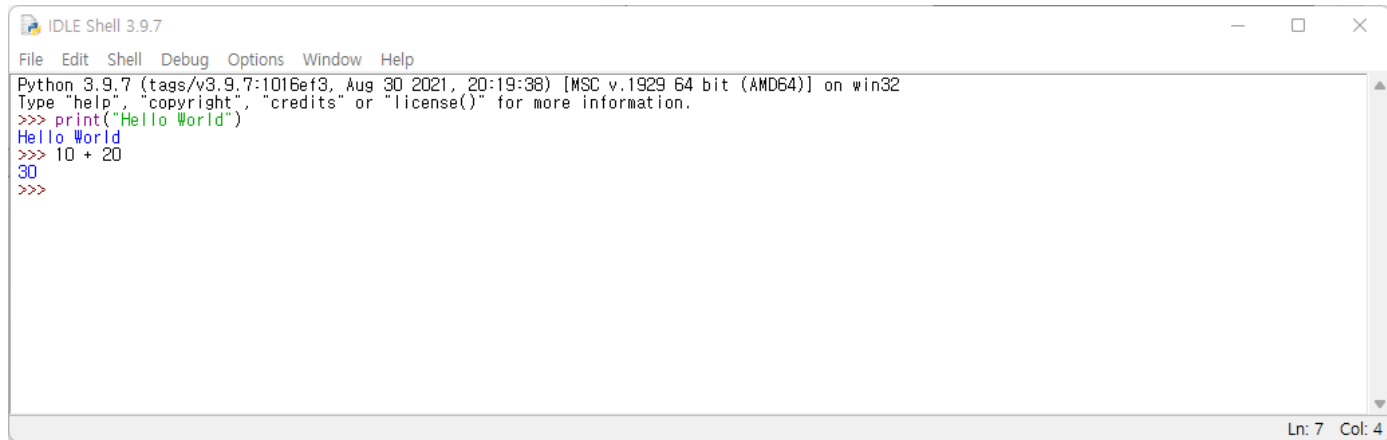
Ln: 5 Col: 4

01

파이썬 코드 입력

10 + 20 을 입력한 뒤 [Enter]을 클릭한다.

10 + 20은 30으로 출력되는 것을 확인 할 수 있다.

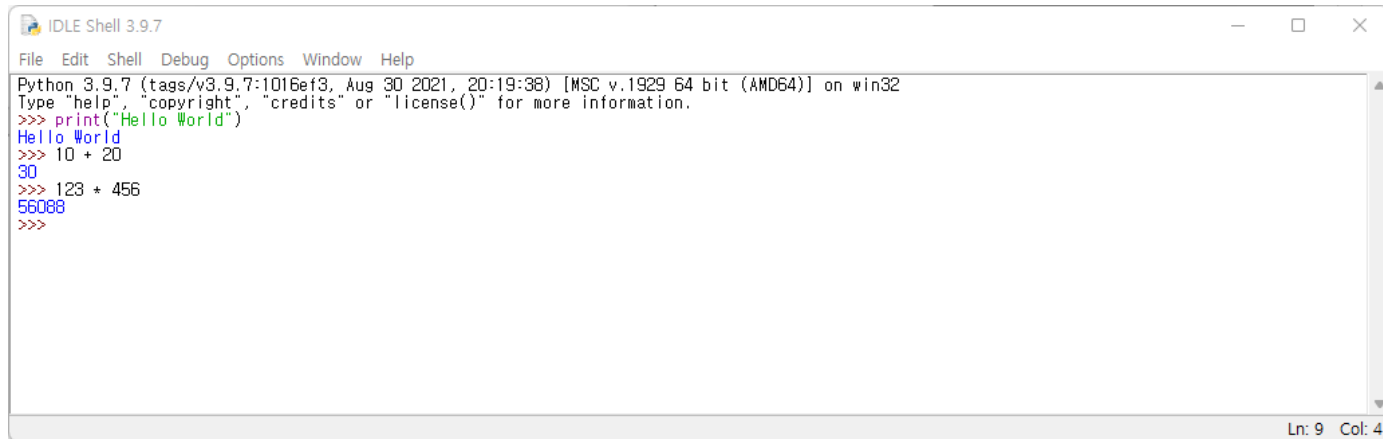
A screenshot of the IDLE Shell 3.9.7 window. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area shows the following text: 'Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32', 'Type "help", "copyright", "credits" or "license()" for more information.', '>>> print("Hello World")', 'Hello World', '>>> 10 + 20', '30', and '>>>'. The status bar at the bottom right indicates 'Ln: 7 Col: 4'.

01

파이썬 코드 입력

123 * 456 을 입력한 뒤 [Enter]을 클릭한다.

56088이 출력되는 것을 확인 할 수 있다.

A screenshot of the Python IDLE Shell 3.9.7 window. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area shows the following text: 'Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32', 'Type "help", "copyright", "credits" or "license()" for more information.', '>>> print("Hello World")', 'Hello World', '>>> 10 + 20', '30', '>>> 123 * 456', '56088', and '>>>'. The status bar at the bottom right shows 'Ln: 9 Col: 4'.

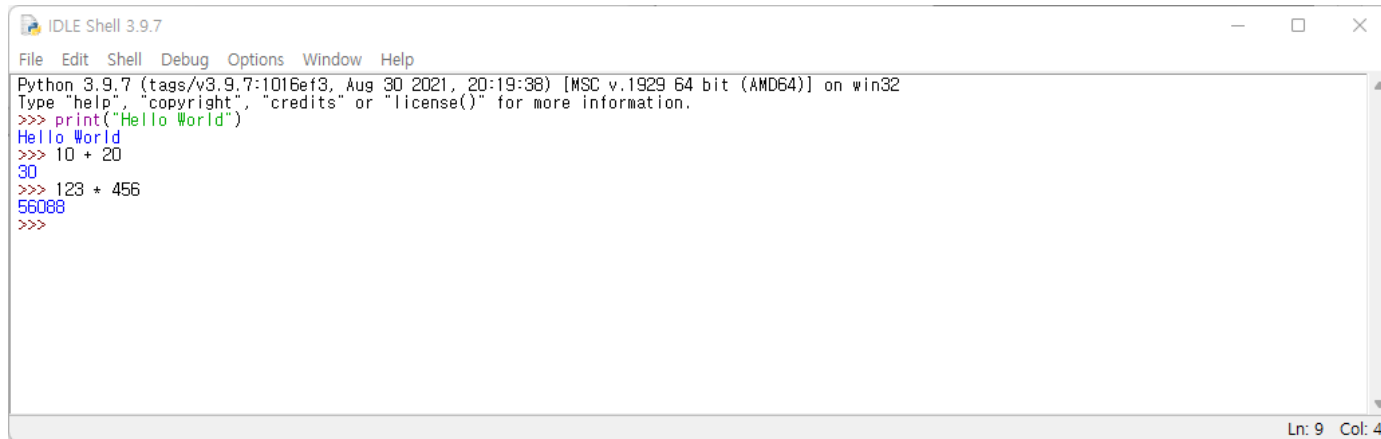
```
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello World")
Hello World
>>> 10 + 20
30
>>> 123 * 456
56088
>>>
```

01

파이썬 코드 입력

123 * 456 을 입력한 뒤 [Enter]을 클릭한다.

56088이 출력되는 것을 확인 할 수 있다.

A screenshot of the Python IDLE Shell 3.9.7 window. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area shows the following text: 'Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32', 'Type "help", "copyright", "credits" or "license()" for more information.', '>>> print("Hello World")', 'Hello World', '>>> 10 + 20', '30', '>>> 123 * 456', '56088', and '>>>'. The status bar at the bottom right shows 'Ln: 9 Col: 4'.

```
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>> print("Hello World")
Hello World
>>> 10 + 20
30
>>> 123 * 456
56088
>>>
```



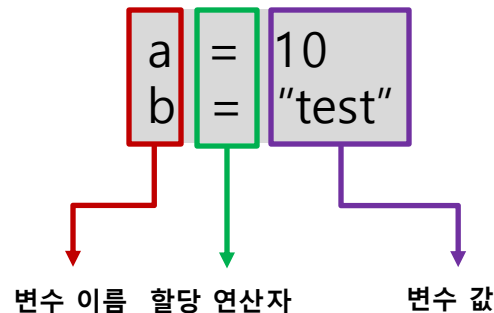
변수

02

변수(Variables)란?

변수의 사전적인 의미는 메모리에 데이터를 저장하는데 사용되는 공간의 이름이다.

변수에 저장되는 데이터 또는 객체는 프로그램 안에서 계속 변경이 가능하다.



02

변수(Variables)란?

변수의 이름은 자유롭게 만들어도 되지만, 몇 가지 유의 사항이 있다.

- _나 영문자로 시작해야 한다.
- 숫자로 시작할 수 없다.
- 특수문자를 사용할 수 없다.
- 변수 이름에는 공백이 존재하면 안 된다.(공백은 _로 표시)
- 파이썬의 예약어(if, while, for등)는 사용할 수 없다.

02

변수 선언

파이썬에서는 변수 선언과 동시에 값을 할당을 해주어야 한다.

```
a = 10  
b = "test"  
  
print(a)  
print(type(a))  
  
print(b)  
print(type(b))
```

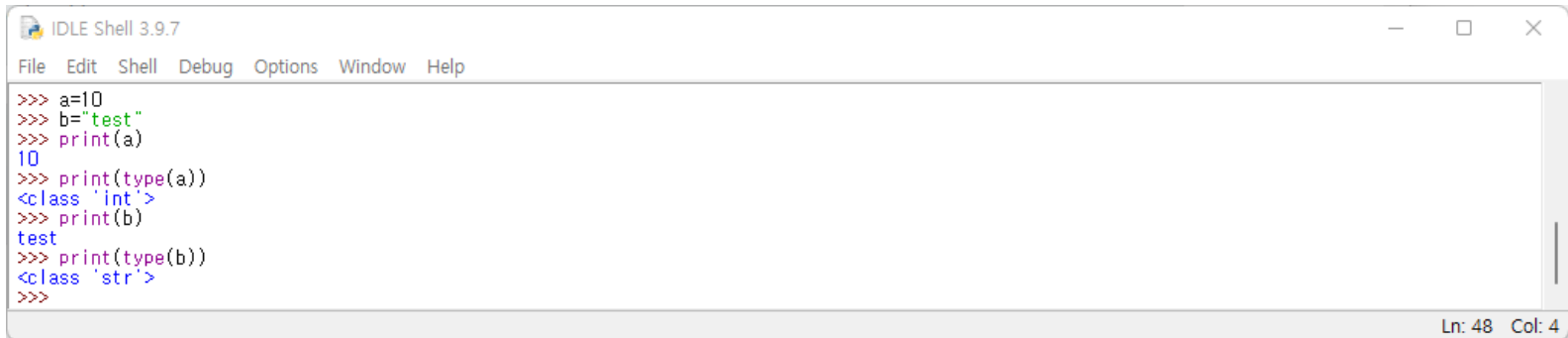
a라는 변수 이름에 숫자 10이라는 값을 할당하였다.

b라는 변수 이름에 문자 "test"라는 값을 할당하였다.

파이썬에서는 변수의 자료형을 따로 지정을 하지 않더라도 할당되는 값에 따라서 자동으로 변수의 형태가 지정이 된다.

02

변수 선언



```
IDLE Shell 3.9.7
File Edit Shell Debug Options Window Help
>>> a=10
>>> b="test"
>>> print(a)
10
>>> print(type(a))
<class 'int'>
>>> print(b)
test
>>> print(type(b))
<class 'str'>
>>>
```

Ln: 48 Col: 4

print(a)는 변수 a의 값인 10이 출력이 되고
print(type(a))는 a의 변수 타입이 숫자이기 때문에 int가 출력된다.

print(b)는 변수 b의 값인 "test"가 출력이 되고
print(type(b))는 b의 변수 타입이 문자이기 때문에 str이 출력된다.

02

변수 선언

앞에서 설명에서 변수에 할당된 값을 변경이 가능하다.

```
a = 10

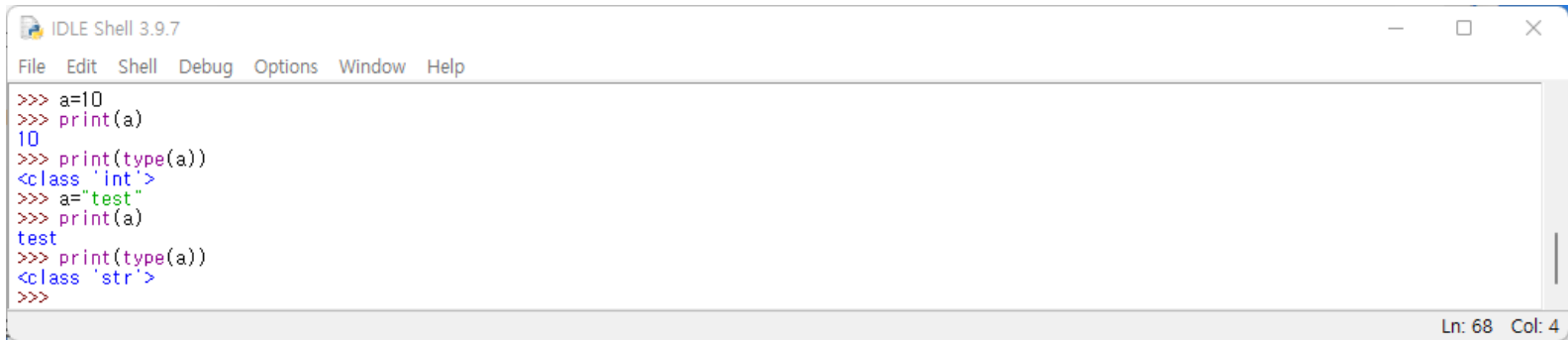
print(a)
print(type(a))

a = "test"

print(a)
print(type(a))
```

02

변수 선언



```
>>> a=10
>>> print(a)
10
>>> print(type(a))
<class 'int'>
>>> a="test"
>>> print(a)
test
>>> print(type(a))
<class 'str'>
>>>
```

Ln: 68 Col: 4

print(a)는 변수 a의 값인 10이 출력이 되고
print(type(a))는 a의 변수 타입이 숫자이기 때문에 int가 출력된다.

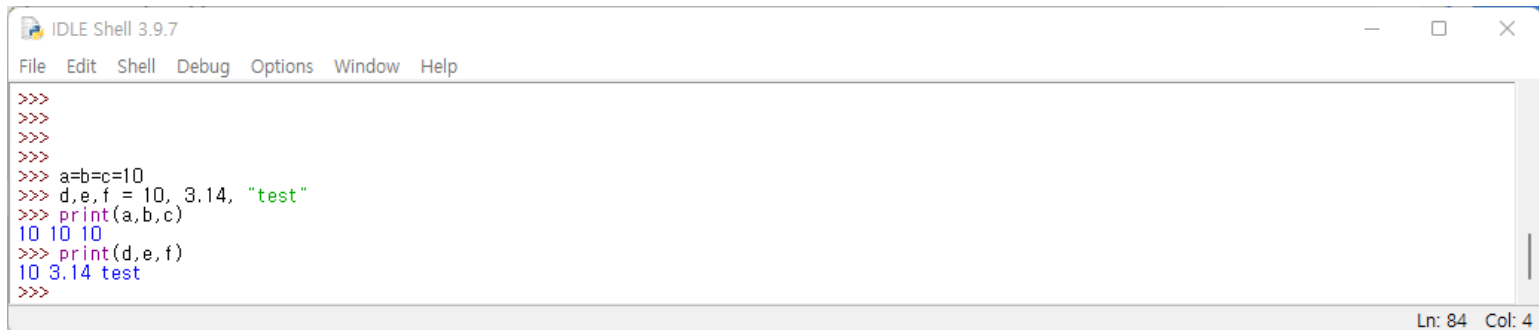
변수 a를 "test"라는 값으로 변경을 한 후
print(a)는 변경된 "test"라는 값이 출력이 되고
print(type(a)) 역시 값의 데이터 타입이 str으로 변경되었기 때문에
str이 출력이 된다.
(이전의 a = 10을 출력할 수는 없다.)

02

변수 선언

같은 값을 여러 변수에 동시에 지정할 수 있고,
각각 변수들에 다른 값들을 동시에 지정할 수 있다.

```
a = b = c = 10  
  
d, e, f = 10, 3.14, "test"  
  
print(a, b, c)  
  
print(d, e, f)
```



```
IDLE Shell 3.9.7  
File Edit Shell Debug Options Window Help  
>>>  
>>>  
>>>  
>>>  
>>> a=b=c=10  
>>> d,e,f = 10, 3.14, "test"  
>>> print(a,b,c)  
10 10 10  
>>> print(d,e,f)  
10 3.14 test  
>>>  
Ln: 84 Col: 4
```

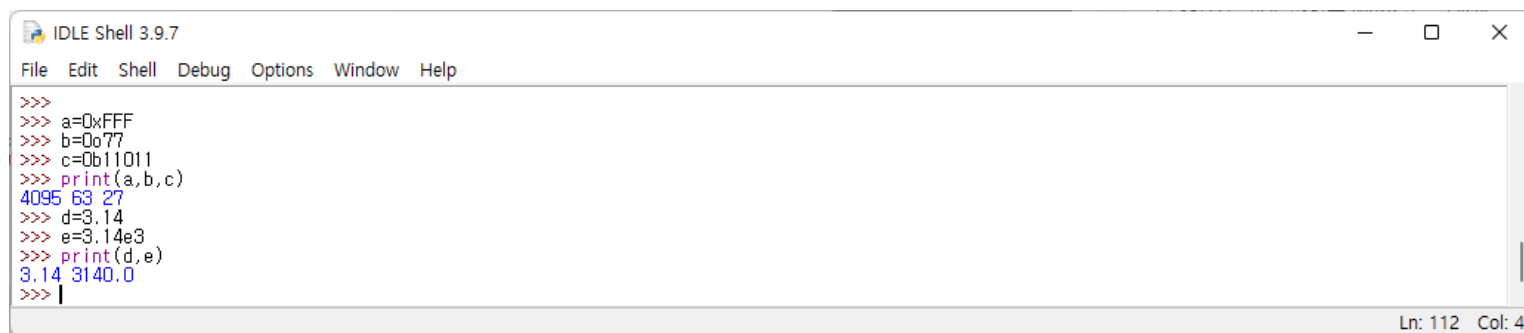

02

변수 데이터형(숫자형)

정수형은 16진수, 8진수, 2진수도 사용이 가능하다.
실수형은 소수점 데이터를 뜻한다.

```
a = 0xFF  
b = 0o77  
c = 0b11011  
print(a, b, c)
```

```
d = 3.14  
e = 3.14e3  
print(d, e)
```



```
IDLE Shell 3.9.7  
File Edit Shell Debug Options Window Help  
>>>  
>>> a=0xFF  
>>> b=0o77  
>>> c=0b11011  
>>> print(a,b,c)  
4095 63 27  
>>> d=3.14  
>>> e=3.14e3  
>>> print(d,e)  
3.14 3140.0  
>>> |
```

Ln: 112 Col: 4

3.14e3은 3.14×10^3 을 뜻한다.

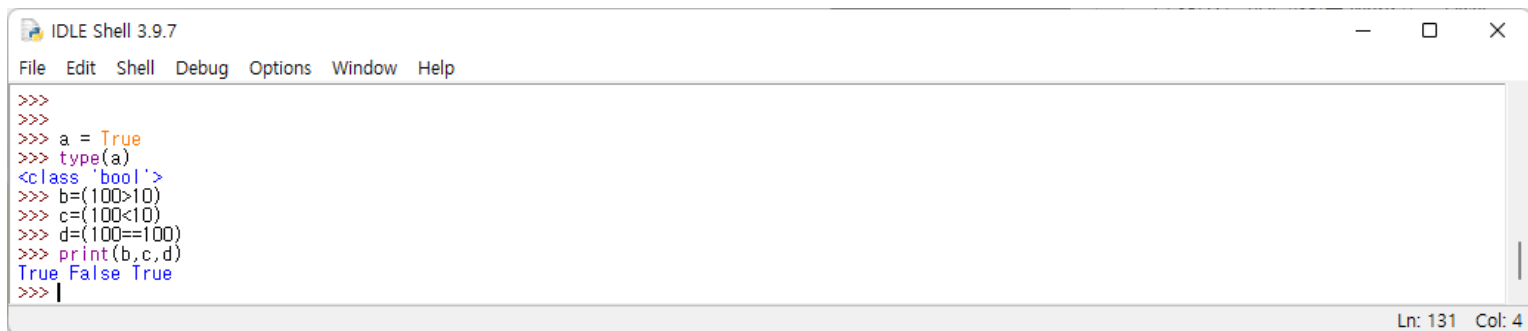
02

변수 데이터형(불형)

불(bool)형은 True, False의 2가지의 값만 존재한다.
비교의 결과를 참이나 거짓으로 출력하여 사용하기도 한다.

```
a = True
type(a)

b = (100 > 10)
c = (100 < 10)
d = (100 == 100)
print(b, c, d)
```



```
IDLE Shell 3.9.7
File Edit Shell Debug Options Window Help
>>>
>>>
>>> a = True
>>> type(a)
<class 'bool'>
>>> b=(100>10)
>>> c=(100<10)
>>> d=(100==100)
>>> print(b,c,d)
True False True
>>> |
```

Ln: 131 Col: 4

02

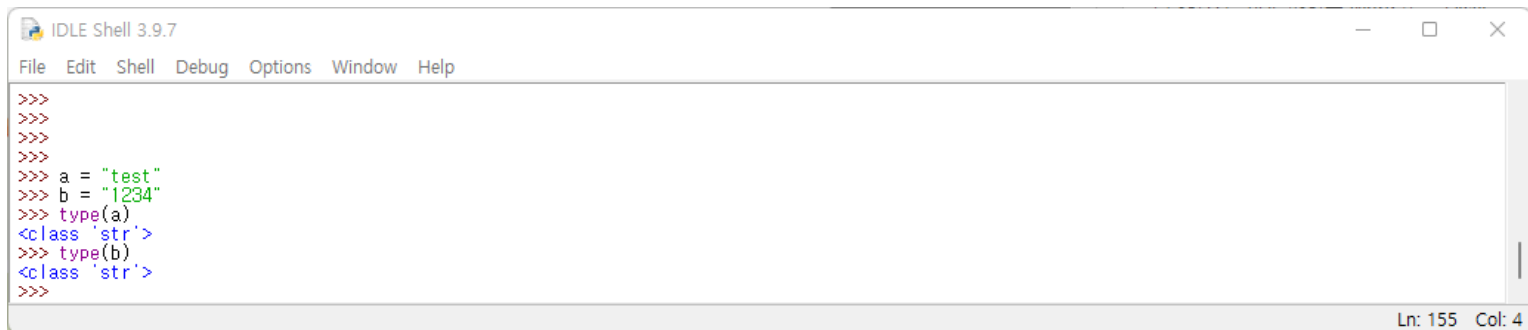
변수 데이터형(문자형)

문자형은 문자의 집합을 의미한다.

문자열은 ""(큰따옴표) '(작은따옴표)로 양쪽을 감싸야 한다.

```
a = "test"  
b = "1234"
```

```
type(a)  
type(b)
```



```
IDLE Shell 3.9.7  
File Edit Shell Debug Options Window Help  
>>>  
>>>  
>>>  
>>> a = "test"  
>>> b = "1234"  
>>> type(a)  
<class 'str'>  
>>> type(b)  
<class 'str'>  
>>>
```

Ln: 155 Col: 4

숫자형 데이터를 ""로 묶으면 문자형으로 인식한다.



연산자

03

산술 연산자

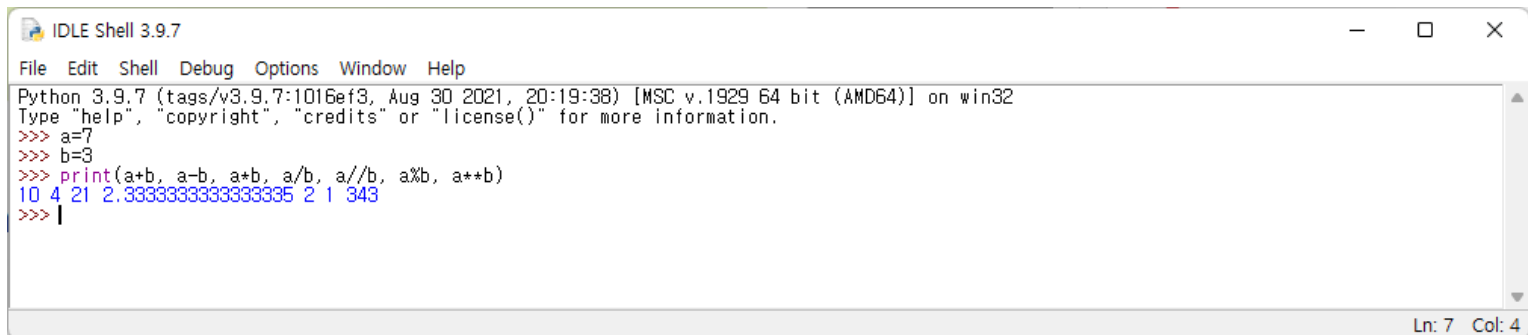
산술 연산자는 사칙연산을 다루는 기본적인면서 가장 많이 사용되는 연산자이다.

연산자	의미	예시	설명
=	대입 연산자	<code>a = 10</code>	변수 a에 10을 할당한다.
+	덧셈	<code>a = 5 + 3</code>	변수 a에 5와 3을 더한 값(8)을 할당한다.
-	뺄셈	<code>a = 5 - 3</code>	변수 a에 5에서 3을 뺀 값(2)을 할당한다.
*	곱셈	<code>a = 5 * 3</code>	변수 a에 5와 3을 곱한 값(15)을 할당한다.
/	나눗셈	<code>a = 5 / 3</code>	변수 a에 5에서 3을 나눈 값(1.666...)을 할당한다.
//	나눗셈(몫)	<code>a = 5 // 3</code>	변수 a에 5에서 3을 나눈 값에서 소수점을 버린 값(1)을 할당한다.
%	나머지 값	<code>a = 5 % 3</code>	변수 a에 5에서 3을 나눈 후 나머지 값(2)을 할당한다.
**	제곱	<code>a = 5 ** 3</code>	변수 a에 5의 3제곱 값(125)을 할당한다.

03

산술 연산자

```
a = 7  
b = 3  
  
print(a+b, a-b, a*b, a/b, a//b, a%b, a**b)
```



```
IDLE Shell 3.9.7  
File Edit Shell Debug Options Window Help  
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>> a=7  
>>> b=3  
>>> print(a+b, a-b, a*b, a/b, a//b, a%b, a**b)  
10 4 21 2.3333333333333335 2 1 343  
>>> |
```

Ln: 7 Col: 4

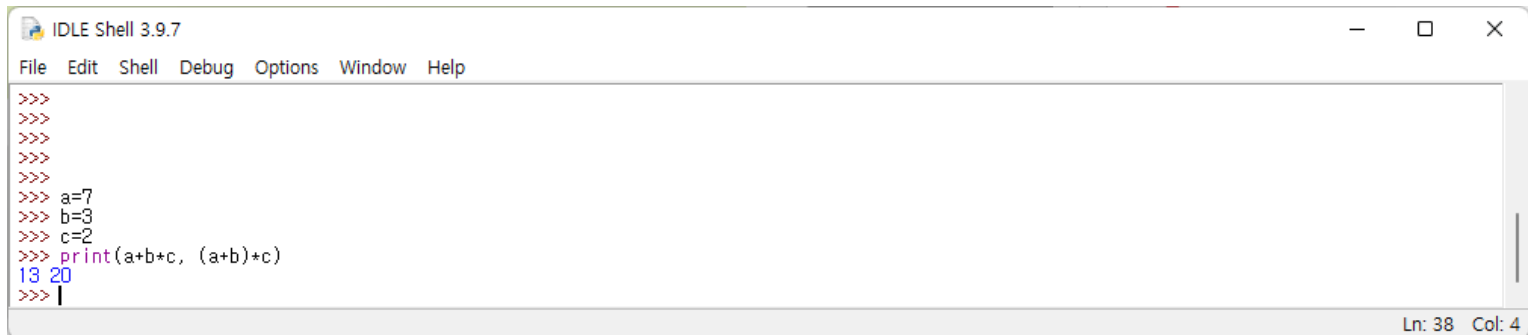
03

산술 연산자 우선순위

연산자가 여러 개일 경우에는 일반적인 사칙연산과 마찬가지로 우선 순위를 잡아서 계산한다.

```
a = 7
b = 3
c = 2

print(a + b * c , (a + b) * c)
```



```
IDLE Shell 3.9.7
File Edit Shell Debug Options Window Help
>>>
>>>
>>>
>>>
>>> a=7
>>> b=3
>>> c=2
>>> print(a+b*c, (a+b)*c)
13 20
>>> |
```

Ln: 38 Col: 4

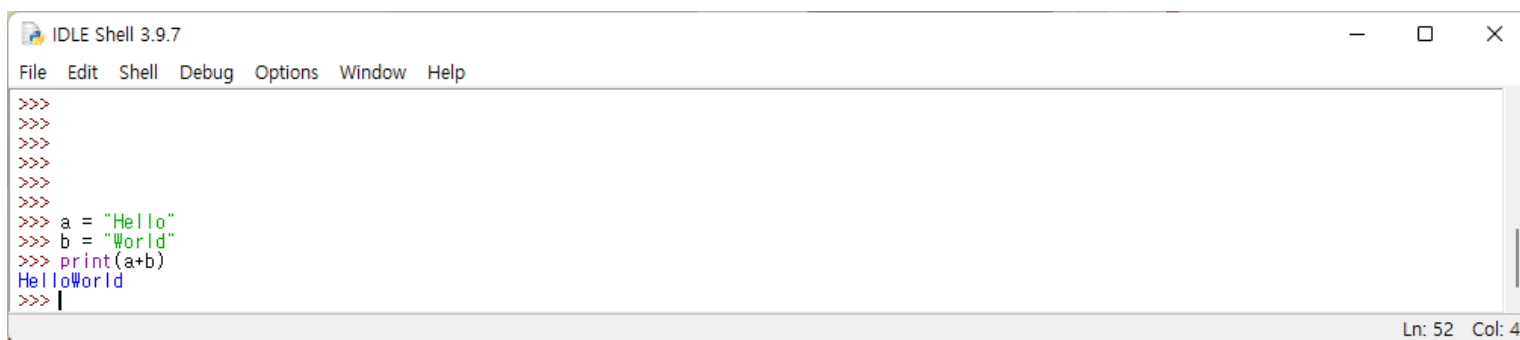
$a+b*c \rightarrow 7+3*2$ 로 $3*2$ 가 먼저 실행이 되고 7이 더해짐으로 13
 $(a+b)*c \rightarrow (7+3)*2$ 로 7과 3이 먼저 더해지고 2가 곱해짐으로 20

03

산술 연산자

데이터의 형태가 숫자가 아닌 문자열에 산술 연산자를 사용하게 되면 어떻게 되는지 확인해보자

```
a = "hello"  
b = "world"  
  
print(a + b)
```



```
IDLE Shell 3.9.7  
File Edit Shell Debug Options Window Help  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>>  
>>> a = "Hello"  
>>> b = "World"  
>>> print(a+b)  
HelloWorld  
>>> |
```

Ln: 52 Col: 4

문자열은 문자끼리 연결이 된다.

03

대입 연산자

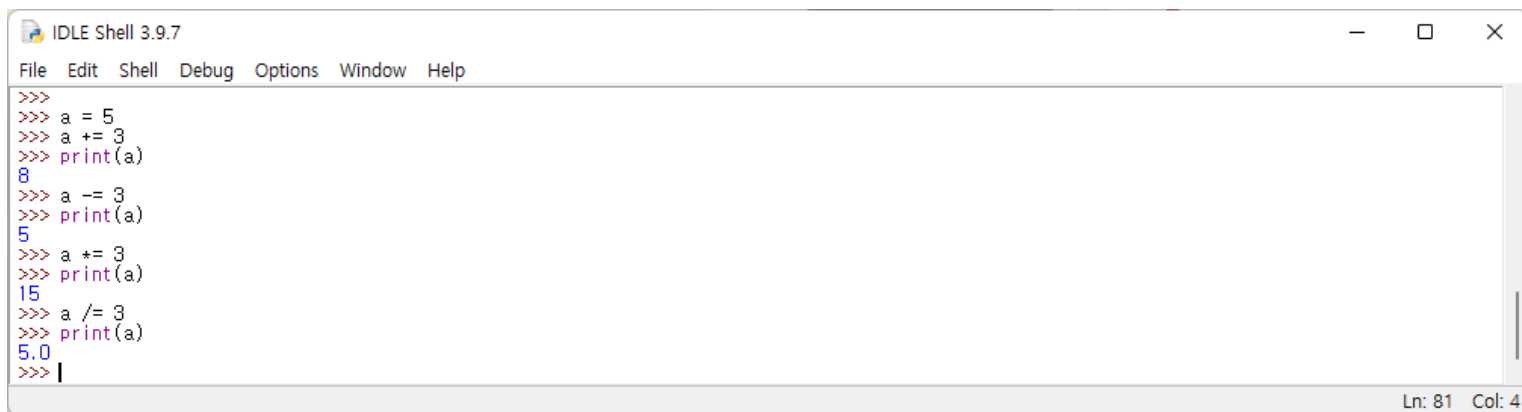
산술 연산자에서 = 이라는 대입 연산자 외에 다른 대입 연산자들이 존재한다.

연산자	예시	설명
+=	a += 3	a = a + 3 과 동일하다
-=	a -= 3	a = a - 3 과 동일하다
*=	a *= 3	a = a * 3 과 동일하다
/=	a /= 3	a = a / 3 과 동일하다
//=	a //= 3	a = a // 3 과 동일하다
%=	a %= 3	a = a % 3 과 동일하다
**=	a **= 3	a = a ** 3 과 동일하다

03

대입 연산자

```
a = 5  
a += 3  
print(a)  
a -= 3  
print(a)  
a *= 3  
print(a)  
a /= 3  
print(a)
```



```
IDLE Shell 3.9.7  
File Edit Shell Debug Options Window Help  
>>>  
>>> a = 5  
>>> a += 3  
>>> print(a)  
8  
>>> a -= 3  
>>> print(a)  
5  
>>> a *= 3  
>>> print(a)  
15  
>>> a /= 3  
>>> print(a)  
5.0  
>>> |  
Ln: 81 Col: 4
```

03

관계 연산자

대소 관계와 상등 관계를 나타내는 연산자로 결과 값은 논리 값으로 나타난다. (논리값은 불형 (True & False))
조건문과 반복문에서 자주 사용된다.

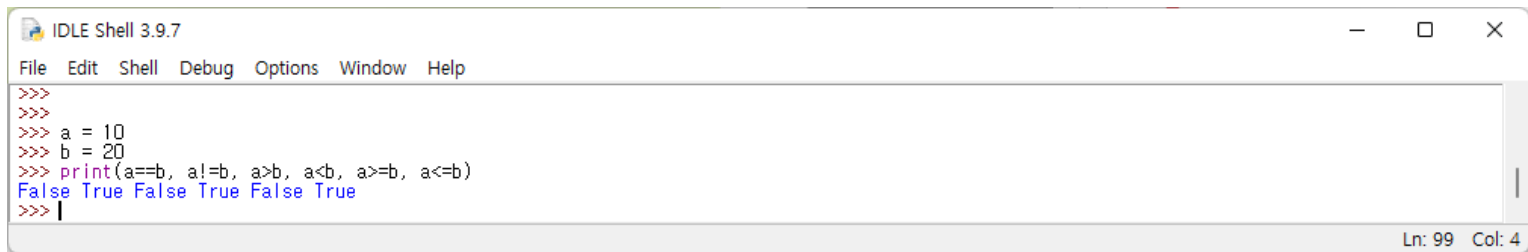
연산자	의미	설명
==	같다	두 값이 동일하면 참(True)
!=	같지 않다	두 값이 동일하지 않으면 참(True)
>	크다	왼쪽의 값이 크다면 참(True)
<	작다	왼쪽의 값이 작다면 참(True)
>=	크거나 같다	왼쪽의 값이 크거나 같으면 참(True)
<=	작거나 같다	왼쪽의 값이 작거나 같으면 참(True)

03

관계 연산자

```
a = 10
b = 20

print(a == b, a != b, a > b, a < b, a >= b, a <= b)
```



```
IDLE Shell 3.9.7
File Edit Shell Debug Options Window Help
>>>
>>>
>>> a = 10
>>> b = 20
>>> print(a==b, a!=b, a>b, a<b, a>=b, a<=b)
False True False True False True
>>> |
```

Ln: 99 Col: 4

03

논리 연산자

논리 값을 판단하는 연산자이다.

연산자	의미	설명
and	~이고, 그리고	두 논리값이 참이어야 참(True)
or	~이거나, 또는	두 논리값 중 하나만 참이어도 참(True)
not	~아니다, 부정	참이면 거짓(False), 거짓이면 참(True)

03

논리 연산자

```
a = 25  
(a > 30) and (a < 50)  
(a > 30) and (a < 50)  
not(a == 30)
```



The screenshot shows the IDLE Shell 3.9.7 window with a menu bar (File, Edit, Shell, Debug, Options, Window, Help) and a command prompt area. The following code is entered and executed:

```
>>>  
>>>  
>>> a=25  
>>> (a>30)and(a<50)  
False  
>>> (a>30)or(a<50)  
True  
>>> not(a==30)  
True  
>>> |
```

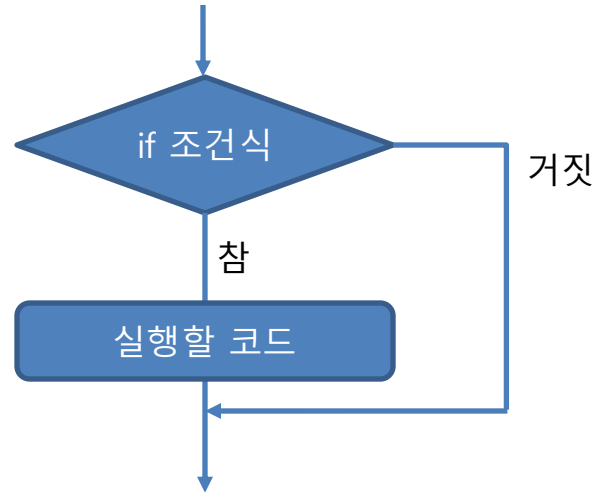
The status bar at the bottom right indicates "Ln: 111 Col: 4".



조건문

04

if문



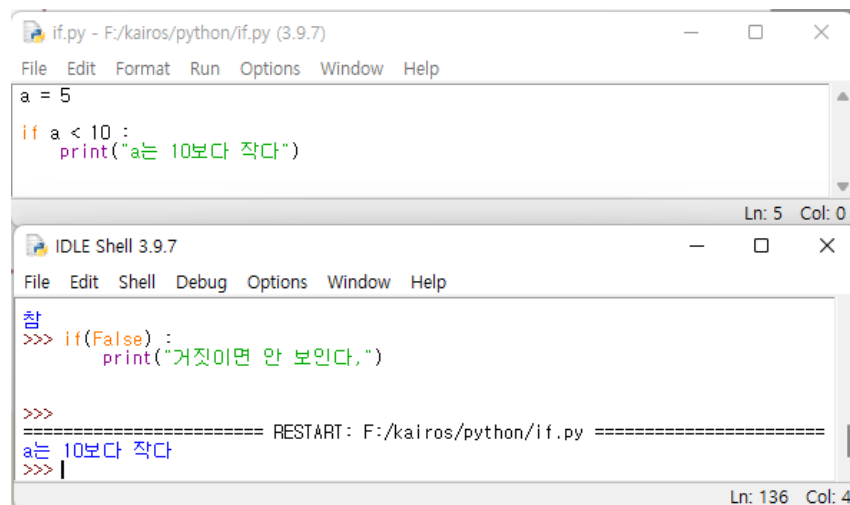
조건문은 if 뒤 조건식이 참이면 실행할 코드를 실행하고, 거짓이라면 코드를 실행하지 않는다. 파이썬에서는 실행할 코드를 들여쓰기를 하여 작성해야 정상적으로 작동을 한다.

04

if문

```
a = 5

if a < 10 :
    print("a는 10보다 작다")
```



The screenshot shows the Python IDLE 3.9.7 environment. The top window, titled 'if.py - F:/kairos/python/if.py (3.9.7)', contains the following code:

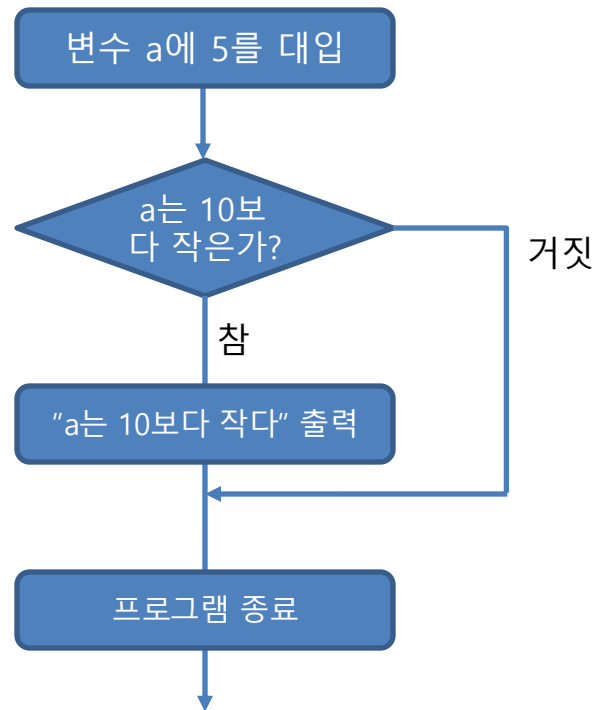
```
a = 5
if a < 10 :
    print("a는 10보다 작다")
```

The bottom window, titled 'IDLE Shell 3.9.7', shows the execution of the script. It displays the prompt '참' (True) and the output 'a는 10보다 작다' (a is less than 10). Below the output, it shows a restart message: '===== RESTART: F:/kairos/python/if.py ====='.

a의 값을 10보다 큰 수로 변경한 뒤 코드를 실행하면 아무 값도 출력
이 되지 않는다.

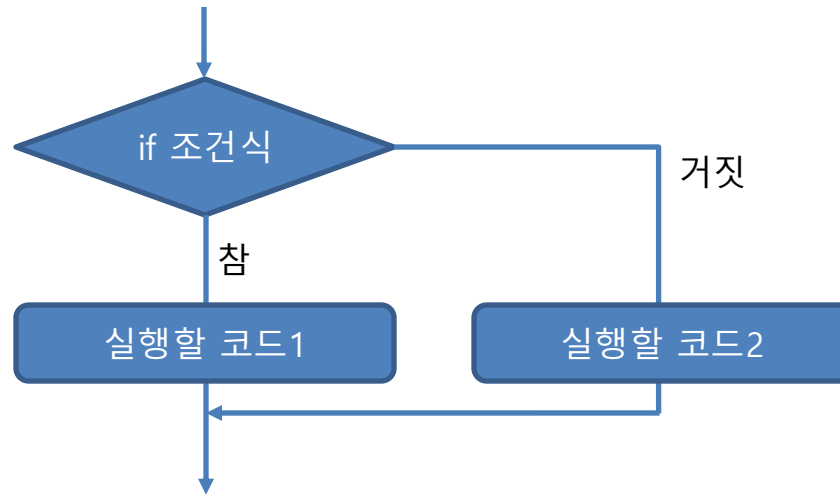
04

if문



04

if~ else문



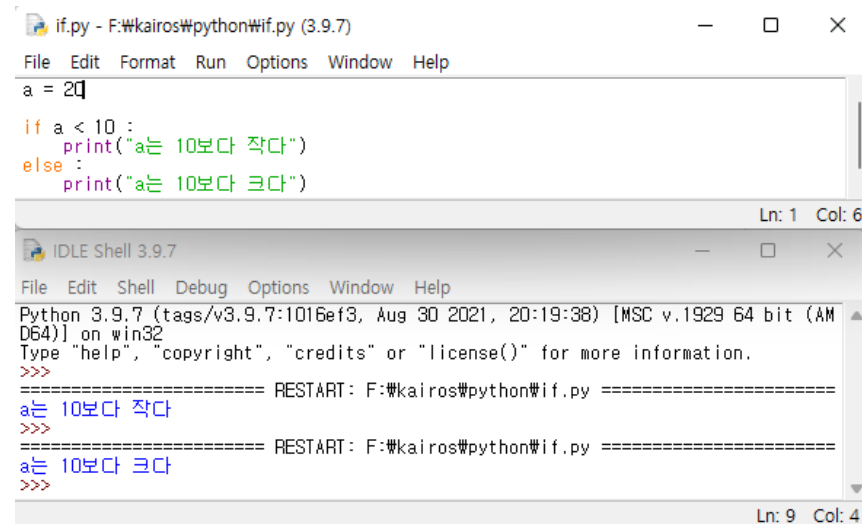
if~ else문은 조건식이 참인 경우 실행할 코드와 거짓인 경우 실행할 코드를 작성한다.

04

if~ else문

```
a = 20

if a < 10 :
    print("a는 10보다 작다")
else :
    print("a는 10보다 크다")
```



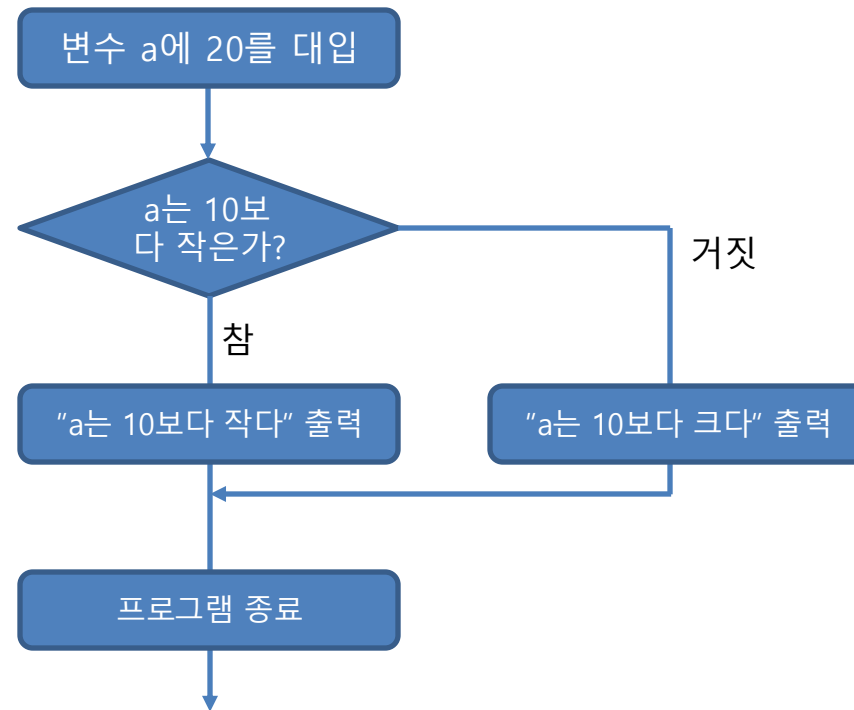
```
if.py - F:\kairos\python\if.py (3.9.7)
File Edit Format Run Options Window Help
a = 20
if a < 10 :
    print("a는 10보다 작다")
else :
    print("a는 10보다 크다")
Ln: 1 Col: 6

IDLE Shell 3.9.7
File Edit Shell Debug Options Window Help
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32
Type "help", "copyright", "credits" or "license()" for more information.
>>>
===== RESTART: F:\kairos\python\if.py =====
a는 10보다 작다
>>>
===== RESTART: F:\kairos\python\if.py =====
a는 10보다 크다
>>>
Ln: 9 Col: 4
```

a의 값이 조건식인 $a < 10$ 에 거짓임으로 else 밑에 있는 코드가 실행이 된다.

04

if~ else문



04

if~ elif~ else문

if~ elif~ else문을 통하여 조건식을 다중으로 사용할 수 있다.

```
score = int(input("점수를 입력하세요 : "))

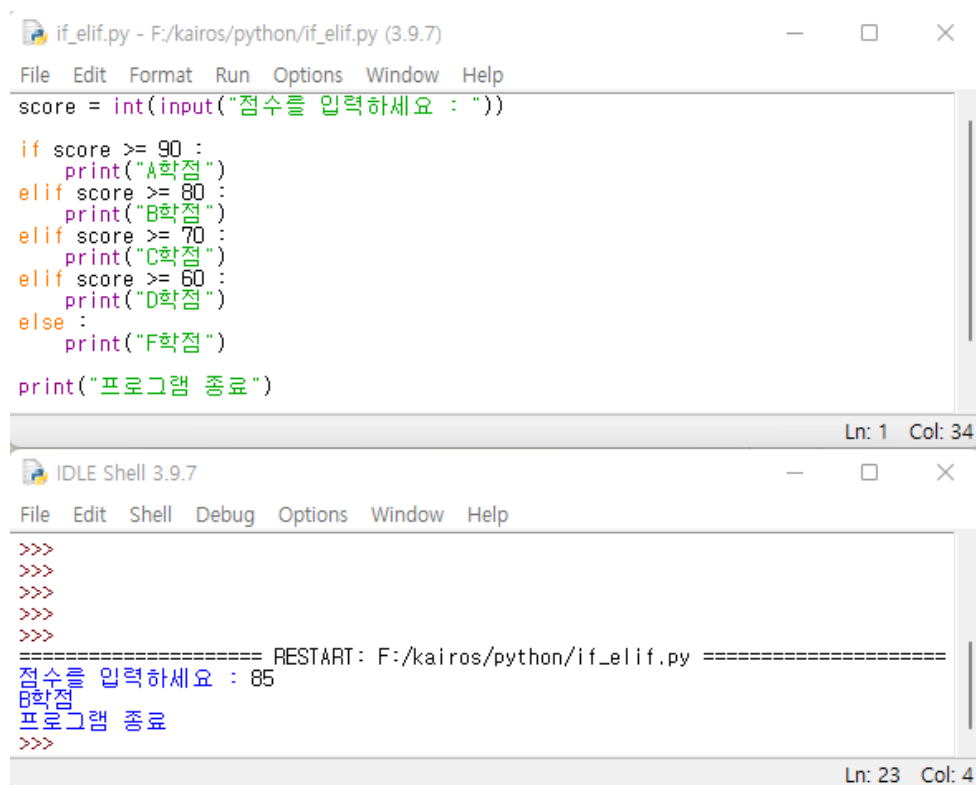
if score >= 90 :
    print("A학점")
elif score >= 80 :
    print("B학점")
elif score >= 70 :
    print("C학점")
elif score >= 60 :
    print("D학점")
else :
    print("F학점")

print("프로그램 종료")
```

04

if~ elif~ else문

if~ elif~ else문을 통하여 조건식을 다중으로 사용할 수 있다.



```
if_elif.py - F:/kairos/python/if_elif.py (3.9.7)
File Edit Format Run Options Window Help
score = int(input("점수를 입력하세요 : "))

if score >= 90 :
    print("A학점")
elif score >= 80 :
    print("B학점")
elif score >= 70 :
    print("C학점")
elif score >= 60 :
    print("D학점")
else :
    print("F학점")

print("프로그램 종료")

Ln: 1 Col: 34

IDLE Shell 3.9.7
File Edit Shell Debug Options Window Help
>>>
>>>
>>>
>>>
>>>
===== RESTART: F:/kairos/python/if_elif.py =====
점수를 입력하세요 : 85
B학점
프로그램 종료
>>>

Ln: 23 Col: 4
```



반복문

05

for문

for문은 지정한만큼 코드를 반복적으로 실행하는 문이다.

기본 형태는

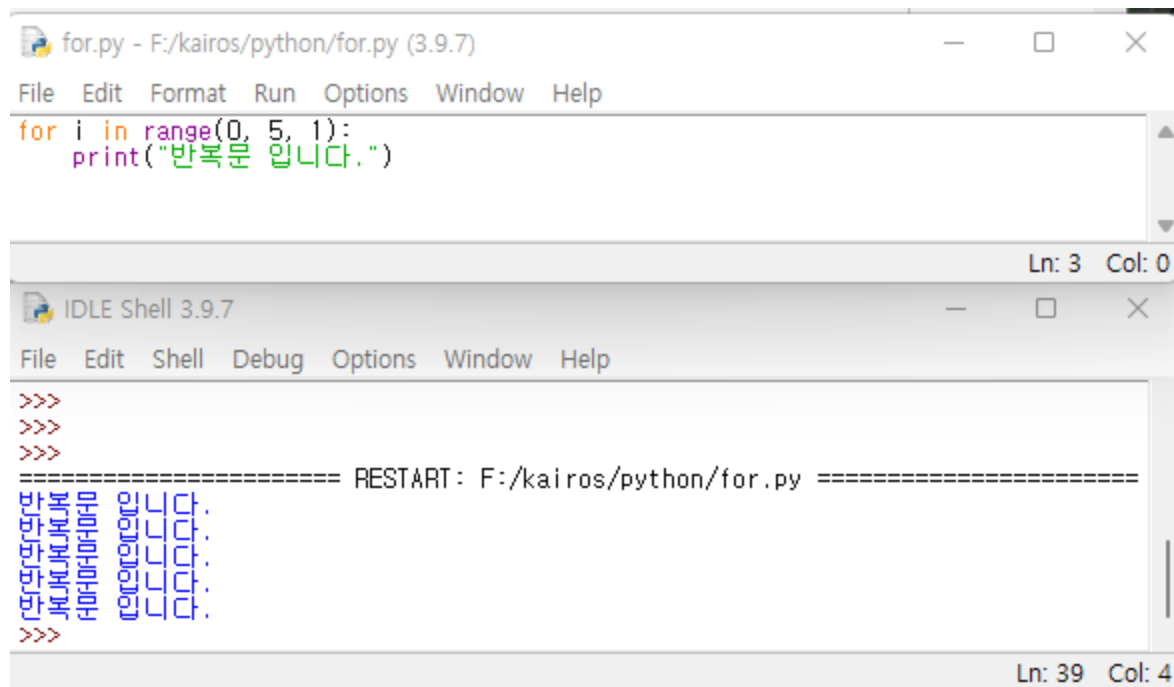
```
for 변수 in range(시작값, 종료값, 증가값) :  
    반복 실행할 코드
```

시작 값을 기준으로 실행이 될 때마다 증가 값만큼 값이 증가하고 값이 종료 값과 같거나 커지는 경우 반복문은 종료된다.

05

for문

```
for i in range(0, 5, 1) :  
    print("반복문 입니다.")
```



The screenshot displays a Python IDE with two windows. The top window, titled 'for.py - F:/kairos/python/for.py (3.9.7)', contains the following code:

```
File Edit Format Run Options Window Help  
for i in range(0, 5, 1):  
    print("반복문 입니다.")
```

The bottom window, titled 'IDLE Shell 3.9.7', shows the execution output after a restart:

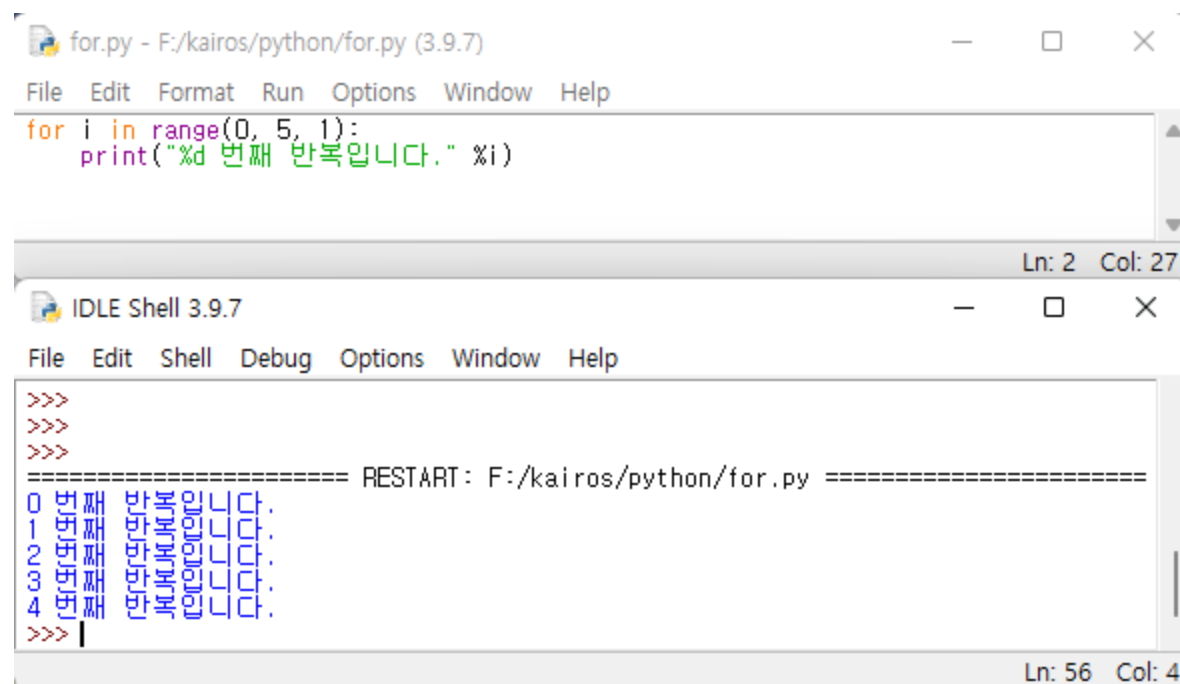
```
File Edit Shell Debug Options Window Help  
>>>  
>>>  
>>>  
===== RESTART: F:/kairos/python/for.py =====  
반복문 입니다.  
반복문 입니다.  
반복문 입니다.  
반복문 입니다.  
반복문 입니다.  
>>>
```

The status bar at the bottom right indicates 'Ln: 39 Col: 4'.

05

for문

```
for i in range(0, 5, 1) :  
    print("%d 번째 반복입니다." %i)
```



The screenshot displays a Python IDE window titled 'for.py - F:/kairos/python/for.py (3.9.7)'. The code editor shows a for loop that iterates from 0 to 4, printing the iteration number. Below the code editor is the 'IDLE Shell 3.9.7' window, which shows the output of the program: five lines of text, each indicating a specific iteration from 0 to 4. The shell window also shows a 'RESTART' message and the file path.

```
for.py - F:/kairos/python/for.py (3.9.7)  
File Edit Format Run Options Window Help  
for i in range(0, 5, 1):  
    print("%d 번째 반복입니다." %i)  
Ln: 2 Col: 27  
IDLE Shell 3.9.7  
File Edit Shell Debug Options Window Help  
>>>  
>>>  
>>>  
===== RESTART: F:/kairos/python/for.py =====  
0 번째 반복입니다.  
1 번째 반복입니다.  
2 번째 반복입니다.  
3 번째 반복입니다.  
4 번째 반복입니다.  
>>> |  
Ln: 56 Col: 4
```

for문의 변수 i는 다른 변수명으로 바뀌서 사용해도 무관하다.

05

for문

숫자 1에서 10까지의 합을 구하는 방법을 생각해보자.

반복문 없이 합을 구하려면 변수 10개를 지정하여 합을 구해야 한다.

for문을 사용하면 간단하게 합을 구할 수 있다.

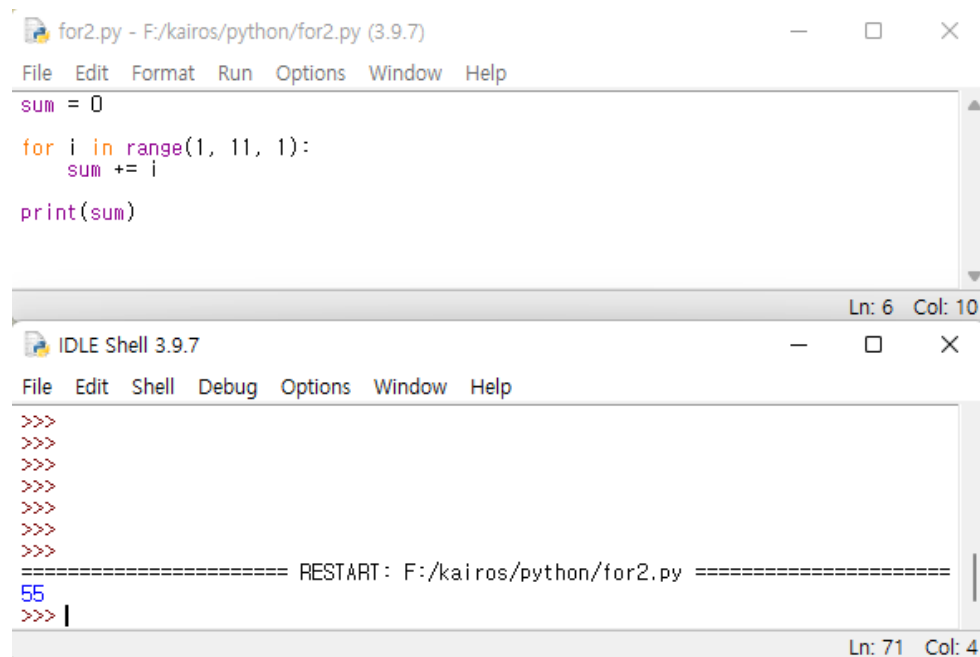
05

for문

```
sum = 0

for i in range(1, 11, 1):
    sum += i

print(sum)
```



The screenshot displays two windows from a Python IDE. The top window, titled 'for2.py - F:/kairos/python/for2.py (3.9.7)', contains the following Python code:

```
File Edit Format Run Options Window Help
sum = 0
for i in range(1, 11, 1):
    sum += i
print(sum)
```

The bottom window, titled 'IDLE Shell 3.9.7', shows the execution of the script. It features a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The shell contains several prompt lines ('>>>') and a restart message: '===== RESTART: F:/kairos/python/for2.py ====='. The output of the script, '55', is displayed on the line following the restart message. The status bar at the bottom indicates 'Ln: 71 Col: 4'.

05

다중 for문

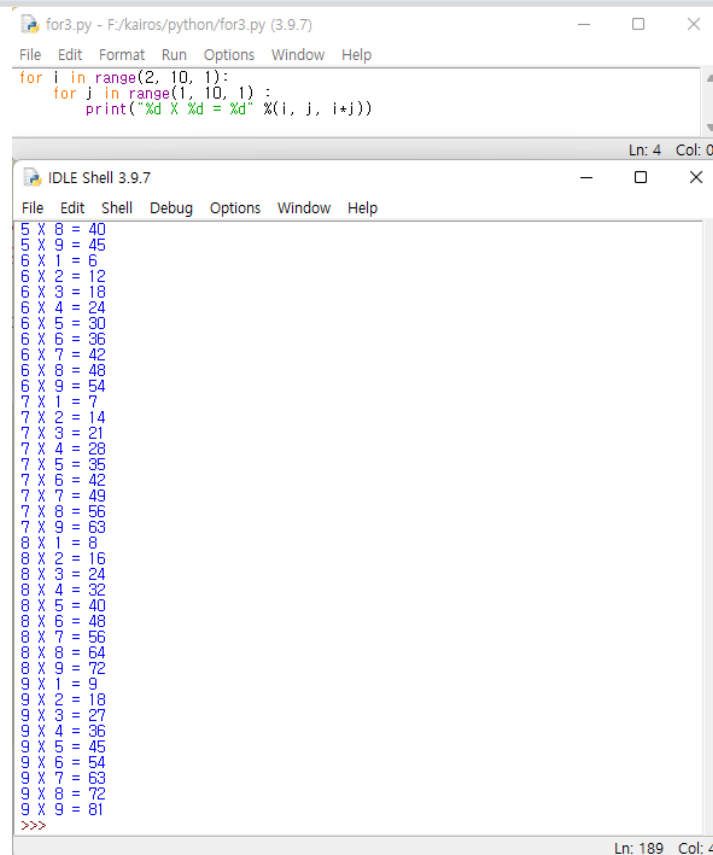
다중 for문은 for문 안에 또 하나의 for문을 사용하여 중복문을 2번 돌리는 방법을 말한다.

```
for 변수1 in range(시작 값, 종료 값, 증가 값) :  
    for 변수2 in range(시작 값, 종료 값, 증가 값) :  
        반복 실행할 코드
```

05

다중 for문

```
for i in range(2, 10, 1):  
    for j in range(1, 10, 1):  
        print("%d X %d = %d" %(i, j, i*j))
```



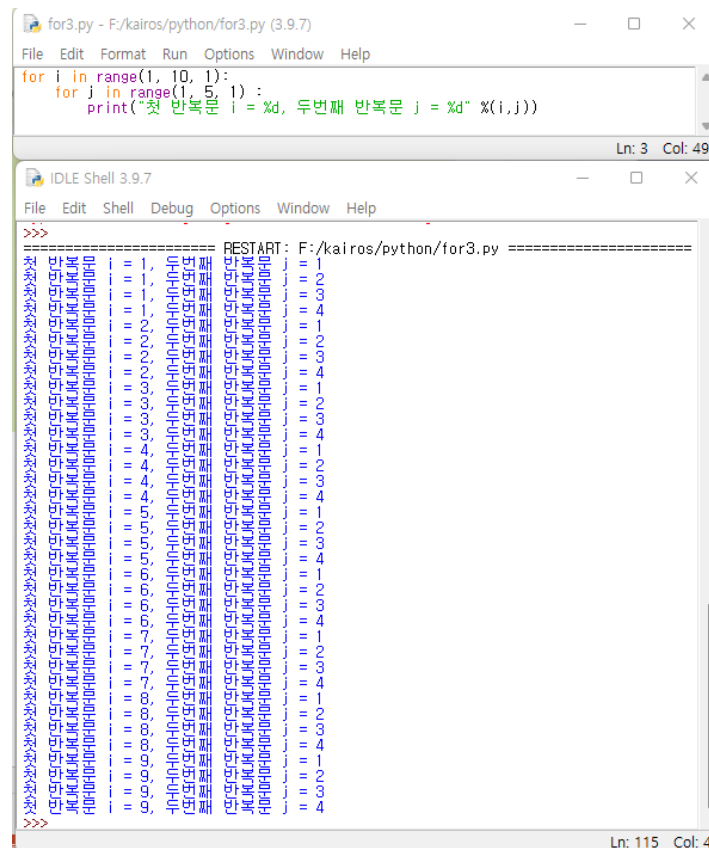
The screenshot shows a Python IDLE 3.9.7 window with a file named 'for3.py'. The code in the editor is a nested for loop that prints multiplication tables for numbers 2 through 9. The output window shows the results of the program, displaying each multiplication from 2x1 to 9x9 on a new line.

```
for3.py - F:/kairos/python/for3.py (3.9.7)  
File Edit Format Run Options Window Help  
for i in range(2, 10, 1):  
    for j in range(1, 10, 1):  
        print("%d X %d = %d" %(i, j, i*j))  
Ln: 4 Col: 0  
IDLE Shell 3.9.7  
File Edit Shell Debug Options Window Help  
5 X 1 = 5  
5 X 2 = 10  
5 X 3 = 15  
5 X 4 = 20  
5 X 5 = 25  
5 X 6 = 30  
5 X 7 = 35  
5 X 8 = 40  
5 X 9 = 45  
6 X 1 = 6  
6 X 2 = 12  
6 X 3 = 18  
6 X 4 = 24  
6 X 5 = 30  
6 X 6 = 36  
6 X 7 = 42  
6 X 8 = 48  
6 X 9 = 54  
7 X 1 = 7  
7 X 2 = 14  
7 X 3 = 21  
7 X 4 = 28  
7 X 5 = 35  
7 X 6 = 42  
7 X 7 = 49  
7 X 8 = 56  
7 X 9 = 63  
8 X 1 = 8  
8 X 2 = 16  
8 X 3 = 24  
8 X 4 = 32  
8 X 5 = 40  
8 X 6 = 48  
8 X 7 = 56  
8 X 8 = 64  
8 X 9 = 72  
9 X 1 = 9  
9 X 2 = 18  
9 X 3 = 27  
9 X 4 = 36  
9 X 5 = 45  
9 X 6 = 54  
9 X 7 = 63  
9 X 8 = 72  
9 X 9 = 81  
>>>  
Ln: 189 Col: 4
```

05

다중 for문

```
for i in range(1, 10, 1):  
    for j in range(1, 5, 1):  
        print("첫 반복문 i = %d , 두번째 반복문 j = %d" %(i, j))
```



The screenshot shows a Python IDE window titled 'for3.py - F:/kairos/python/for3.py (3.9.7)'. The code editor contains the following Python code:

```
for i in range(1, 10, 1):  
    for j in range(1, 5, 1):  
        print("첫 반복문 i = %d , 두번째 반복문 j = %d" %(i, j))
```

The output window, titled 'IDLE Shell 3.9.7', shows the execution results. It starts with a 'RESTART' message and displays the output of the nested loops, which is a 10x5 grid of strings: '첫 반복문 i = 1 , 두번째 반복문 j = 1' through '첫 반복문 i = 10 , 두번째 반복문 j = 5'. The status bar at the bottom indicates 'Ln: 115 Col: 4'.

05

while문

반복문에는 for문 외에 while문이 있다.

```
변수 = 시작 값  
while 변수 < 종료 값 :  
    반복 실행할 코드  
    변수 = 변수 + 증가 값
```

while 뒤에 있는 조건식이 거짓이 되기 전까지 반복한다.

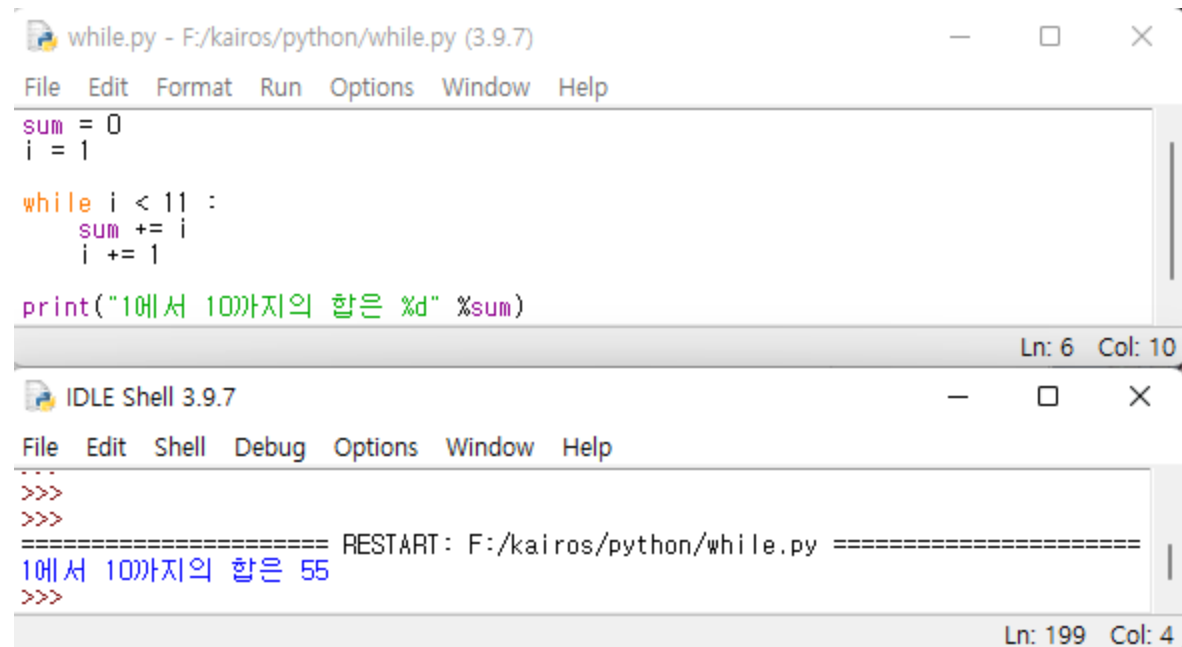
05

while문

```
sum = 0
i = 1

while i < 11 :
    sum += i
    i += 1

print("1에서 10까지의 합은 %d" %sum)
```



The screenshot displays two windows from a Python IDE. The top window, titled 'while.py - F:/kairos/python/while.py (3.9.7)', contains the following code:

```
sum = 0
i = 1

while i < 11 :
    sum += i
    i += 1

print("1에서 10까지의 합은 %d" %sum)
```

The bottom window, titled 'IDLE Shell 3.9.7', shows the execution output. It starts with two empty prompt lines (>>>). Then, it displays a restart message: 'RESTART: F:/kairos/python/while.py'. Finally, it prints the result: '1에서 10까지의 합은 55'.

05

while문

```
while True :  
    print("반복")
```

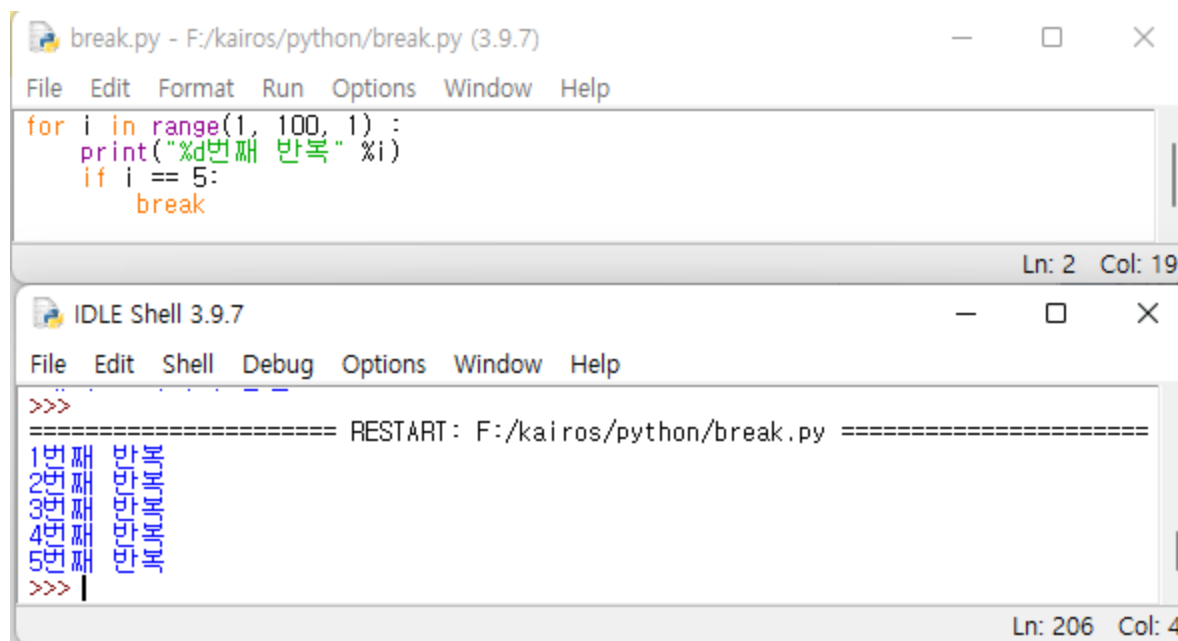
while문에서 조심해야할 점은 조건식 부분에 True가 들어가게 되면 무한 반복이 된다.

05

break문

반복문에서 반복이 되는 도중에 빠져 나가는 break문이 있다.

```
for i in range(1, 100, 1) :  
    print("%d번째 반복" %i)  
    if i == 5 :  
        break
```



The screenshot displays the IDLE 3.9.7 environment. The top window, titled 'break.py - F:/kairos/python/break.py (3.9.7)', contains the following Python code:

```
for i in range(1, 100, 1) :  
    print("%d번째 반복" %i)  
    if i == 5 :  
        break
```

The bottom window, titled 'IDLE Shell 3.9.7', shows the execution output. It begins with a restart message: '===== RESTART: F:/kairos/python/break.py ====='. The output consists of five lines of text, each representing a loop iteration:

```
1번째 반복  
2번째 반복  
3번째 반복  
4번째 반복  
5번째 반복
```

The shell prompt '>>>' is visible at the bottom of the output, indicating the program has finished execution.

05

break문

```
sum = 0
i = 0

for i in range(1, 100, 1) :
    sum += i
    if sum >= 500 :
        break

print("합계가 500 이상이 되는 범위의 숫자는 : %d" %i)
```

The screenshot shows a Python IDE window titled 'break2.py - F:/kairos/python/break2.py (3.9.7)'. The code in the editor is identical to the one in the previous block. Below the editor is the 'IDLE Shell 3.9.7' window. It shows the execution of the script with several empty prompt lines ('>>>') followed by a 'RESTART' message and the output: '합계가 500 이상이 되는 범위의 숫자는 : 32'. The status bar at the bottom of the shell window indicates 'Ln: 220 Col: 4'.

```
break2.py - F:/kairos/python/break2.py (3.9.7)
File Edit Format Run Options Window Help

sum = 0
i = 0

for i in range(1, 100, 1) :
    sum += i
    if sum >= 500 :
        break

print("합계가 500 이상이 되는 범위의 숫자는 : %d" %i)

Ln: 4 Col: 21

IDLE Shell 3.9.7
File Edit Shell Debug Options Window Help

>>>
>>>
>>>
>>>
>>>
>>>
===== RESTART: F:/kairos/python/break2.py =====
합계가 500 이상이 되는 범위의 숫자는 : 32
>>>

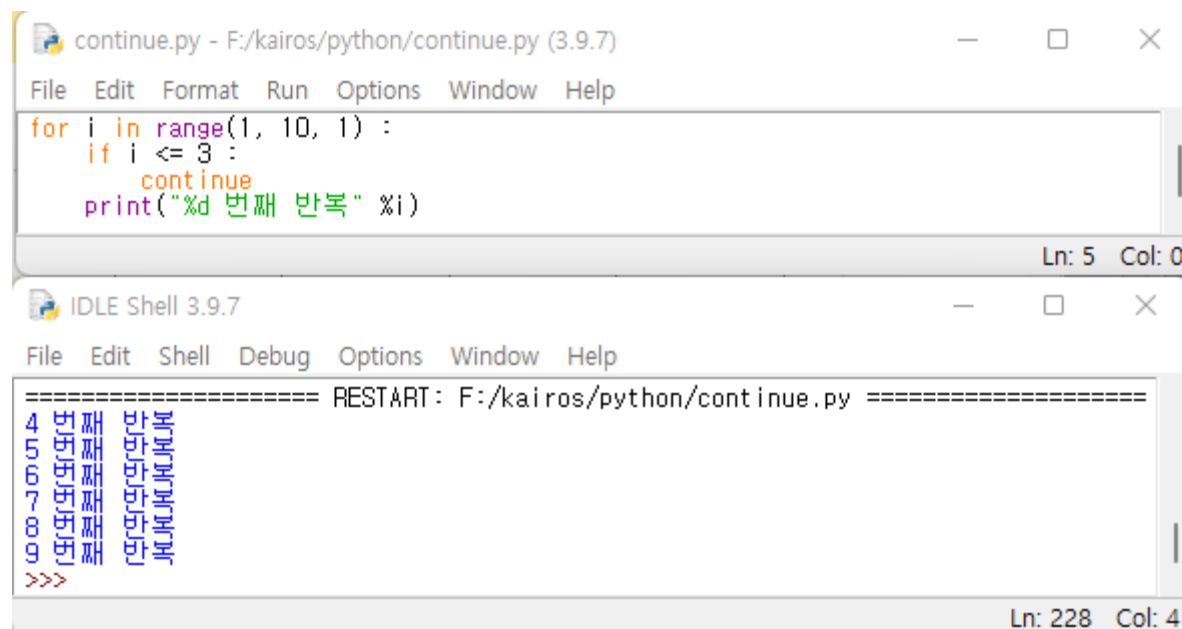
Ln: 220 Col: 4
```

05

continue문

break문이 반복문에서 빠져 나간다면 continue문은 반복문으로 돌아간다.

```
for i in range(1, 10, 1):  
    if i <= 3:  
        continue  
    print("%d 번째 반복" %i)
```



The screenshot shows a Python IDE window titled 'continue.py - F:/kairos/python/continue.py (3.9.7)'. The code in the editor is:

```
for i in range(1, 10, 1):  
    if i <= 3:  
        continue  
    print("%d 번째 반복" %i)
```

Below the editor is the 'IDLE Shell 3.9.7' window. It shows the output of the script, which is a series of vertical bars representing the first three iterations being skipped and the next seven being printed. The output is:

```
===== RESTART: F:/kairos/python/continue.py =====  
4  
5  
6  
7  
8  
9  
>>>
```

The status bar at the bottom of the shell window indicates 'Ln: 228 Col: 4'.

튜플, 리스트,
딕셔너리

06

튜플

앞의 변수 부분에서는 자료형들이 값을 하나만 넣고 저장을 했지만 튜플, 리스트, 딕셔너리는 여러 데이터를 저장할 수 있다.

이 중 튜플은 가장 단순한 자료형으로 소괄호(())로 데이터를 감싸서 저장한다.

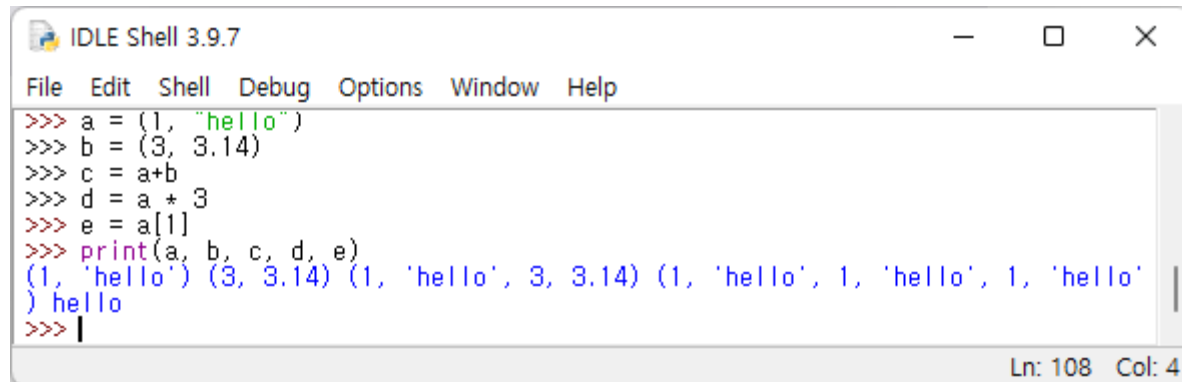
튜플은 한번 생성 되면 내부의 원소를 삭제하거나 수정이 불가능하다.

06

튜플

```
a = (1, "hello")
b = (3, 3.14)
c = a + b
d = a * 3
e = a[1]

print(a, b, c, d, e)
```



```
IDLE Shell 3.9.7
File Edit Shell Debug Options Window Help
>>> a = (1, "hello")
>>> b = (3, 3.14)
>>> c = a+b
>>> d = a * 3
>>> e = a[1]
>>> print(a, b, c, d, e)
(1, 'hello') (3, 3.14) (1, 'hello', 3, 3.14) (1, 'hello', 1, 'hello', 1, 'hello')
>>> |
```

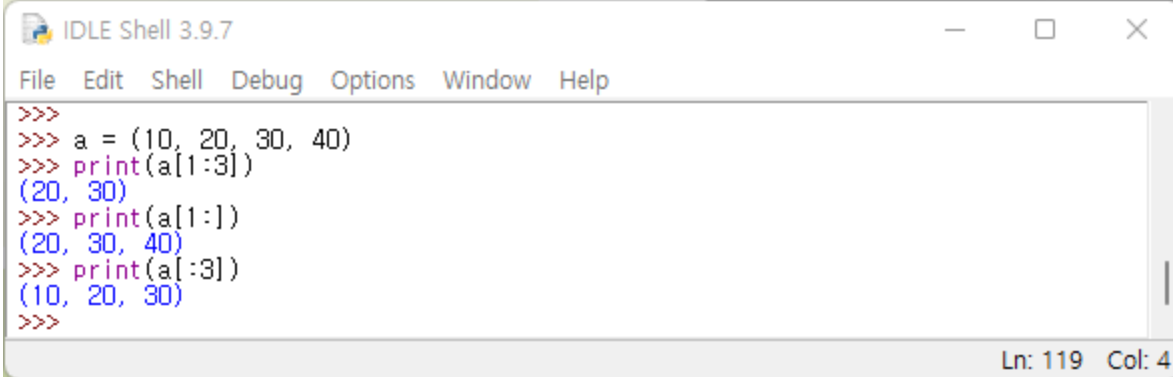
Ln: 108 Col: 4

06

튜플

```
a = (10, 20, 30, 40)
```

```
print(a[1 : 3])  
print(a[1 : ])  
print(a[ : 3])
```

A screenshot of the IDLE Shell 3.9.7 window. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area shows the following Python code and its output:

```
>>>  
>>> a = (10, 20, 30, 40)  
>>> print(a[1:3])  
(20, 30)  
>>> print(a[1:])  
(20, 30, 40)  
>>> print(a[:3])  
(10, 20, 30)  
>>>
```

The status bar at the bottom right indicates 'Ln: 119 Col: 4'.

리스트

리스트는 원소들이 연속적으로 저장되는 형태이다.
리스트는 대괄호([])안에 요소들을 저장하며 요소의 개수는 0개 이상이면 가능하다.

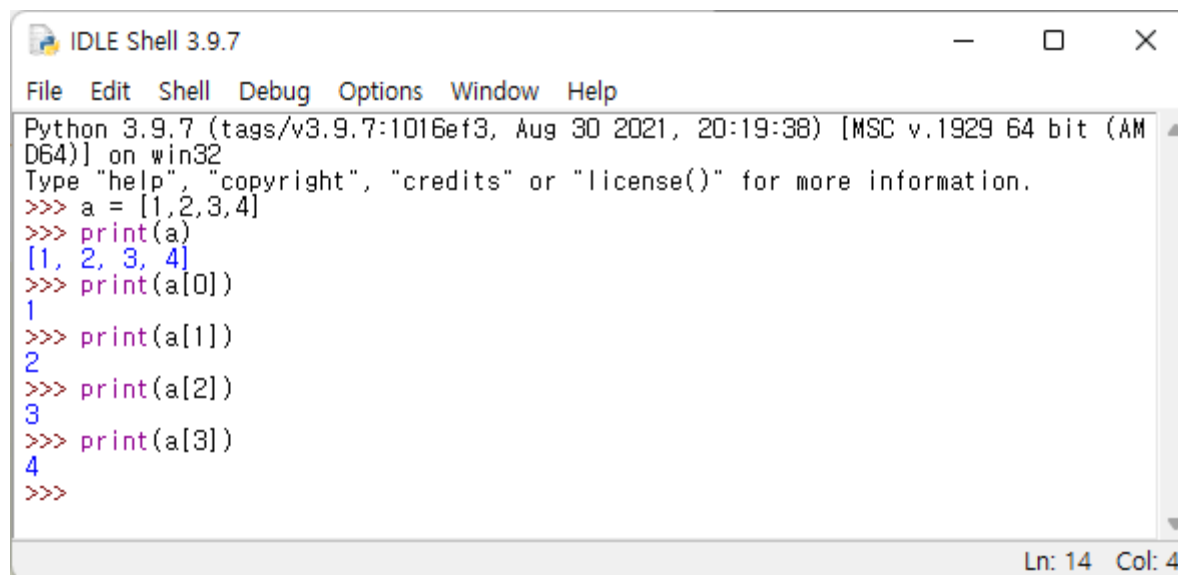
리스트명 = [요소1, 요소2, 요소3, ...]

06

리스트

```
a = [1, 2, 3, 4]
```

```
print(a)  
print(a[0])  
print(a[1])  
print(a[2])
```



```
IDLE Shell 3.9.7  
File Edit Shell Debug Options Window Help  
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>> a = [1,2,3,4]  
>>> print(a)  
[1, 2, 3, 4]  
>>> print(a[0])  
1  
>>> print(a[1])  
2  
>>> print(a[2])  
3  
>>> print(a[3])  
4  
>>>  
Ln: 14 Col: 4
```

06

리스트 내부 함수

함수	설명	코드
append	리스트 가장 뒤에 항목을 추가한다.	리스트명.append(값)
pop	리스트 맨 뒤 항목을 삭제한다.	리스트명.pop()
sort	리스트 항목을 오름차순으로 정렬한다.	리스트명.sort()
reverse	리스트 항목을 역순으로 정렬한다.	리스트명.reverse()
index	지정한 값의 위치를 반환한다.	리스트명.index(값)
insert	지정된 위치에 값을 삽입한다.	리스트명.insert(위치, 값)
remove	리스트에서 지정한 값을 삭제한다.	리스트명.remove(값)
extend	리스트 뒤에 새로운 리스트를 삽입한다.	리스트명.extend(추가할 리스트)
count	리스트의 해당 값의 개수를 출력한다.	리스트명.count(값)
clear	리스트의 내용을 모두 삭제한다.	리스트명.clear()
del	리스트에서 해당 위치의 항목을 삭제한다.	del 리스트명[위치]
len	리스트의 전체 개수를 출력한다.	len(리스트명)
copy	리스트의 내용을 새로운 리스트에 복사한다.	새 리스트 = 리스트명.copy()
sorted	리스트 항목을 정렬하여 새로운 리스트에 대입한다.	새 리스트 = sorted(리스트명)

리스트 내부함수

```
a = [20, 10, 40, 30]  
b = [12, 99, 24, 55]
```

```
a.sort()  
b.reverse()
```

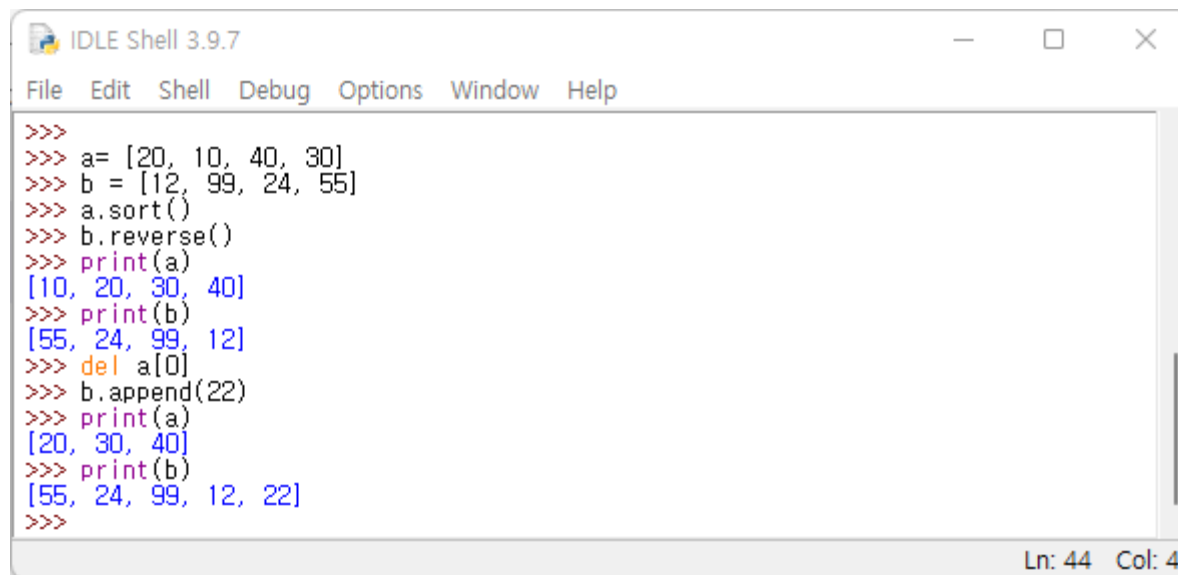
```
print(a)  
print(b)
```

```
del a[0]  
b.append(22)
```

```
print(a)  
print(b)
```

06

리스트 내부함수

A screenshot of the IDLE Shell 3.9.7 window. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area contains the following Python code:

```
>>>  
>>> a= [20, 10, 40, 30]  
>>> b = [12, 99, 24, 55]  
>>> a.sort()  
>>> b.reverse()  
>>> print(a)  
[10, 20, 30, 40]  
>>> print(b)  
[55, 24, 99, 12]  
>>> del a[0]  
>>> b.append(22)  
>>> print(a)  
[20, 30, 40]  
>>> print(b)  
[55, 24, 99, 12, 22]  
>>>
```

The status bar at the bottom right shows 'Ln: 44 Col: 4'.

06

2차원 리스트

리스트의 원소들에는 또 다른 리스트를 포함할 수 있다.

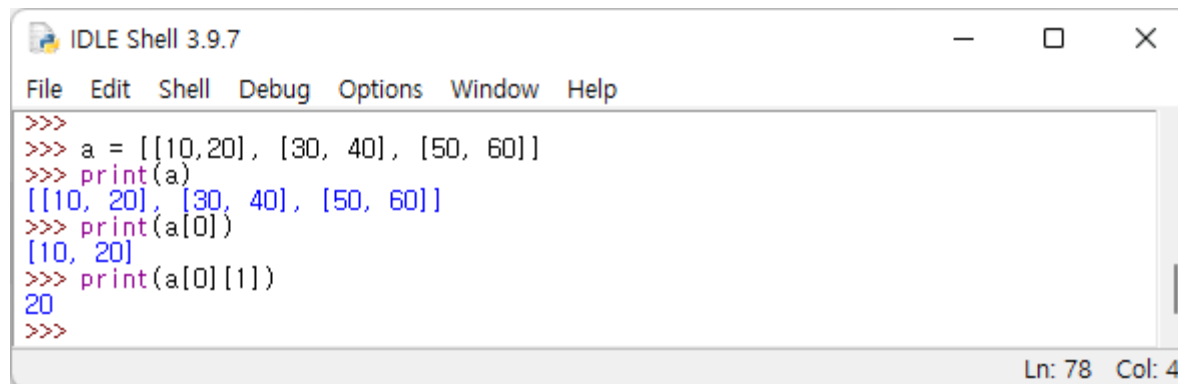
리스트명 = [[요소1, 요소2], [요소3, 요소4], [요소5, 요소6]]

```
a = [[10,20], [30, 40], [50, 60]]
```

```
print(a)
```

```
print(a[0])
```

```
print(a[0][1])
```



```
IDLE Shell 3.9.7
File Edit Shell Debug Options Window Help
>>>
>>> a = [[10,20], [30, 40], [50, 60]]
>>> print(a)
[[10, 20], [30, 40], [50, 60]]
>>> print(a[0])
[10, 20]
>>> print(a[0][1])
20
>>>
```

Ln: 78 Col: 4

06

딕셔너리

딕셔너리란 사전형 데이터를 의미하며, key와 value를 1대1로 대응시킨 형태이다. 하나의 key에는 하나의 value만이 대응된다.

key 값은 절대로 변하지 않으며 value 값은 변경할 수 있다.

튜플과 다르게 key-value 쌍 자체를 수정하거나 삭제할 수 있기 때문에 유용하게 사용할 수 있다.

딕셔너리명 = {key1 : value1, key2 : value2, key3 : value3}

06

딕셔너리

```
a = {"name" : "test", "age" : 20, "phone" : "01012345678"}

print(a)
print(a["name"])

a["area"] = "Seoul"

print(a)
```



```
IDLE Shell 3.9.7
File Edit Shell Debug Options Window Help
>>> a = {"name" : "test", "age" : 20, "phone" : "01012345678"}
>>> print(a)
{'name': 'test', 'age': 20, 'phone': '01012345678'}
>>> print(a["name"])
test
>>> a["area"] = "Seoul"
>>> print(a)
{'name': 'test', 'age': 20, 'phone': '01012345678', 'area': 'Seoul'}
>>> |
```

Ln: 145 Col: 4

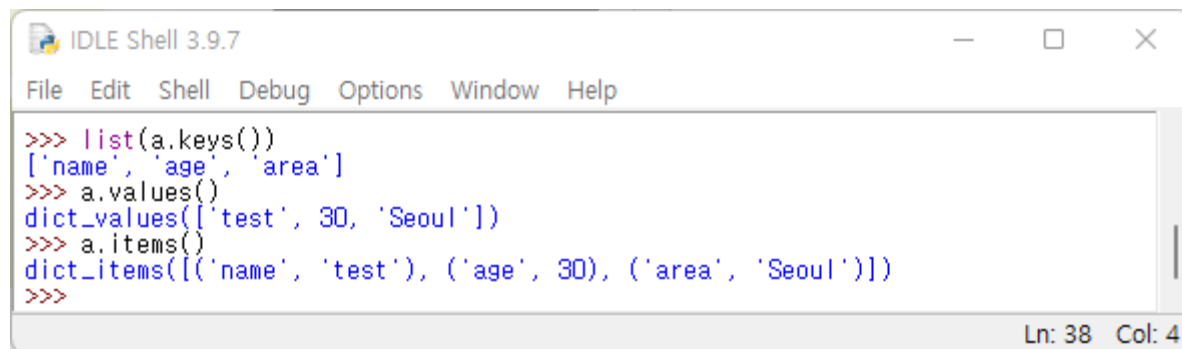
06

딕셔너리

```
list(a.keys())
```

```
a.values()
```

```
a.items()
```

A screenshot of the IDLE Shell 3.9.7 window. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The shell area shows the following code and output:

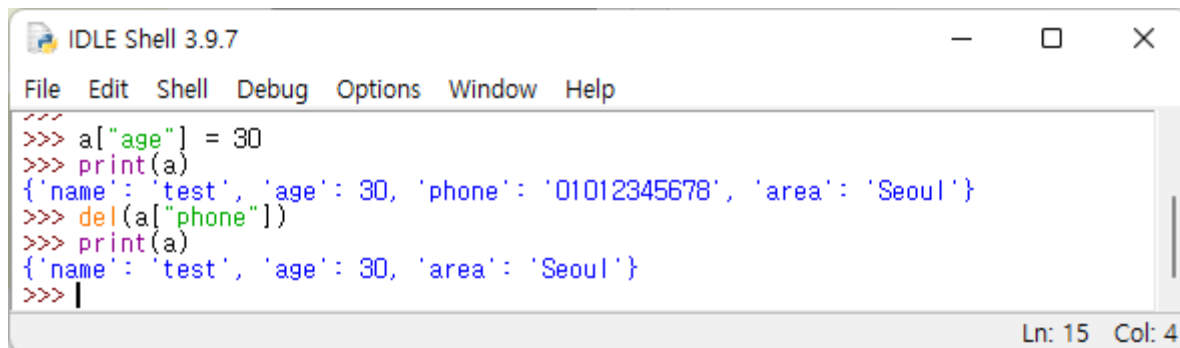
```
>>> list(a.keys())  
['name', 'age', 'area']  
>>> a.values()  
dict_values(['test', 30, 'Seoul'])  
>>> a.items()  
dict_items([('name', 'test'), ('age', 30), ('area', 'Seoul')])  
>>>
```

The status bar at the bottom right indicates 'Ln: 38 Col: 4'.

06

딕셔너리

```
a["age"] = 30  
print(a)  
  
del(a["phone"])  
print(a)
```



```
IDLE Shell 3.9.7  
File Edit Shell Debug Options Window Help  
>>> a["age"] = 30  
>>> print(a)  
{'name': 'test', 'age': 30, 'phone': '01012345678', 'area': 'Seoul'}  
>>> del(a["phone"])  
>>> print(a)  
{'name': 'test', 'age': 30, 'area': 'Seoul'}  
>>> |
```

Ln: 15 Col: 4

문자열 함수

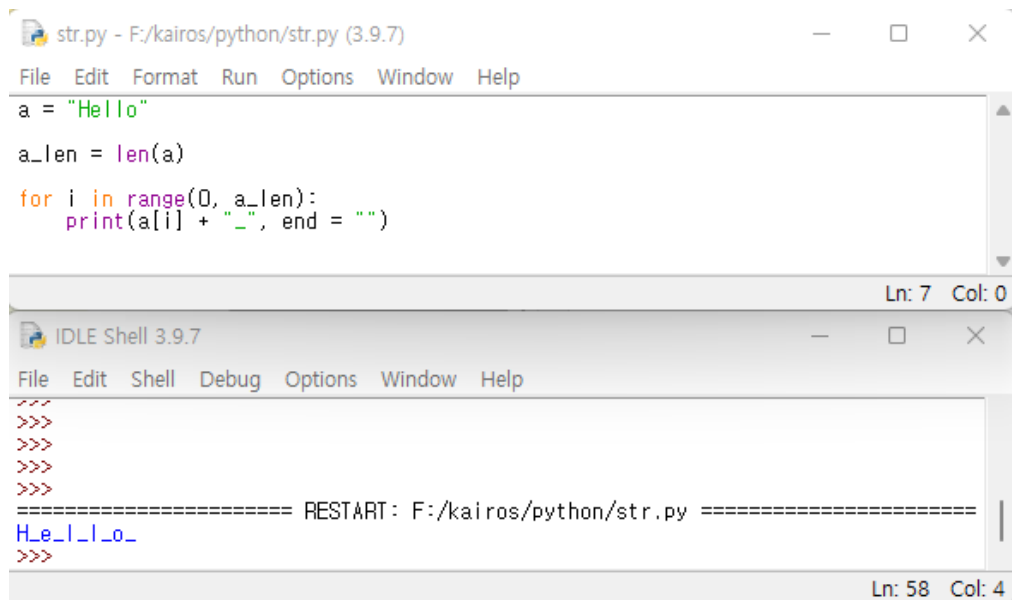
문자열 함수

len() : 문자열의 길이를 출력

```
a = "Hello"

a_len = len(a)

for i in range(0, a_len):
    print(a[i] + "_", end="")
```



The screenshot shows a Python IDE window titled 'str.py - F:/kairos/python/str.py (3.9.7)'. The code in the editor is:

```
a = "Hello"
a_len = len(a)
for i in range(0, a_len):
    print(a[i] + "_", end="")
```

Below the editor is the 'IDLE Shell 3.9.7' window. It shows the output of the script: 'H_e_l_l_o_'. The shell also displays a restart message: '===== RESTART: F:/kairos/python/str.py ====='.

문자열 함수

upper() : 문자를 모두 대문자로 변환

lower() : 문자를 모두 소문자로 변환

swapcase() : 대문자를 소문자로, 소문자를 대문자로 변환

title() : 단어의 첫 문자만 대문자로 변환

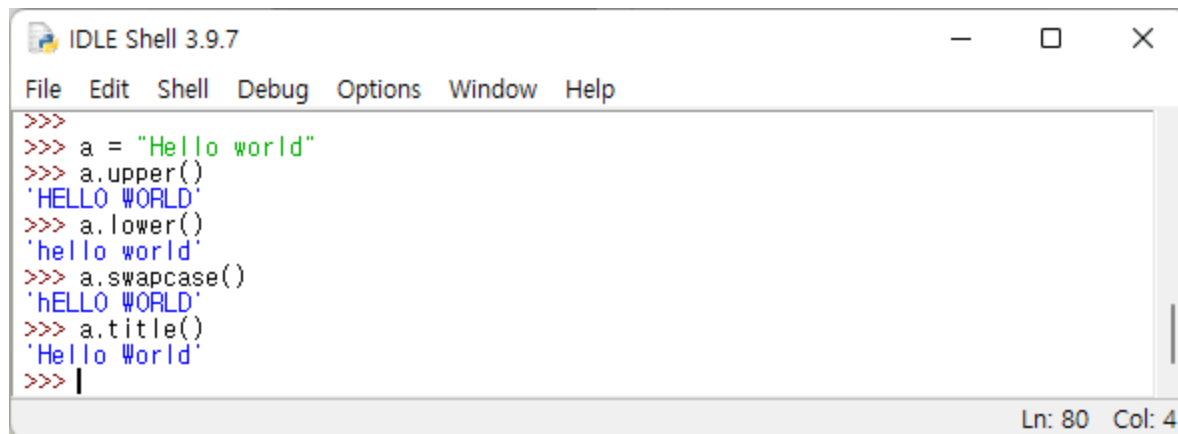
```
a = "Hello world"
```

```
a.upper()
```

```
a.lower()
```

```
a.swapcase()
```

```
a.title()
```

A screenshot of the IDLE Shell 3.9.7 window. The window has a menu bar with 'File', 'Edit', 'Shell', 'Debug', 'Options', 'Window', and 'Help'. The main text area shows the following Python code and its output:

```
>>>
>>> a = "Hello world"
>>> a.upper()
'HELLO WORLD'
>>> a.lower()
'hello world'
>>> a.swapcase()
'hELLO wORLD'
>>> a.title()
'Hello World'
>>> |
```

The status bar at the bottom right indicates 'Ln: 80 Col: 4'.

문자열 함수

count() : 문자열의 개수 출력

find() : 문자열 앞부터 시작하여 처음으로 나온 위치를 출력

index() : find와 같이 위치를 출력하지만 찾는 단어가 없을 시 에러 발생

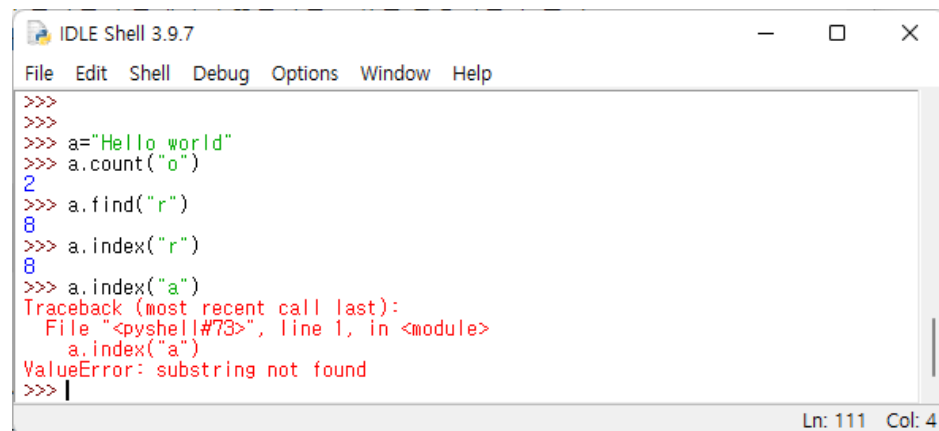
```
a = "Hello world"
```

```
a.count("o")
```

```
a.find("r")
```

```
a.index("r")
```

```
a.index("a")
```



```
IDLE Shell 3.9.7
File Edit Shell Debug Options Window Help
>>>
>>>
>>> a="Hello world"
>>> a.count("o")
2
>>> a.find("r")
8
>>> a.index("r")
8
>>> a.index("a")
Traceback (most recent call last):
  File "<pyshell#73>", line 1, in <module>
    a.index("a")
ValueError: substring not found
>>> |
```

Ln: 111 Col: 4

07

문자열 함수

lstrip() : 문자열 왼쪽의 공백을 지운다.

rstrip() : 문자열 오른쪽의 공백을 지운다.

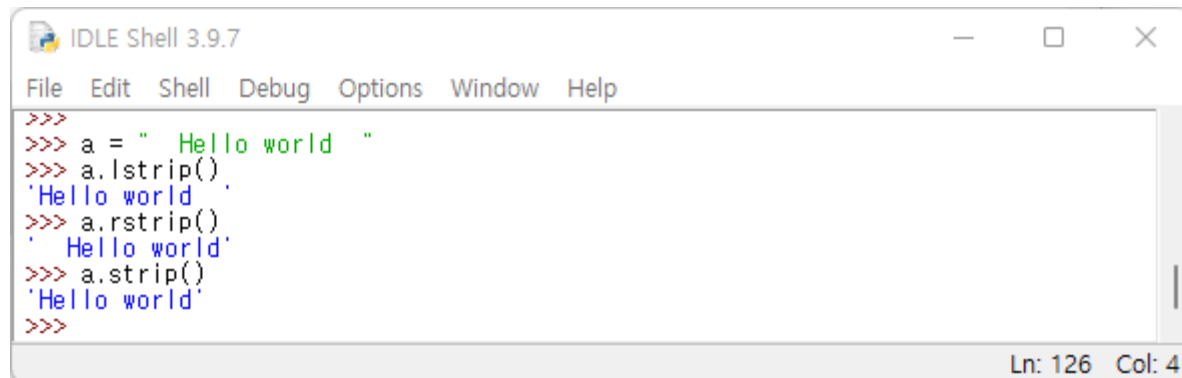
strip() : 문자열 양쪽의 공백을 지운다.

```
a = " Hello world "
```

```
a.lstrip()
```

```
a.rstrip()
```

```
a.strip()
```



```
IDLE Shell 3.9.7
File Edit Shell Debug Options Window Help
>>>
>>> a = " Hello world "
>>> a.lstrip()
'Hello world'
>>> a.rstrip()
' Hello world'
>>> a.strip()
'Hello world'
>>>
```

Ln: 126 Col: 4

07

문자열 함수

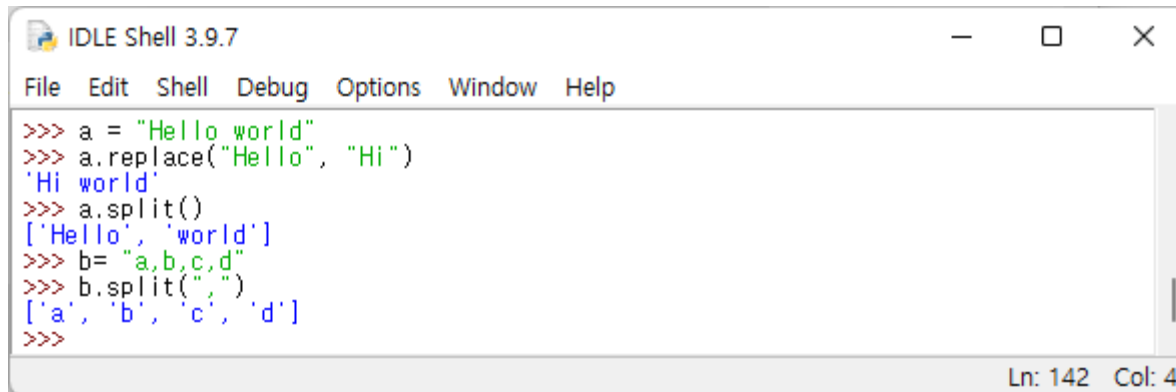
replace() : 문자열 안의 특정한 값을 다른 값으로 치환한다.

split() : 문자열을 괄호 안에 값으로 구분하여 나누어준다.(괄호 안에 값을 넣지 않으면 공백을 기준으로 한다.)

```
a = "Hello world"

a.replace("Hello", "Hi")
a.split()

b= "a,b,c,d"
b.split(",")
```



```
IDLE Shell 3.9.7
File Edit Shell Debug Options Window Help
>>> a = "Hello world"
>>> a.replace("Hello", "Hi")
'Hi world'
>>> a.split()
['Hello', 'world']
>>> b= "a,b,c,d"
>>> b.split(",")
['a', 'b', 'c', 'd']
>>>
```

Ln: 142 Col: 4



함수

08

함수

함수는 어떠한 입력 값을 가지고 작업을 실행하고 결과물을 출력하는 것이 함수가 하는 일이다.

파이썬에서는 def 키워드를 이용하여 함수를 정의한다.

```
def 함수명 (매개변수) :  
    실행할 코드1  
    실행할 코드2  
    return 출력 값
```

함수명(인자) → 함수 호출하는 부분

함수를 사용 하는 이유는 반복적으로 실행 해야되는 코드가 있을 시 코드를 반복해서 사용하게 되면 코드가 길어질 뿐만 아니라 코드가 길어짐으로 코드를 보는데 가독성이 떨어진다.

매개변수, 인자

매개 변수와 인자는 사람들이 혼용해서 사용하는 경우가 많다.

매개변수는 함수에 입력으로 전달된 값을 받는 변수이고

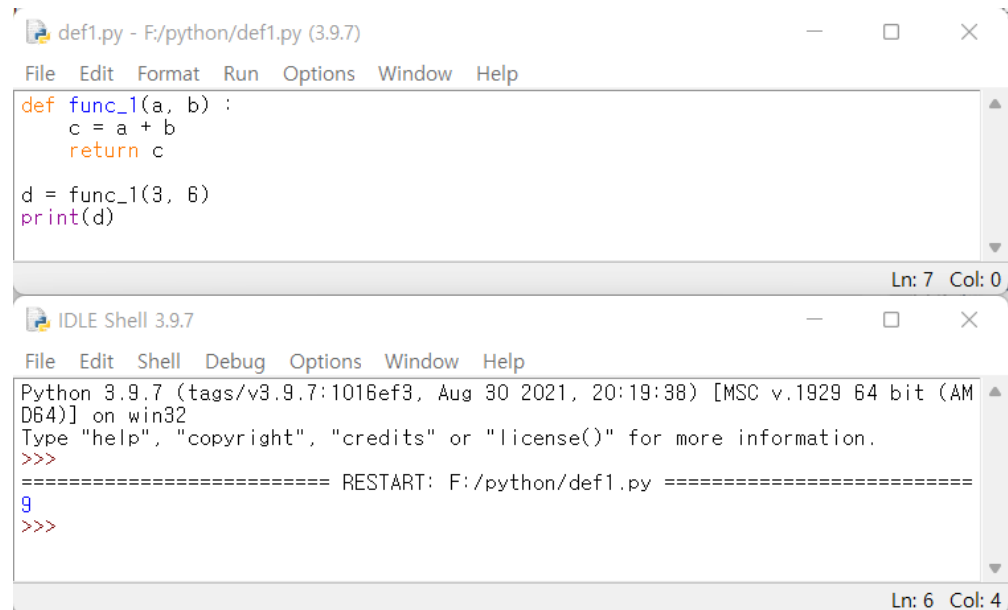
인자는 함수를 호출할 때 전달하는 입력 값을 의미한다.

함수에서 매개변수와 인자를 사용하지 않는 경우도 존재한다.

08

함수

```
def func_1(a, b) :  
    c = a + b  
    return c  
  
d = func_1(3, 6)  
  
print(d)
```



The screenshot displays a Python IDE with two windows. The top window, titled 'def1.py - F:/python/def1.py (3.9.7)', contains the following code:

```
def func_1(a, b) :  
    c = a + b  
    return c  
  
d = func_1(3, 6)  
print(d)
```

The bottom window, titled 'IDLE Shell 3.9.7', shows the execution output:

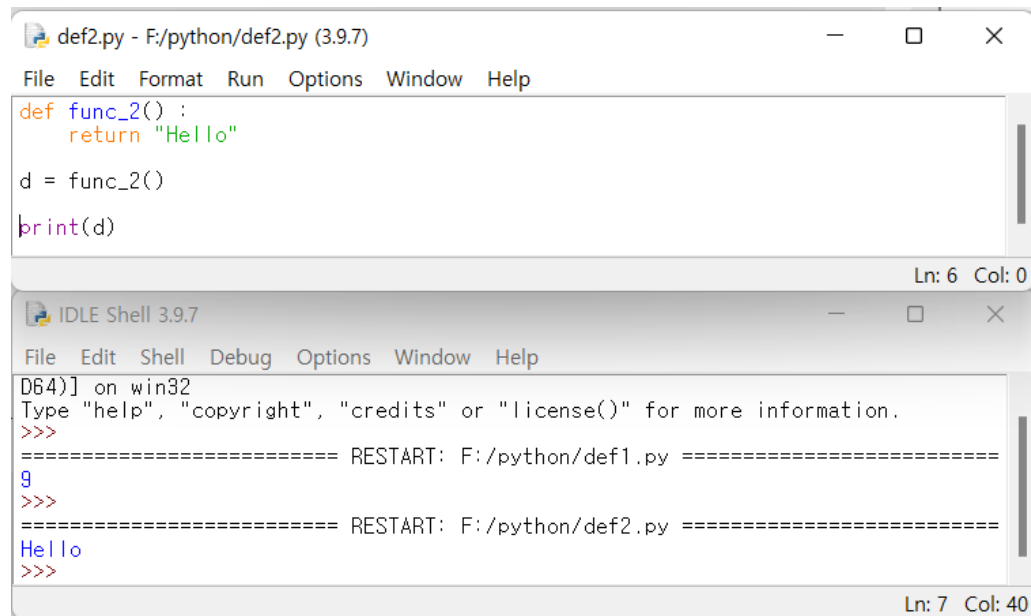
```
Python 3.9.7 (tags/v3.9.7:1016ef3, Aug 30 2021, 20:19:38) [MSC v.1929 64 bit (AMD64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: F:/python/def1.py =====  
9  
>>>
```

The status bar at the bottom of the shell window indicates 'Ln: 6 Col: 4'.

08

함수

```
def func_2() :  
    return "Hello"  
  
d = func_2()  
  
print(d)
```



The screenshot displays the Python IDLE 3.9.7 environment. The top window, titled 'def2.py - F:/python/def2.py (3.9.7)', contains the following Python code:

```
def func_2() :  
    return "Hello"  
  
d = func_2()  
  
print(d)
```

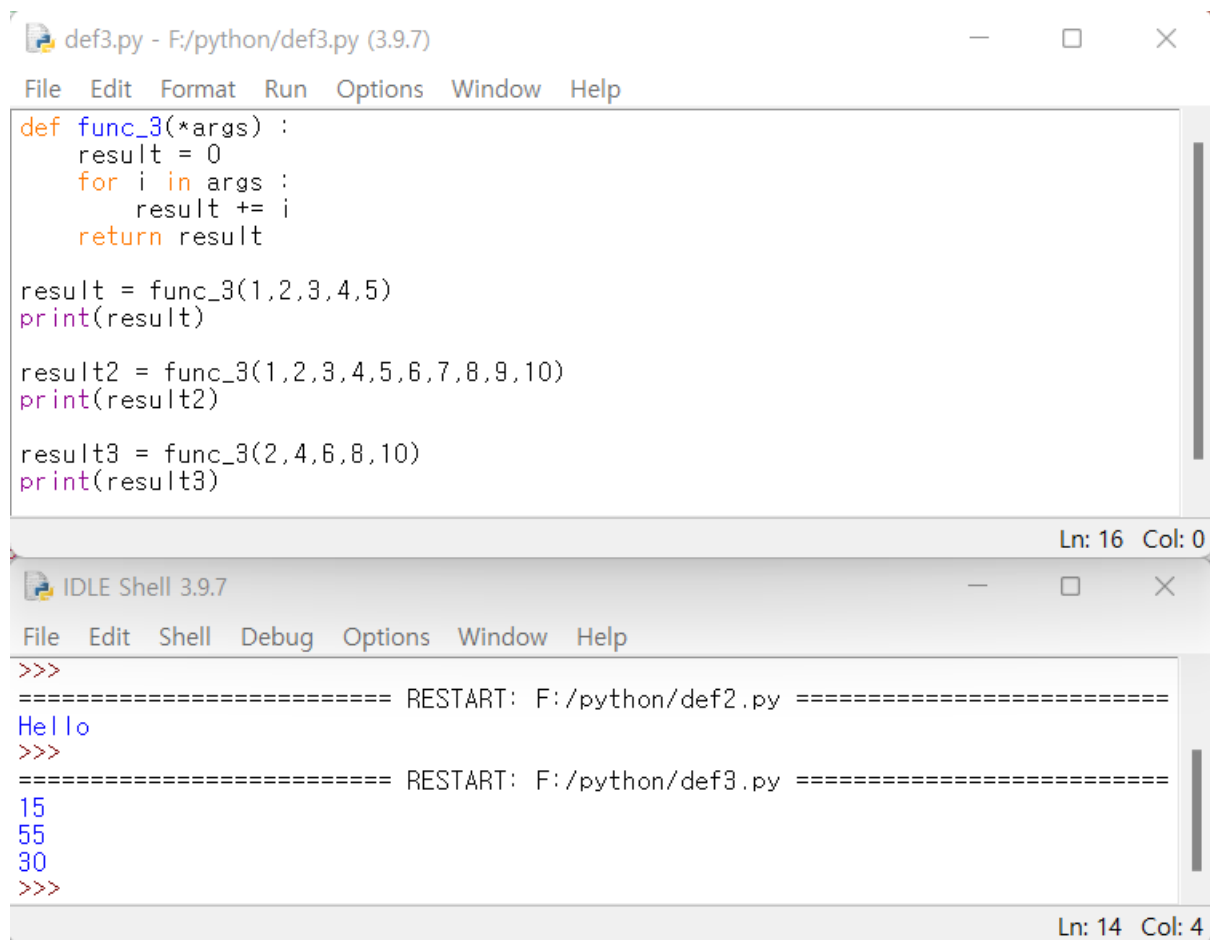
The bottom window, titled 'IDLE Shell 3.9.7', shows the execution output. It includes a prompt 'D64)] on win32', a help message, and two 'RESTART' lines. The first restart shows the output 'Hello'.

```
D64)] on win32  
Type "help", "copyright", "credits" or "license()" for more information.  
>>>  
===== RESTART: F:/python/def1.py =====  
9  
>>>  
===== RESTART: F:/python/def2.py =====  
Hello  
>>>
```

함수

```
def func_3(*args) :  
    result = 0  
    for i in args :  
        result += i  
    return result  
  
result = func_3(1,2,3,4,5)  
print(result)  
  
result2 = func_3(1,2,3,4,5,6,7,8,9,10)  
print(result2)  
  
result3 = func_3(2,4,6,8,10)  
print(result3)
```


함수



The screenshot displays two windows from the Python IDLE 3.9.7 environment. The top window, titled 'def3.py - F:/python/def3.py (3.9.7)', contains a Python script. The script defines a function named `func_3` that takes an arbitrary number of arguments (`*args`), calculates their sum, and returns the result. Below the function definition, three lines of code call `func_3` with different sets of arguments and print the results. The bottom window, titled 'IDLE Shell 3.9.7', shows the output of the script. It indicates a restart of the shell, followed by the output of the previous script ('Hello') and the results of the current script: 15, 55, and 30.

```
def func_3(*args) :  
    result = 0  
    for i in args :  
        result += i  
    return result  
  
result = func_3(1,2,3,4,5)  
print(result)  
  
result2 = func_3(1,2,3,4,5,6,7,8,9,10)  
print(result2)  
  
result3 = func_3(2,4,6,8,10)  
print(result3)
```

```
>>>  
===== RESTART: F:/python/def2.py =====  
Hello  
>>>  
===== RESTART: F:/python/def3.py =====  
15  
55  
30  
>>>
```



클래스

클래스란?

클래스가 무엇인가는 대개 붕어빵에 비유하여 많이들 표현한다.

Class → 붕어빵을 찍어내는 틀

Object → 붕어빵

클래스는 무언가를 똑같이 만들어주는 설계도와 같은 의미

클래스 구성

클래스는 크게 2가지로 구성

1. 속성
2. 매서드

속성 : 클래스가 가지는 변수

매서드 : 클래스가 가지는 동작

클래스 선언

파이썬에서 클래스를 선언하는 방식은

```
class 클래스이름:
```

Class라는 키워드와 클래스의 이름을 나열하여 선언한다.

파이썬에서 클래스의 이름은 일반적으로 대문자로 표시

생성자 선언

클래스를 선언한 후 `__init__` 이라는 함수를 이용하여 생성자를 선언한다.

`__init__` 함수의 첫번째 매개변수는 `self`로 선언하여야 한다.

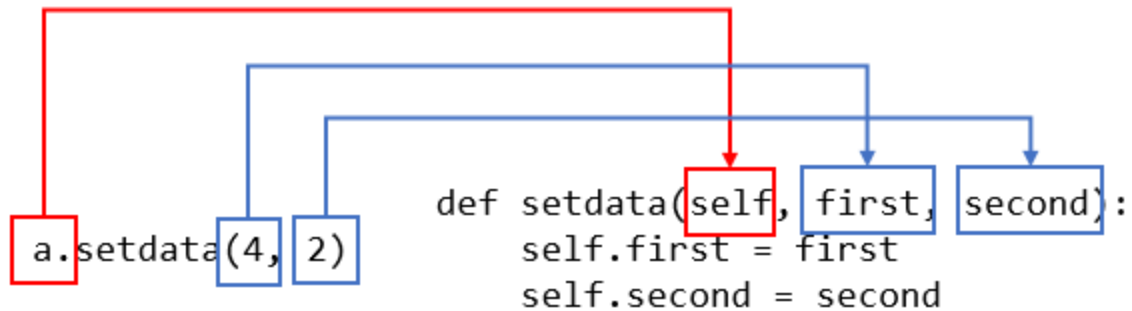
```
class Test_class_2 ():  
    def __init__(self, a_input, b_input):  
        self.a = a_input  
        self.b = b_input
```

11

`__init__` ? `self` ?

`__init__` 은 클래스를 생성시 초기화를 하며 실행되는 함수이다.

`self`는 객체 자기 자신을 참조하는 매개변수이다.



11

클래스 변수

같은 클래스에서 생성된 객체들은 서로 독립적이다.

이러한 객체들끼리 서로의 정보를 공유하는 방법으로 클래스 변수가 있다.

```
class test_class_4():
    _list = []

    def __init__(self, name):
        self.name = name

    def add_list(self, phone):
        self._list.append(phone)

c = test_class_4("test")
d = test_class_4("test2")
c.add_list("01012345678")
d.add_list("01098765432")
c._list
```


상속

상속이란 사전적 의미로는 " 물려받다" 라는 의미이다.

클래스에서 상속은 A 클래스가 있고 B 클래스가 있는데 B클래스에서 A클래스의 기능을 물려받을 수 있게 하는것이다.

```
class B클래스(A클래스)
```

다음과 같은 형식으로 클래스의 상속이 가능하다.

상속을 사용하는 이유는 기존 클래스를 변경하지 않고 기능을 추가하거나 변경할 때 사용한다.

판다스, 넘파이,
맷플로립

판다스(pandas)

판다스는 파이썬의 데이터를 처리하기 위한 라이브러리이다.
데이터 분석과 같은 작업에서 판다스는 필수 라이브러리로 알려져 있다.

라이브러리를 설치하기 위해서는 CMD(명령 프롬프트)에서 다음 명령어를 입력한다.

```
C:\  
λ pip install pandas
```

판다스(pandas)의 구조

판다스의 구조는 크게 3가지로 나누어 진다.

스리즈 (Series)

데이터프레임 (DataFrame)

패널 (Panel)

이 중 데이터프레임을 가장 많이 사용하며 기본적인 시리즈와 데이터프레임을 다루어보자

09

판다스(Pandas) - 시리즈 (Series)

판다스의 시리즈는 1차원 배열의 값과 그 값에 대응하는 인덱스를 부여할 수 있다.

```
import pandas as pd

pd_list = pd.Series([5000, 6000, 6500, 6500],
                    index=["아메리카노", "카페라떼", "카페모카", "카푸치노"])

print(pd_list)
```

✓ 0.7s

Python

아메리카노	5000
카페라떼	6000
카페모카	6500
카푸치노	6500

dtype: int64

(설치된 라이브러리를 불러오는 방법으로
import 라이브러리명 as 별칭)

09

판다스(Pandas) - 시리즈 (Series)

시리즈의 값과 인덱스 값은 따로 출력이 가능하다.

변수.values → 시리즈의 값만 출력

변수.index → 시리즈의 인덱스 값만 출력

```
▶ print("시리즈의 값 : " , pd_list.values)
print("시리즈의 인덱스 값 : " , pd_list.index)
[7] ✓ 0.4s
... 시리즈의 값 : [5000 6000 6500 6500]
시리즈의 인덱스 값 : Index(['아메리카노', '카페라떼', '카페모카', '카푸치노'],
dtype='object')
```

09

판다스(Pandas) – 데이터프레임(DataFrame)

스리즈는 1차원 배열이면 데이터프레임은 2차원 리스트이다.
2차원이므로 행방향의 인덱스와 열방향의 인덱스가 존재한다.
쉽게 말하면 행렬의 구조와 같다.

```
values = [[1,2,3], [4,5,6], [7,8,9]]
index = ['a', 'b', 'c']
columns = ['A', 'B', 'C']

pd_dataframe = pd.DataFrame(values, index=index, columns=columns)

print(pd_dataframe)
```

✓ 0.8s

Python

	A	B	C
a	1	2	3
b	4	5	6
c	7	8	9

09

판다스(Pandas) – 데이터프레임(DataFrame)

데이터프레임도 시리즈와 마찬가지로 값과 인덱스 값을 따로 출력 가능하다.

```
print("데이터프레임의 인덱스값 : ", pd_dataframe.index)
print("데이터프레임의 컬럼 값 : ", pd_dataframe.columns)
print("데이터프레임의 값 : ", pd_dataframe.values)
```

✓ 0.5s

Python

데이터프레임의 인덱스값 : Index(['a', 'b', 'c'], dtype='object')

데이터프레임의 컬럼 값 : Index(['A', 'B', 'C'], dtype='object')

데이터프레임의 값 : [[1 2 3]

[4 5 6]

[7 8 9]]

09

판다스(Pandas) – 데이터프레임(DataFrame)

데이터프레임은 앞에서 배운 데이터형 리스트와 딕셔너리를 이용하여 데이터프레임을 생성할 수 있다.

```
list_data = [{"A", "남", "010-1234-5678"},  
             [{"B", "여", "010-4321-8765"},  
             [{"C", "여", "010-9876-1234"}]
```

```
pd_dataframe_2 = pd.DataFrame(list_data)
```

```
print(pd_dataframe_2)
```

✓ 0.5s

Python

	0	1	2
0	A	남	010-1234-5678
1	B	여	010-4321-8765
2	C	여	010-9876-1234

가장 기본적인 형태로 인덱스 값들을 지정하지 않았기 때문에 자동으로 0,1,2.... 로 지정된다.

09

판다스(Pandas) – 데이터프레임(DataFrame)

행이나 열의 값을 지정할 수 있다.

```
pd_dataframe_2 = pd.DataFrame(list_data, columns=["이름", "성별", "전화번호"])
```

```
print(pd_dataframe_2)
```

✓ 0.4s

Python

	이름	성별	전화번호
0	A	남	010-1234-5678
1	B	여	010-4321-8765
2	C	여	010-9876-1234

09

판다스(Pandas) – 데이터프레임(DataFrame)

데이터형 딕셔너리형은 key : value 로 데이터가 구성되어있기때문에 key의 값이 columns의 값으로 지정이 된다.

```
list_data_2 = {  
    "이름" : ["A", "B", "C"],  
    "성별" : ["남", "여", "여"],  
    "전화번호" : ["010-1234-5678", "010-4321-8765", "010-9876-1234"]  
}
```

```
pd_dataframe_3 = pd.DataFrame(list_data_2)
```

```
print(pd_dataframe_3)
```

✓ 0.5s

Python

	이름	성별	전화번호
0	A	남	010-1234-5678
1	B	여	010-4321-8765
2	C	여	010-9876-1234

09

판다스(Pandas) – 데이터프레임(DataFrame) 조회

판다스에서는 데이터프레임에서 원하는 구간만 조회가 가능하다.

데이터프레임명.head(n) → 앞 부분을 n개만 출력

데이터프레임명.tail(n) → 뒤 부분을 n개만 출력

데이터프레임명['열 이름'] → 해당되는 열의 값만 출력

09

판다스(Pandas) – 데이터프레임(DataFrame) 조회

```
print(pd_dataframe_3.head(2))
```

✓ 0.3s

Python

	이름	성별	전화번호
0	A	남	010-1234-5678
1	B	여	010-4321-8765

```
print(pd_dataframe_3.tail(2))
```

✓ 0.5s

Python

	이름	성별	전화번호
1	B	여	010-4321-8765
2	C	여	010-9876-1234

```
print(pd_dataframe_3["성별"])
```

✓ 0.3s

Python

0 남

1 여

2 여

Name: 성별, dtype: object

09

판다스(Pandas) – 데이터프레임(DataFrame)

판다스는 이렇게 데이터를 데이터프레임으로 생성할 수 있지만 외부의 데이터 파일(csv, text, Excel, SQL, JSON)을 읽어서 데이터프레임으로 생성할 수 있다.

```
csv_data = pd.read_csv('example.csv')
```

```
print(csv_data)
```

✓ 0.5s

Python

	Name	Gender	Phone
0	A	male	010-1234-5678
1	B	female	010-1234-5679
2	C	male	010-1234-5680
3	D	female	010-1234-5681
4	E	male	010-1234-5682
5	F	female	010-1234-5683
6	G	male	010-1234-5684
7	H	female	010-1234-5685
8	I	male	010-1234-5686
9	J	female	010-1234-5687

넘파이(Numpy)

넘파이는 수치 데이터를 다루는 라이브러리이다.
다차원 행렬 자료 구조인 ndarray를 통하여 벡터 및 행렬을 사용하는
선형 대수 계산에서 주로 사용된다.
넘파이는 편의성도 좋지만 속도면에서도 우수하다.

넘파이도 설치를 하려면 CMD(명령 프롬프트)에서 다음 명령어를 입력한다.

```
C:\  
λ pip install numpy
```

09

넘파이(Numpy) – array()

넘파이의 핵심인 array()는 리스트, 튜플, 배열로 부터 ndarray를 생성한다. 1차원 배열이든 2차원 배열이든 상관 없이 array를 생성할 수 있다.

```
import numpy as np
```

```
np_array_1 = np.array([1,2,3,4,5])  
print(np_array_1)
```

✓ 0.1s

Python

```
[1 2 3 4 5]
```

```
np_array_2 = np.array([[1,2,3], [4,5,6]])  
print(np_array_2)
```

✓ 0.4s

Python

```
[[1 2 3]  
 [4 5 6]]
```


09

넘파이(Numpy) – array()

np.array()로 만든 데이터의 타입은 두개 다 ndarray로 나타나게 된다.

```
print("array_1의 데이터 타입 : ", type(np_array_1))  
print("array_2의 데이터 타입 : ", type(np_array_2))
```

✓ 0.5s

Python

```
array_1의 데이터 타입 : <class 'numpy.ndarray'>
```

```
array_2의 데이터 타입 : <class 'numpy.ndarray'>
```

09

넘파이(Numpy) – array()

np.array()를 사용하게 되면 배열의 축의 개수(ndim)와 크기(shape)를 출력할 수 있다.

```
print("array_1의 축의 개수 : " , np_array_1.ndim)
print("array_1의 크기 : ", np_array_1.shape)
```

✓ 0.4s

Python

```
array_1의 축의 개수 : 1
array_1의 크기 : (5,)
```

```
print("array_2의 축의 개수 : ", np_array_2.ndim)
print("array_2의 크기 : ", np_array_2.shape)
```

✓ 0.5s

Python

```
array_2의 축의 개수 : 2
array_2의 크기 : (2, 3)
```

09

넘파이(Numpy) – zeros()

zeros()는 모든 원소의 값이 0인 배열을 생성한다.

```
zero_array = np.zeros((3,3))
```

```
print(zero_array)
```

✓ 0.5s

Python

```
[[0. 0. 0.]  
 [0. 0. 0.]  
 [0. 0. 0.]]
```

09

넘파이(Numpy) – ones()

ones()는 모든 원소 값이 1인 배열을 생성한다.

```
one_array = np.ones((2,3))
```

```
print(one_array)
```

✓ 0.5s

Python

```
[[1. 1. 1.]
```

```
 [1. 1. 1.]]
```

09

넘파이(Numpy) – full()

full()은 모든 원소의 값을 지정한 값으로 행렬을 생성한다.

```
full_array = np.full((3,4), 5)
```

```
print(full_array)
```

✓ 0.4s

Python

```
[[5 5 5 5]  
 [5 5 5 5]  
 [5 5 5 5]]
```

09

넘파이(Numpy) – eye()

eye()는 단위 행렬을 생성한다.

```
eye_array = np.eye(3)
```

```
print(eye_array)
```

✓ 0.7s

Python

```
[[1.  0.  0.]  
 [0.  1.  0.]  
 [0.  0.  1.]]
```

09

넘파이(Numpy) – random.random()

random.random()은 임의의 값을 원소로 하여 행렬을 생성한다.

```
random_array = np.random.random((2,2))
```

```
print(random_array)
```

✓ 0.8s

Python

```
[[0.18010694 0.03847656]  
 [0.31585709 0.19115222]]
```

09

넘파이(Numpy) – arange()

arange(n)은 0부터 n-1까지의 원소 값을 가지는 배열을 생성한다.

```
arange_array = np.arange(10)
```

```
print(arange_array)
```

✓ 0.5s

Python

```
[0 1 2 3 4 5 6 7 8 9]
```


09

넘파이(Numpy) – arange()

arange(i,j,k)는 i부터 시작해서 j-1까지 k씩 증가하는 배열을 생성한다.

```
arange_array_2 = np.arange(1, 20, 2)
```

```
print(arange_array_2)
```

✓ 0.2s

Python

```
[ 1  3  5  7  9 11 13 15 17 19]
```

09

넘파이(Numpy) – reshape()

reshape()는 배열 내부의 데이터는 변경하지 않고 배열의 구조를 변경한다.

```
arange_array_3 = np.arange(30)
reshape_array = arange_array_3.reshape((5,6))
```

```
print(reshape_array)
```

✓ 0.4s

Python

```
[[ 0  1  2  3  4  5]
 [ 6  7  8  9 10 11]
 [12 13 14 15 16 17]
 [18 19 20 21 22 23]
 [24 25 26 27 28 29]]
```

넘파이(Numpy) – 연산

넘파이를 사용하면 배열 간의 연산을 쉽게 수행 가능하다.

덧셈	$a+b$	<code>np.add()</code>
뺄셈	$a-b$	<code>np.subtract()</code>
곱셈	$a*b$	<code>np.multiply()</code>
나눗셈	a/b	<code>np.divide()</code>

09

넘파이(Numpy) – 연산(덧셈)

```
x = np.array([1,2,3])  
y = np.array([4,5,6])
```

```
result = x + y  
result2 = np.add(x,y)
```

```
print(result)  
print(result2)
```

✓ 0.4s

Python

```
[5 7 9]
```

```
[5 7 9]
```

09

넘파이(Numpy) – 연산(뺄셈)

```
x = np.array([1,2,3])
y = np.array([4,5,6])

result = x - y
result2 = np.subtract(x,y)

print(result)
print(result2)
```

✓ 0.3s

Python

[-3 -3 -3]

[-3 -3 -3]

09

넘파이(Numpy) – 연산(곱셈)

```
x = np.array([1,2,3])
y = np.array([4,5,6])

result = x * y
result2 = np.multiply(x,y)

print(result)
print(result2)
```

✓ 0.4s

Python

[4 10 18]

[4 10 18]

09

넘파이(Numpy) – 연산(나눗셈)

```
x = np.array([1,2,3])
y = np.array([4,5,6])

result = x / y
result2 = np.divide(x,y)

print(result)
print(result2)
```

✓ 0.4s

Python

```
[0.25 0.4  0.5 ]
```

```
[0.25 0.4  0.5 ]
```

09

넘파이(Numpy) – 연산(행렬곱)

앞의 곱셈인 `np.multiply()`는 요소별 곱이다. 넘파이는 요소별 곱만이 아닌 벡터와 행렬의 곱 또는 행렬곱을 하기 위해서는 `dot()`를 사용한다.

```
x = np.array([[1,2], [3,4]])  
y = np.array([[5,6], [7,8]])
```

```
result = np.dot(x,y)
```

```
print(result)
```

✓ 0.3s

Python

```
[[19 22]  
 [43 50]]
```


09

맷플롯립(Matplotlib)

맷플롯립은 파이썬에서 데이터를 차트나 플롯(plot)으로 그려주는 라이브러리이다. 흔히 데이터 시각화 패키지로 알려져있다
라인 플롯, 바 차트, 파이 차트, Scatter 등 다양한 차트와 플롯을 지원한다.

```
C:\>pip install matplotlib
```

09

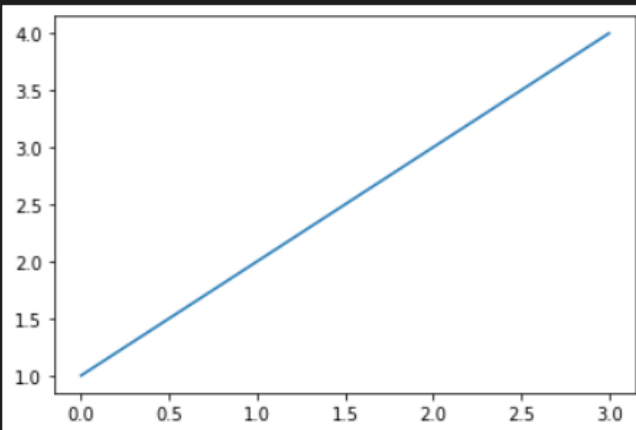
맷플롯립(Matplotlib) – plot()

.plot() 함수에 하나의 숫자로 이루어진 리스트를 입력하여 그래프를 지정한다.

리스트의 값들은 y값으로 나타나고 x축은 자동으로 0,1,2,3 순으로 지정된다.

show() 함수를 통하여 그래프를 화면에 출력한다.

```
import matplotlib.pyplot as plt  
  
plt.plot([1,2,3,4])  
plt.show()
```

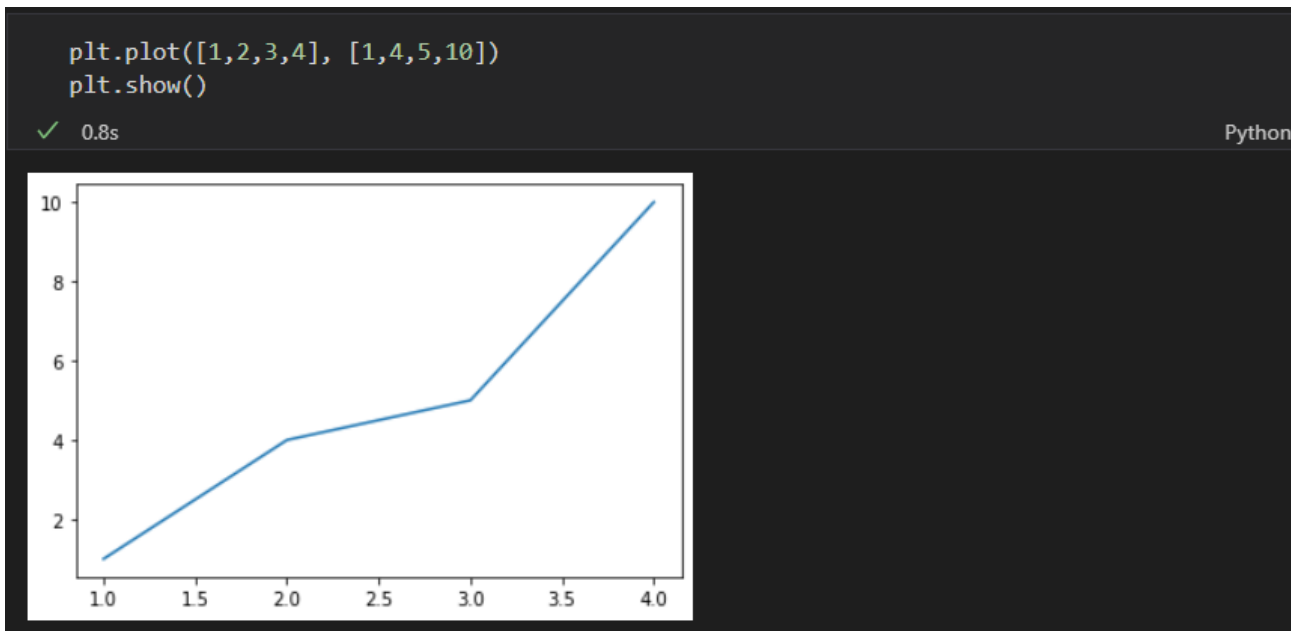


09

맷플롯립(Matplotlib) – plot()

.plot() 함수에 두개의 숫자로 이루어진 리스트를 입력하여 그래프를 지정한다.

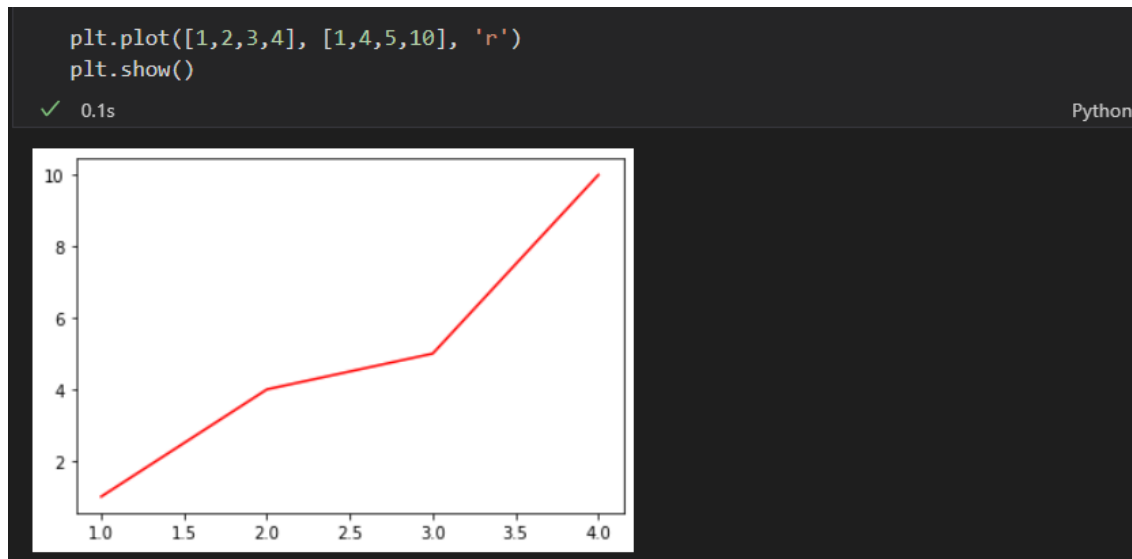
첫 리스트의 값이 x축 두번째 리스트의 값이 y축의 값이 된다.



09

맷플롯립(Matplotlib) – 선 색 변경

`plot('x축 리스트', 'y축 리스트', '옵션 ')` → 옵션 부분에 색상 인자 값이나 헥스 값을 넣어주면 선의 색이 변경이 가능하다.



b	blue
g	green
r	red
c	cyan
m	magenta
y	yellow
k	black
w	white

09

맷플롯립(Matplotlib) – 선 종류 변경

`plot('x축 리스트', 'y축 리스트', '옵션 ')` → 옵션 부분에 선 종류 인자 값을 넣어주면 선의 종류가 변경된다.

-	Solid line	o	Circle marker
--	Dashed line	v	Triangle_down marker
-.	Dash-dot line	*	Star marker
.	Point marker	x	X marker
,	Pixel marker	D	Diamond marker

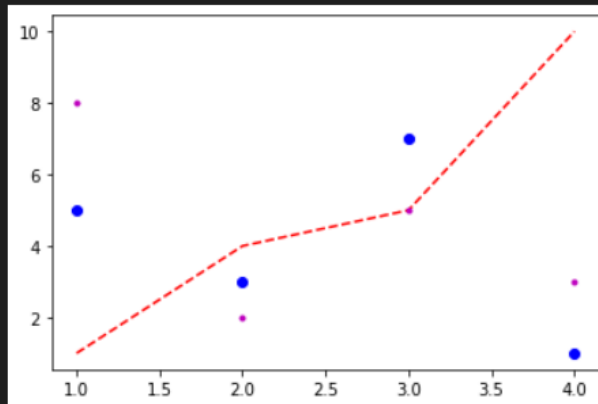
09

맷플롯립(Matplotlib) – 선 종류 변경

```
plt.plot([1,2,3,4], [1,4,5,10], "--r")  
plt.plot([1,2,3,4], [5,3,7,1], "ob")  
plt.plot([1,2,3,4], [8,2,5,3], ".m")  
plt.show()
```

✓ 0.1s

Python



09

맷플롯립(Matplotlib) – 레이블, 타이틀 추가

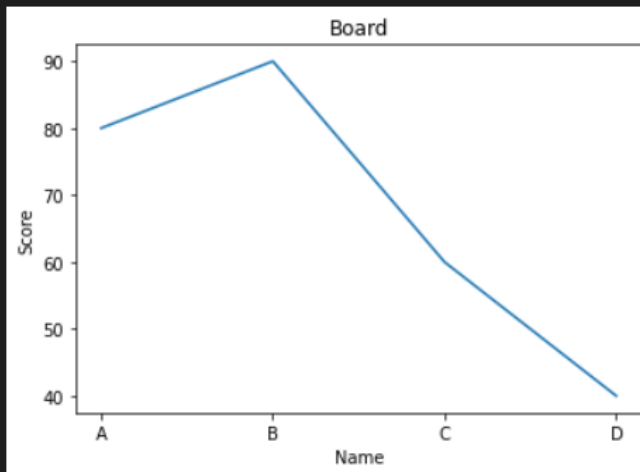
plt.xlabel(), plt.ylabel() → x,y축 레이블 추가

plt.title() → 그래프 제목 추가

```
plt.plot(["A", "B", "C", "D"], [80, 90, 60, 40])  
plt.xlabel("Name")  
plt.ylabel("Score")  
plt.title("Board")  
plt.show()
```

✓ 0.8s

Python



09

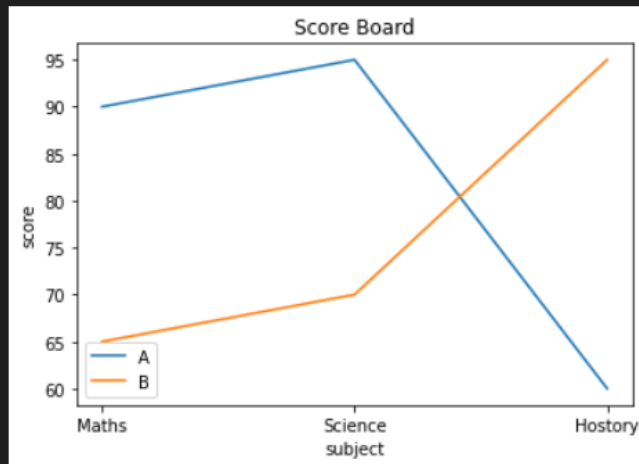
맷플롯립(Matplotlib) – 범례 추가

plt.legend() → 그래프 라인의 범례를 추가

```
plt.plot(["Maths", "Science", "Hostory"], [90,95,60])  
plt.plot(["Maths", "Science", "Hostory"], [65,70,95])  
plt.xlabel("subject")  
plt.ylabel("score")  
plt.title("Score Board")  
plt.legend(["A", "B"])  
plt.show()
```

✓ 0.1s

Python



09

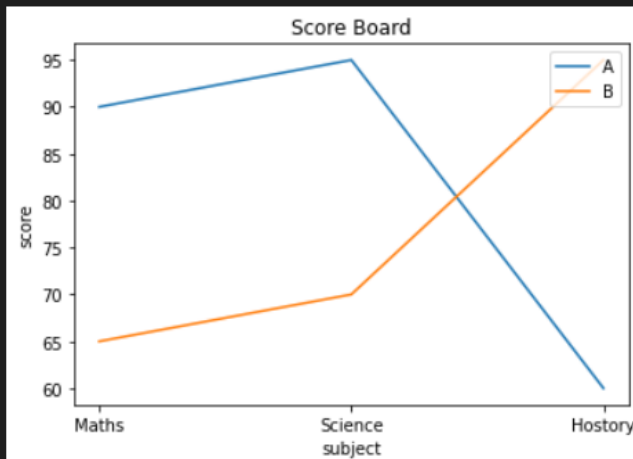
맷플롯립(Matplotlib) – 범례 추가

plt.legend() → loc="위치" 를 지정하면 지정된 위치에 범례가 나타난다.

```
plt.plot(["Maths", "Science", "History"], [90,95,60])  
plt.plot(["Maths", "Science", "History"], [65,70,95])  
plt.xlabel("subject")  
plt.ylabel("score")  
plt.title("Score Board")  
plt.legend(["A", "B"], loc="upper right")  
plt.show()
```

✓ 0.1s

Python



09

맷플롯립(Matplotlib) – 범례 추가

`plt.legend()` → `loc="위치"` 를 지정하면 지정된 위치에 범례가 나타난다.(위치는 string이나 code 값으로 지정 가능)

string	code	string	code	string	code
upper left	2	upper center	9	upper right	1
center left	6	center	10	center right	7
lower left	3	lower center	8	lower right	4
best	0			right	5

09

맷플롯립(Matplotlib) – 서브플롯

여러 개의 그래프를 그리는 경우 사용한다.
그래프의 위치를 격자형으로 지정하여 그래프를 출력한다.

```
plt.subplot(nrow, ncol, pos)
```

nrow : 행의 수

ncol : 열의 수

pos : 위치

예를 들어 plt.subplot(3,4,1) 지정하면

1	2	3	4
5	6	7	8
9	10	11	12

1의 위치에 그래프가 출력이 된다.

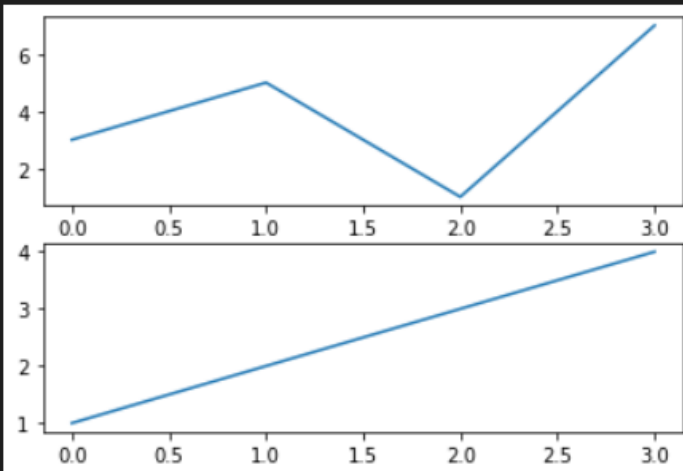
09

맷플롯립(Matplotlib) – 서브플롯

```
plt.subplot(2,1,1)  
plt.plot([3,5,1,7])  
plt.subplot(2,1,2)  
plt.plot([1,2,3,4])  
plt.show()
```

✓ 0.1s

Python



09

맷플롯립(Matplotlib) – 바형 그래프

선 그래프는 `plot()`을 사용하였고 바형 그래프를 출력하기 위해서는 `bar()` 함수를 사용하면 바형 그래프를 출력 할 수 있다.

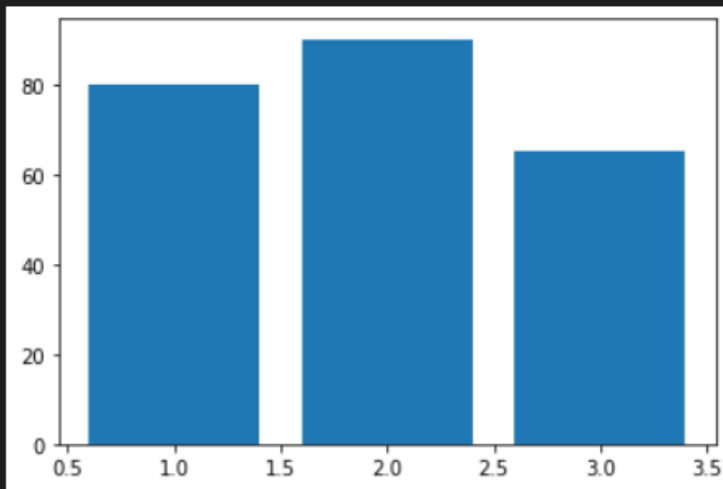
```
x = [1,2,3]
```

```
plt.bar(x, [80, 90, 65])
```

```
plt.show()
```

✓ 0.1s

Python



09

맷플롯립(Matplotlib) – 바형 그래프

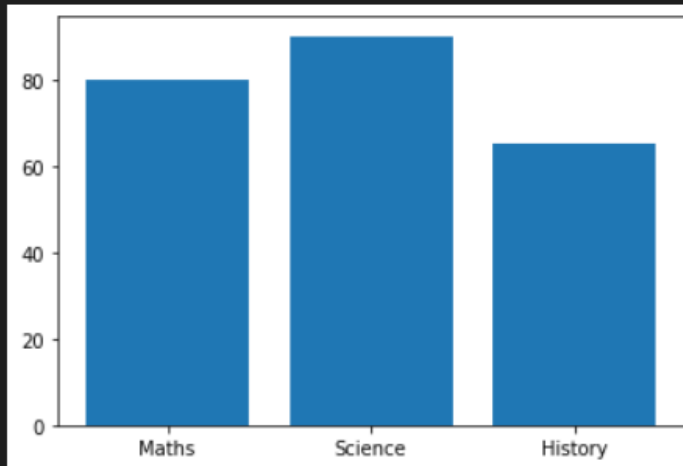
xticks()를 이용하여 x축의 레이블 값의 변경이 가능하다.

```
x = [1,2,3]

plt.bar(x, [80, 90, 65])
plt.xticks(x, ["Maths", "Science", "History"])
plt.show()
```

✓ 0.9s

Python



09

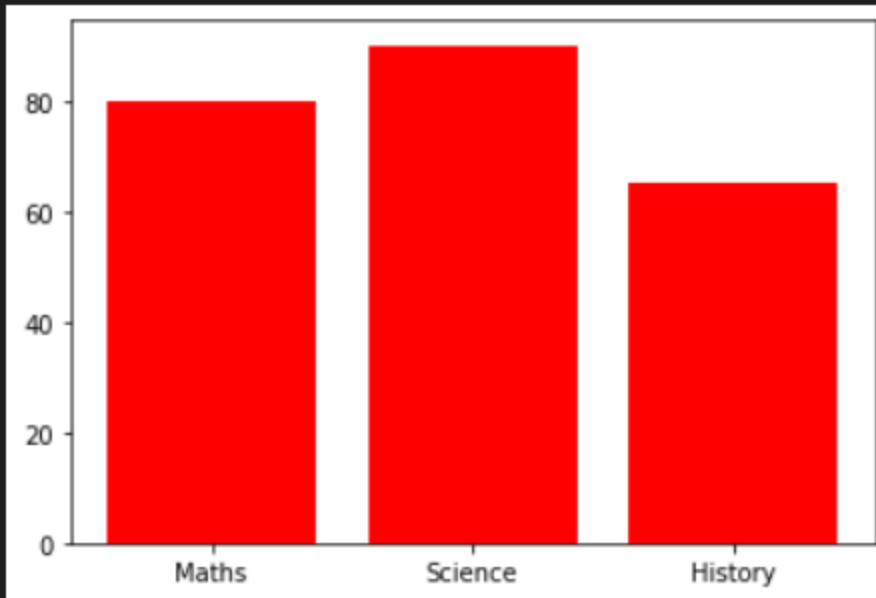
맷플롯립(Matplotlib) – 바형 그래프 색상 변경

```
x = [1,2,3]

plt.bar(x, [80, 90, 65], color="r")
plt.xticks(x, ["Maths", "Science", "History"])
plt.show()
```

✓ 0.9s

Python



09

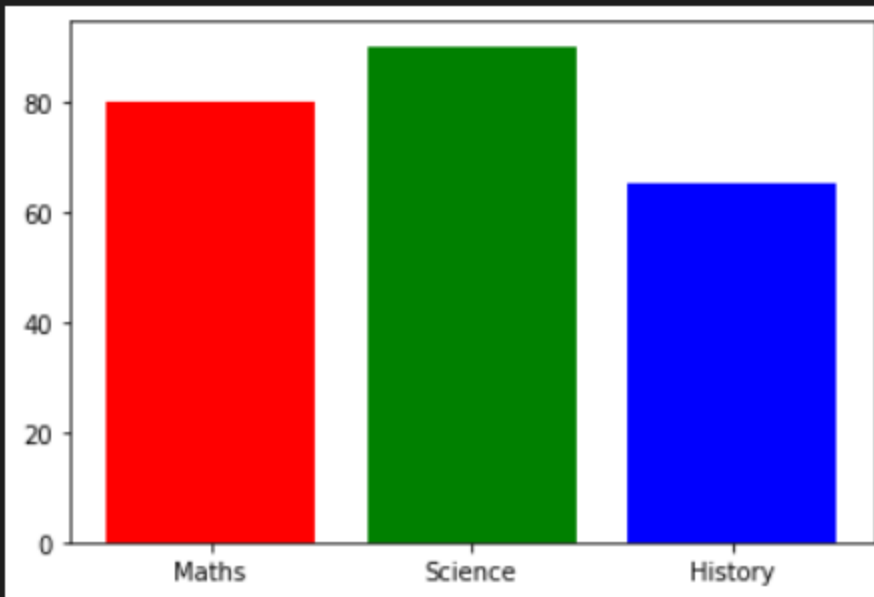
맷플롯립(Matplotlib) – 바형 그래프 색상 변경

```
x = [1,2,3]

plt.bar(x, [80, 90, 65], color=["r","g","b"])
plt.xticks(x, ["Maths", "Science", "History"])
plt.show()
```

✓ 0.8s

Python



09

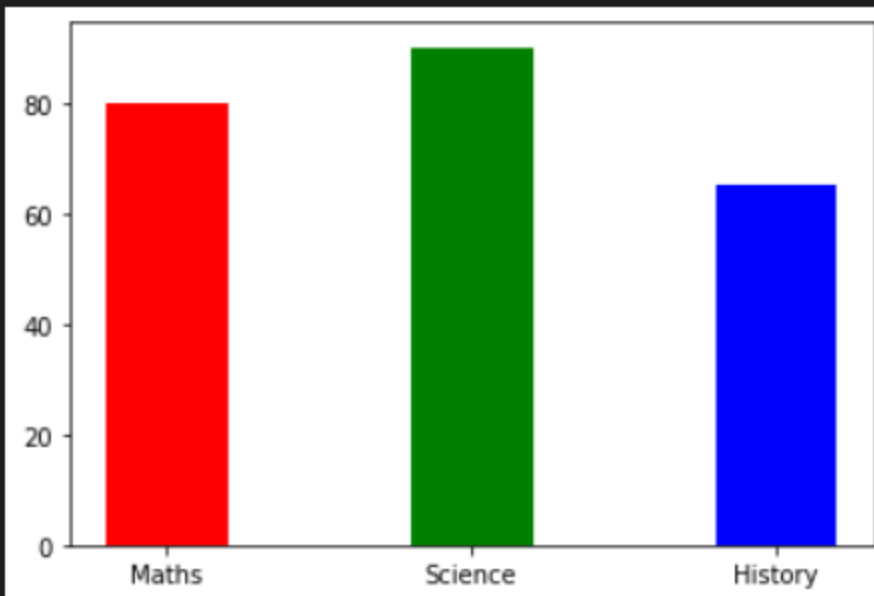
맷플롯립(Matplotlib) – 바형 그래프 폭 지정하기

```
x = [1,2,3]

plt.bar(x, [80, 90, 65], color=["r","g","b"], width=0.4)
plt.xticks(x, ["Maths", "Science", "History"])
plt.show()
```

✓ 0.1s

Python



09

맷플롯립(Matplotlib) – 산점도

산점도를 출력하기 위해서는 matplotlib의 scatter()를 사용하면 출력이 가능하다.

```
import numpy as np
```

```
x = np.random.rand(50)
```

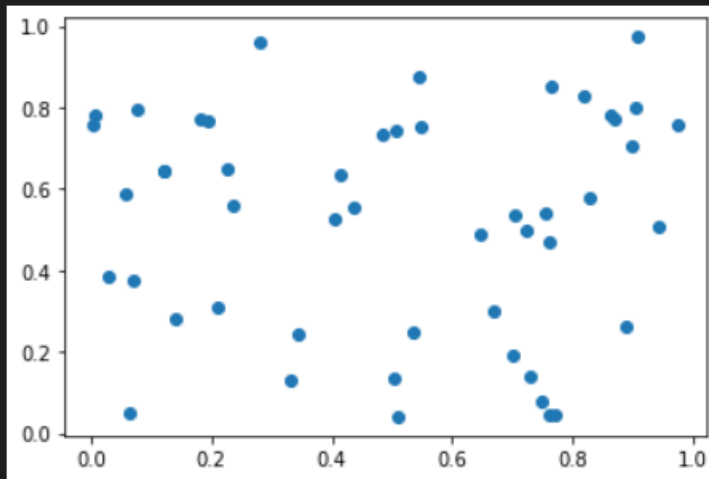
```
y = np.random.rand(50)
```

```
plt.scatter(x,y)
```

```
plt.show()
```

✓ 0.1s

Python



09

맷플롯립(Matplotlib) – 산점도 크기 및 색상 지정

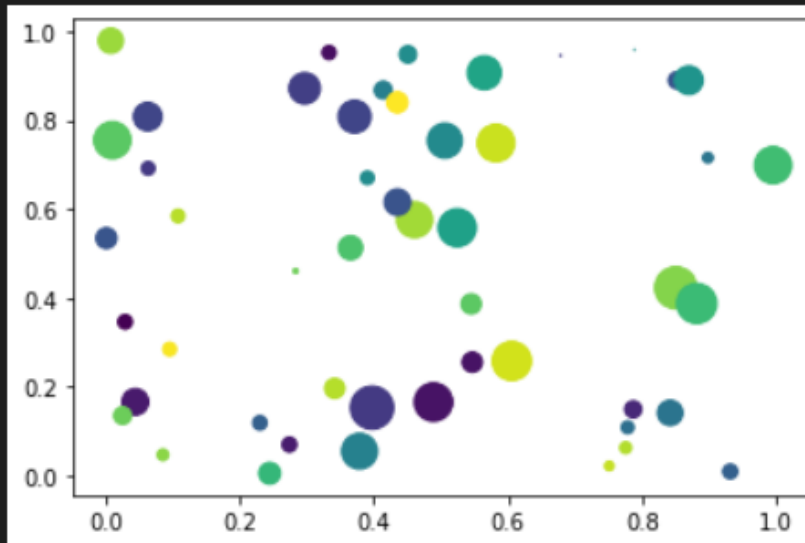
```
import numpy as np

x = np.random.rand(50)
y = np.random.rand(50)
area = (20 * np.random.rand(50))**2
colors = np.random.rand(50)

plt.scatter(x,y, s=area, c=colors)
plt.show()
```

✓ 0.1s

Python



09

맷플롯립(Matplotlib) – 산점도 투명도와 컬러맵 설정

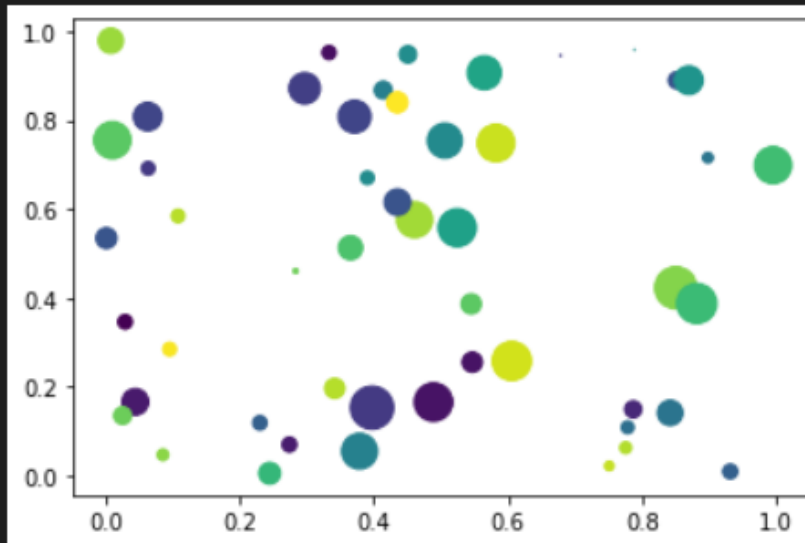
```
import numpy as np

x = np.random.rand(50)
y = np.random.rand(50)
area = (20 * np.random.rand(50))**2
colors = np.random.rand(50)

plt.scatter(x,y, s=area, c=colors)
plt.show()
```

✓ 0.1s

Python





플라스크

Flask란?

Flask는 2004년 오스트리아의 오픈소스 개발자 아르민 로나허가 만든 웹 프레임워크이다.

Flask는 홈페이지에서도 "micro" 프레임워크라는 점을 강조하고 있는데 최소한의 구성 요소와 요구 사항을 제공하기 때문에 시작하기 쉽고 필요에 따라 유연하게 사용이 가능하다.

웹 서버란?

웹 서버는 하드웨어 , 소프트웨어 두 가지 측면으로 구분 할 수 있다.

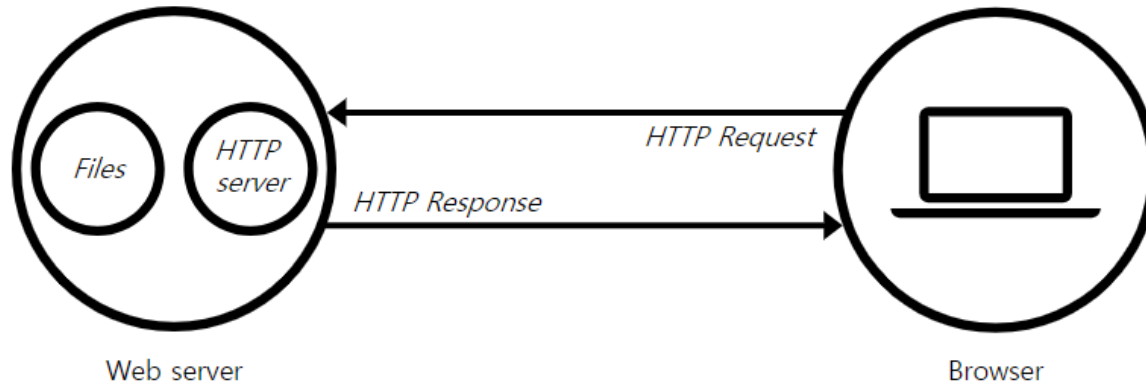
- 하드웨어

웹사이트의 컴포넌트 파일들을 저장하는 컴퓨터
컴포넌트 파일에는 HTML, Image, CSS, Javascript가 존재
컴포넌트 파일을 인터넷을 통해 클라이언트에게 전달

- 소프트웨어

사용자가 어떻게 호스트 파일들에 접근하는지 관리
웹 서버는 주소 HTTP 프로토콜을 사용하여 클라이언트의
요청을 처리 및 응답

웹 서버란?



브라우저가 웹 서버에서 불러진 파일을 필요로 할 때, 브라우저는 HTTP를 통해 파일을 요청한다. 요청이 올바른 웹 서버(하드웨어)에 도달하였을 때, HTTP 서버(소프트웨어)는 요청된 문서를 HTTP를 이용하여 보낸다.

정적 / 동적 웹 페이지

정적 웹 페이지

웹 서버에 미리 저장된 파일이 그대로 전달되는 웹 페이지
유저는 서버에 저장된 데이터가 변경되지 않는 한 고정된 페이지를 보게 된다.

동적 웹 페이지

웹 서버에 있는 데이터들을 스크립트에 의해 가공 처리 된 후 생성되어 전달되는 웹 페이지
유저는 상황, 시간, 요청 등에 따라 달라지는 웹페이지를 보게 된다.

정적 / 동적 웹 페이지

종류	장점	단점
정적 웹 페이지	속도가 빠르다 비용이 적게 든다	서비스가 한정적이다. 페이지의 수정, 추가, 삭제 등 작업이 모두 수동임으로 관리가 힘들다
동적 웹 페이지	서비스가 다양하다. 웹 사이트의 구조에 따라 추가, 수정, 삭제 작업이 용이하여 관리가 쉽다.	정적 웹 페이지에 비해 상대적으로 느리다. 웹 서버 외의 어플리케이션 서버가 필요함으로 추가 비용이 발생한다.

HTML

HTML (HyperText Markup Language)은 웹페이지를 기술하기 위한 마크업 언어이다. 조금 더 자세히 말하면 웹페이지의 내용(content)과 구조(structure)을 담당하는 언어로써 HTML 태그를 통해 정보를 구조화하는 것이다.

마크업 언어 - 특별한 기호나 표기를 사용하여 글의 서식과 스타일을 정해주는 언어 (프로그래밍 언어 X)

HTML

```
<!DOCTYPE html>
<html>
<head>
<title>Page Title</title>
</head>
<body>

<h1>This is a Heading</h1>
<p>This is a paragraph.</p>

</body>
</html>
```

HTML 문서는 반드시
<!DOCTYPE html>으로 문서의
형식을 지정해주어야 한다.

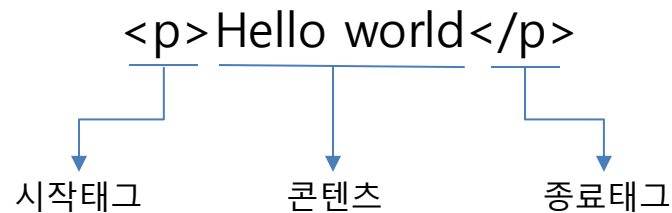
실제 html document는 2행부터
시작되고 <html>...</html>에 기
술한다.

<head>...</head>에는 title, 외
부 파일 참조, 메타데이터 설정
등이 위치하며 웹 브라우저에는
표시되지 않는다.

웹 브라우저에 표시가 되는 부분
은
<body> ...</body> 이다.

HTML 문법

- 요소(Element)



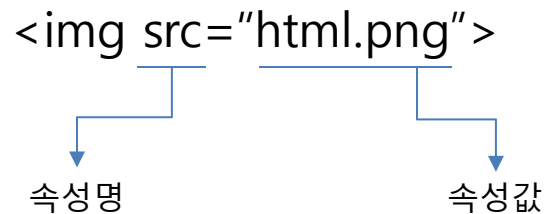
HTML 요소는 시작태그와 종료태그 사이에 위한 콘텐츠로 구성된다.

실제 HTML document는 요소들의 집합으로 이루어진다.

html 요소 (<https://www.w3schools.com/tags/default.asp>)

HTML 문법

- 속성(Attribute)



속성(Attribute)이란 요소의 성질, 특징을 정의하는 명세이다. 요소는 속성을 가질 수 있으며 요소에 추가적 정보(예를 들어 이미지 파일의 경로, 크기 등)를 제공한다.

속성은 시작 태그에 위치해야 하며 이름과 값의 쌍을 이룬다.

html 속성 (https://www.w3schools.com/tags/ref_attributes.asp)

HTML 문법

- 주석(comments)

```
<!-- 이부분이 주석입니다-->  
<h1>This is a Heading</h1>  
<p>This is a paragraph.</p>
```

주석(comment)는 주로 개발자에게 코드를 설명하기 위해 사용되며 브라우저는 주석을 화면에 표시하지 않는다.

TAG

`<!DOCTYPE html>` : 문서 형식 정의 태그는 출력할 웹 페이지의 형식을 브라우저에게 전달한다. 문서의 최상위에 위치해야 하며 대소문자를 구별하지 않는다.

`<html>` : 전체 HTML 문서를 감싸는 태그, 브라우저에게 HTML 코드가 해당 태그 내부에 존재한다고 알려준다. 하나만 존재해야 하고 HTML 바깥에 DOCTYPE을 제외한 다른 태그가 있으면 안 된다.

`<head>` : HTML 문서에 대한 정보를 나타내는 부분이며 주로 외부 소스를 참조해야 할 경우 사용한다. 하나만 존재해야 하고, HTML 바로 아래에 있어야 한다.

`<body>` : HTML 문서에서 실제로 보여지는 부분이며 하나만 존재해야 하고, html 바로 아래, head 다음에 위치해야 한다.

head Tag

title Tag

- 문서의 제목을 정의. 정의된 제목은 브라우저의 탭에 표시된다.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
  </head>
  <body>

  </body>
</html>
```

head Tag

style Tag

- HTML 문서의 style 정보를 정의한다.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
    <style>
      body {
        background-color : black;    //배경화면의 색 지정
        color : white;              //body의 폰트 색 지정
      }
    </style>
  </head>
  <body>
    <p>Test</p>
  </body>
</html>
```

head Tag

link Tag

- 외부 리소스와의 연계 정보를 정의한다. 주로 외부 CSS파일을 연계 하는데 사용한다.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
    <link rel="stylesheet" href="style.css">
  </head>
  <body>
    <p>Test</p>
  </body>
</html>
```

head Tag

script Tag

- 데이터와 실행 가능한 코드를 문서에 포함할 때 사용하며 보통 JavaScript 코드와 함께 사용한다.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
    <link rel="stylesheet" href="style.css">
    <script>
      document.addEventListener('click', function () {
        alert('Clicked!');
      });
    </script>
  </head>
  <body>
    <p>Test</p>
  </body>
</html>
```

head Tag

script Tag

- src 속성을 사용하여 외부의 Javascript 파일을 로드 할 수 있다.

```
<!DOCTYPE html>
<html>
  <head>
    <title>Page Title</title>
    <link rel="stylesheet" href="style.css">
    <script src="test.js"></script>
  </head>
  <body>
    <p>Test</p>
  </body>
</html>
```

body Tag

Headings Tag (제목 태그)

- 제목을 나타낼 때 사용하는 태그로 h1부터 h6까지 태그가 존재한다.
- h1이 가장 중요한 제목을 뜻하며 글자의 크기도 가장 크다.

```
<!DOCTYPE html>
<html>
  <body>
    <h1>heading 1</h1>
    <h2>heading 2</h2>
    <h3>heading 3</h3>
    <h4>heading 4</h4>
    <h5>heading 5</h5>
    <h6>heading 6</h6>
  </body>
</html>
```

heading 1

heading 2

heading 3

heading 4

heading 5

heading 6

body Tag

Text Tag (글자 태그)

- 굵은 글씨

```
<!DOCTYPE html>
<html>
  <body>
    <p>일반</p>
    <b>b태그</b>
    <p style='font-weight:bold'>style bold</p>
    <strong>strong 태그</strong>
  </body>
</html>
```

일반

b태그

style bold

strong 태그

body Tag

p Tag

- 단락을 지정하는 태그로 본문 내용에서 많이 사용된다.

```
<!DOCTYPE html>
<html>
  <body>
    <h1>
      HTML이란?
    </h1>
    <p>
      1980년, 유럽 입자 물리 연구소(CERN)의 계약자였었던 물리학자 팀 버너스리가 HTML의 원형인 인콰이어를 제안하였
다.
      인콰이어는 CERN의 연구원들이 문서를 이용하고 공유하기 위한 체계였다.
      1989년에 팀 버너스리는 인터넷 기반 하이퍼텍스트 체계를 제안하는 메모를 작성했다.
      버너스 리는 1990년 말에 HTML을 명시하고, 브라우저와 서버 소프트웨어를 작성했다.
      그 해에 버너스리와 CERN 데이터 시스템 엔지니어 로버트 카일리아우와 함께 CERN측에 자금 지원을 요청하였지만,
이 프로젝트는 CERN으로부터 정식으로 채택 받지 못했다. 버너스리의 개인적인 기록에 1990년부터 "하이퍼텍스트가
사용되는 여러 분야의 일부"를 열거했고
      백과사전을 그 목록의 첫 번째로 두었다.
    </p>
    <p>
      HTML 최초의 일반 공개 설명은 1991년 말에 버너스리가 처음으로 인터넷에서 문서를 "HTML 태그"(HTML tag)로 부르
면서 시작되었다.
      그것은 머릿글자로 이루어진 20개의 요소를 기술하였고, 상대적으로 HTML의 단순한 디자인이었다.
      하이퍼링크를 제외한 HTML 태그들은 CERN 자체의 SGML 기반 문서화 포맷인 SGML GUID에 강하게 영향을 받았다.
      이 요소 중 13개는 HTML 4 버전에서도 여전히 존재한다.
      HTML은 동적인 웹 페이지의 웹 브라우저를 통한 문자와 이미지 양식이다.
      문자 요소의 대부분은 1988년 ISO 기술 보고서 9537 SGML을 이용한 기법에서 찾을 수 있다.
      하지만 SGML 개념의 일반적인 마크업은 단지 개별 효과 보다는 요소 기반이고 또한 구조와 처리의 분리(?)
      (HTML은 CSS와 함께 이 방향으로 점진적으로 이동해 왔다.)
    </p>
  </body>
</html>
```

HTML이란?

1980년, 유럽 입자 물리 연구소(CERN)의 계약자였었던 물리학자 팀 버너스리가 HTML의 원형인 인콰이어를 제안하였다. 인콰이어는 CERN의 연구원들이 문서를 이용하고 공유하기 위한 체계였다. 1989년에 팀 버너스리는 인터넷 기반 하이퍼텍스트 체계를 제안하는 메모를 작성했다. 버너스 리는 1990년 말에 HTML을 명시하고, 브라우저와 서버 소프트웨어를 작성했다. 그 해에 버너스리와 CERN 데이터 시스템 엔지니어 로버트 카일리아우와 함께 CERN측에 자금 지원을 요청하였지만, 이 프로젝트는 CERN으로부터 정식으로 채택 받지 못했다. 버너스리의 개인적인 기록에 1990년부터 "하이퍼텍스트가 사용되는 여러 분야의 일부"를 열거했고 백과사전을 그 목록의 첫 번째로 두었다.

HTML 최초의 일반 공개 설명은 1991년 말에 버너스리가 처음으로 인터넷에서 문서를 "HTML 태그"(HTML tag)로 부르면서 시작되었다. 그것은 머릿글자로 이루어진 20개의 요소를 기술하였고, 상대적으로 HTML의 단순한 디자인이었다. 하이퍼링크를 제외한 HTML 태그들은 CERN 자체의 SGML 기반 문서화 포맷인 SGML GUID에 강하게 영향을 받았다. 이 요소 중 13개는 HTML 4 버전에서도 여전히 존재한다. HTML은 동적인 웹 페이지의 웹 브라우저를 통한 문자와 이미지 양식이다. 문자 요소의 대부분은 1988년 ISO 기술 보고서 9537 SGML을 이용한 기법에서 찾을 수 있다. 하지만 SGML 개념의 일반적인 마크업은 단지 개별 효과 보다는 요소 기반이고 또한 구조와 처리의 분리(?) (HTML은 CSS와 함께 이 방향으로 점진적으로 이동해 왔다.)

Hyperlink

Hyperlink 란?

하이퍼텍스트 문서 안에서 직접 모든 형식의 자료를 연결하고 가리킬 수 있는 참조 고리이다. 이를테면 동영상, 음악, 그림, 프로그램, 파일, 글 등의 특정 위치를 지정할 수 있다. 이는 하이퍼텍스트의 핵심 개념이며, HTML을 비롯한 마크업 언어에서 구현하고 있다. 이 용어는 단순히 링크(link, 고리)라고 줄여 말하기도 한다. 한마디로 누르면 웹 사이트나 프로그램 등으로 이동하는 것이다.

하이퍼텍스트는 한 문서에서 다른 문서로 즉시 접근할 수 있는 텍스트이다.

Hyperlink

디렉토리

루트 디렉토리	파일 시스템 계층 최상위 디렉토리	C:\W
홈 디렉토리	시스템의 사용자에게 할당된 개별 디렉토리	C:\WUsers\W{계정명}
작업 디렉토리	작업 중인 파일의 위치한 디렉토리	./
부모 디렉토리	작업 디렉토리의 부모 디렉토리	../

파일의 경로

절대 경로	현재 작업 디렉토리와 관계없이 특정 파일의 절대적인 위치를 지정한다.	http://www.google.com c:\Wusers\Wtest\WDesktop\Windex.html
상대 경로	현재 작업 디렉토리를 기준으로 특정 파일의 상대적인 위치를 지정한다.	./index.html ../route/second.html

Hyperlink

href 속성에서 사용 가능한 값

절대 URL	웹사이트의 URL(<code>http://www.google.com</code>)
상대 URL	자신의 위치를 기준으로 한 대상 URL (<code>html/index.html</code>)
Fragment identifier	페이지 내의 특정 id 요소에 대한 링크 (<code>#{id값}</code>)
메일	<code>mailto:</code>
script	<code>javascript:alert('test');</code>

list Tag

비순서형 리스트

```
<!DOCTYPE html>
<html>
  <body>
    <h2>
      비순서형 리스트
    </h2>
    <ul>
      <li>아메리카노</li>
      <li>카페라떼</li>
      <li>카페모카</li>
    </ul>
  </body>
</html>
```

비순서형 리스트

- 아메리카노
- 카페라떼
- 카페모카

list Tag

순서형 리스트

```
<!DOCTYPE html>
<html>
  <body>
    <h2>
      순서형 리스트
    </h2>
    <ol>
      <li>아메리카노</li>
      <li>카페라떼</li>
      <li>카페모카</li>
    </ol>
  </body>
</html>
```

순서형 리스트

1. 아메리카노
2. 카페라떼
3. 카페모카

table Tag

table tag는 웹상 테이블을 생성 시킨다.

table : 표를 감싸는 태그

tr : 표의 행을 나타내는 태그

th: 표의 열을 나타내는 태그 중 제목을 표현

td: 표의 열을 나타내는 태그

table Tag

```
<!DOCTYPE html>
<html>
  <body>
    <table border="1">
      <tr>
        <th>태그</th>
        <th>설명</th>
      </tr>
      <tr>
        <td>table</td>
        <td>표를 감싸는 태그</td>
      </tr>
      <tr>
        <td>tr</td>
        <td>표의 행을 나타내는 태그</td>
      </tr>
      <tr>
        <td>th</td>
        <td>표의 열을 나타내는 태그 중 제목을 표현</td>
      </tr>
      <tr>
        <td>td</td>
        <td>표의 열을 나타내는 태그</td>
      </tr>
    </table>
  </body>
</html>
```

태그	설명
table	표를 감싸는 태그
tr	표의 행을 나타내는 태그
th	표의 열을 나타내는 태그 중 제목을 표현
td	표의 열을 나타내는 태그

image

웹 페이지에 이미지를 삽입하는 경우 img tag를 사용한다.


img 속성

src	이미지 파일의 경로
alt	이미지 파일이 없을 경우 나타나는 메시지
width	이미지의 너비
height	이미지의 높이

image

```
<!DOCTYPE html>
<html>
  <body>
    
    
  </body>
</html>
```



이미지가 존재하지 않습니다

form Tag

form Tag는 사용자가 입력한 데이터를 수집하기 위하여 사용된다.
입력 방식으로는 input, textarea, button, select, checkbox, radio
button, submit button 등 태그들이 있다

- 속성

action : 입력 데이터가 전송될 URL 지정

method : 입력 데이터 전달 방식 지정 (get / post)

form Tag

- get

전송 URL에 입력 데이터를 쿼리스트링 형식으로 보내는 방식
전송 URL 바로 뒤에 '?'를 통하여 데이터의 시작을 알리고 'key=value'
형태의 데이터를 추가한다.

URL에 전송 데이터가 노출되기 때문에 보안에 문제가 있을 수 있으며 전송할 수 있는 데이터의 한계가 존재한다. (최대 255자)

- post

get 형식과 다르게 request body에 데이터를 담아 보내는 방식
URL에 전송 데이터가 노출되지 않아 보안적으로는 뛰어나지만 get
형식에 비해 속도가 느리다.

input Tag

form 태그 중에서 사용자로 부터 데이터를 입력 받기 위해 사용되는 태그이다.

form 태그 내에 존재하여야 입력 데이터를 전송할 수 있다.

서버에 전송되는 데이터는 name이라는 속성을 키로 value 속성을 값으로 하여 'key=value' 형태로 전송된다.

input Tag

Button	버튼 생성
Checkbox	Checkbox 생성
Color	색상 선택 생성
Date	날짜(년월일) 생성
Email	이메일 입력 폼 생성
File	파일 선택 폼 생성
Image	이미지로 된 버튼 생성
Morth	월 선택 생성
Number	숫자 입력 생성
Password	비밀번호 입력 생성
Radio	Radio 버튼 생성
Range	범위 선택 생성

Reset	초기화 버튼 생성
Search	검색어 입력 생성
Submit	제출 버튼 생성
Tel	전화번호 입력 생성
Text	텍스트 입력 생성
Time	시간 선택 생성
url	url 입력 생성
Week	주 선택 생성

input Tag

```
<!DOCTYPE html>
<html>
  <body>
    <h3>button</h3>
    <input type="button" value="click" onclick="alert('button click')">
    <br>
    <h3>checkbox</h3>
    <input type="checkbox" name="coffee1" value="coffee1" checked>아메리카노
    <input type="checkbox" name="coffee2" value="coffee2">카페라떼
    <input type="checkbox" name="coffee3" value="coffee3">카페모카
    <br>
    <h3>date</h3>
    <input type="date" name="date">
    <br>
    <h3>email</h3>
    <input type="email" name="email">
    <br>
    <h3>file</h3>
    <input type="file" name="upload">
    <br>
    <h3>number</h3>
    <input type="number" name="num" min="2" max="8" step="2" value="2">
    <br>
    <h3>password</h3>
    <input type="password" name="pwd">
    <br>
    <h3>radio</h3>
    <input type="radio" name="gender" value="male" checked>남자
    <input type="radio" name="gender" value="female">여자
    <br>
    <h3>submit</h3>
    <input type="submit" value="submit">
    <br>
    <h3>text</h3>
    <input type="text" name="text">
    <br>
    <h3>time</h3>
    <input type="time" name="time">
  </body>
</html>
```

button

checkbox

☒ 아메리카노 ☐ 카페라떼 ☐ 카페모카

date

email

file

 파일 선택 | 선택된 파일 없음

number

password

radio

☒ 남자 ☐ 여자

submit

text

time

select Tag

여러 개의 리스트 중 여러 개의 아이템을 선택할 때 사용된다.
서버에 전송되는 데이터는 select 요소의 name을 속성의 키 값으로,
option 요소의 value를 key 값으로 하여 key=value 형태로 전송된다.

select에서 사용하는 태그

Select	Select 생성
Option	Option 생성
optgroup	Option을 그룹화하여 표시

select Tag

```
<!DOCTYPE html>
<html>
  <body>
    <select name="cafe">
      <optgroup label="coffee">
        <option value="americano" selected>아메리카노</option>
        <option value="latte">카페라떼</option>
        <option value="moka">카페모카</option>
      </optgroup>
      <optgroup label="Non-coffee">
        <option value="greentea">녹차</option>
        <option value="choco">핫초코</option>
        <option value="lemonade">레몬에이드</option>
      </optgroup>
    </select>

  </body>
</html>
```

아메리카노 ▼
coffee
아메리카노
카페라떼
카페모카
Non-coffee
녹차
핫초코
레몬에이드

CSS 적용 방법

Link Style

외부에 있는 CSS 파일을 로드하는 방식

```
<!DOCTYPE html>
<html>
  <head>
    <link rel="stylesheet" href="css/style.css">
  </head>
  <body>
    <h1>Hello World</h1>
    <p>This is a web page.</p>
  </body>
</html>
```

CSS 적용 방법

Embedding Style

HTML 내부에 CSS를 포함시키는 방식

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      h1 { color: red; }
      p { background: aqua; }
    </style>
  </head>
  <body>
    <h1>Hello World</h1>
    <p>This is a web page.</p>
  </body>
</html>
```

CSS 적용 방법

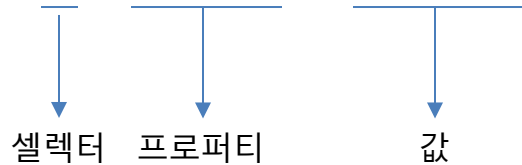
Inline Style

HTML요소의 style에 CSS를 기술하는 방식

```
<!DOCTYPE html>
<html>
  <body>
    <h1 style="color: red">Hello World</h1>
    <p style="background: aqua">This is a web page.</p>
  </body>
</html>
```

CSS 문법

p{ color : white; font-size : 12px; }



p태그의 글자의 크기는 12px, 글자의 색은 white로 지정
셀렉터는 HTML 요소를 지정한다.

복수의 셀렉터를 지정하는 경우는 (,)로 구분

h1, p { color : white; }

CSS 셀렉터(selector)

* -> 문서내의 모든 요소에 적용

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      * { color: red; }
    </style>
  </head>
  <body>
    <h1>Hello World</h1>
    <p>This is a web page.</p>
  </body>
</html>
```

Hello World

This is a web page.

CSS 셀렉터(selector)

태그명 -> 지정된 태그명을 가지는 요소만 적용

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      p { color: red; }
    </style>
  </head>
  <body>
    <h1>Hello World</h1>
    <p>This is a web page1.</p>
    <p>This is a web page2.</p>
    <p>This is a web page3.</p>
  </body>
</html>
```

Hello World

This is a web page1.

This is a web page2.

This is a web page3.

CSS 셀렉터(selector)

#id 값 -> id 속성 값과 일치하는 요소에 적용. id 속성값은 중복될 수 없는 유일한 값이다.

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      #p1 { color: red; }
    </style>
  </head>
  <body>
    <h1>Hello World</h1>
    <p id="p1">This is a web page1.</p>
    <p id="p2">This is a web page2.</p>
    <p id="p3">This is a web page3.</p>
  </body>
</html>
```

Hello World

This is a web page1.

This is a web page2.

This is a web page3.

CSS 셀렉터(selector)

.class 값 -> class 값이 일치하는 요소에 적용

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      .container { color: red; }
    </style>
  </head>
  <body>
    <h1>Hello World</h1>
    <div class="container">
      <p id="p1">This is a web page1.</p>
      <p id="p2">This is a web page2.</p>
    </div>
    <p id="p3">This is a web page3.</p>
  </body>
</html>
```

Hello World

This is a web page1.

This is a web page2.

This is a web page3.

CSS 셀렉터(selector)

class 값을 이용한 셀렉터는 여러 개 지정이 가능.

```
<!DOCTYPE html>
<html>
  <head>
    <style>
      .text-center { text-align: center; }
      .text-large { font-size: 200%;}
      .text-red {color : red}
      .text-blue {color : blue}
    </style>
  </head>
  <body>
    <h1>Hello World</h1>
    <p class="text-center">This is a web page1.</p>
    <p class="text-large text-red">This is a web page2.</p>
    <p class="text-center text-large text-blue">This is a web page3.</p>
    <p class="text-center text-large text-red text-blue">This is a web page3.</p>
  </body>
</html>
```

Hello World

This is a web page1.

This is a web page2.

This is a web page3.

This is a web page3.



**THANK
YOU**