

파이썬 프로그래밍 강의 노트 #02

자료형

자료형(Data Type)

- 전에 작성했던 코드를 다시 살펴봄

```
print("안녕하세요. 첫 번째 파이썬 프로그램입니다.")
```

```
2 + 5
```

```
2 - 5
```

```
2.3 + 5.2
```

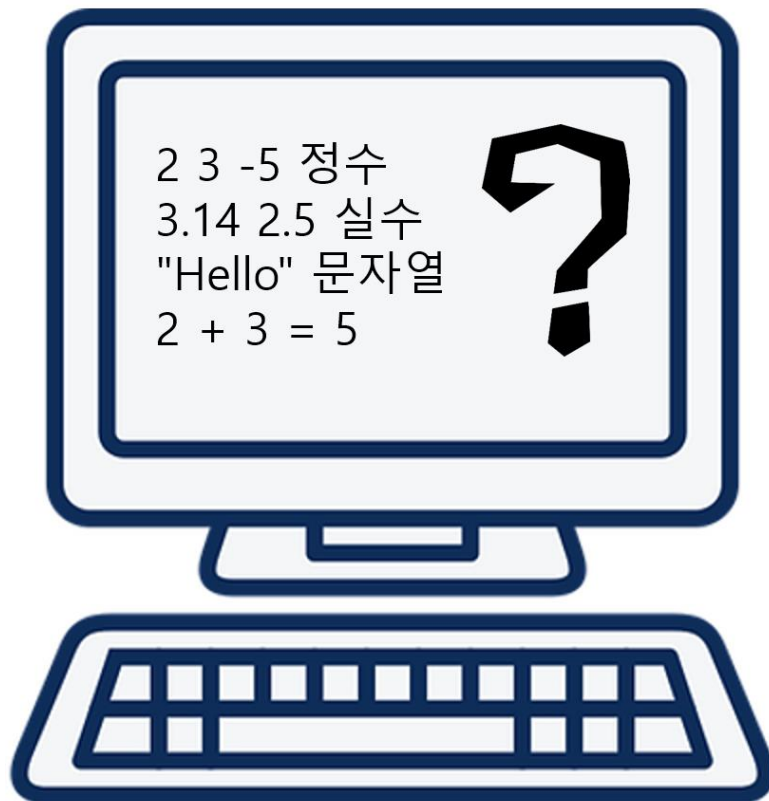
```
3 / 2
```

```
3 // 2
```

- 여기서 "안녕하세요. 첫 번째 ..." 는 문자열 값
- 2, 5는 숫자 값(정수값)
- 2.3, 5.2는 숫자 값(실수값)

자료형(Data Type)

- 어떻게 문자열인지 숫자인지 구별할까?
 - 사람에게는 쉬운 일
 - 하지만 컴퓨터라면?
 - 컴퓨터에서는 모든 것을 이진수로 표현
 - 이진수를 어떻게 문자열과 숫자로 구분?



자료형(Data Type)

- 파이썬을 비롯한 많은 프로그래밍 언어들은 다양한 목적을 위해 자료형을 사용함
 - 값의 종류를 구별
 - 문자열, 숫자, 불린값(boolean value)
 - 값의 종류가 표현할 수 있는 범위
 - 메모리에 값을 어떻게 저장할 지 결정
 - 특정 자료형에 적용시킬 수 있는 연산의 종류와 사용할 수 있는 연산자를 제한
 - 연산 결과가 어떤 종류의 값으로 나올 것인지 파악

자료형(Data Type)

- 파이썬에서는 인터프리터가 코드를 실행시키면서 사용된 값들의 자료형을 결정함
- 예
 - 코드에 따옴표가 있으면 문자열로 구분
 - 숫자만으로 구성된 내용은 정수 또는 실수로 인식
 - 2, 5는 정수형
 - 2.3, 5.2는 실수형
 - "안녕하세요. 첫 번째 ..."는 문자열형 (줄여서 문자열)

자료형에 따라 가능한 연산

▣ 자료형에 적용할 수 있는 연산 또는 연산자의 일부

피연산자1의 자료형	피연산자2의 자료형	적용 가능 연산 또는 연산자
정수	정수	덧셈, 뺄셈, 곱셈, 정수의 나눗셈, 나머지 연산
정수	실수	덧셈, 뺄셈, 곱셈, 실수의 나눗셈
실수	실수	덧셈, 뺄셈, 곱셈, 실수의 나눗셈
문자열	문자열	덧셈 (문자열 연결)

자료형에 따라 가능한 연산

▣ 만약 문자열에서 문자열을 빼다면?

```
>>> print("가나다" - "가나")
Traceback (most recent call last):
  File "<pyshell#7>", line 1, in <module>
    print(s1 - s2)
TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

파이썬의 자료형

▣ 파이썬의 기본 자료형

종류	자료형	메모리 공간 크기	표현 범위
정수	int	제한 없음	제한 없음
실수	float	8 바이트(byte)	$4.9 \times 10^{-324} \sim 1.8 \times 10^{308}$
문자열	str	제한 없음	제한 없음

자료형(Data Type)

□ 정수형 (int)

- 영어로 integer
- 소수점 없는 숫자를 표현
- 파이썬에서는 int형으로 표시 (C/Java에서도 같음)
 - C/Java의 정수형과 파이썬의 정수형은 좀 다름
 - 표현 범위의 크기 제한이 없음(unbounded)

```
2932839232923939 + 2382929292939393
2932839232923939 * 2382929292939393
2932839232923939 // 2382929292939393
```

자료형(Data Type)

□ 실수형 (float)

- 부동 소수점(floating-point number)

$0.001 + 0.0001$

$0.001 * 0.00001$

$0.001 * 0.0001$

$0.8323234243343 * 0.000000000001$

- 숫자e승수 또는 숫자E승수 형태로 사용해서 숫자 * $10^{\text{승수}}$ 표시 가능

예: $1.0e3$ # 1000.0 을 의미함

$1.0e-3$ # 0.001 을 의미함

- 일반적인 수학에서의 실수와 컴퓨터에서 저장하는 방법이 다름 → 미세한 오차 발생 가능 (산술 연산으로 이동할 것)

자료형(Data Type)

□ 문자열

- 한 쌍의 작은 따옴표(' ') 또는 큰 따옴표(" ")안에 있는 글자 또는 글자들
- 문자열은 반드시 같은 종류의 따옴표로 묶여 있어야 함
- 여러 줄의 문자열 표현 방법
 - """ 또는 ''' 사용

"""

여러 줄에 걸친
문자열을
표현하려면
따옴표 세 개로 묶으면 된다
"""

이스케이프 시퀀스

이스케이프 시퀀스	설명
\n	줄바꿈 문자
\t	탭(Tab) 문자
\\	백슬래시, \
\'	작은 따옴표, '
\"	큰 따옴표, "

```
print("백슬래시를 표현하려면 \\를 써야 합니다.")
print("첫\t번째 문장\n새로운 문장")
print("문자열 상수에서 큰 따옴표 \" 표현 방법")
print('문자열 상수에서 큰 따옴표 " 표현 방법')
print('문자열 상수에서 작은 따옴표 \' 표현 방법')
print("문자열 상수에서 작은 따옴표 ' 표현 방법")
```

문자 인코딩(Character Encoding)

- 키보드로 숫자 또는 문자를 입력하면 컴퓨터가 이해할 수 있는 0과 1의 조합으로 변환해서 전달
- 어떻게 키보드의 문자들이 이진수로 변환될까?
- 변환표가 있다고 생각하면 됨
- 인코딩
 - 문자를 컴퓨터가 이해할 수 있는 값으로 변환하는 것
- 파이썬은 UTF-8 인코딩 방식을 사용함
 - utf-8를 사용하면 한 개의 문자는 1~3바이트의 이진수로 변환됨

불린 자료형

□ 불린 자료형

- 참 또는 거짓을 나타내는 True 또는 False 값을 표현
- 코드를 작성할 때 불린 값을 생성하는 조건식을 사용해서 True일 때 혹은 False일 때 특정 작업을 실행시킬 수 있음

```
2 > 3      # False
2 < 3      # True
```

None 상수값

□ None

- NoneClass 자료형을 나타내는 상수값
 - NoneClass는 None이라는 값을 제공하기 위해 존재

□ None의 사용 목적

- 값이 존재하지 않음을 표현
- 값을 정확하게 알 수 없는 경우를 표현
- 값이 정의되지 않은 경우를 표현

type() 명령

□ type() 명령

- 값이 어떤 자료형인지 확인 가능
- 파이썬 쉘에서 type() 명령어 사용해보기

```
>>> type("안녕하세요")
<class 'str'>
>>> type('안녕하세요')
<class 'str'>
>>> type(2)
<class 'int'>
>>> type(2.3)
<class 'float'>
>>> type(a < 2)
<class 'bool'>
>>> type(True)
<class 'bool'>
```


리터럴(Literal) 또는 리터럴 상수(Literal Constant)

□ 리터럴

- 리터럴 상수라고도 함
- 코드에서 직접적으로 명시되는 숫자값이나 문자열

```
>>> 5 + 7.4  
12.4  
>>> print("안녕하세요")  
안녕하세요
```

int(), float(), str() 명령

□ int()

- 실수값 또는 소수점 없는 숫자로만 구성된 문자열을 정수로 변환해서 반환

```
>>> int("123") # 정수 문자열을 정수로 변환
123
```

```
>>> int(123.6) # 소수점 이하 버림 (반올림 안함)
123
```

- 잘못된 문자열이나 숫자가 전달되면 오류 발생

```
>>> int("123.0")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base 10: '123.0'
```

int(), float(), str() 명령

```
>>> int("123a")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base
10: '123a'
>>> int("a123")
Traceback (most recent call last):
  File "<stdin>", line 1, in <module>
ValueError: invalid literal for int() with base
10: 'a123'
```

int(), float(), str() 명령

□ float() 명령

- 정수값 또는 숫자들로 구성된 문자열을 실수로 변환

```
>>> float("123.2") # 실수형 문자열을 변환  
123.2
```

```
>>> float("123") # 정수형 문자열도 실수값으로 변환  
123.0
```

```
>>> float(123) # 정수를 전달해도 실수값으로 변환  
123.0
```

int(), float(), str() 명령

- 잘못된 문자열이 전달되면 오류 발생

```
>>> float("123.0a")
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
ValueError: could not convert string to float:
'123.0a'
>>> float("a123.0")
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
ValueError: could not convert string to float:
'a123.0'
```

int(), float(), str() 명령

- str() 명령은 주어진 값을 문자열로 변환

```
>>> str(123.4)
'123.4'
>>> str(123)
'123'
```

- 불린 값을 "True" 또는 "False" 문자열로 변환할 수 있음

```
>>> str(True) # 불린값을 문자열로 변환
'True'
>>> str(3 < 2)
'False'
```

표현식과 문장

□ 표현식(expression)

- 값을 만들어내는 코드
- 상수, 연산(계산) 결과, 변수, 함수 호출 등

□ 명령문(statement)

- 컴퓨터에 작업을 수행하라고 명령하는 코드
- 표현식 또는 명령문으로 구성 가능
- 대입문, 조건문, 반복문 등

상수

□ 상수 (constant)

- 프로그램 실행 중에 변하지 않는 값/내용
- 코드에 사용된 숫자, 문자열
- 아래에서 보면
 - "안녕하세요. 첫 번째 파이썬 프로그램입니다."
 - 2, 5, 2.3, 5.2

```
print("안녕하세요")
```

```
2 + 5
```

```
2 - 5
```

```
2.3 + 5.2
```

```
2.3 - 5.2
```


산술 연산

□ 산술 연산 (arithmetic operations)

- 정수 연산 결과는 정수, 실수 연산 결과는 실수,
- 정수, 실수 연산 결과는 실수
- 연산자 앞뒤에 붙는 공백문자는 연산에 영향 없음

산술 연산자	설명	사용 예
+	덧셈	$2 + 3$, $2.1 + 3.2$
-	뺄셈	$2 - 3$, $2.1 - 3.2$
*	곱셈	$2 * 3$, $2.1 * 3.2$
/	실수 나눗셈	$2 / 3$, $2.1 / 3.2$
//	정수의 나눗셈	$2 // 3$, $5 // 2$, $3.1 // 2.1$
%	정수의 나머지	$2 \% 3$, $5 \% 2$, $3.1 \% 2.1$
**	거듭제곱	$2 ** 3$, $2.1 ** 3.2$

산술 연산

▣ 다음을 연산하고 결과를 확인하자

```
2 + 5
2 - 5
2 * 5
5 / 2
5 // 2
5 % 2
2 / 5
2 // 5
2 ** 5
```

```
2.3 + 5.2
2.3 - 5.2
2.3 * 5.2
5.2 / 2.3
5.2 // 2.3
5.2 % 2.3
2.3 / 5.2
2.3 // 5.2
2.3 ** 5.2
```

산술 연산 순서 (연산자 우선 순위)

□ 산술 연산자 우선 순위

- 기본적인 사칙연산에 대해서는 수학과 같음
- 같은 우선 순위에서는 왼쪽에서 오른쪽 방향으로 계산
- 불확실하면 괄호를 쓸 것

산술연산자 우선 순위	연산자
괄호	()
거듭제곱	**
곱셈, 나눗셈, 정수의 나눗셈, 나머지	*, /, //, %
덧셈, 뺄셈	+, -

- 오른쪽 결과 값은?

$2 + 3 * 4 ** 2$ $((2 + 3) * 4) ** 2$

정수 계산하기

값을 정수로 만들기

계산 결과가 실수로 나왔을 때 강제로 정수로 만들어보기
int에 괄호를 붙이고 숫자 또는 계산식을 넣으면 됨
int에 문자열을 넣어도 정수로 만들 수 있음
단, 정수로 된 문자열 이라야 함

- int(숫자)
- int(계산식)
- int('문자열')

```
>>> int(3.3)
3
>>> int(5 / 2)
2
>>> int('10')
10
```

- int는 정수(integer)를 뜻하며 값을 정수로 만들어 줌(소수점 이하는 버림)

실수 계산하기

실수 계산하기

실수끼리 계산해보자

>>>에 3.5 + 2.1를 입력해보자

```
>>> 3.5 + 2.1  
5.6
```

덧셈의 결과 → 5.6

뺄셈, 곱셈, 나눗셈 실행

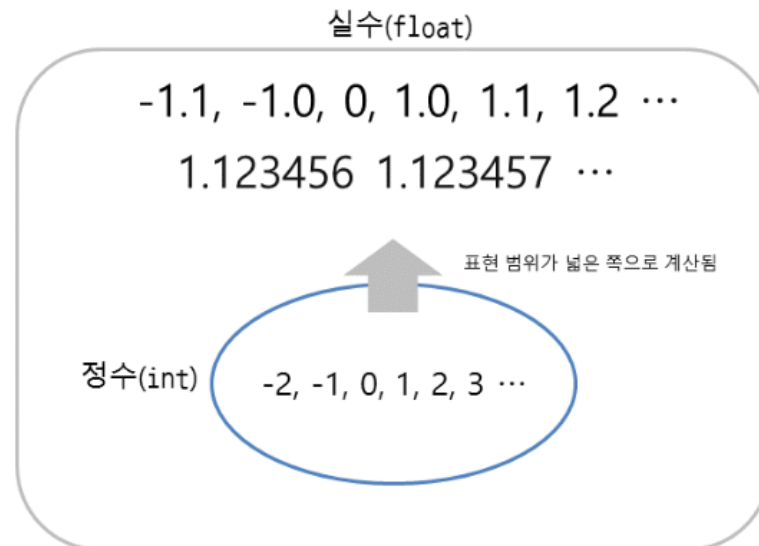
```
>>> 4.3 - 2.7  
1.5999999999999996  
>>> 1.5 * 3.1  
4.65  
>>> 5.5 / 3.1  
1.7741935483870968
```

실수 계산하기

실수와 정수를 함께 계산해보자

```
>>> 4.2 + 5  
9.2
```

실수와 정수를 함께 계산하면 표현
범위가 넓은 실수로 계산됨



실수 계산하기

□ 값을 실수로 만들기

- 숫자 또는 계산 결과를 강제로 실수로 만들려면 어떻게 해야 할까요?
- float에 괄호를 붙이고 숫자 또는 계산식을 넣으면 됨
- float에 문자열을 넣어도 실수로 만들 수 있음
- 단, 실수 또는 정수로 된 문자열이어야 함

- float(숫자)
- float(계산식)
- float('문자열')

```
>>> float(5)
5.0
>>> float(1 + 2)
3.0
>>> float('5.3')
5.3
```

실수 계산하기

□ 복소수

- 파이썬에서는 실수부와 허수부로 이루어진 복소수(complex number)도 사용할 수 있음
- 이때 허수부는 숫자 뒤에 j를 붙임(수학에서는 허수를 i로 표현하지만 공학에서는 j를 사용)

```
>>> 1.2+1.3j  
(1.2+1.3j)
```

두 실수를 복소수로 만들 때는 complex를 사용

```
>>> complex(1.2, 1.3)  
(1.2+1.3j)
```


실습문제 1

□ $\frac{22.44 \times 30}{13.10 + 40}$ 를 을 계산하는 코드를 작성하고 결과를 화면에 출력

□ 코드



실습문제 1

□ $\frac{22.44 \times 30}{13.10 + 40}$ 를 을 계산하는 코드를 작성하고 결과를 화면에 출력

□ 코드

```
print((22.44 * 30) / (13.10 + 40))
```

실습문제 2

▣ 다음 연산의 결과는?

$$a = 50$$

$$b = 25$$

$$c = 10$$

$$d = 5$$

$$\text{결과} = (a + b) * c / d - b$$

실습문제 3

▣ 다음 연산의 결과는?

$$a = 15$$

$$b = 6$$

$$c = 3$$

$$d = 2$$

$$\text{결과} = a * (b + c) / d + b - c$$

실습문제 4

▣ 다음 연산의 결과는?

$$a = 8$$

$$b = 4$$

$$c = 2$$

$$d = 1$$

$$\text{결과} = ((a + b) * c) / (d + b) - a$$

실습문제 5

▣ 다음 거듭 제곱의 결과는?

$$a = 2$$

$$b = 3$$

$$c = 4$$

$$\text{결과} = a ** b + b ** c - (a + c)$$

실습문제 6

▣ 다음 연산의 나머지 결과는?

$$a = 100 \quad b = 7 \quad c = 3$$

$$\text{결과} = (a \% b) * c + b // c$$

실습문제 7

▣ 다음 혼합 연산의 결과는?

$a = 12$

$b = 5$

$c = 3$

$d = 2$

결과 = $(a // b + c) ** d - a \% c$

실습문제 8

▣ 다음 소수점 연산의 결과는?

$$a = 15.5$$

$$b = 2.3$$

$$c = 4.1$$

$$\text{결과} = (a / b) + (c * b) - (a // c)$$

실습문제 9

▣ 다음 중첩 연산의 결과는?

$$a = 9$$

$$b = 3$$

$$c = 2$$

$$d = 4$$

$$e = 5$$

$$\text{결과} = ((a + b * c) // d) ** e - a \% b$$

문자열과 덧셈/곱셈 연산자

□ 문자열을 피연산자로 취하는 덧셈 연산자

- 피연산자에 해당되는 문자열을 연결해서 새로운 문자열 생성

```
>>> "가나다" + "abc"  
'가나다abc'
```

□ 문자열과 곱셈 연산자

- 문자열 * 정수
- 문자열을 정수 피연산자 개수 만큼 연결해서 새로운 문자열 생성

```
>>> "가나다" * 3  
'가나다가나다가나다'
```

문자열과 덧셈/곱셈 연산자

- 산술연산자의 결합 순서와 우선 순위가 그대로 적용됨
- 왼쪽부터 오른쪽 순서대로 계산

```
>>> "가나다" + "abc" * 2  
'가나다abcab'
```

- 산술연산자의 우선 순위가 적용됨

- 곱셈이 먼저 사용됨

```
>>> "가나다" + "abc" * 2  
'가나다abcab'
```

- 괄호가 먼저 사용됨

```
>>> ("가나다" + "abc") * 2  
'가나다abcfg나다abc'
```

실습문제 1

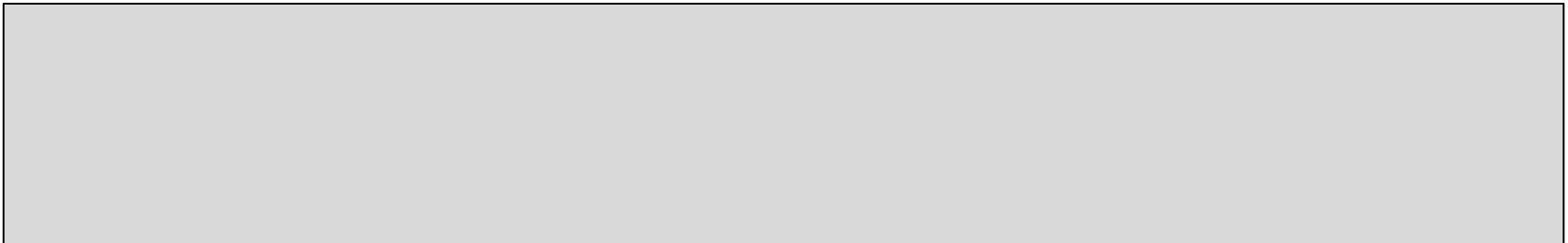
□ 문제

- 가로 3 x 세로 3 인 정방 행렬을 문자열 *로 표시해보자,
- Sep 옵션 활용하여 개행 표시

□ 요구사항

- 문자열에 대한 덧셈 또는 곱셈 연산자 활용

□ 코드



실습문제 1

□ 문제

- 가로 3 x 세로 3 인 정방 행렬을 문자열 *로 표시해보자,
- Sep 옵션 활용하여 개행 표시

□ 요구사항

- 문자열에 대한 덧셈 또는 곱셈 연산자 활용

□ 코드

```
print("*" * 3, "*" * 3, "*" * 3, sep="\n")
```

실습문제 2

- ▣ 다음 두 문자열을 덧셈 연산하여 하나의 문자열로 만드세요.

```
str1 = "Python"  
str2 = "Programming"
```



실습문제 3

- ▣ 다음 두 문자열을 덧셈 연산하여 하나의 문자열로 만드는데, 두 문자열 사이에 공백을 추가하세요.

```
str1 = "Data"  
str2 = "Science"
```



실습문제 4

- ▣ 다음 두 숫자를 문자열로 변환하여 덧셈 연산을 수행하세요.

num1 = 123

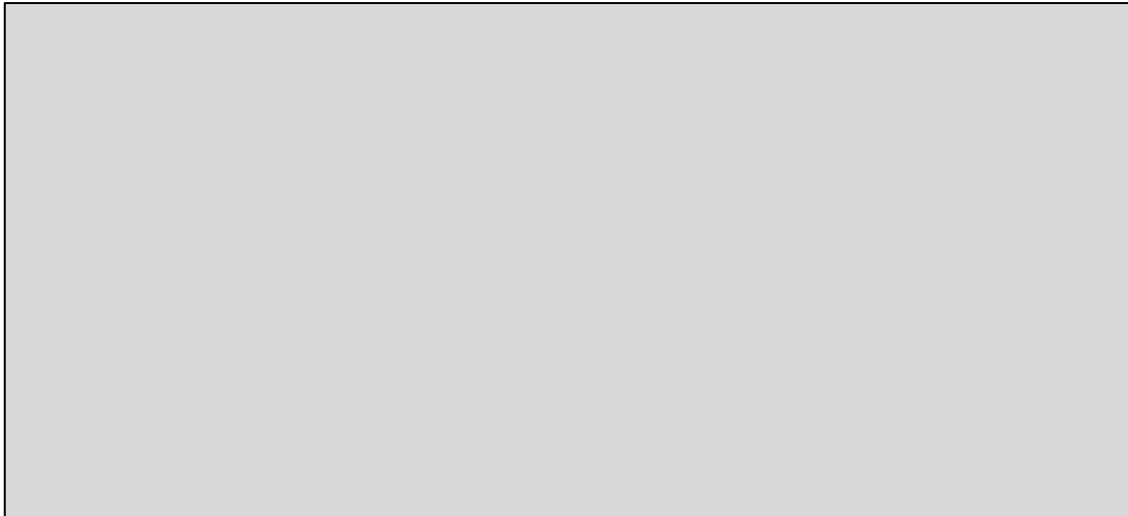
num2 = 456



실습문제 5

- ▣ 다음 문자열을 5번 반복하여 하나의 문자열로 만드세요.

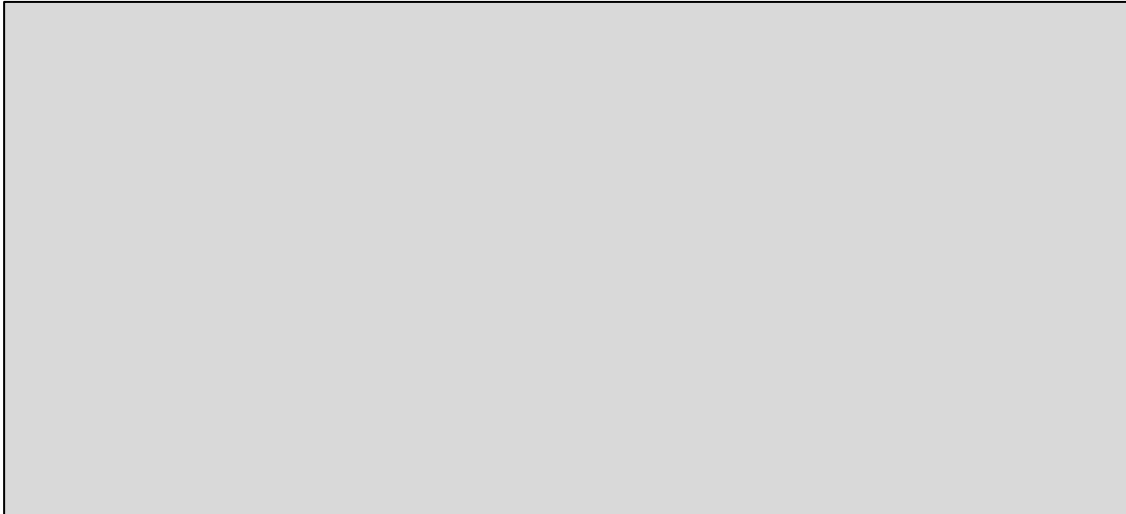
```
str1 = "Repeat"
```



실습문제 6

- ▣ 다음 문자열을 3번 반복하고, 각 반복 사이에 공백을 추가하세요.

```
str1 = "Echo"
```



실습문제 7

- ▣ 다음 문자열을 4번 반복하고, 각 반복 사이에 줄바꿈 문자를 추가하세요.

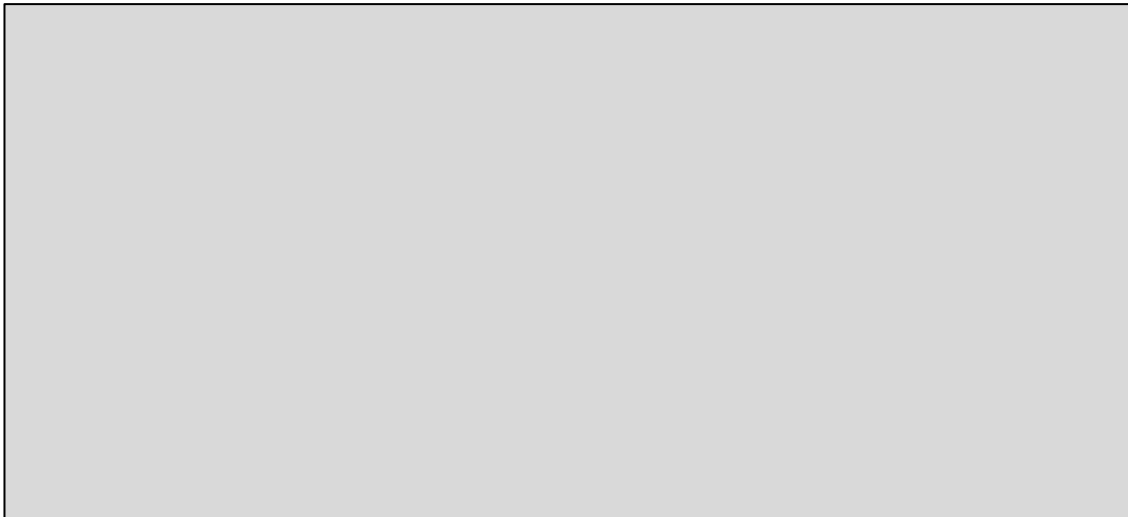
```
str1 = "Line"
```



실습문제 8

- ▣ 다음 문자열을 2번 반복하고, 그 결과를 다른 문자열과 덧셈 연산을 수행하세요.

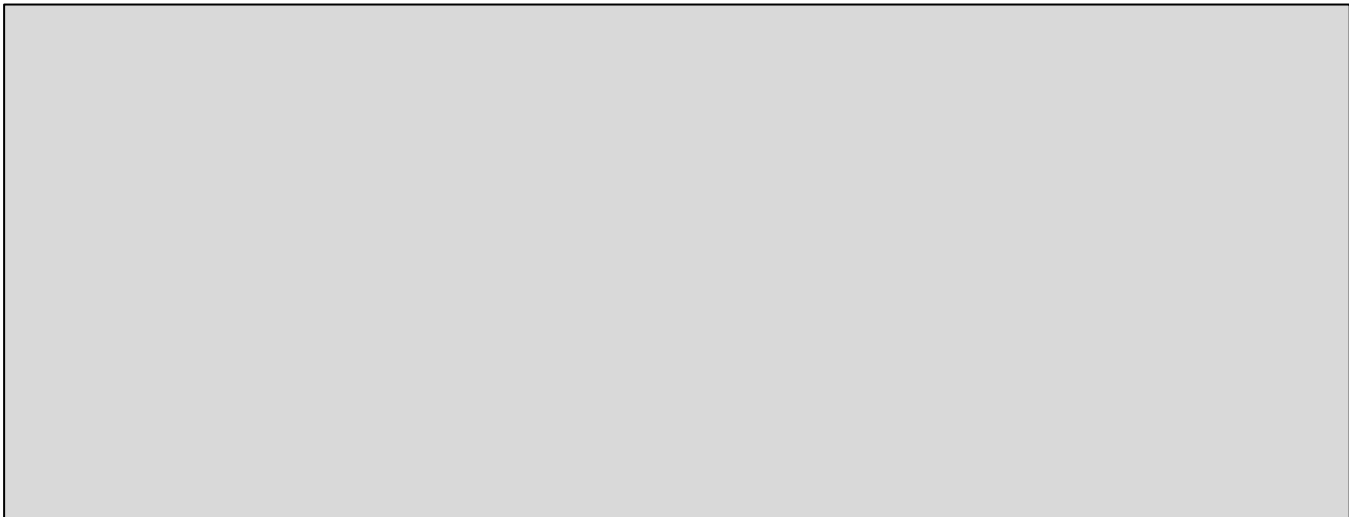
```
str1 = "Hello"  
str2 = "World"
```



실습문제 9

- ▣ 다음 문자열을 3번 반복하고, 다른 문자열과 덧셈 연산을 수행한 후, 결과 문자열을 다시 2번 반복하세요.

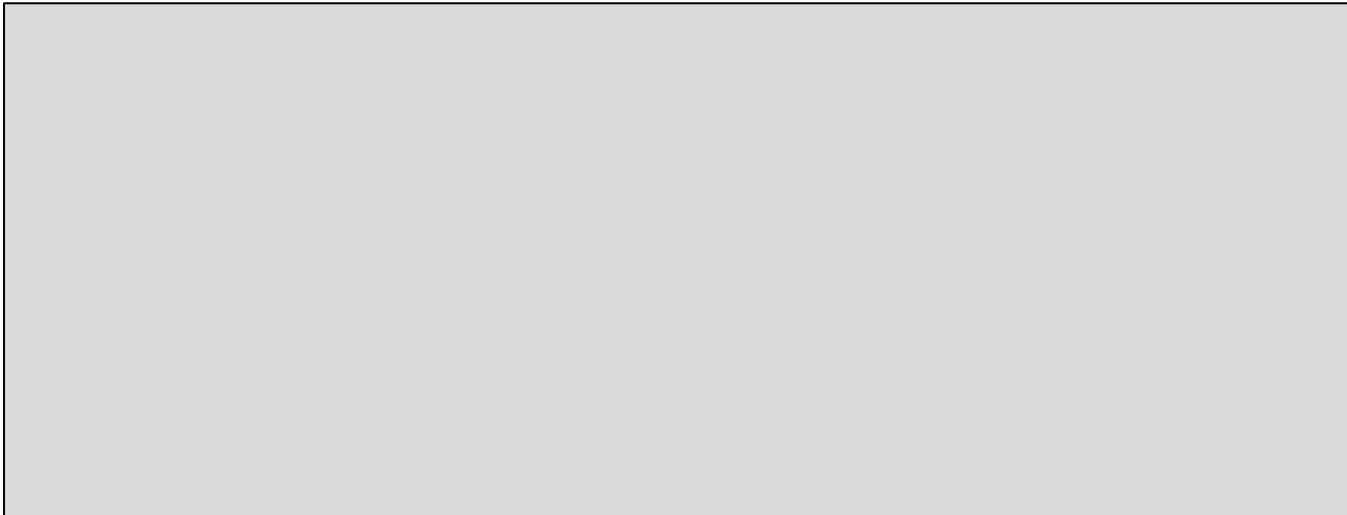
```
str1 = "Fun"  
str2 = "Python"
```



실습문제 10

- ▣ 다음 문자열을 주어진 수만큼 반복하고, 반복한 문자열을 다른 문자열과 연결하세요.

```
str1 = "Python"  
n = 3  
str2 = "Rocks"
```



실습문제 11

- 다음 두 문자열을 각각 주어진 수만큼 반복하고, 두 결과 문자열을 연결하세요.

```
str1 = "AB"
```

```
n1 = 2
```

```
str2 = "CD"
```

```
n2 = 3
```



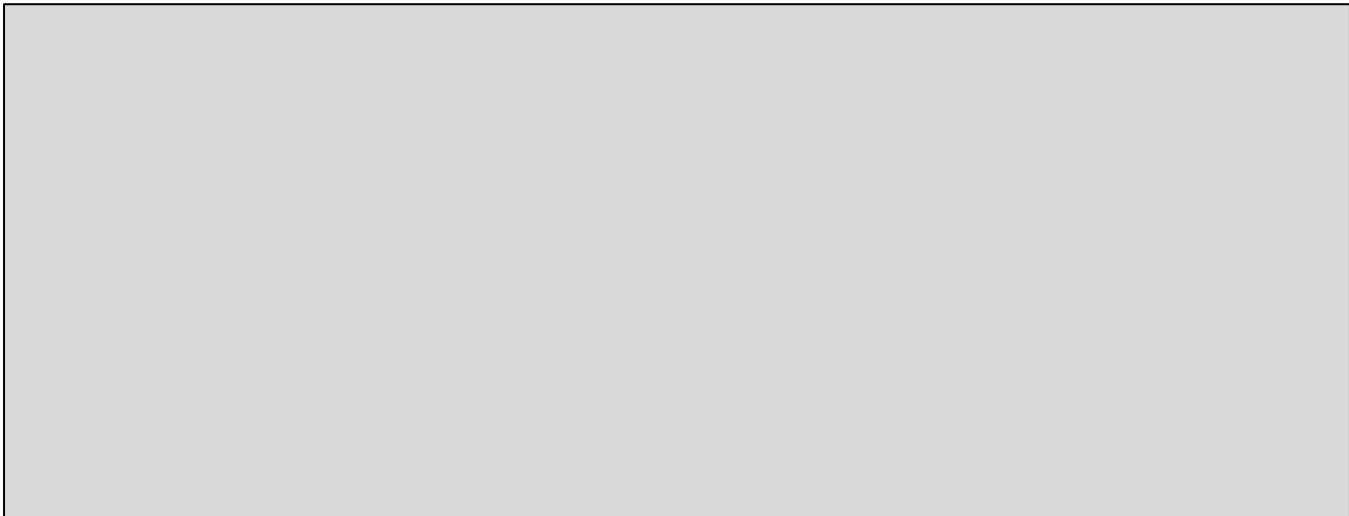
실습문제 12

- ▣ 다음 문자열을 반복하고, 그 사이에 다른 문자열을 삽입하여 결과 문자열을 만드세요.

```
str1 = "123"
```

```
n = 4
```

```
str2 = "-"
```



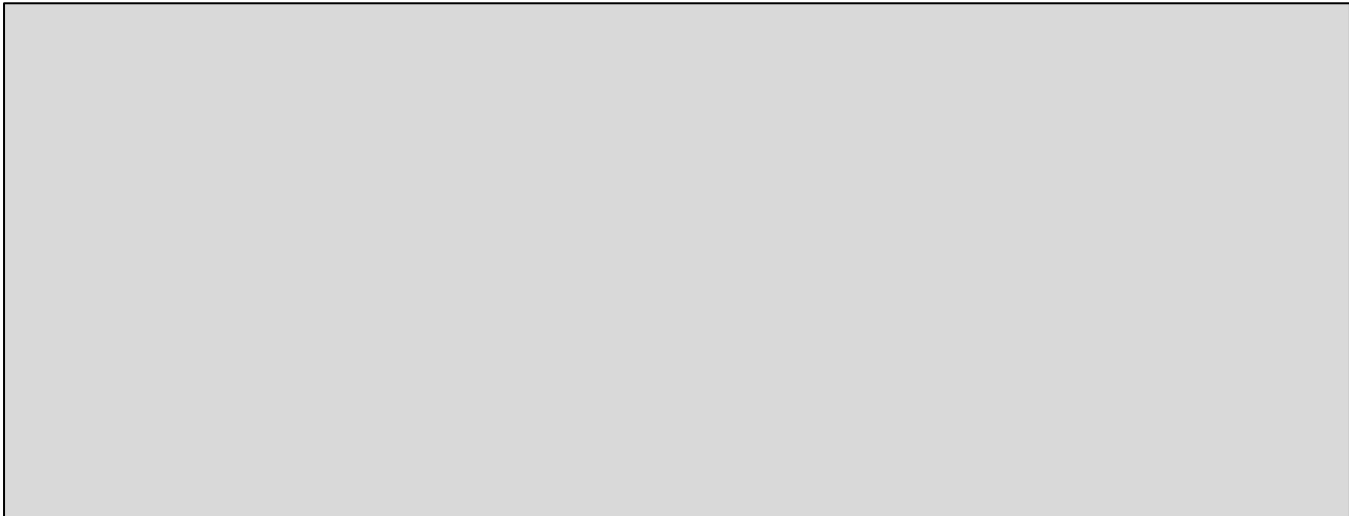
실습문제 13

- 다음 문자열을 반복한 후, 각 반복된 문자열 사이에 특정 문자를 추가한 결과 문자열을 만드세요.

```
str1 = "Cat"
```

```
n = 5
```

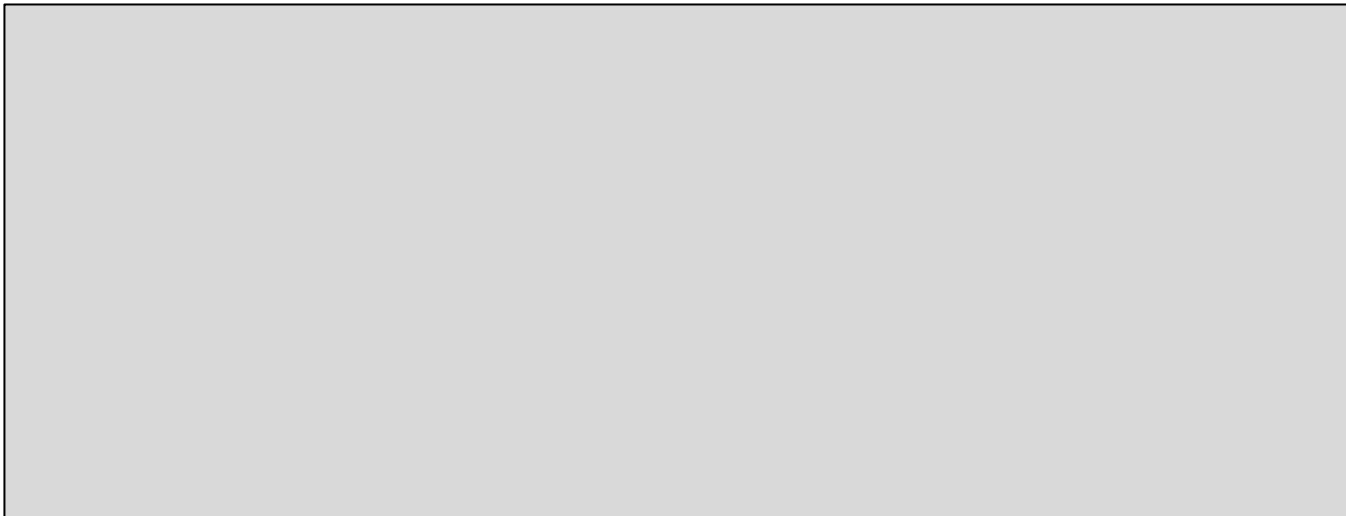
```
char = "*"
```



실습문제 4

- ▣ 다음 문자열을 반복하고, 다른 문자열을 덧셈 연산으로 추가한 후, 전체 결과를 다시 반복하여 하나의 문자열로 만드세요.

```
str1 = "Go"  
n1 = 3  
str2 = "Stop"  
n2 = 2
```



How to learn Python programming | Guido van Rossum

