



문화 A0019

파이썬프로그래밍

김 태 완

kimtwan21@dongduk.ac.kr

print 함수

- print 함수는 가장 기본!

```
print("hello")  
print("데이터사이언스")
```

```
print("100")  
print("%d" % 100)
```

```
print("100+100")  
print("%d" % (100+100))
```

```
print("%d / %d = %d" % (200, 100, 200/100))
```

print 함수

- Print 함수에서 사용할 수 있는 서식
 - %d : 정수
 - %f : 실수 (소수점이 붙은 수)
 - %s : 문자열

```
print("%d" % 10)
```

```
print("%f" % 3.141592)
```

```
print("%s" % "데이터")
```

print 함수

- 이스케이프 문자

이스케이프 문자	역할	설명
\n	새로운 줄로 이동	<code>Enter</code> 를 누른 효과
\t	다음 탭으로 이동	<code>Tab</code> 을 누른 효과
\b	뒤로 한 칸 이동	<code>Backspace</code> 를 누른 효과
\'	'를 출력	
\"	"를 출력	
\\	\를 출력	

```
print("안녕하세요\n")
print("안녕하세요\t 저는...")
print("\\를 출력")
print("\'를 출력")
print("\"를 출력")
```

데이터형

- 데이터형 (data type)
 - 자료를 기능과 역할에 따라 구분한 것 (자료의 형식)
 - 4가지 기본 데이터형
 - int: 정수형 / float: 실수형 / str: 문자형 / bool: 불형

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

print(type(var1))
print(type(var2))
print(type(var3))
print(type(var4))
```

데이터형 - 숫자형

- 정수와 정수의 연산은 정수 / 실수와 실수의 연산은 실수
- 정수와 실수의 연산은 실수 / 정수 나누기 정수는 실수

```
x = 10
```

```
y = 5.5
```

```
z = x+y
```

```
print(z)
```

```
print(type(z))
```

```
x = 10
```

```
y = 5
```

```
z = x/y
```

```
print(z)
```

```
print(type(z))
```

데이터형 – 문자형

- 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
 - 더하기(+) 연산자 사용 : 띄어쓰기 없이 문자열이 연결됨 (뿔셈, 곱셈, 나눗셈은 오류 발생)
 - 단 문자열에 숫자를 곱하는 것은 가능함

```
a = "동덕여대 데이터사이언스"  
print(a)  
print(type(a))
```

```
b = "동덕" + "여대" + " 데이터" + "사이 언 스!"  
print(b)
```

데이터형 - 문자형

- 불형 (bool)
 - 참(True)이나 거짓(False)만 저장할 수 있는 데이터 형식
 - 논리형이라고도 함

```
x = (100 > 10)
print(x)
```

```
y = (2 < 1)
print(y)
```

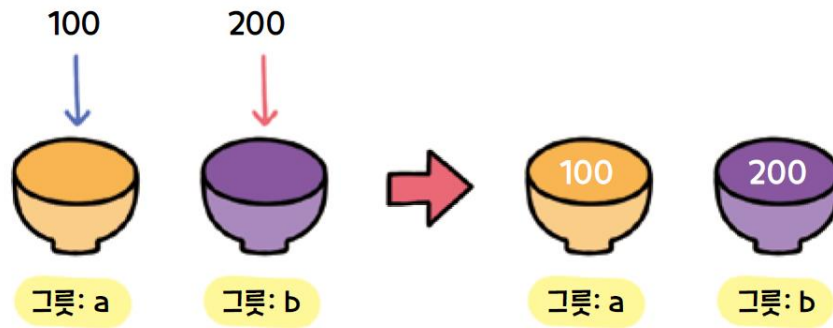
```
z = True
print(z)
```


변수

- 변수
 - 값을 저장하는 메모리 공간
 - 좀 더 쉽게 무엇을 담는 그릇이라 생각할 수 있음
 - 한 번 사용되고 사라지는 코드

```
print(100+200)
```

- 100과 200을 저장하기 위해서는 100과 200을 담을 그릇(변수)가 필요함



변수

- 변수
 - 그릇(변수) a, b를 선언하고 100, 200을 대입하기
 - 대입 연산자(=) 사용
 - 수학에서 =는 양변이 같음을 의미하지만, 프로그램에서는 할당 ASSIGNMENT를 의미

```
a=100
b=200
<----- 아무 것도 나오지 않음
```

- 두 그릇(변수) a, b에 들어있는 값을 더해 새로운 그릇 c에 담기

```
a=100
b=200
c=a+b
```

변수

- 올바른 값의 대입
 - 대입 연산자인 =이 나오면 무조건 =의 오른쪽 부분이 모두 계산된 후에 왼쪽으로 대입됨
 - 따라서 모든 코드에서 =의 왼쪽에는 변수가 있어야 함
 - =의 오른쪽이 모두 변수일 필요는 없음
 - 변수-변수의 연산 / 값-값의 연산 / 변수-값의 연산 모두 가능
 - 대입 연산자의 왼쪽에 기존에 사용했던 변수가 나오면 이전 값은 없어지고, 새로운 값으로 덮어 씌워짐

```
a = 1
b = 2

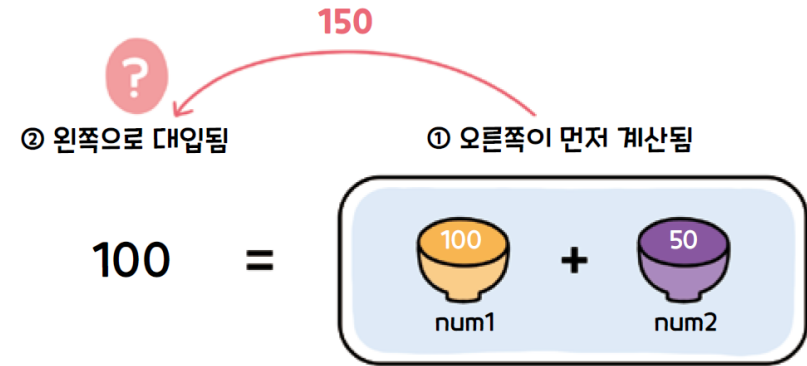
b = a+3
print(b)
```

변수

- 잘못된 값의 대입 예
 - 대입 연산자 =의 왼쪽이 변수가 아니라면 오류 발생

```
100 = a+b
```

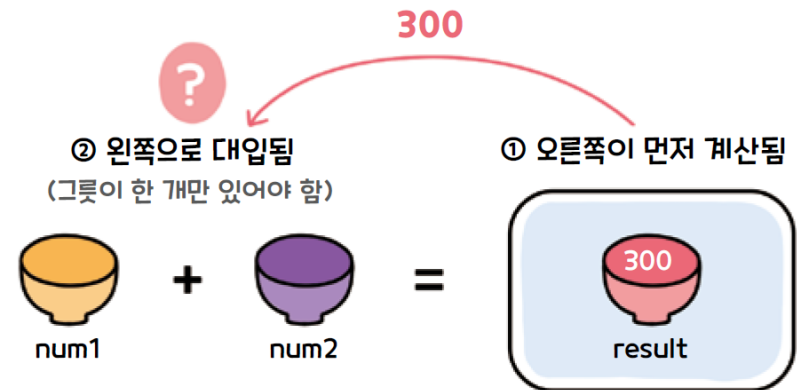
```
SyntaxError: cannot assign to literal
```



- 대입 연산자 =의 왼쪽 변수는 한 개만 존재해야 함

```
a+b=c
```

```
SyntaxError: cannot assign to operator
```



변수

- 변수 이름 짓기

- 변수명은 영문 및 숫자만 사용할 수 있음 (영어는 대소문자 구분 – love, Love, lOvE, LoVE 서로 다른 변수명)
- 변수의 이름만 보고도 의미를 파악할 수 있어야 함
 - 띄어쓰기 허용x: 변수명이 길어지면 의미 파악이 어려움
 - 띄어쓰기 대신 언더바 (_)를 많이 사용함
- 변수명은 영문 및 숫자만 사용할 수 있으며, 영문과 숫자를 섞어서 사용 가능

```
c = 100
C = 200
zz_zz = 300
P1234 = 400
its4you = 600
```

```
first_num = 100    <----- First Number의 약자
num_input = 200    <----- 입력된 숫자
inputDate = 30     <----- 입력된 날짜
```

- 단, 영문으로 시작해야 함

```
>>> 333 = 100    <----- 숫자로만 이루어져 오류 발생
SyntaxError: cannot assign to literal

>>> 3abcd = 200  <----- 숫자로 시작해 오류 발생
SyntaxError: invalid syntax
```

데이터형 - 문자형

- 변수 이름 짓기
 - 변수명에 예약어를 사용할 수 없음
 - 예약어: 이미 파이썬 문법에 정의되어 사용되는 단어(if, else 등)
 - 변수명으로 예약어를 사용하면 오류 발생함

```
if = 100  
SyntaxError: invalid syntax
```

- 함수명을 변수로 사용할 경우 문법상 오류 발생 x
 - 다만 본래의 기능을 상실

```
print=100  
Traceback (most recent call last): ...  
error message
```

True	False	None	if	elif
continue	def	finally	else	for
pass	while	with	try	except
break	class	return	import	as

변수 값을 입력 받는 함수

- input () 함수
 - 키보드로 입력 받도록 도와주는 함수
 - input()으로 입력 받는 데이터는 모두 문자열 (str)로 인식
 - input() 함수로 값을 입력 받고 변수에 저장하지 않으면 화면에 출력한 후, 그냥 사라짐
 - 따라서 input() 함수는 입력된 값을 변수에 저장하는 것이 일반적임

```
>>> input()
100
'100'
```

```
>>> num1 = input()
100          <----- 사용자 입력
              <----- 아무 메시지도 나오지 않음
```

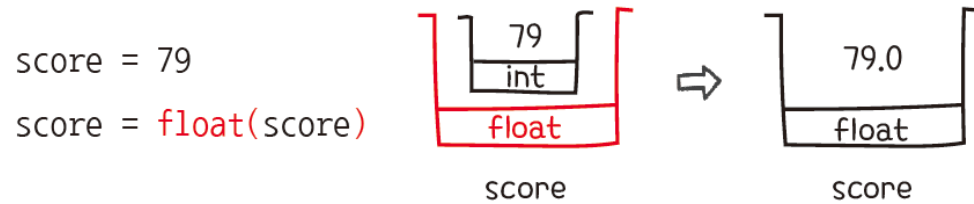
```
>>> num2 = input("숫자 ==> ") <----- 괄호안에 메시지를 넣어 입력 가이드하기
숫자 ==> 200      <----- 사용자 입력
              <----- 아무 메시지도 나오지 않음
```

```
>>> result1 = num1 + num2
          <----- 아무 메시지도 나오지 않음
>>> print(result1)
숫자 ==> 100200
```

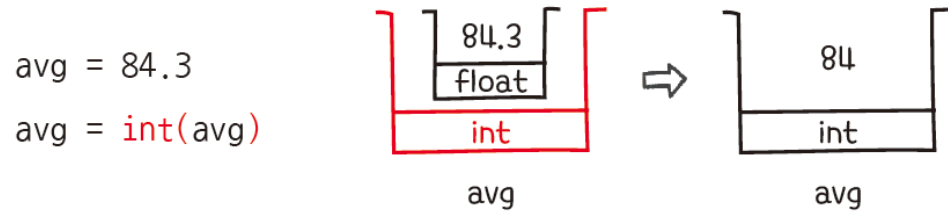
변수 값을 입력 받는 함수

- 데이터 타입 바꾸기

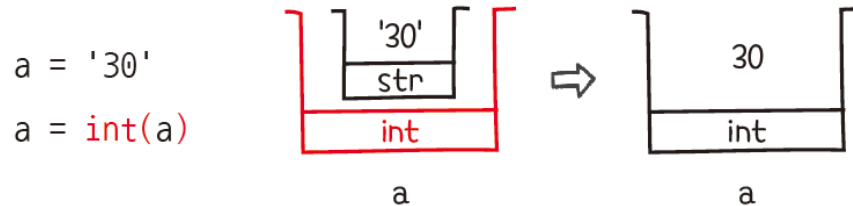
- 정수형 데이터를 실수형으로 변환



- 실수형 데이터를 정수형으로 변환



- 문자열 데이터를 정수형으로 변환



```
>>> num1 = int(input("숫자1 ==>"))
숫자1 ==> 100
>>> num2 = int(input("숫자2 ==>"))
숫자2 ==> 200
>>> result = num1 + num2
>>> print(result)
300
```


변수 값을 입력 받는 함수

- 학생들에게 이름과 학번을 입력 받아 출력하는 프로그램

```
>>> name = input("이름 ? : ")
이름 ? : 김태완
>>> univ_id = input("학번 ? : ")
학번 ? : 20221234
>>> print("제 이름은 ", name, "이고, 제 학번은", univ_id, " 입니다.")
제 이름은 김태완이고, 제 학번은 20221234 입니다.
```

- 실습: 택배 정보를 입력 받아 배송비와 함께 출력하는 프로그램을 만들어 봅시다.

실행 결과

```
## 택배를 보내기 위한 정보를 입력하세요. ##
받는 사람 : 김난생 ●———— 사용자 입력
주소 : 서울 영등포구 여의도동 88 ●———— 사용자 입력
무게(g) : 721 ●———— 사용자 입력
** 받는 사람 ==> 김난생
** 주소 ==> 서울 영등포구 여의도동 88
** 배송비 ==> 3605 원
```



연습 문제

01 다음 코드의 실행 결과를 고르시오.

```
a = 200  
b = 300  
c = a + b  
print( a, '+', b, '=', c)
```

- ① $a + b = c$
- ② $200 + 300 = 500$
- ③ 200, +, 300, =, 500
- ④ 200 '+' 300 '=' 500

02 다음 코드를 실행했을 때 result 변수에 최종적으로 저장되는 값을 고르시오.

```
number1 = 200  
number2 = 300  
result = number1 + 200
```

- ① 200
- ② 300
- ③ 400
- ④ 500

연습 문제

03 다음 중 문법상 오류가 발생하는 코드를 고르시오.

- ① `a = 100`
- ② `b = 200`
- ③ `a + b = 300`
- ④ `a = b + 300`

04 다음 코드를 실행했을 때 `result1`과 `result2`에 들어갈 값을 차례대로 고르시오.

```
number1 = 10  
number2 = 2  
result1 = number1 * number2  
result2 = number1 / number2
```

- ① 20, 5.0
- ② 5.0, 20
- ③ 20, 20
- ④ 5.0, 5.0

연습 문제

05 다음은 문자열의 덧셈이다. 실행 결과를 고르시오.

```
string1 = "안녕"  
string2 = "2"  
print(string1 + string2)
```

- ① 안녕안녕
- ② 안녕
- ③ 22
- ④ 안녕2

06 다음 코드를 실행한 후에, 키보드로 200과 300을 입력했다면 출력될 결과를 고르시오.

```
number1 = input("숫자1 ==> ")  
number2 = input("숫자2 ==> ")  
print(number1 + number2)
```

- ① 500
- ② 200300
- ③ 200
- ④ 300

산술 연산자

- 파이썬에서 사용되는 기본적인 산술 연산자

연산자	의미	사용 예	설명
=	대입 연산자	a = 3	정수 3을 a에 대입
+	더하기	a = 5 + 3	5와 3을 더한 값을 a에 대입
-	빼기	a = 5 - 3	5에서 3을 뺀 값을 a에 대입
*	곱하기	a = 5 * 3	5와 3을 곱한 값을 a에 대입
/	나누기	a = 5 / 3	5를 3으로 나눈 값을 a에 대입
//	나누기(몫)	a = 5 // 3	5를 3으로 나눈 후 소수점을 버리고 값을 a에 대입
%	나머지값	a = 5 % 3	5를 3으로 나눈 후 나머지값을 a에 대입
**	제곱	a = 5 ** 3	5의 3제곱을 a에 대입

- 예시 : n1 = 200, n2 = 150

```
>>> res = n1 + n2
>>> print(res)
350
```

```
>>> res = n1 * n2
>>> print(res)
30000
```

```
>>> res = n1 / n2
>>> print(res)
1.33333333333
```

```
>>> q = 5 // 3
>>> r = 5 % 3
>>> print(q, r)
1 2
```

```
>>> num = 5 ** 3
>>> print(num)
125
```

산술 연산자

- 예시 : $n1 = 200$, $n2 = 150$

```
>>> res = n1 + n2
>>> print(res)
350
```

```
>>> res = n1 * n2
>>> print(res)
30000
```

```
>>> res = n1 / n2
>>> print(res)
1.33333333333
```

```
>>> q = 5 // 3
>>> r = 5 % 3
>>> print(q, r)
1 2
```

```
>>> num = 5 ** 3
>>> print(num)
125
```

산술 연산자

- 산술 연산자의 우선순위 : 우리가 알고 있는 사칙연산과 동일
 - 왼쪽부터 순서대로 계산하되 괄호가 있을 경우 또는 곱하기, 나누기가 우선 연산
- 실제 현업에서는 대부분 계산 순서대로 괄호 삽입 (나중에 의미 파악도 쉬움)

```
>>> a, b, c = 3, 4, 5
>>> print(a + b - c)
2
>>> print(a - c + b)
2
>>> print(-c + a + b)
2
```

```
>>> a, b, c = 2, 4, 6
>>> print(a / b * c)
3.0
>>> print(a * c / b)
3.0
>>> print(c / a * b)
3.0
```

```
>>> a, b, c = 3, 4, 5
>>> print(a * b + c)
17
>>> print(c + a * b)
17
```

산술 연산자

- 산술 연산을 하는 문자열과 숫자의 상호 변환
 - 문자열이 숫자로 구성되어 있을 때, `int()` 또는 `float()` 함수 사용해서 정수나 실수로 변환
 - 문자열을 `int()` 함수가 정수로, `float()` 함수가 실수로 변경

[illegible]

출력 결과

```
101 101.123 10000000000000000000000000000000
```

- 숫자를 문자열로 변환하려면 str() 함수 사용
- a와 b가 문자열로 변경되어 100+1이 아닌 문자열의 연결인 '1001'과 '100.1231'

```
a = 100; b = 100.123
str(a) + '1'; str(b) + '1'
```

출력 결과

```
'1001'
```

```
'100.1231'
```


대입 연산자

- 오른쪽의 값이나 계산 결과를 왼쪽으로 대입하라는 의미로, '=' 연산자가 가장 기본적인 대입 연산자

```
num = 100
num = 100 * 200
num = int("100") + int("200")
```

- 여러 개의 대입 연산자
 - 콤마(,)로 분리해서 왼쪽에 변수가 2개 이상 나올 수도 있음
 - 그런 경우에는 오른쪽도 반드시 콤마로 분리된 2개의 숫자, 수식, 문자열 등이 와야 함

```
cnum1, num2 = 100, 200
num1, num2 = 100*200, 100+200
num1, num2 = int("100"), 100//5
```

- 오류

```
num1, num2, num3 = 100, 200
num1, num2 = 100
num1 = 100, 200
```

복합 대입 연산자

- 복합 대입 연산자

연산자	사용 예	설명
<code>+=</code>	<code>a += 3</code>	<code>a = a + 3</code> 과 동일
<code>-=</code>	<code>a -= 3</code>	<code>a = a - 3</code> 과 동일
<code>*=</code>	<code>a *= 3</code>	<code>a = a * 3</code> 과 동일
<code>/=</code>	<code>a /= 3</code>	<code>a = a / 3</code> 과 동일
<code>//=</code>	<code>a //= 3</code>	<code>a = a // 3</code> 과 동일
<code>%=</code>	<code>a %= 3</code>	<code>a = a % 3</code> 과 동일
<code>**=</code>	<code>a **= 3</code>	<code>a = a ** 3</code> 과 동일

`a = a + 1`

이 수식을 계산하여
왼쪽 변수에 대입한다.

```
>>> num = 20
>>> num += 3 ; print(num)
23
>>> num -= 3 ; print(num)
20
>>> num *= 3 ; print(num)
60
>>> num /= 3 ; print(num)
20.0
>>> num //= 3 ; print(num)
6.0
>>> num %= 3 ; print(num)
0.0
>>> num **= 3 ; print(num)
0.0
```

복합 대입 연산자

- 예시 : a가 10에서 시작해 프로그램이 진행될수록 값이 누적

```
a = 10
a += 5; print(a)
a -= 5; print(a)
a *= 5; print(a)
a /= 5; print(a)
a //= 5; print(a)
a %= 5; print(a)
a **= 5; print(a)
```


출력 결과

15 10 50 10.0 2.0 2.0 32.0

복합 대입 연산자

- 예시

나눠지는 수 ==> 25
나누는 수 ==> 10



사용자 입력

25 을(를) 10 (으)로 나눈 몫은 2 입니다.

25 을(를) 10 (으)로 나눈 나머지는 5 입니다.

비교 연산자

- 어떤 것이 큰 지, 작은 지, 같은 지를 비교하는 연산자
- 결과는 참을 의미하는 True와 거짓을 의미하는 False로 표시함 (Boolean)
- 비교 연산자를 단독으로 사용하는 경우는 거의 없음 (조건문이나 반복문과 함께 사용함)
- 주의 : 비교 연산자 (==)와 대입 연산자 (=) 의 의미 다름
 - Ex) print(n1 = 1000)

연산자	의미	설명
==	같다	두 값이 동일하면 참(True)
!=	같지 않다	두 값이 다르면 참(True)
>	크다	왼쪽이 크면 참(True)
<	작다	왼쪽이 작으면 참(True)
>=	크거나 같다	왼쪽이 크거나 같으면 참(True)
<=	작거나 같다	왼쪽이 작거나 같으면 참(True)

```
>>> n1 = 100
>>> n2 = 200
>>> print(n1 == n2 , n1 != n2)
False True
>>> print(n1 > n2 , n1 < n2)
False True
>>> print(n1 >= n2 , n1 <= n1)
False True
```

비교 연산자

- 예시

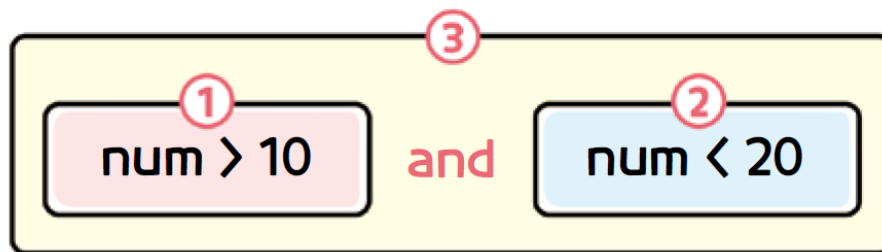
```
>>> print(10 == 100)
False
>>> print(10 != 100)
True
>>> print(10 < 100)
True
>>> print(10 > 100)
False
>>> print(10 <= 100)
True
>>> print(10 >= 100)
False
```

```
>>> print("가방" == "가방")
True
>>> print("가방" != "하마")
True
>>> print("가방" < "하마")
True
>>> print("가방" > "하마")
False
```

논리 연산자

- 비교 연산자가 여러 번 필요할 때 사용함
- 예시 : 숫자가 10과 20 사이에 있어야 한다.
 - 조건 1 : 숫자 num은 10보다 커야 함
 - 조건 2 : 숫자 num이 20보다 작아야 함

`(num > 10) and (num < 20)`



연산자	의미	설명	사용 예
and(논리곱)	~이고, 그리고	둘 다 참이어야 참	<code>(a > 100) and (a < 200)</code>
or(논리합)	~이거나, 또는	둘 중 하나만 참이어도 참	<code>(a == 100) or (a == 200)</code>
not(논리부정)	~아니다, 부정	참이면 거짓, 거짓이면 참	<code>not(a < 100)</code>

논리 연산자

- and 연산자

좌변	우변	결과
True	True	True
True	False	False
False	True	False
False	False	False

- or 연산자

좌변	우변	결과
True	True	True
True	False	True
False	True	True
False	False	False

- not 연산자

- 좌변 True \rightarrow False

- 좌변 False \rightarrow True

논리 연산자

- 예시

```
>>> num = 99
>>> (num > 100) and (num < 200)
False
>>> (num == 99) or (num == 100)
True
>>> not(num == 100)
True
```

```
if(1234) : print("참이면 보여요")
if(0) : print("거짓이면 안 보여요")
```

감사합니다

kimtwan21@dongduk.ac.kr

김 태 완