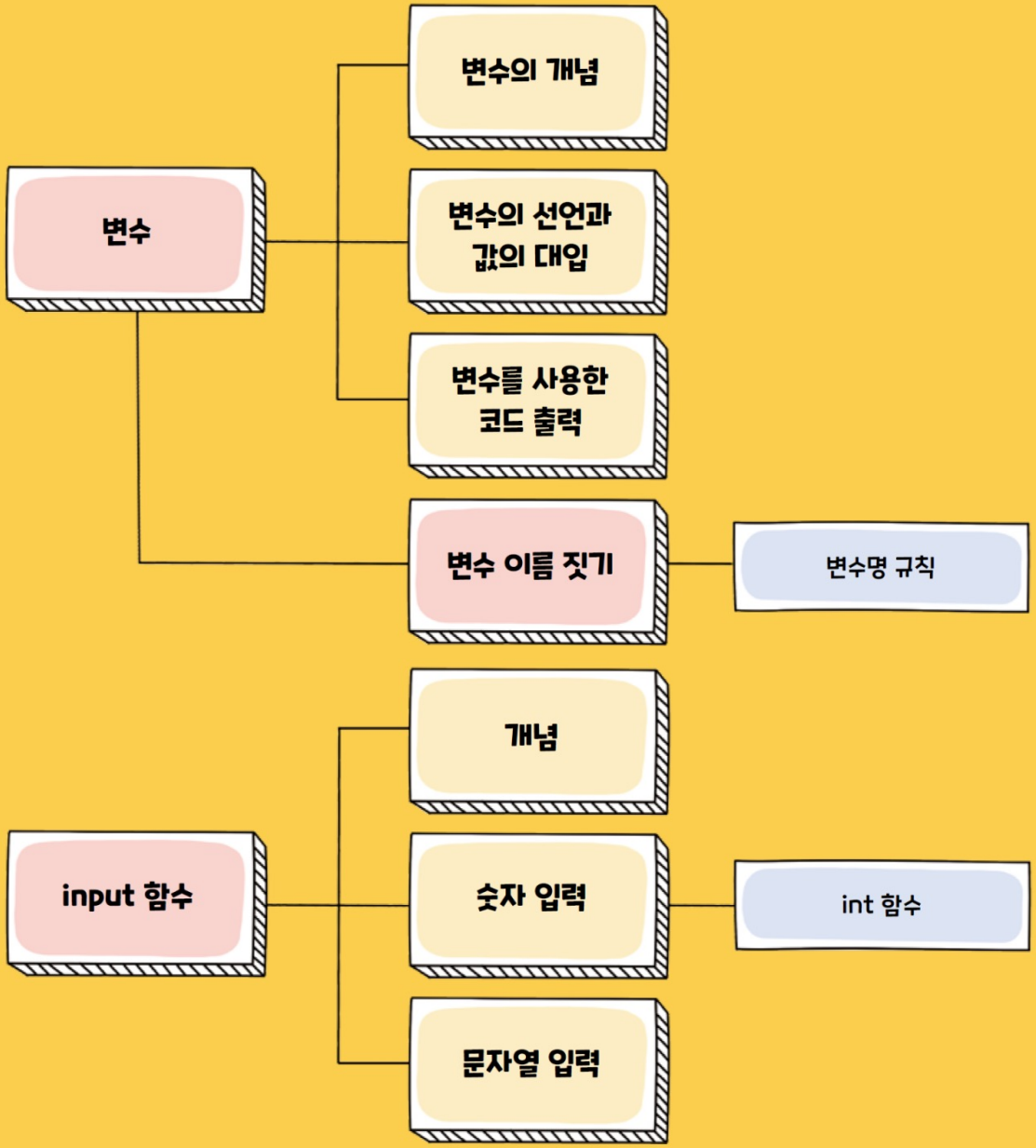
Chapter 2
변수

임경태

Chapter 02
변수

목차

Preview


변수
변수의 개념
변수의 선언과 값의 대입
변수를 사용한 코드 출력
변수 이름 짓기
변수명 규칙
input 함수
개념
숫자 입력
int 함수
문자열 입력

학습목표

- 간단한 계산기를 만들며 변수의 개념을 이해합니다.
- 변수의 개념과 사용법을 익힙니다.
- 키보드로 값을 입력받는 방법에 대해서 학습합니다.

Section 01
변수란?

변수의 개념

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

```
>>> print(100 + 200)
300
```

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

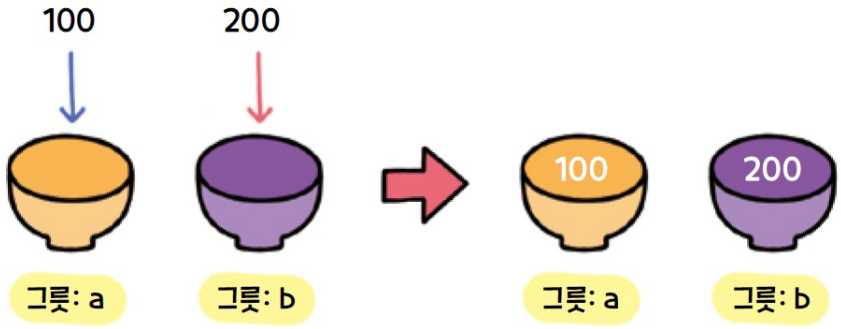


그림 2-1 그릇에 값 넣기

변수의 개념

- 그릇(변수) a, b를 선언하고 100, 200을 대입하기
  - 대입 연산자(=) 사용

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

  - 위 코드의 의미

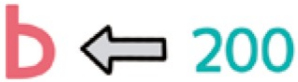


그림 2-2 대입 연산자(=)의 의미

변수의 개념

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

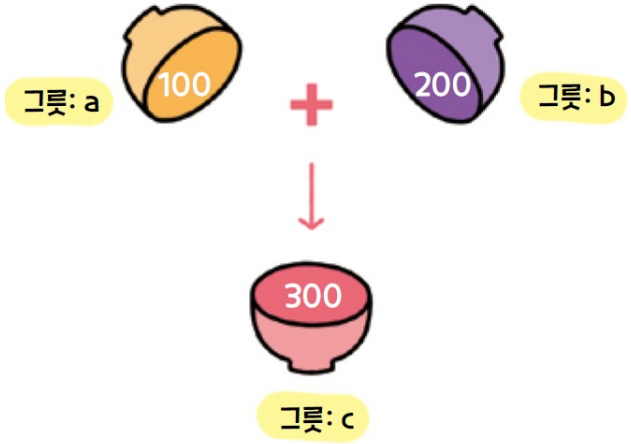


그림 2-3 변수에 담긴 값 더하기

  - 코드로 표현하기

```
>>> c = a + b
```

변수의 개념

- 계산식을 포함한 전체를 출력하기
  - print() 함수 안에서 여러 개를 출력하고 싶다면 콤마(,)로 구분해야 함

```
>>> print(a, “+”, b, “=”, c)
100 + 200 = 300
```

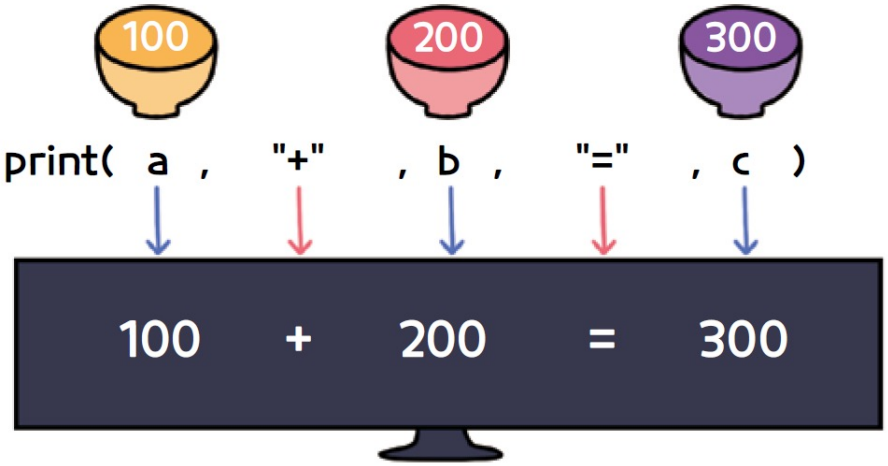


그림 2-4 print() 함수로 전체 출력

변수의 개념

확인문제

1. 다음 빈칸에 들어갈 단어를 채우시오.

변수와 그릇은 비슷한 개념으로, a = 100은 변수 [ ]에 값 [ ](을)를 대입하라는 의미이다.

2. 다음 빈칸에 들어갈 단어를 채우시오.

화면에 변수 또는 값을 출력하는 함수는 [ ]이다.

정답

Click!

변수의 선언과 값의 대입

- 올바른 값의 대입

```
>>> num1 = 100
>>> num2 = 50
>>> result = num1 + num2
```

  - 대입 연산자인 =이 나오면 무조건 =의 오른쪽 부분이 모두 계산된 후에 왼쪽으로 대입됨

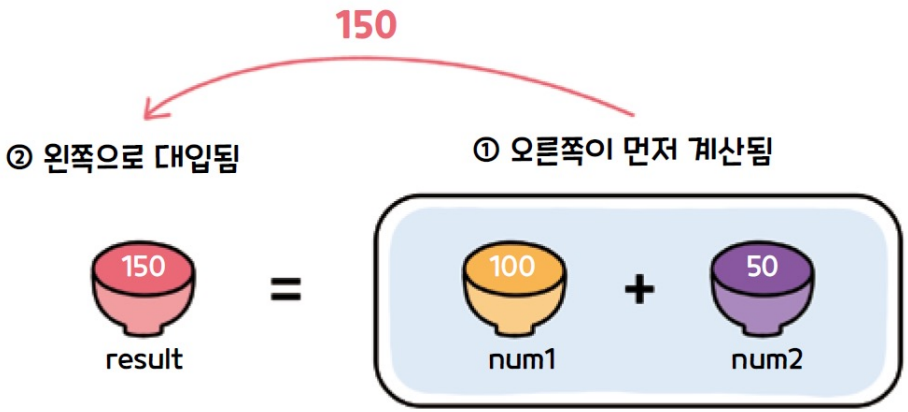


그림 2-5 정상적인 대입 연산자의 구조 1

변수의 선언과 값의 대입

올바른 값의 대입

- 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

```
>>> result = num1 + 200      <------- 변수와 값의 연산
```

- result의 값은? Click!
  - 대입 연산자의 왼쪽에 기존에 사용했던 변수가 나오면 이전 값은 없어지고, 새로운 값으로 덮어 씌워짐

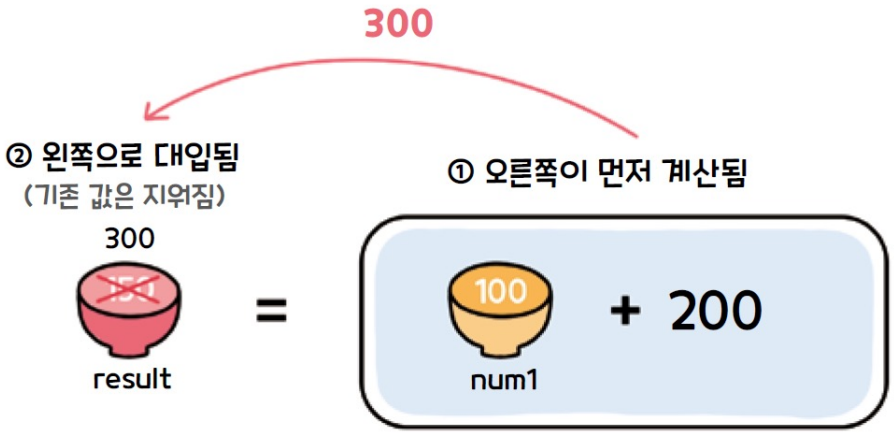


그림 2-6 정상적인 대입 연산자의 구조 2

변수의 선언과 값의 대입

잘못된 값의 대입

- 대입 연산자 =의 왼쪽이 변수가 아니라면 오류 발생

```
>>> 100 = num1 + num2
SyntaxError: cannot assign to literal
```

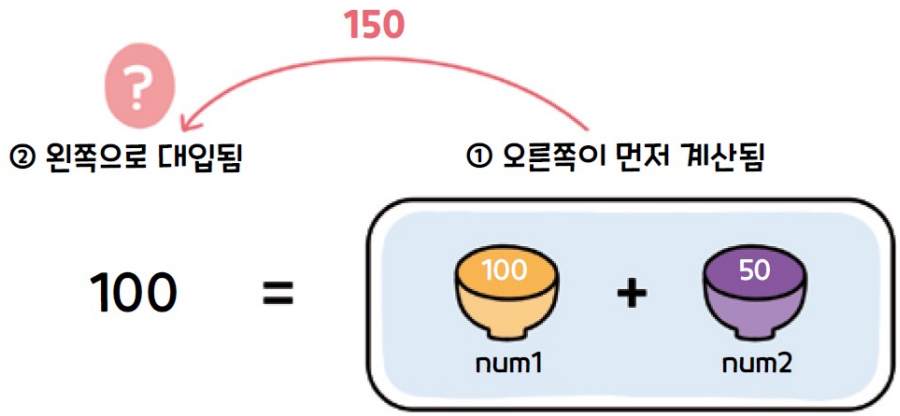


그림 2-7 비정상적인 대입 연산자의 구조 1

변수의 선언과 값의 대입

- 잘못된 값의 대입
  - 대입 연산자 =의 왼쪽 변수는 한 개만 존재해야 함

```
>>> num1 + num2 = result
SyntaxError: cannot assign to operator
```

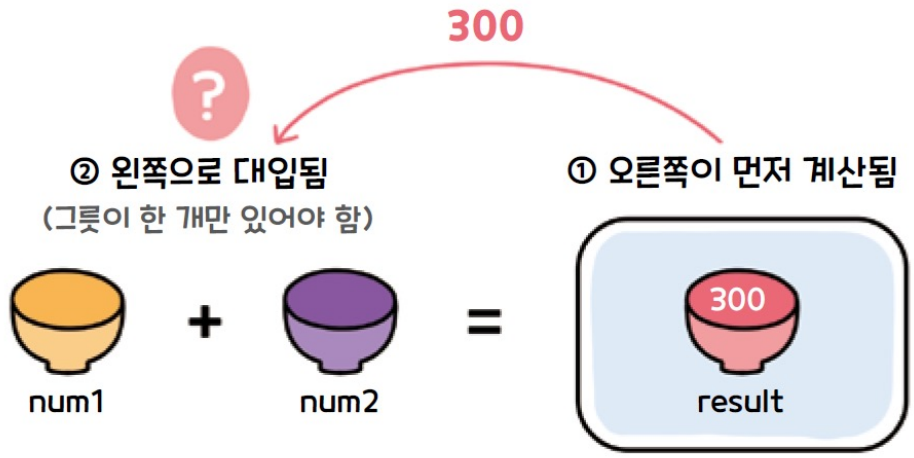


그림 2-8 비정상적인 대입 연산자의 구조 2

변수의 개념

확인문제

다음 보기 중에서 오류가 발생하는 것을 모두 고르시오.

① `100 = 150 + 200`

② `result = 100 + 50`

③ `result + 100 = 50`

④ `result = num1 + num2`

정답

Click!

변수를 사용한 코드 출력1: 숫자

뺄셈 연산

- 두 숫자의 빼기 연산(num1: 100, num2: 50)

```
>>> result = num1 - num2
>>> print(num1 , “-” , num2 , “=” , result)
100 - 50 = 50
```

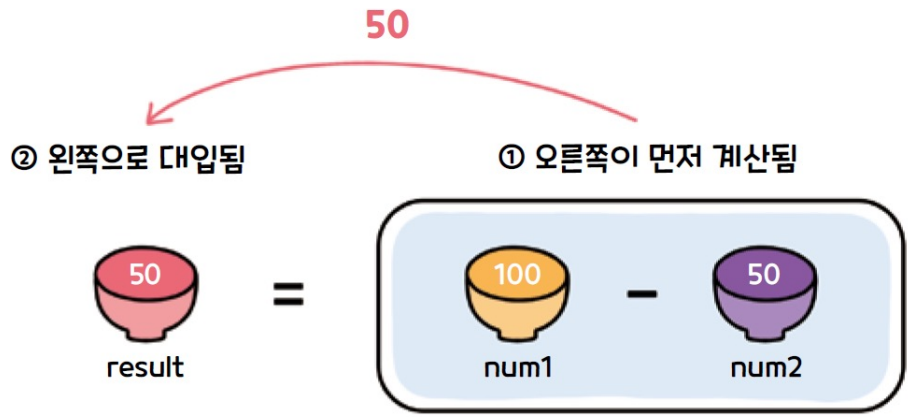


그림 2-9 뺄셈 연산

변수를 사용한 코드 출력1: 숫자

곱셈 연산

- 두 숫자의 곱셈 연산(num1: 100, num2: 50)
- 곱셈 연산자: *

```
>>> result = num1 * num2
>>> print(num1 , “x” , num2 , “=” , result)
100 x 50 = 5000
```

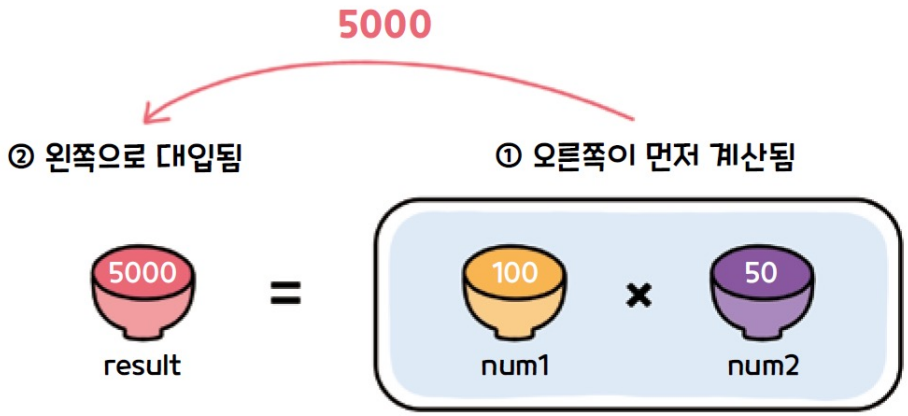


그림 2-10 곱셈 연산

변수를 사용한 코드 출력1: 숫자

- 나눗셈 연산
  - 두 숫자의 나눗셈 연산(num1: 100, num2: 50)
  - 나눗셈 연산자: /

```
>>> result = num1 / num2
>>> print(num1 , "÷" , num2 , "=" , result)
100 ÷ 50 = 2.0
```

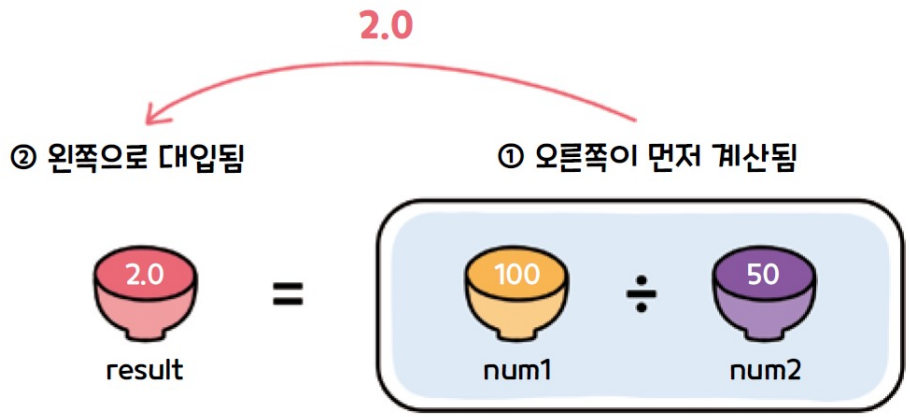


그림 2-11 나눗셈 연산

변수를 사용한 코드 출력2: 문자열

문자열을 변수에 대입

- 숫자와 동일하게 변수를 선언하고 큰따옴표 또는 작은따옴표로 묶은 문자열을 대입하는 방식

```
>>> str1 = “난생처음”
>>> str2 = “파이썬”
```

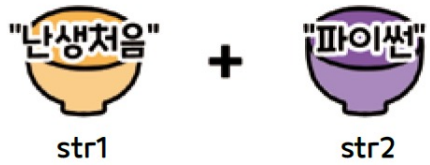


그림 2-12 변수에 문자열을 대입

- 여러 개의 문자열을 한 줄에 출력하기
  - 콤마(,)로 분리

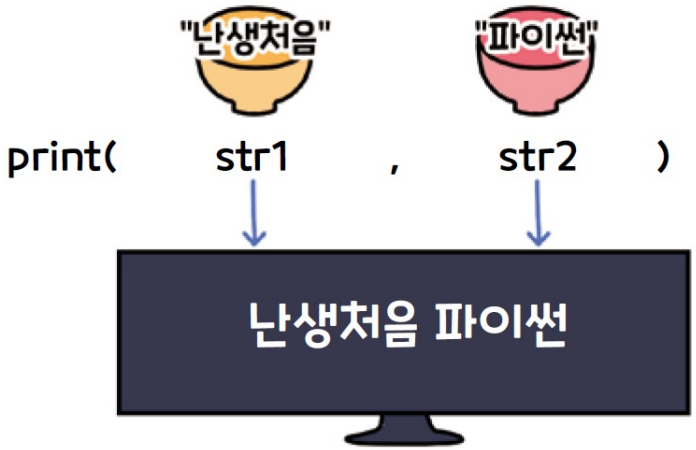


그림 2-13 print()로 문자열을 출력

변수를 사용한 코드 출력2: 문자열

문자열의 더하기 연산과 빼기 연산

- 문자열의 더하기 연산 : 문자열을 이어줌(띄어쓰기 없이 붙음)

```
>>> result = str1 + str2
>>> print(result)
난생처음파이썬
```

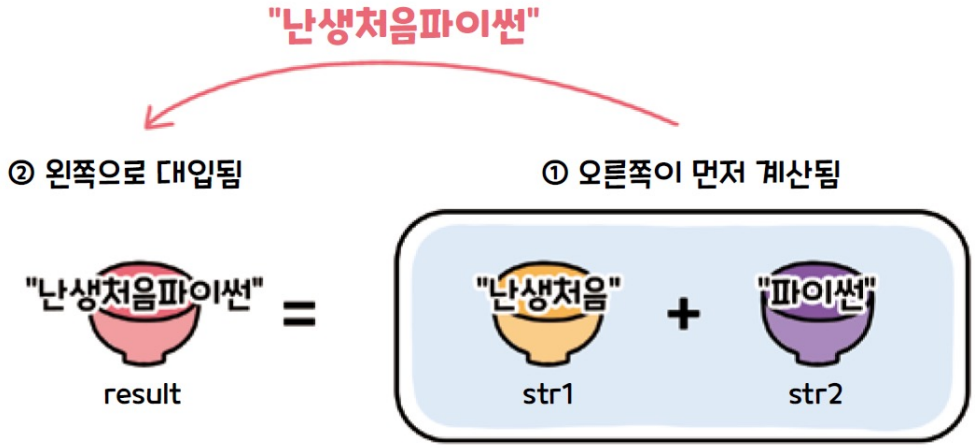


그림 2-14 문자열의 더하기 연산

변수를 사용한 코드 출력2: 문자열

- 문자열의 더하기 연산과 빼기 연산
  - 문자열의 빼기 연산 : 오류 발생
    - 곱하기와 나누기도 문자열에서 사용할 수 없음

```
>>> result = str1 - str2
>>> print(result)
Traceback (most recent call last):
    File "<pyshell#20>", line 1, in <module>
        result = str1 - str2
TypeError: unsupported operand type(s) for -: 'str' and 'str'
```

변수를 사용한 코드 출력2: 문자열

확인문제

다음 중 오류가 발생하지 않는 것은 무엇인가?

① res = "안녕" + "하세요"
② res = "안녕" - "하세요"
③ res = "안녕" * "하세요"
④ res = "안녕" / "하세요"

정답

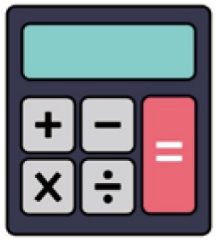
Click!

LAB 간단한 사칙연산 계산기 만들기

이제 변수에 대한 사용법 및 개념을 완전히 익혔으므로, 이를 활용해서 간단히 사칙연산(더하기, 빼기, 곱하기, 나누기)이 가능한 계산기를 만들어 봅시다.

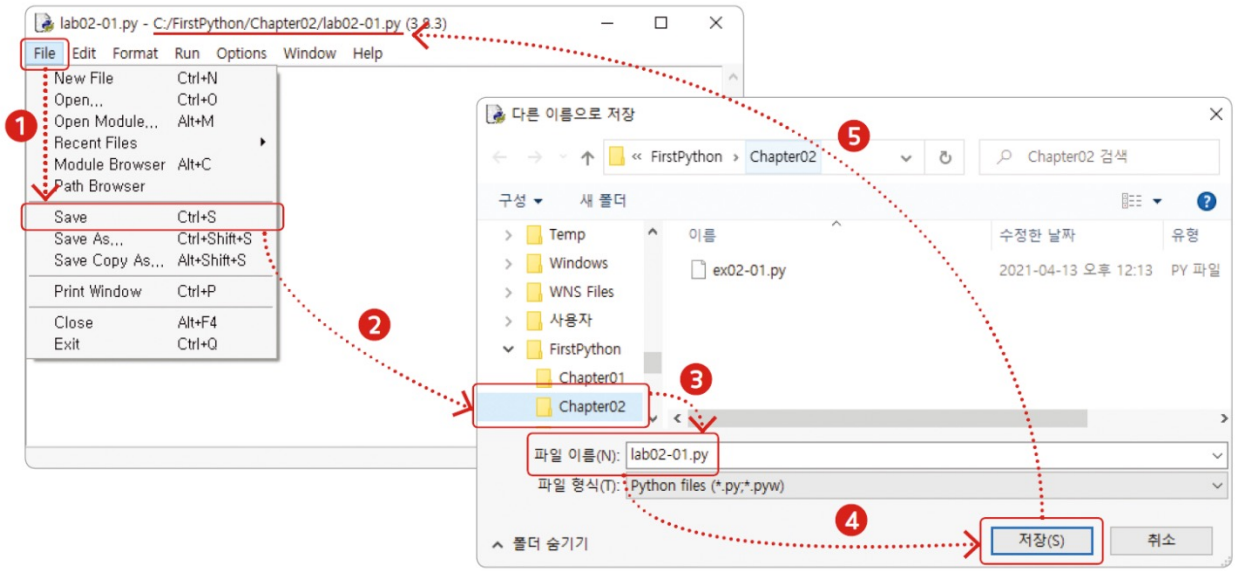
실행 결과

```
100 + 200 = 300
100 - 200 = -100
100 * 200 = 20000
100 / 200 = 0.5
```



1. IDLE Shell 창의 메뉴 [File]-[New File]을 선택해서 빈 파이썬 파일을 준비하기.
   그리고 상단의 [File]-[Save]를 선택하거나, <Ctrl>+<S> 를 눌러서 먼저 저장하기.

   - 파일 저장 경로: C:\FirstPython\Chapter02\lab02-01.py



LAB 간단한 사칙연산 계산기 만들기

2. 두 변수를 준비하고 숫자를 대입하기
   - 스크립트 모드이므로 실행되지는 않고 메모장처럼 계속 입력만 가능함

```
num1 = 100
num2 = 200
```

3. 두 수를 사칙연산하는 코드를 그 아래에 추가하기
   - 두 수를 더한 결과는 result1~ result4라는 변수에 저장

```
result1 = num1 + num2
result2 = num1 - num2
result3 = num1 * num2
result4 = num1 / num2
```

LAB 간단한 사칙연산 계산기 만들기

4. 마지막으로 사칙연산 결과를 출력하는 코드를 추가하기

```
print(num1 , "+" , num2 , "=" , result1)
print(num1 , "-" , num2 , "=" , result2)
print(num1 , "*" , num2 , "=" , result3)
print(num1 , "/" , num2 , "=" , result4)
```

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

Section 02
변수 이름 짓기

변수명 규칙

- 변수명은 영문 및 숫자만 사용할 수 있음
  - 변수명은 영문 및 숫자만 사용할 수 있으며, 영문과 숫자를 섞어서 사용할 수 있음

```
a = 100
abcd = 200
zzzz = 300
p1234 = 400
its4you = 600
```

  - 단, 영문으로 시작해야 함

```
>>> 333 = 100
SyntaxError: cannot assign to literal
>>> 3abcd = 200
SyntaxError: invalid syntax
```

숫자로만 이루어져 오류 발생

숫자로 시작해 오류 발생

변수명 규칙

- 병수명에 언더바(_)를 사용할 수 있음
  - 변수의 이름으로 띄어쓰기가 허용되지 않기 때문에 변수명이 길어지면 의미 파악이 어려워짐. 이럴 때 띄어쓰기 대신 언더바(_)를 사용
  - 언더바(_)는 중간, 제일 앞, 제일 뒤 위치에 관계없이 사용할 수 있음

```
input_number = 100
_number = 200
_data_ = 300
__my__ = 400
_1234 = 500
```

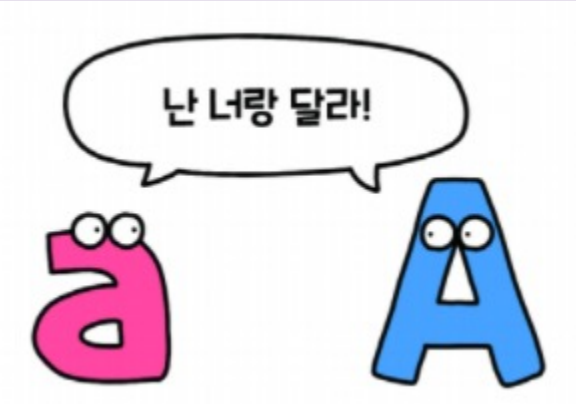
하나 더 알기 **언더바를 2개 쓰는 경우**

파이썬 내부에서 사용하는 일부 예약어의 앞뒤에 언더바를 2개씩 붙인 것이 있습니다. 예를 들어 __init__() 함수, __main__ 변수 등이 있는데, 이에 대해서는 차후에 다시 언급하겠습니다.

변수명 규칙

- 변수명은 대문자와 소문자를 구분함
  - 파이썬은 대문자와 소문자를 완전히 다른 것으로 취급함
  - 따라서 변수명의 대문자와 소문자는 다른 변수임
  - 대문자와 소문자를 섞으면 언더바 대신에 의미를 부여하기에 좋음
  - 다음은 모두 올바르며, 서로 다른 변수임

```
mynumber = 100
MYNUMBER = 200
Mynumber = 300
myNumber = 400
```



변수명 규칙

- 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
```

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

그림 2-15 파이썬의 예약어

변수명 규칙

하나 더 알기 **함수명을 변수로 사용할 경우**

함수명을 변수로 사용해도 문법상 오류가 발생하지는 않습니다. 그러나 함수의 본래 기능을 잃어버리기 때문에 프로그램이 엉망으로 작동할 수도 있으므로 권장하지 않습니다.
예를 들어 지금까지 사용한 print() 함수의 이름인 print를 변수명으로 사용해 보겠습니다. 다음 코드는 오류를 발생하지 않습니다.

```
>>> print = 100
>>> print = 200 + 300
```

그렇다면 print() 함수의 본래 기능을 사용해서 "난생처음" 글자를 출력해 보겠습니다.

```
>>> print("난생처음")
Traceback (most recent call last):
    File "<pyshell#49>", line 1, in <module>
        print("난생처음")
TypeError: 'int' object is not callable
```

이제는 print() 함수를 사용할 수가 없습니다. print를 변수로 선언하고 사용해 더 이상 우리가 알고 있는 출력을 해주는 함수가 아니게 된 것입니다. 그러므로 함수의 이름을 변수로 사용하는 것은 바람직하지 않습니다. 파이썬 IDLE을 재시작하면, 다시 원래의 print() 함수를 사용할 수 있습니다.

변수명 규칙

확인문제

변수명에 대한 설명으로 올바른 것은 무엇인가?

① 변수명은 영문 및 숫자로 시작해야 한다.

② 변수명은 언더바(_)로 시작할 수 있다.

③ 변수명은 띄어쓰기를 허용한다.

④ 변수명으로 예약어를 사용할 수 있다.

정답

Click!

좋은 변수 이름이란?

- 변수의 이름만 보고도 의미를 파악할 수 있어야 함
  - 문법상 문제는 없으나, 변수의 용도를 파악할 수 없는 경우

```
a = 100
bbb = 200
asdf = 300
AA = 400
```

- 변수의 의미가 파악되더라도 너무 길어지면 안 됨

```
first_number_of_input_values = 100
second_number_of_input_values = 200
```

좋은 변수 이름이란?

- 좋은 변수 이름은 짧으면서도 그 의미를 파악할 수 있어야 함
  - 언더바로 구분해주기
  - 대문자와 소문자를 섞어 사용하기
  - 의미를 파악하기 쉽도록 하기

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

좋은 변수 이름이란?

확인문제

다음 코드 중에서 문법상 잘못된 것을 모두 고르고 이유를 작성하시오.

① num1 = 10

② 2num = 20

③ num1 + num2 = res

④ last_number_of_input_values = 200

정답

Click!

Section 03
값을 입력받는 input() 함수

input() 함수의 개념

- input() 함수
  - 키보드로 입력받도록 도와주는 함수

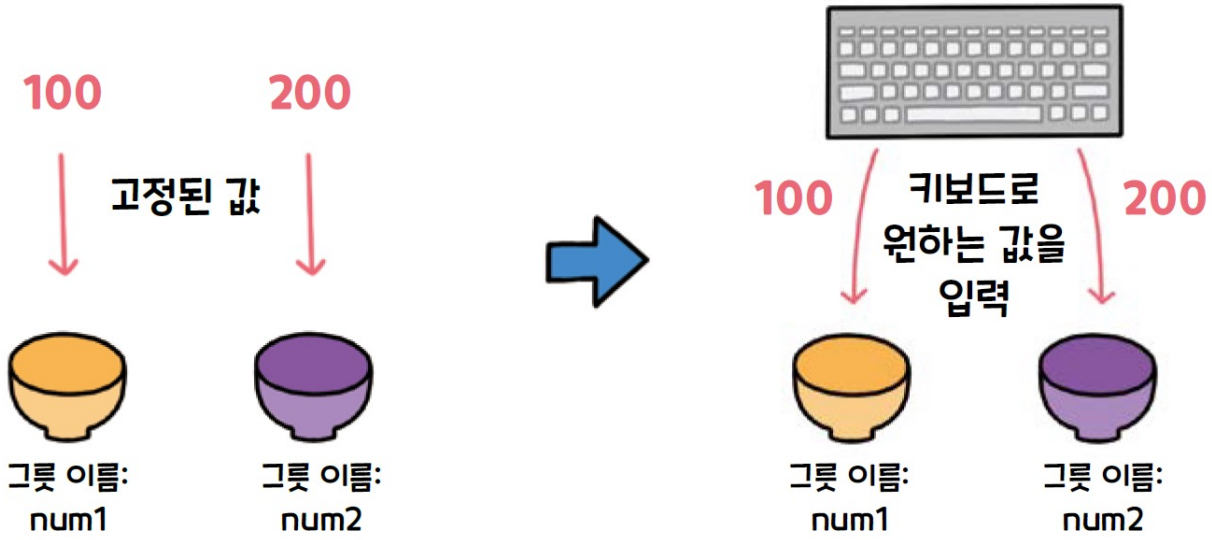


그림 2-16 변수에 키보드로 직접 값을 입력

사용자에게 숫자 값을 입력받아 출력하기

- input() 함수의 활용
  - 파이썬 셸에서 사용자 입력받기

```
>>> input()
100     <------ 사용자 입력
'100'
```

  - input() 함수에서 입력받을 때 주의할 점
    - input() 함수로 값을 입력받고 변수에 저장하지 않으면 화면에 출력한 후, 그냥 사라짐
    - 따라서 input() 함수는 입력된 값을 변수에 저장하는 것이 일반적임

```
>>> num2 = input("숫자 ==> ")
숫자 ==> 100     <------ 사용자 입력
  <------ 아무 메시지도 나오지 않음
```

사용자에게 숫자 값을 입력받아 출력하기

- input() 함수의 활용
  - input() 함수로 입력받은 값을 변수에 저장하기

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

  - input() 함수는 괄호 안에 메시지를 넣어 사용자에게 가이드하기

```
>>> num2 = input("숫자 ==> ")
숫자 ==> 100    <------- 사용자 입력
 <------- 아무 메시지도 나오지 않음
```

  - 입력받은 num1, num2를 더해서 result1 변수에 넣기

```
>>> result1 = num1 + num2
 <------- 아무 메시지도 나오지 않음
```

사용자에게 숫자 값을 입력받아 출력하기

- input() 함수의 활용
  - result1 출력해서 결과값 확인

```
>>> result1 = num1 + num2
```

Click!

  - input() 함수는 입력받은 값을 모두 문자열로 취급함
  - 즉 num1, num2에 입력된 100, 100은 숫자가 아닌 문자열임

사용자에게 숫자 값을 입력받아 출력하기

정수로 변환하는 int() 함수

- input()로 입력받은 값을 숫자로 사용하기 위해서는 입력받은 문자열을 다시 int() 함수를 이용하여 정수로 변환해야 함
- int(값)
  - 값이 무엇이든지 정수로 변환함

그림 2-17 정수로 변환하는 int() 함수

```
>>> int("100")
100
```
<-- 문자열을 정수로 변경

```
>>> int(100.12)
100
```
<-- 실수를 정수로 변경

사용자에게 숫자 값을 입력받아 출력하기

- 다시 num1, num2를 입력받고 더해 정수 출력하기

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

사용자에게 문자열을 입력받아 출력하기

- 사용자에게 이름과 전화번호를 입력받아 출력하는 프로그램

[코드 2-1]

```
userName = input("이름 ==> ")
userPhone = input("전화번호 ==> ")
print("제 이름은 ", userName, "이고, 연락처는", userPhone, "입니다.")
```

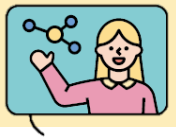
[실행결과]

```
이름 ==> 난처음
전화번호 ==> 010-1234-1234
제 이름은 난처음이고, 연락처는 010-1234-1234입니다.
```

사용자 입력

사용자 입력

사용자에게 문자열을 입력받아 출력하기



확인문제

다음 빈칸에 들어갈 단어를 채우시오.

(1) 문자열 또는 실수를 정수로 변환해 주는 함수는 (　　)이다.

(2) 키보드로 값을 입력받는 함수 이름은 (　　)이다.

(3) input( ) 함수는 기본적으로 모든 값을 (　　)(으)로 입력받는다.

정답

Click!

LAB 택배 배송 정보 입력하기

엄마의 심부름으로 난생이는 편의점에서 택배를 보내려 합니다. 택배를 보낼 때, 받는 사람과 주소 를 입력하고 택배의 무게를 입력합니다. 택배 무게는 그램(g)당 5원이며, 자동으로 계산됩니다.

택배 정보를 입력받아 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
```

LAB 택배 배송 정보 입력하기

1. lab02-02.py 파일을 만들고, 받는 사람, 주소, 무게를 저장할 변수를 선언하기

```
print("## 택배를 보내기 위한 정보를 입력하세요. ##")
personName = input("받는 사람 : ")
personAddr = input("주소 : ")
weight = int(input("무게(g) : "))
```

2. print( ) 함수로 받는 사람과 주소를 출력하기

```
print("** 받는 사람 ==>", personName)
print("** 주소 =>", personAddr)
```

3. 배송비는 무게의 그램(g)당 5원이므로 무게에 5를 곱하여 출력하기

```
print("** 배송비 ==>", weight*5, "원")
```

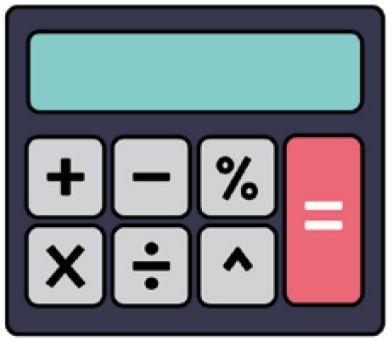
4. <Ctrl>+<S>를 눌러서 변경된 내용을 저장하고, <F5>를 눌러 실행 결과 확인하기

LAB 숫자를 입력하는 계산기 만들기

사용자가 키보드로 2개의 숫자를 입력하면 두 숫자의 덧셈, 뺄셈, 곱셈, 나눗셈, 나머지, 제곱을 계 산하는 계산기를 만들어 봅시다.

실행 결과

```
숫자1 ==> 100        사용자 입력
숫자2 ==> 10         사용자 입력
100 + 10 = 110
100 - 10 = 90
100 * 10 = 1000
100 / 10 = 10.0
100 % 10 = 0
100 ** 10 = 100000000000000000000
```



LAB 숫자를 입력하는 계산기 만들기

1. lab02-03.py 파일을 만든 후, 두 변수를 준비하고 키보드로 입력받을 수 있도록 코딩하기

```
num1 = int(input("숫자1 ==>"))
num2 = int(input("숫자2 ==>"))
```

2. 두 수를 계산하는 코드를 그 아래에 추가하기

- 이 때 나머지 값은 % 연산자를, 제곱값은 ** 연산자를 사용함
  - 나머지 값과 제곱값에 대한 내용은 3장에서 좀 더 자세히 학습함

```
result1 = num1 + num2
result2 = num1 - num2
result3 = num1 * num2
result4 = num1 / num2
result5 = num1 % num2
result6 = num1 ** num2
```

LAB 숫자를 입력하는 계산기 만들기

3. 마지막으로 사칙연산 결과를 출력하는 코드를 추가합니다.

```
print(num1 , "+" , num2 , "=" , result1)
print(num1 , "-" , num2 , "=" , result2)
print(num1 , "*" , num2 , "=" , result3)
print(num1 , "/" , num2 , "=" , result4)
print(num1 , "%" , num2 , "=" , result5)
print(num1 , "**" , num2 , "=" , result6)
```

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

[실전 예제]
거북이가 나오는 프로그램

실전 예제 거북이가 나오는 프로그램

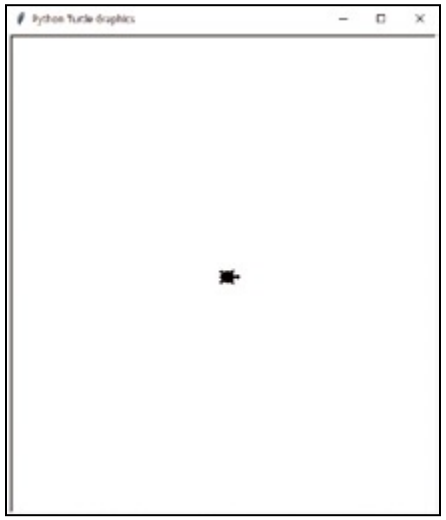
파이썬은 거북이 그래픽 환경을 제공해서, 재미있는 프로그래밍이 가능하도록 지원함

[거북이 그래픽 시작하기]

1. 거북이 그래픽을 터틀 그래픽(Turtle Graphic)으로도 부름.

2. 그래픽 화면으로 거북이가 나오기 위한 코드

```
>>> import turtle <-- turtle 기능을 포함함
>>> turtle.shape('turtle')<-- 윈도우 창이 나오면서 동시에 거북이 모양 결정함
```

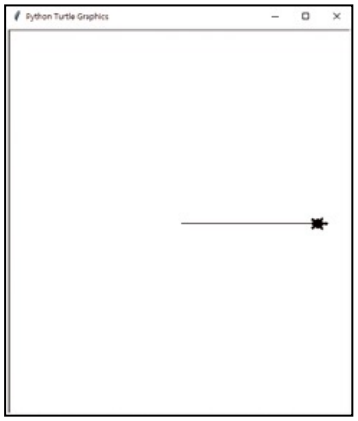


실전 예제 거북이가 나오는 프로그램

[거북이 이동 및 회전시키기]

1. 거북이를 앞으로 200만큼 이동시키기

```
>>> turtle.forward(200)
```

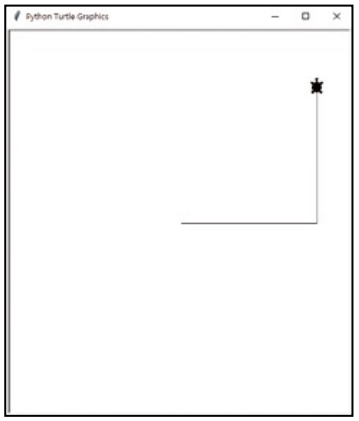


- 200은 화면의 픽셀(Pixel, 점)의 개수를 의미함

2. 거북이를 위쪽으로 200만큼 이동시키기

- 거북이를 왼쪽으로 90도 회전시킨 후, 200만큼 이동

```
>>> turtle.left(90)
>>> turtle.forward(200)
```

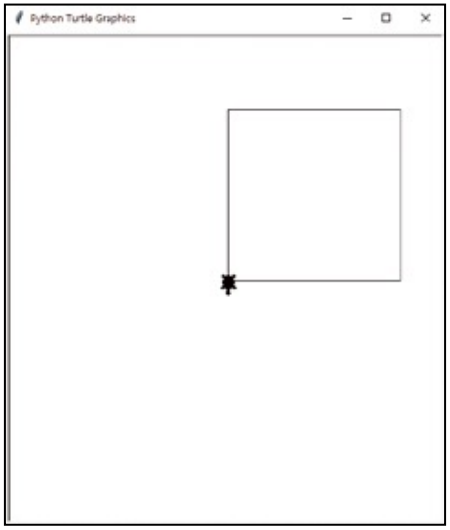


실전 예제 **거북이가 나오는 프로그램**

[거북이 이동 및 회전시키기]

3. 2번 코드를 두 번 더 반복하면 거북이가 완전한 네모를 그린 후에, 원래 위치로 돌아옴

   - 다음 장부터는 더욱 다양한 형태의 거북이 그래픽을 다룸



Preview

감사합니다 :)