
CHAP.2

이번에는 키보드에서 입력을 받아보자

- 숫자, 문자를 구분할 수 있어야 한다.

결과를 예측해 보라

```
>>>a=1
>>>b=2
>>>c=a+b
>>>print(c)
```

```
>>>a='1'
>>>b='2'
>>>c=a+b
>>>print(c)
```

결과가 다르다.

```
>>>a=1
>>>b=2
>>>c=a+b
>>>print(c)
3
```

```
>>>a='1'
>>>b='2'
>>>c=a+b
>>>print(c)
12
```

1과 '1'은 다르다

어떤 면에서 ?

type

type이라는 명령이 있다.
데이터의 타입(type); 유형을 알려준다.

```
>>>a=1
>>>type(a)
<class 'int'>          integer; 정수
>>>b='1'
>>>type(b)
<class 'str'>          string; 문자열
>>>c='123'
>>>type(c)
<class 'str'>
```

```
>>>1 == '1'   정수 1과 문자열 1은 같습니다.
False
>>>1 != '1'   정수 1과 문자열 1은 같지 않습니다.
True


>>>type(False)
<class 'bool'>
>>>type(True)
<class 'bool'>
```

키보드 입력 받기

```
a = input(“Input a number: ”)
b = input(“Input a number: ”)
c = a + b
print(c)
```

```
Input a number: 1
Input a number: 2
12
```

```
a = input(“Input a number: ”)
b = input(“Input a number: ”)
a = int(a)
c = a + int(b)
print(c)
```

```
Input a number: 1
Input a number: 2
3
```

입력 받은 단의 구구단 출력

```
def dan(d):
    a = 0
    for i in range(9):
        a = a + 1
        b = d * a
        print(d, 'x', a, '=', b)
    print()

d = input("Input dan: ")
```

```
d = int(d)
dan(d)
```

```
Input dan: 2
2 x 1 = 2
2 x 2 = 4
2 x 3 = 6
```

int, float, str

왜 float... 동영상...참고

http://www.kocw.net/home/cview.do?lid=02accf95cd2ad793

```
>>>type(1)
<class 'int'>
>>>type(1.1)
<class 'float'>
>>>type('1')
<class 'str'>
```

```
>>>int('1')
1
>>>int(1.1)
1
>>>float(1)
1.0
>>>str(1)
'1'
```

나온 김에 성격은 좀 다르지만...

```
>>>round(1.2)
1
>>>round(1.6)
2
>>>2**3
8
```

```
import math
>>>math.floor(1.6)
1
>>>math.ceil(1.2)
2
>>>math.pow(2, 3)
8.0
>>>math.sqrt(9)
3.0
```

2의 12승까지 출력

```
import math
a = 0
for i in range(12):
    a = a + 1
    #b = math.pow(2, a)
    b = 2**a
    print(b, end=' ')
```

이 정도는 알아두자.

```
2 4 8 16 32 64 128 256 512 1024 2048 4096
```

조건문

```
print("A")
a = 1
if a==1:
    print("o")
print("B")
```

```
A
O
B
```

```
print("A")
a = 1
if a==2:
    print("o")
print("B")
```

```
A
B
```

```
print("A")
a = 1
if a==2:
    print("o")
else:
    print("x")
print("B")
```

```
A
x
B
```

덧셈문제 출제하고 맞추기

메모장 시연

```
a = 10
b = 15
c = a + b
print(a,'+',b,'=', end=' ')
ans = input()
ans = int(ans)
if (ans == c):
    print('O')
else:
    print('x')
```

```
10 + 15 = 25
o
```

무작위로 숫자 꺼내 오기

메모장 시연

```
import random
a=random.randint(0,5)
print(a)
```

```
4
```

```
import random
for i in range(5):
    a=random.randint(0,5)
    print(a)
```

```
4
3
3
5
4
```

알고 있는 무작위

메모장 시연

```
import random
random.seed(123)
for i in range(5):
    a=random.randint(0,5)
    print(a)
```

```
0
2
0
3
2
```

엄밀한 설명은 아니지만,
일단은 이렇게 이해하면 편하다.

숫자가 무작위로 나열된 표가 있다.
그 표를 하나 정해 표에 있는 숫자를
차례대로 꺼내 오면
무작위 숫자를 얻을 수 있다.

무작위표를 하나 정해 버리면,
매번 똑 같은 결과를 얻을 수 있다.

학생 실습

무작위로 덧셈문제 출제하고 맞추기

```
import random
a=random.randint(10,100)
b=random.randint(10,100)
c = a + b
print(a,'+',b,'=', end=' ')
```

```
ans = input()
ans = int(ans)
if (ans == c):
    print(‘O’)
else:
    print(‘x’)
```

산술 연산자

계산 결과는 숫자

+, -, /, * ⇒ 숫자

```
>>>1+1
2
>>>2-1
1
>>>2*3
6
```

```
>>>4/2
2.0
>>>5/2
2.5
>>>5//2
2
>>>5%2
1
```

논리 연산자 계산 결과는 True 또는 False

==, !=, <, <=, >, >=, ⇒ True 또는 False

```
a=1
b=2
c=(a==b)
print(c)
c=(a!=b)
print(c)
```

```
False
True
```

```
c=(a<b); print(c)
c=(a<=b); print(c)
c=(a>b); print(c)
c=(a>=b); print(c)
```

```
True
True
False
False
```

식을 한 단계씩 계산해 보자

정말 기본 중에 기본
이거 모르는 사람 많다.

```
1 + 2 * 3
→ 1 + 6
```

```
a = 1 + 2 * 3
→ a = 1 + 6
```

식을 한 단계씩 계산해 보자

```
1 + 2 * 3
→ 1 + 6
→ 7
```

계산 끝

```
a = 1 + 2 * 3
→ a = 1 + 6
→ a = 7
```

계산 끝 ?

식을 한 단계씩 계산해 보자

```
1 + 2 * 3
→ 1 + 6
→ 7
```

계산 끝
연산자가 2개
계산단계도 2번

```
a = 1 + 2 * 3
→ a = 1 + 6
→ a = 7
```

계산 끝 ?
연산자가 3개
계산단계도 3번 이어야 한다.

아직 계산이 끝나지 않았다.

```
1 + 2 * 3
→ 1 + 6
→ 7
```

연산자는 계산하고, 그 결과를 남긴다.

계산결과를 남기지 않으면, 연속 계산이 불가능.

```
1 + 2 * 3
→ 1 + ?
→
```

```
a = 1 + 2 * 3
→ a = 1 + 6
→ a = 7
→
```

어떤 값을 남길까 ?

호랑이는 가죽을 남기고, 연산자는 계산결과를 남긴다.

```
1 + 2 * 3
→ 1 + 6
→ 7
```

연산자는 계산하고, 그 결과를 남긴다.

계산결과를 남기지 않으면, 연속 계산이 불가능.

```
1 + 2 * 3
→ 1 + ?
→
```

```
a = 1 + 2 * 3
→ a = 1 + 6
→ a = 7
→ 7
```

계산결과가 남는다는 점을 알고 보면 명확하게 이해된다.

```
a=1
b=2
c=(a==b)
print(c)
c=(a!=b)
print(c)
```

→ c=(False)
→ c=False
→ False

```
False
True
```

```
a=1
b=2


print(a==b)


print(a!=b)
```

```
False
True
```

함수 정의

코드 블록에 이름을 붙인다. 나중에 또 사용하려고.

아래 코드 블록을 add라는 이름으로 정의한다.

```
a=1
b=2
c=a+b
print(c)
print("end")
```

같은 칸을 띄운 코드

```
3
end
```

```
def add():
    a=1
    b=2
    c=a+b
    print(c)
add()
print("end")
```

'add()함수 정의' 라고 말함

코드 블록

'add()함수 호출'이라고 말함

```
3
end
```

함수의 코드 블록 안에 있던 print를 밖으로 빼내면,

```
def add(a, b):
    c=a+b
    print(c)
a=2
b=3
add(a, b)
print("end")
```

```
5
end
```

```
def add(a, b):
    c=a+b    이 c는 함수 add()안에 있는 놈
    print(c)
a=2
b=3
add(a, b)
print(c)    한번도 전에 알려지지 않았던 c
print("end")
```

```
error
```

```
def add(a, b):
    c=a+b
    print(c)
a=2
b=3
add(a, b)
print("end")
```

```
5
end
```

최종코드1

```
def add(a, b):
    c=a+b
    return c
a=2
b=3
c=add(a, b)
print(c)
print("end")
```

c값을 남기시오.

함수 ~ 연산자

바로 여기에

```
5
end
```

함수의 코드 블록 안에 있던
c가 함수 밖의 코드 블록에 있는
c라는 것을 알려준다.

변수가 할당 연산자에 의해
값이 바뀌고
그 결과를 함수 밖의 변수에 반영할 때

최종코드2-부득이한 경우에 사용

```
def add(a, b):
    global c
    c = a + b

c=0
a=2
b=3
add(a, b)
print(c)
print("end")
```

이 블록에서 나오는 c는 함수 밖의 c다.

```
5
end
```

할당 연산자를 쓰지 않는 경우는,
global 키워드를 쓰지 않아도 된다.

값을 할당하지 않는 변수

```
def add(a, b):
    global c
    c=a+b
    print(d)
c=0
a=2
b=3
d=4
add(a, b)
print(“end”)
```

```
4
end
```

```
d=1
def add(a, b):
    c=a+b
    print(d)

add(1,2)
print(c)
```

c는 c와 다르다.
밖에는 c가 없다.
그래서, 에러

c는 담장 안에 있다.

```
c=1
def add(a, b):
    c=a+b

add(1,2)
print(c)
```

c는 add() 함수 안에서 새로 생성한다.

* 함수형 프로그래밍의 근본적인 특징 중 하나.

a=1
a=2

```
c=1
def add(a, b):
    global c
    c=a+b

add(1,2)
print(c)
```

global의 역할

c를 이 함수 안에서 새로 생성하지 마시오.

밖에 있는 c라는 의미

```
def add(a, b):
    c=a+b
    return c

c = add(1, 2)
print(c)
```

return

값(?)을 남긴다.

함수를 종료하고, 호출한 곳으로 돌아간다.