

# 변수, 표현식, 명령문

## 제2장



모두를 위한 파이썬  
[www.py4e.com](http://www.py4e.com)



# 상수

- 값이 변하지 않아서 숫자, 글자, 문자열과 같은 고정 값을 “상수”라고 함
- 숫자 상수
- 문자열 상수는 작은따옴표나 (') 큰따옴표로(") 표시

```
>>> print(123)
123
>>> print(98.6)
98.6
>>> print('Hello world')
Hello world
```

# 예약어

예약어를 변수 이름/식별자로 사용할 수 없음

```
False  class  return  is  finally  
None   if     for    lambda  continue  
True   def    from   while  nonlocal  
and    del    global  not    with  
as     elif   try     or     yield  
assert else  import  pass  
break  except in    raise
```

# 변수

- 변수는 이름이 주어진 메모리로 변수 이름을 통해 데이터를 저장하고 검색 가능
- 프로그래머가 변수 이름을 지정
- 대입문을 통해 변수값을 변경 가능

x = 12.2

y = 14

x

12.2

y

14

# 변수

- 변수는 이름이 주어진 메모리로 변수 이름을 통해 데이터를 저장하고 검색 가능
- 프로그래머가 변수 이름을 지정
- 대입문을 통해 변수값을 변경 가능

x = 12.2

y = 14

x = 100

x ~~12.2~~ 100

y 14

# 파이썬 변수 이름 규칙

- 글자나 밑줄로 시작
- 글자, 숫자, 밑줄로 이루어짐
- 대소문자 구분

좋은 예: spam eggs spam23 \_speed  
나쁜 예: 23spam #sign var.12  
모두 다름: spam Spam SPAM

# 연상 기호 변수 이름

- 변수 이름은 프로그래머 마음대로 지정할 수 있다 - 모범 기준을 따라야 함
- 변수값과 연관 지어 변수 이름을 지정 (연상 기호)
- 변수 이름이 키워드인 것 같아 보일 수 있기 때문에 초급자는 헛갈릴 수 있음

<http://en.wikipedia.org/wiki/Mnemonic>

```
x1q3z9ocd = 35.0  
x1q3z9afd = 12.50  
x1q3p9afd = x1q3z9ocd * x1q3z9afd  
print(x1q3p9afd)
```

무엇을 하는 코드  
입니까?



```
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd * x1q3z9afd
print(x1q3p9afd)
```

```
a = 35.0
b = 12.50
c = a * b
print(c)
```

무엇을 하는 코드  
입니까?

```
x1q3z9ocd = 35.0
x1q3z9afd = 12.50
x1q3p9afd = x1q3z9ocd * x1q3z9afd
print(x1q3p9afd)
```

```
a = 35.0
b = 12.50
c = a * b
print(c)
```

무엇을 하는 코드  
입니까?

```
hours = 35.0
rate = 12.50
pay = hours * rate
print(pay)
```

# 문장/줄

x = 2	←	대입문
x = x + 2	←	대입문 + 표현식
print(x)	←	출력문

변수

연산자

상수

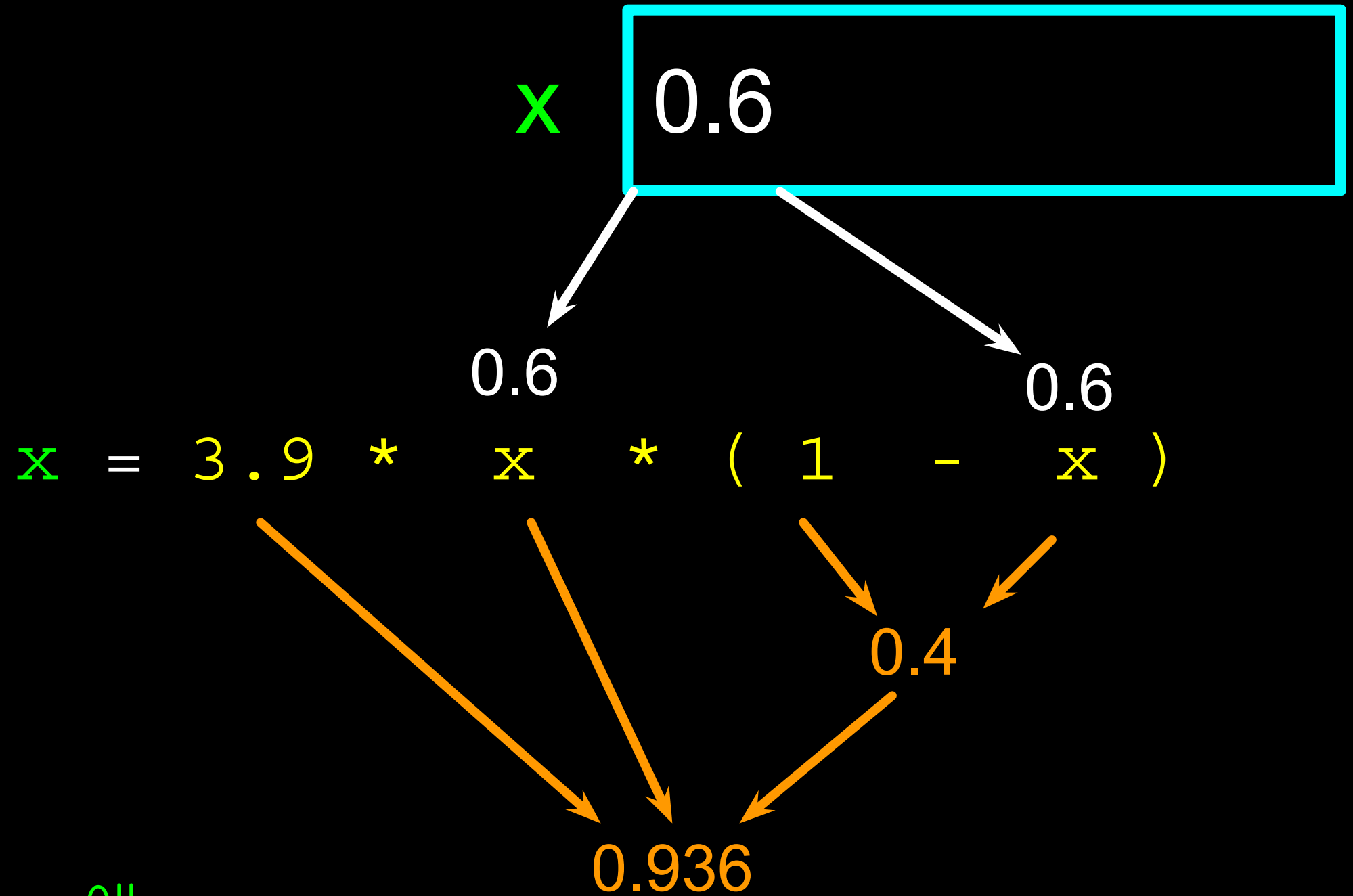
함수

# 대입문

- 대입문 (=)으로 변수값을 지정
- 대입문은 **오른편에 있는 표현식**과 결과값이 저장되는 **변수**로 구성

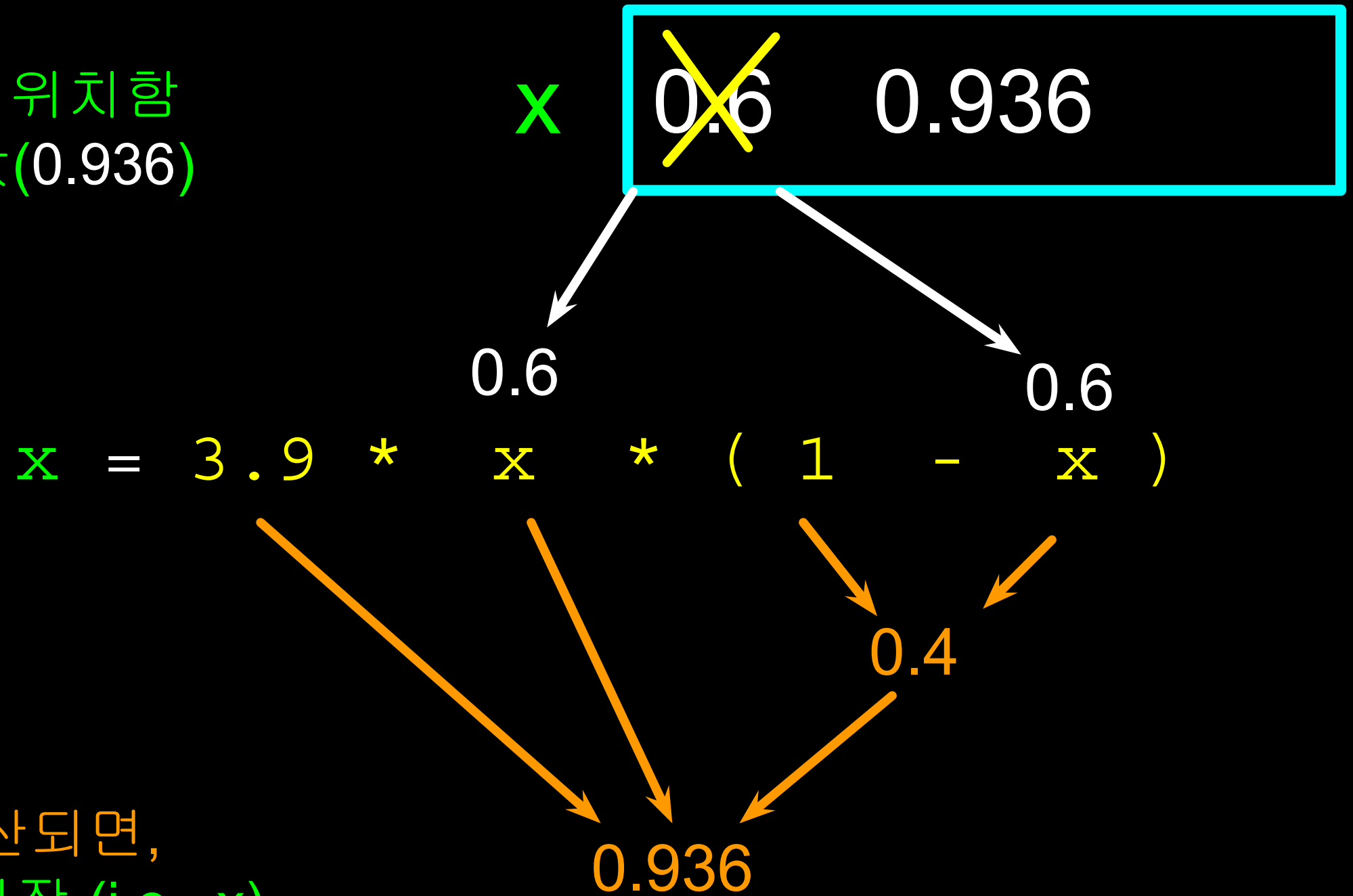
$$x = 3.9 * x * ( 1 - x )$$

변수는 값 (0.6)을 저장하는  
메모리상 위치함



오른편은 표현식이고  
표현식이 계산되면, 결과값은  $x$ 에  
저장됨

변수는 값을 저장하는 메모리상 위치함  
변수의 예전 값(0.6)을 새로운 값(0.936)  
으로 업데이트 할 수 있음



오른편은 표현식. 표현식이 계산되면,  
결과값은 왼편에 있는 변수에 저장 (i.e., x)

표현식

# 숫자 표현식

- 컴퓨터 키보드에 수학 기호가 부족하기 때문에 “computer-speak”을 사용해 연산
- 별표 (\*)는 곱하기
- 별표 별표 (\*\* )는 거듭제곱

연산자	연산
+	더하기
-	빼기
*	곱하기
/	나누기
**	거듭제곱
%	나머지



# 숫자 표현식

```
>>> xx = 2
>>> xx = xx + 2
>>> print(xx)
4
>>> yy = 440 * 12
>>> print(yy)
5280
>>> zz = yy / 1000
>>> print(zz)
5.28
```

```
>>> jj = 23
>>> kk = jj % 5
>>> print(kk)
3
>>> print(4 ** 3)
64
```

4 R 3

$$\begin{array}{r} 5 \overline{) 23} \\ \underline{20} \\ 3 \end{array}$$

연산자	연산
+	더하기
-	빼기
*	곱하기
/	나누기
**	거듭제곱
%	나머지

# 계산 순서

- 긴 연산을 할 때 파이썬은 무슨 연산을 먼저 해야 하는지 알아야 함
- 이것을 “연산 순위”라고 함
- 무슨 연산을 먼저 해야 할까?


$$x = 1 + 2 * 3 - 4 / 5 ** 6$$

# 연산 순위 규칙

높은 우선 순위에서 낮은 우선 순위:

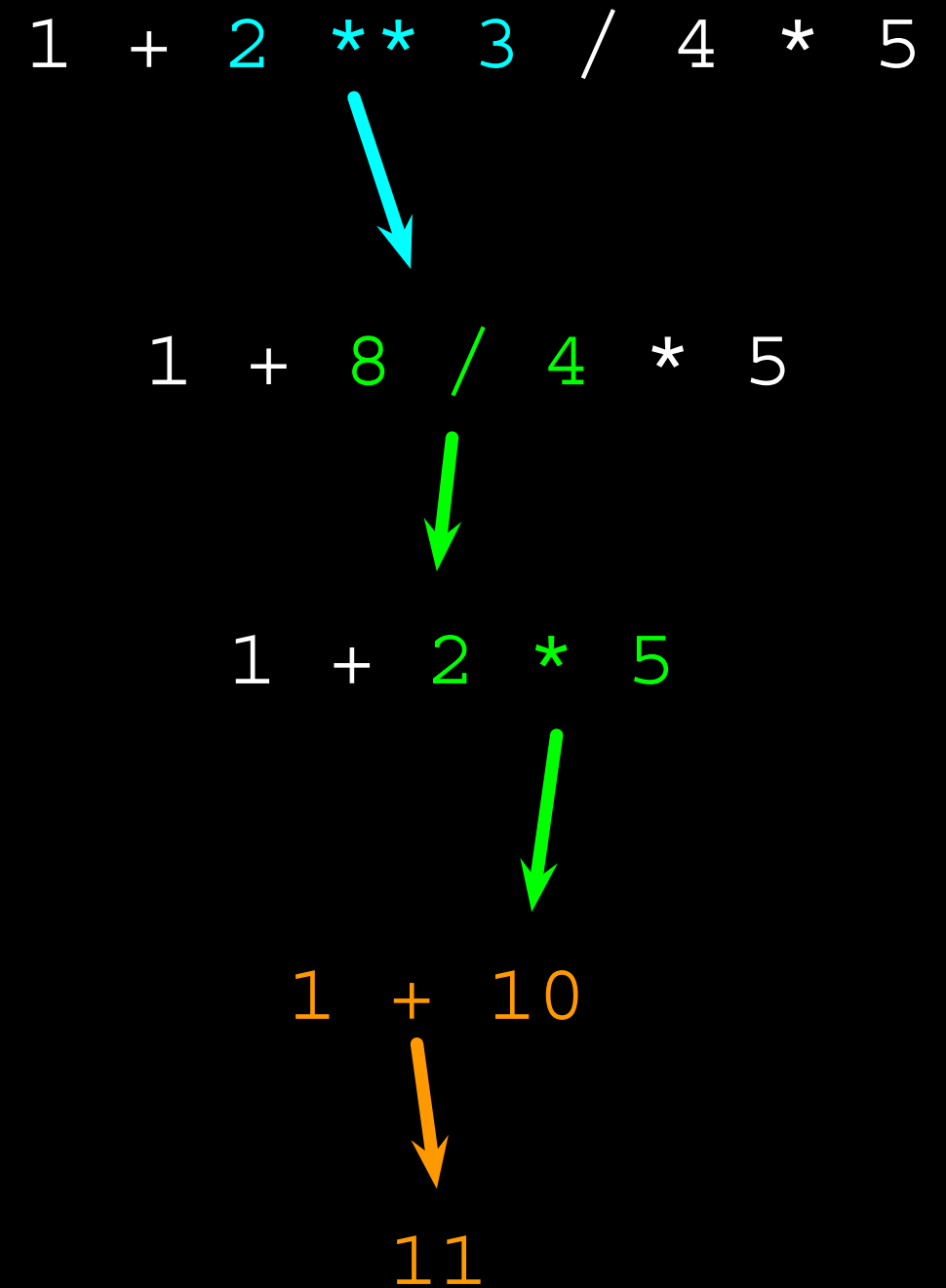

- 괄호가 항상 우선
- 거듭제곱
- 곱셈, 나눗셈, 나머지
- 덧셈, 뺄셈
- 왼쪽에서 오른쪽

괄호  
거듭제곱  
곱하기  
더하기  
왼쪽에서 오른쪽

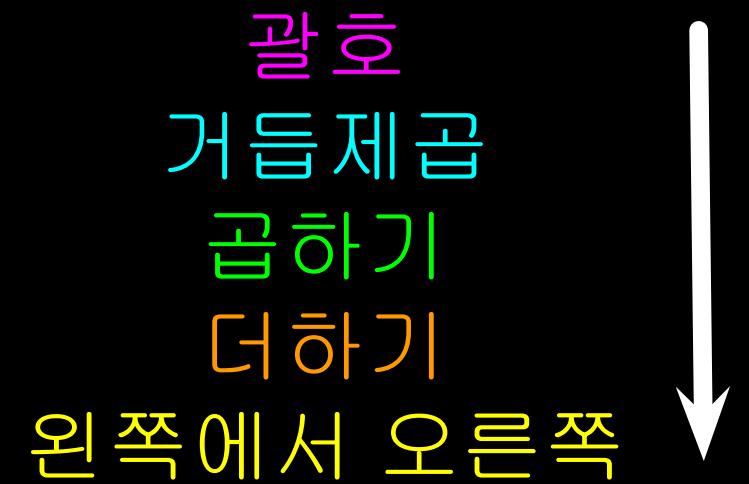


```
>>> x = 1 + 2 ** 3 / 4 * 5
>>> print(x)
11.0
>>>
```

괄호  
거듭제곱  
곱하기  
더하기  
왼쪽에서 오른쪽



# 연산 순위



- 위에서 아래순서
- 코드를 쓸 때 - 괄호를 사용
- 코드를 쓸 때 - 이해하기 쉽도록 간단한 연산식을 쓰기
- 긴 연산식은 간단하게 쪼개기

# “자료형”의 뜻

- 파이썬에는 변수, 문자, 상수라는 “자료형”이 있음
- 파이썬은 정수와 문자열의 차이를 앎
- 예를 들어 “+” 는 숫자를 “덧셈”하고 문자열을 “연결”

```
>>> ddd = 1 + 4
>>> print(ddd)
5
>>> eee = 'hello ' + 'there'
>>> print(eee)
hello there
```

# 자료형

- 파이썬은 “자료형”을 구분
- 어떤 연산은 금지되어 있음
- 예를 들어 문자열에 1을 더할 수 없음
- `type()` 함수를 써서 자료형을 알 수 있음

```
>>> eee = 'hello ' + 'there'
>>> eee = eee + 1
Traceback (most recent call last):
File "<stdin>", line 1, in
<module>TypeError: Can't convert
'int' object to str implicitly
>>> type(eee)
<class'str'>
>>> type('hello')
<class'str'>
>>> type(1)
<class'int'>
>>>
```

# 숫자 자료형

- 숫자는 크게 두 자료형으로 존재

- 정수

-14, -2, 0, 1, 100, 401233

- 부동 소수점 수

-2.5, 0.0, 98.6, 14.0

- 이 외 다른 숫자 자료형이 있음 - 정수와 부동 소수점 수의 변형

```
>>> xx = 1
>>> type (xx)
<class 'int'>
>>> temp = 98.6
>>> type(temp)
<class 'float'>
>>> type(1)
<class 'int'>
>>> type(1.0)
<class 'float'>
>>>
```



# 자료형 변환

- 표현식에 정수와 부동 소수점 수가 있으면 **암묵적으로** 정수를 부동 소수점으로 변환됨
- 내장된 함수 `int()` 와 `float()` 으로 자료형을 변환을 할 수 있음

```
>>> print(float(99) + 100)
199.0
>>> i = 42
>>> type(i)
<class'int'>
>>> f = float(i)
>>> print(f)
42.0
>>> type(f)
<class'float'>
>>>
```

# 정수 나눗셈

정수 나눗셈은 부동 소수점 수를 반환

```
>>> print(10 / 2)
5.0
>>> print(9 / 2)
4.5
>>> print(99 / 100)
0.99
>>> print(10.0 / 2.0)
5.0
>>> print(99.0 / 100.0)
0.99
```

파이썬 2.x와 다른 부분

# 문자열 변환

- `int()` 와 `float()` 으로 문자열과 정수를 변환
- 문자열에 숫자가 없을 경우 에러가 발생

```
>>> sval = '123'
>>> type(sval)
<class 'str'>
>>> print(sval + 1)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
TypeError: Can't convert 'int' object
to str implicitly
>>> ival = int(sval)
>>> type(ival)
<class 'int'>
>>> print(ival + 1)
124
>>> nsv = 'hello bob'
>>> niv = int(nsv)
Traceback (most recent call last):
File "<stdin>", line 1, in <module>
ValueError: invalid literal for int()
with base 10: 'x'
```

# 사용자 입력

- **input()** 함수로 멈추고  
사용자의 입력값을 받을 수  
있음

```
nam = input('Who are you? ')
print('Welcome', nam)
```

- **input()** 함수는 문자열을 반환

```
Who are you? Chuck
Welcome Chuck
```

# 사용자 입력값 변환



- 자료형 변환 함수를 사용해 문자열을 숫자로 바꾼 후 사용자가 입력한 숫자를 읽을 수 있음

```
inp = input('Europe floor?')  
usf = int(inp) + 1  
print('US floor', usf)
```

- 이후 변환이 불가능한 입력값에 대해 알아보기로 함

Europe floor? 0  
US floor 1

# 주석

- 파이썬은 # 다음 모든 내용을 무시
- 주석의 역할
  - 코드가 어떤 일을 하는지 설명
  - 코드를 쓴 사람 등 부가 설명
  - 임시적으로 코드를 비활성화

```
# Get the name of the file and open it
name = input('Enter file:')
handle = open(name, 'r')

# Count word frequency
counts = dict()
for line in handle:
    words = line.split()
    for word in words:
        counts[word] = counts.get(word, 0) + 1

# Find the most common word
bigcount = None
bigword = None
for word, count in counts.items():
    if bigcount is None or count > bigcount:
        bigword = word
        bigcount = count

# All done
print(bigword, bigcount)
```

# 요약

- 자료형
- 예약어
- 변수 (연산 기호)
- 연산자
- 연산자 순위
- 정수 나눗셈
- 자료형 변환
- 사용자 입력
- 주석 (#)



## 연습 문제

사용자에게 시간(Hours)과 시급(Rate)을  
입력값으로 받아 급여를 계산하는 프로그램을  
만들기

```
Enter Hours: 35
```

```
Enter Rate: 2.75
```

```
Pay: 96.25
```



# Acknowledgements / Contributions



These slides are Copyright 2010- Charles R. Severance ([www.dr-chuck.com](http://www.dr-chuck.com)) of the University of Michigan School of Information and made available under a Creative Commons Attribution 4.0 License. Please maintain this last slide in all copies of the document to comply with the attribution requirements of the license. If you make a change, feel free to add your name and organization to the list of contributors on this page as you republish the materials.

Initial Development: Charles Severance, University of Michigan School of Information

Contributor:

- Seung-June Lee ([plusjune@gmail.com](mailto:plusjune@gmail.com))
- Connect Foundation

Translator:

- Saejin Park
- Jeungmin Oh ([tangza@gmail.com](mailto:tangza@gmail.com))