파이썬 프로그래밍

수치형 자료형,
문자열 자료형

한국기술교육대학교
온라인평생교육원

▣ 수치 자료형

1. 정수형 상수

```
a = 23 # 10진 정수
b = 023 # 8진 정수
c = 0x23 # 16진 정수
print type(a), type(b), type(c)
print a, b, c
```

```
<type 'int'> <type 'int'> <type 'int'>
23 19 35
```

- 정수형 상수: 소수 영역이 없는 수
- a=23 → a에 23이라는 정수형 상수 할당(23은 10진 정수)
- 숫자 앞에 숫자 0을 붙이면 8진 정수
- 숫자 앞에 0x(숫자 0, 문자 x)를 붙이면 16진 정수
- a, b, c의 타입 → int(정수형 상수)
- print a, b, c → a, b, c에 할당된 정수가 출력됨
- 10진 정수는 그대로 출력
- 8진·16진 정수는 10진 정수로 값이 바뀌어 출력

```
import sys
print sys.maxint  # 최대 정수 값 확인
```

```
9223372036854775807
```

- 그렇다면 정수형 상수로 나타낼 수 있는 최대 정수는?
- sys 모듈의 maxint를 확인하면 알 수 있음

▣ 수치 자료형

2. 실수형 상수

```
a = 1.2
b = 3.5e3
c = -0.2e-4
print type(a), type(b), type(c)
print a, b, c
```

```
<type 'float'> <type 'float'> <type 'float'>
1.2 3500.0 -2e-05
```

- 실수형 상수: 소수 영역이 있는 수
- 3.5e3=3.5*10³=3500(e3 → 10³ = 1000)
- 실수형 상수는 3500이 아닌 3500.0
- e-4 → 10^{-4} =1/10000
- -0.2e-4=-0.2*(1/10000)=-2*(1/100000)=-2e-5
- a, b, c의 타입 → float(실수형 상수)
- print a, b, c → 각 표현 방식에 따라 출력됨
- 3.5e3 → 3500.0
- -0.2e-4 → -2e-5
- print로 인한 변화는 str(repr) 내장 함수에 의한 것

▣ 수치 자료형

3. 롱형 상수

```
h1 = 12345678901234567890123456789OL
# 마지막에 L을 붙여서 명시적으로 long 형이라고 알려도 되고
print type(h1)
print h1 * h1
print
```

```
<type 'long'>
15241578753238836750495351562536198787501905199875019052100
```

- 롱형: 정수형 상수의 최대 범위를 벗어난 수
- 롱형은 메모리가 허용하는 한 유효자리 수가 무한
- 롱형을 명시하는 방법 → 숫자 끝에 L을 붙여줌
- print h1*h1 → 메모리가 허용하는 한에서 출력됨

```
h2 = 1234567890123456789012345678900
# L을 붙이지 않아도 int형이 담을 수 있는 수치를 초과하면 자동으로 long형이 된다.
print type(h2)
print h2 * h2
print
```

```
<type 'long'>
15241578753238836750495351562536198787501905199875019052100
```

- 숫자 끝에 L을 붙이지 않아도 maxint를 넘으면 롱형
- print h2*h2 → 메모리가 허용하는 한에서 출력됨

▣ 수치 자료형

3. 롱형 상수

```
h3 = 123L
print type(h3)
print

h4 = 123
print type(h4)
```

```
<type 'long'>

<type 'int'>
```

- 작은 수여도 숫자 끝에 L을 붙이면 롱형
- 숫자 끝에 L을 안 붙이면 정수형 상수(int)로 인식
- 숫자 끝에 L을 붙이지 않아도 maxint를 넘으면 롱형
- 자동으로 L이 나타나는 것은 ipython 노트북의 기능
- 일반적인 코딩 환경에서는 나타나지 않음

▣ 수치 자료형

4. 복소수형 상수

```
a = 10 + 20j
print a

b = 10 + 5j
print a + b
```

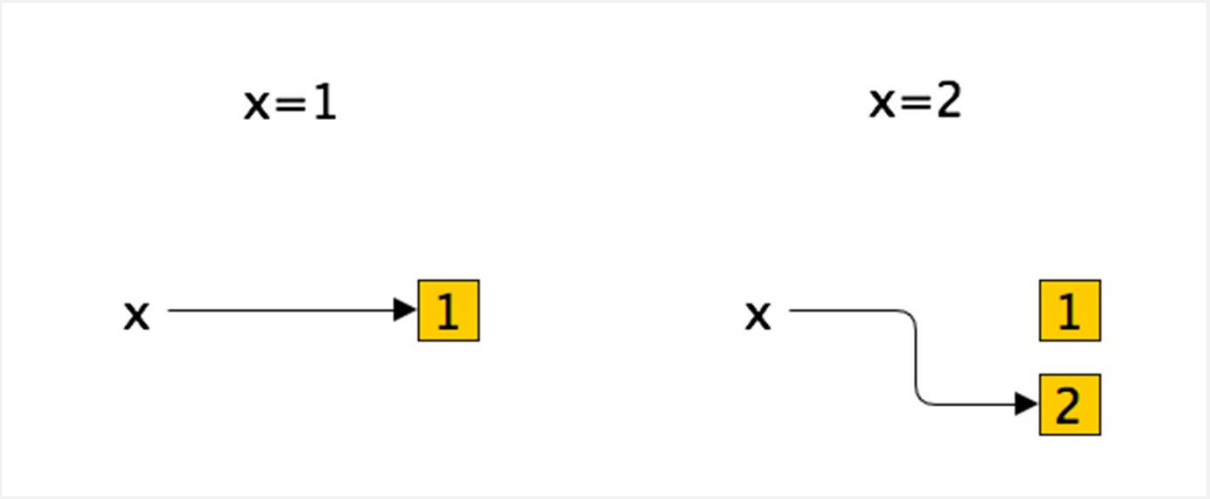
```
(10+20j)
(20+25j)
```

- 복소수형 상수: 실수부(ex. 10)와 허수부(ex. 20j)로 구성
- 복소수형 상수는 실수부와 허수부가 각각 계산됨

▣ 수치 자료형

5. 수치 자료형의 치환

```
x = 1
x = 2
```



- x=1로 입력한 후 x=2로 치환할 때
- x=1 입력 → x가 1을 가리키도록 하는 주소(레퍼런스) 생김
- 그 후 x=2 입력 → x가 2를 가리키도록 하는 주소(레퍼런스) 생김
- 그리고 기존에 1을 가리키던 주소는 없어짐
- 이 때 1은 쓰레기(garbage)가 되어 수집 가능

▣ 수치 자료형

6. 수치 연산과 관련된 내장 함수

```
print abs(-3)
print int(3.141592)
print int(-3.1415)
print long(3)
print float(5)
print complex(3.4, 5)
print complex(6)
```

```
3
3
-3
3
5.0
(3.4+5j)
(6+0j)
```

- print abs(숫자) → 숫자의 절대값이 출력됨
- print int(숫자) → 숫자의 정수형이 출력됨
- -3.9999가 -4에 가깝지만 정수형은 -3
- print long(숫자) → 숫자의 롱형이 출력됨(L은 나타나지 않음)
- print float(숫자) → 숫자의 실수형이 출력됨
- print complex(a, b) → (a+bj)와 같이 복소수형이 출력됨
- print complex(a) → (a+0j)와 같이 실수부에만 숫자 나타남

▣ 수치 자료형

6. 수치 연산과 관련된 내장 함수

```
print divmod(5, 2)
print pow(2, 3)
print pow(2.3, 3.5)
```

```
(2, 1)
8
18.4521691056
```

- print divmod(a, b) → (a를 b로 나눈 몫, a를 b로 나눈 나머지)
- print pow(a, b) → a의 b승 값이 출력됨
- pow 함수는 실수(소수 부분이 있는 수)도 입력이 가능함

▣ 수치 자료형

7. math 모듈의 수치 연산 함수

```
import math
print math.pi
print math.e
print math.sin(1.0) # 1.0 라디안에 대한 사인 값
print math.sqrt(2) # 제곱근
```

```
3.14159265359
2.71828182846
0.841470984808
1.41421356237
```

- math 모듈은 수학적으로 정의된 변수, 함수 지원됨
- math.pi → 파이 값 / math.e → 지수 값
- math.sin → 싸인 값 / math.sqrt → 제곱근 값

▣ 수치 자료형

7. math 모듈의 수치 연산 함수

```
r = 5.0 # 반지름
a = math.pi * r * r # 면적

degree = 60.0
rad = math.pi * degree / 180.0 # 각도를 라디안으로 변환
print math.sin(rad), math.cos(rad), math.tan(rad) #sin, cos, tan
```

```
0.866025403784 0.5 1.73205080757
```

- 반지름이 5인 원 a의 면적을 구하는 공식
- 각도가 60도일 때 라디안 값을 구하는 공식
- 그 후 라디안 값을 싸인, 코싸인, 탄젠트 값으로 출력
- math 모듈로 여러 가지 수치 연산 활용 가능

파이썬 프로그래밍

수치형 자료형,
문자열 자료형

한국기술교육대학교
온라인평생교육원

▣ 문자열

1. 문자열 형식

```
print 'Hello World!'
print "Hello World!"
```

```
Hello World!
Hello World!
```

- 문자열을 만들 때에는 단일 따옴표 또는 이중 따옴표를 사용
- 두 따옴표는 사용 시 차이 없으므로 선택적으로 사용 가능
- 두 따옴표를 모두 사용할 수 있도록 만든 이유는?
- Hello “World”를 출력하려면 단일 따옴표로 문자열 생성
- Hello ‘World’를 출력하려면 이중 따옴표로 문자열 생성

▣ 문자열

1. 문자열 형식

```
multiline = '''
To be, or not to be
that is the question
'''
print multiline

multiline2 = """
To be, or not to be
that is the question
"""
print multiline2
```

```
To be, or not to be
that is the question

To be, or not to be
that is the question
```

- 여러 줄의 문자열을 만들 때에는 ''' ''' 또는 """ """ 사용
- 첫 번째 ''' 뒤에 개행 문자가 있음
- 문자열 첫 번째 줄 바꿈이 된 이유는 개행 문자 때문
- 마지막 ''' 앞에 개행 문자가 있음
- 문자열 마지막 줄 바꿈이 된 이유도 개행 문자 때문
- 이중 따옴표 세 개로 만든 여러 줄의 문자열
- 마찬가지로 개행 문자가 처음과 끝에 있음

▣ 문자열

2. 인덱싱과 슬라이싱

인덱싱

```
s = "Hello world!"
print s[0]
print s[1]
print s[-1]
print s[-2]
```

```
H
e
!
d
```

- 문자열은 시퀀스(순차자료)형으로 인덱스가 붙음
-

문자	H	e	l	l	o	(공백)	w	o	r	l	d	!
인덱스	0	1	2	3	4	5	6	7	8	9	10	11

- 인덱싱: 꺽쇠괄호([]) 안에 인덱스 번호 입력
- print s[0] → 문자열 s에서 인덱스 0인 문자(H) 출력
- print s[1] → 문자열 s에서 인덱스 1인 문자(e) 출력
- print s[11] → 문자열 s에서 인덱스 11인 문자(!) 출력
- 인덱스 앞에 '–(마이너스)' 붙이면 뒤에서부터 셈

▣ 문자열

2. 인덱싱과 슬라이싱

슬라이싱

```
s = "Hello world!"
print s[1:3]
print s[0:5]
```

```
el
Hello
```

- 슬라이싱: 꺽쇠괄호([]) 안에 인덱스 번호와 콜론(:) 입력
- [시작할 인덱스(포함) : 끝낼 인덱스(제외) : 스텝]
- S[1:3] → 인덱스 1인 문자(포함)부터 3인 문자(제외)까지
- 슬라이싱이란 자르기
- [1:3] → 인덱스 1(e)부터 인덱스 3 앞(l)까지 자르기
- 인덱스 3(두 번째 l)은 제외, 인덱스 2(첫 번째 l)까지만 출력
- [0:5] → 인덱스 0(H)부터 인덱스 5 앞(o)까지 자르기

▣ 문자열

2. 인덱싱과 슬라이싱

슬라이싱

```
s = 'Hello'
print s[1:]
print s[:3]
print s[:]
```

```
ello
Hel
Hello
```

- 콜론(:)만 있고 값이 없을 경우 → 기본값으로 사용
- [1:] → 인덱스 1(H)부터 마지막(자료형의 크기)까지
- 자료형의 크기는 5 → [1:]는 [1:5]와 같음
- [1:5] → 인덱스 1(e)부터 인덱스 5앞(o)까지 자르기
- [:3] → 처음(인덱스 0)부터 인덱스 3앞(l)까지
- [:] → 전체 문자열(슬라이싱 하지 않음)

▣ 문자열

2. 인덱싱과 슬라이싱

슬라이싱

```
s = 'abcd'
print s[::2]
print s[::-1]
```

```
ac
dcba
```

- [::값] → 해당 값으로 스텝
- H(인덱스 1)와 e(인덱스 2)의 차이는 1 ← 이 차이가 스텝
- 기본적인 인덱스의 차이(스텝)는 1
- 스텝 2 → 인덱스 차이가 2씩 나도록 슬라이싱
- [::2] → 처음부터 마지막까지 인덱스 차이가 2씩 나도록 슬라이싱
- 인덱스 0(a)을 가져온 다음 인덱스 2(c)를 가져옴
- 인덱스 2(c)를 가져온 다음 인덱스 4를 가져옴(없으므로 끝)
- 스텝의 값만큼 문자의 인덱스 차이가 나도록 슬라이싱
- 스텝의 값이 마이너스 → 인덱스의 차이가 마이너스
- 문자열을 거꾸로 가져오게 됨
- 2(c의 인덱스) - 3(d의 인덱스) = -1
- 1(b의 인덱스) - 2(c의 인덱스) = -1
- [::-1] → 문자열을 거꾸로 뒤집어 가져옴

▣ 문자열

2. 인덱싱과 슬라이싱

슬라이싱

```
s = 'Hello World'
s[0] = 'h'
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-88-28020f3d59a5> in <module>()
      1 s = 'Hello World'
----> 2 s[0] = 'h'

TypeError: 'str' object does not support item assignment
```

- 문자열 자료형은 내부 내용 변경 불가능
- s[0] = h → 문자열 s의 인덱스 0을 h로 변경한다는 의미
- s[0] = h → 수행해보면 에러 발생
- 문자열 내의 문자를 바꿀 수 없으므로 주의

▣ 문자열

2. 인덱싱과 슬라이싱

슬라이싱

```
s = 'Hello World'
s = 'h' + s[1:]
s
```

```
'hello World'
```

- 문자열을 바꾸기 위해서는 슬라이싱을 활용
- 'h' + s[1:] → 문자 h와 인덱스 1부터 슬라이싱한 결과를 합함
- 'h' + s[1:] = h + ello World = hello World
- 문자열의 기존 내용이 바뀐 것이 아님
- 새로운 문자열이 구성되어 할당된 것
- 따라서 기존 문자열은 쓰레기(garbage)

▣ 문자열

3. 문자열 연산

```
print 'Hello' + '' + 'World'
print 'Hello' * 3
print '-' * 60
```

```
HelloWorld
HelloHelloHello
------------------------------------------------------------
```

- 문자열 + 문자열 → 순서 그대로 두 문자열 연결
- print 'Hello' + ''(공백없음) + 'World' → HelloWorld
- 문자열 * 숫자 → 숫자만큼 문자열 반복
- print 'Hello' * 3 → HelloHelloHello
- print '-' * 60 → -가 60번 반복되어 나타남

▣ 문자열

4. 문자열의 길이

```
s = 'Hello World'
len(s)
```

```
11
```

- len(s) → s의 문자열 길이를 의미
- 주의! 이클립스에서는 print를 반드시 추가해야 결과값 출력됨

▣ 문자열

5. 문자열내 포함 관계 여부

```
s = 'Hello World'
print 'World' in s
print 'World' not in s
```

```
True
False
```

- 'A' in B → B 안에 A가 있다(결과는 True or False)
- print 'World' in s → 문자열 s 안에 World가 있다 → True
- 'A' not in B → B 안에 A가 없다(결과는 True or False)
- print 'World!' in s → 문자열 s 안에 World!가 있다 → False
- print ' ' in s → 문자열 s 안에 공백이 있다 → True