Python 과학 프로그래밍 기초
Python 문법 기초 (2)

한림대학교 소프트웨어융합대학
박섭형

2021년 1학기

2. Python 문법 기초 (2)

- 키워드
- Print() / Input()
- 변수와 바인딩

divmod() 함수

- 정수와 실수만 적용 가능

[1]:
```
divmod(11, 3)
```

[1]:
```
(3, 2)
```

[2]:
```
divmod(11.2, 3.2)
```

[2]:
```
(3.0, 1.5999999999999988)
```

Python 키워드

- 키워드는 상수나, 변수 혹은 다른 식별자들의 이름으로 사용할 수 없다.
- 예약어는 모두 소문자로 구성되어 있다.
- None은 키워드는 아니지만 Python에서 상수로 사용하고 있는 내장 객체이므로, 사용자가 다른 용도로 사용하면 안된다.
- Python 언어의 keyword 리스트

[3]:
```
import sys
sys.version
```

[3]:
```
'3.8.5 (default, Sep  3 2020, 21:29:08) [MSC v.1916 64 bit (AMD64)]'
```

[4]:
```
import keyword
keyword.kwlist
len(keyword.kwlist)
```

[4]:
```
35
```

Print( ): 정보를 화면에 표시하기

[5]:
```
a = 1
b = 2
(a, b)
```

[5]:
```
(1, 2)
```

[6]:
```
a = 1
b = 2
print(a)
print("b =", b)
print(a, b)
print(a, b, '\n', sep=', ')
print(a)
```

```
1
b = 2
1 2
1, 2,

1
```

[7]:
```
a = 1
b = 2
c = 3
print(a, b, end=' ')
print(b, end=' ')
```

```
print(c, end=' ')
```

```
1 2 2 3
```

키보드를 이용한 문자열 입력

[9]:
```
input("Press any key to exit.")
```

```
Press any key to exit.5
```

[9]:
```
'5'
```

[10]:
```
s = input("Enter any string and press enter.")
print(s)
```

```
Enter any string and press enter.5
5
```

변수

- 변수를 사용할 때 변수형을 사전에 선언하지 않는다.
- 변수는 메모리에 있는 객체의 이름
- 변수는 객체의 참조(reference)일 뿐, 변수에 객체를 복사하지 않는다

[11]:
```
a = 3
b = 7
type(a), type(b)
```

[11]:
```
(int, int)
```

a, b는 정수형 객체와 바인딩됐기 때문에 정수형 변수로 지정된다. type(a), type(a)

[12]:
```
c = b / a # c는 실수형으로 처리된다.
c         # 왜 2.3333333333333333이 아닐까?
```

[12]:
```
2.3333333333333335
```

[13]:
```
type(c)
```

[13]:
```
float
```

[14]:
```
d = 2      # d는 정수형 변수로 지정된다.
e = 4.3    # e는 실수형 변수로 지정된다.
f = d + e # 정수형과 실수형을 더한 결과는 실수형이 된다.
f
```

[14]: 6.3

[15]:
```
g = pow(2, 200)
g, type(g)
```

[15]: (1606938044258990275541962092341162602522202993782792835301376, int)

[16]:
```
h = 2 ** 200
h, type(h)
```

[16]: (1606938044258990275541962092341162602522202993782792835301376, int)

여러 변수에 동시에 값을 지정하는 방법

[17]:
```
a = b = c = 3
print(a, b, c)
print(id(a), id(b), id(c))
```

```
3 3 3
140731260741472 140731260741472 140731260741472
```

[18]:
```
b = 4
print(a, b, c)
print(id(a), id(b), id(c))
```

```
3 4 3
140731260741472 140731260741504 140731260741472
```

[19]:
```
c
```

[19]: 3

[20]:
```
a, b, c
```

[20]: (3, 4, 3)

여러 변수에 동시에 값을 지정하는 방법

[21]:
```
a, b, c
```

[21]: (3, 4, 3)

두 변수의 값 교환

[22]:
```
a, b = 3, 5
print("a=", a, ", b=", b)
a, b = b, a
print("a=", a, ", b=", b)
```

```
a= 3 , b= 5
a= 5 , b= 3
```

[23]:
```
a = 3
b = 5
print("a=", a, ", b=", b)
temp = a
a = b
b = temp
a, b
print("a=", a, ", b=", b)
```

```
a= 3 , b= 5
a= 5 , b= 3
```

대화형 모드에서 변수 _

[24]:
```
a = 3
b = 51
c = 12
a * b
```

[24]: 153

[25]:
```
_ * 3
```

[25]: 459

일반 변수 _

[26]:
```
for n in range(4): # 0, 1, 2, 3
    print(n)
```

```
0
1
2
3
```

[27]:
```
for _ in range(4):
    print("Print a sentence repeatedly.")
```

```
Print a sentence repeatedly.
Print a sentence repeatedly.
Print a sentence repeatedly.
Print a sentence repeatedly.
```

Walrus 연산자

- Python 3.8에서 추가
- x := 3
  - x에 3을 할당
  - 3을 반환
- x = 3
  - x에 3을 할당

[28]:
```
x = 3
```

[29]:
```
(x := 3)
```

[29]: 3

[30]:
```
total = 0
print("이 프로그램은 숫자를 입력받아 더한 결과를 출력한다.")
while True:
    number = input("Enter a number digit or 'q' to stop: ")
    if number == "q":
```

```
        break
    total = total + eval(number)
print("Sum = ", total)
```

이 프로그램은 숫자를 입력받아 더한 결과를 출력한다.

```
Enter a number digit or 'q' to stop: 5
Enter a number digit or 'q' to stop: 8
Enter a number digit or 'q' to stop: q
Sum =  13
```

[31]:
```
total = 0
print("이 프로그램은 숫자를 입력받아 더한 결과를 출력한다.")
while (number := input("Enter a number digit or 'q' to stop: ")) != 'q':
    total = total + eval(number)
print("Sum = ", total)
```

이 프로그램은 숫자를 입력받아 더한 결과를 출력한다.

```
Enter a number digit or 'q' to stop: 5
Enter a number digit or 'q' to stop: 8
Enter a number digit or 'q' to stop: q
Sum =  13
```

바인딩 (Binding)

- 어느 'name'이 '객체'를 참조하도록 만드는 과정
- Python에서 다루는 모든 것은 객체(object)

[32]:
```
x = 5
print(id(5))
print(id(x))
print(type(5))
print(type(x))
```

```
140731260741536
140731260741536
<class 'int'>
<class 'int'>
```

[33]:
```
x = 5
y = 5
print(id(x))
print(id(y))
print(x == y)
print(x is y)
```

```
140731260741536
140731260741536
True
True
```

[34]:
```
x = 5.
y = 5.
print(id(x))
print(id(y))
print(x == y)
print(x is y)
```

```
2416530037776
2416530036144
True
False
```

2.3 구문(Statement)과 표현(Expression)

- 표현: print할 수 있거나, 변수에 할당할 수 있는 것
  - 2 + 1
  - x + 1
  - min(2, 3) + 1
  - None
  - True
- 구문: 결과를 만들기 위해 실행된다
  - x = 1
  - y = (x if x > 0 else -x)
  - if x > 5:

자료형

- Python numbers
  - int: 정수형. 메모리가 허락하는 한 무제한의 자리수로 정수를 계산할 수 있음.
  - float: 부동 소수점 형식의 실수형. IEEE 754 형식을 따름.
  - complex: 복소수형
  - bool: int의 subclass로 True, False 두 object만 존재.
- NoneType: None이 유일한 object로, 아무 값도 없는 것을 의미.
- 시퀀스형
  - string: 유니코드 문자열. 변경 불가능(immutable).
    - character: 0 ~ 0x10FFFF 사이 정수인 Unicode code point
  - bytes
    - 0 과 255 사이의 숫자들의 시퀀스로 변경 불가능.
    - 화면에 나타날 때는 ASCII 문자열
  - bytearray: bytes의 변경 가능 버전
  - list: 변경 가능(mutable) 자료형.
  - tuple: 변경 불가능 자료형.
  - range

[35]:
```
code_points = [0x0031, 0x0061, 0x0041, 0x2161, 0x2164, 0x265e,
               0x265f, 0x1f600, 0x1f609, 0xd55c, 0xae00]
for cp in code_points:
    print(f"The glyph of code point 0x{cp:x} (in decimal {cp}) is {chr(cp)}.")
```

```
The glyph of code point 0x31 (in decimal 49) is 1.
The glyph of code point 0x61 (in decimal 97) is a.
The glyph of code point 0x41 (in decimal 65) is A.
The glyph of code point 0x2161 (in decimal 8545) is ⅱ.
The glyph of code point 0x2164 (in decimal 8548) is Ⅴ.
The glyph of code point 0x265e (in decimal 9822) is ♞.
The glyph of code point 0x265f (in decimal 9823) is ♟.
The glyph of code point 0x1f600 (in decimal 128512) is 😀.
The glyph of code point 0x1f609 (in decimal 128521) is 😉.
The glyph of code point 0xd55c (in decimal 54620) is 한.
The glyph of code point 0xae00 (in decimal 44544) is 글.
```

[36]:
```
ex_str = 'Unicode string \ud55c\uae00'
print(ex_str)
alt_str = 'Unicode string 한글'
print(alt_str)
ex_str == alt_str
```

```
Unicode string 한글
Unicode string 한글
```

[36]: True

[37]:
```
n글 = 3
n글
```

[37]: 3

[38]:
```
ex_str = "Unicode string 한글"
ex_bytes = ex_str.encode('utf-8')
print(ex_bytes)
```

```
b'Unicode string \xed\x95\x9c\xea\xb8\x80'
```

[39]:
```
ex_bytes_2 = bytes(ex_str, 'utf-8')
print(ex_bytes_2)
```

```
b'Unicode string \xed\x95\x9c\xea\xb8\x80'
```

[40]:
```
decoded_ex_bytes = ex_bytes.decode('utf-8')
print(decoded_ex_bytes)
```

```
Unicode string 한글
```

[41]:
```
decoded_ex_bytes_2 = str(ex_bytes_2, 'utf-8')
print(decoded_ex_bytes_2)
```

```
Unicode string 한글
```

[42]:
```
decoded_ex_bytes == ex_str
```

[42]:
```
True
```

- dict: 사전형.
- 집합형
  - set: 집합형. 중복을 허락하지 않는다. mutable.
  - frozenset: immutable, hashable.
- function

정수 자료형

- 메모리가 허락하는 한 무제한의 정밀도로 표현 가능
- 10 진수, 2 진수, 8 진수, 16 진수 표현 가능

[43]:
```
x = 111   # 10 진수
z = 0b111 #  2 진수
w = 0o11  #  8 진수
y = 0x11  # 16 진수

11 + 0x11 + 0b111
```

[43]:
```
35
```

[44]:
```
x, y, z, w
```

[44]:
```
(111, 17, 7, 9)
```

비교 연산자

- 복소수 비교 불가능

연산	의미
〉	왼쪽 항이 오른쪽 항보다 큰가?
〉=	왼쪽 항이 오른쪽 항보다 크거나 같은가?
〈	왼쪽 항이 오른쪽 항보다 작은가?
〈=	왼쪽 항이 오른쪽 항보다 작거나 같은가?
==	왼쪽 항과 오른쪽 항이 같은가?
!=	왼쪽 항과 오른쪽 항이 같지 않은가?

[45]:
```
3 > 2, 3 >= 2, 3 < 2, 3 <= 2, 3 == 2, 3 != 2
```

[45]:
```
(True, True, False, False, False, True)
```

[46]:
```
3 <= 4 < 5, 5 < 5 < 8
```

[46]:
```
(True, False)
```

[47]:
```
3 + 2j > 2 - 2j
```

```
---------------------------------------------------------------------------
TypeError                                 Traceback (most recent call last)
<ipython-input-47-5b14319112a8> in <module>
----> 1 3 + 2j > 2 - 2j

TypeError: '>' not supported between instances of 'complex' and 'complex'
```

논리 연산자

- and: 연산자의 좌우 항이 모두 True일 때 True
- or: 연산자의 좌우 항 중에 하나라도 True일 때 True
- not: 피연산자(operand)의 부정

x	y	x and y	x or y	not x
True	True	True	True	False
True	False	False	True	False
False	True	False	True	True
False	False	False	False	True

bool() 값이 False인 객체들

[50]:
```
bool(0), bool(0.0), bool(""), bool([]), bool(tuple()), bool({}), bool(dict()),␣
↪bool(None)
```

[50]:
```
(False, False, False, False, False, False, False, False)
```

not 연산자는 모든 객체에 적용할 수 있다

[51]:
```
print(not 0, not 1, not 2.0, not 1+1j, not 2**3, not -2+2)
```

```
True False False False False True
```

and와 or는 True와 False만을 반환하는 것이 아니라 연산에 사용된 피연산자 중의 하나를 반환

[52]:
```
True and False
```

[52]:
```
False
```

[53]:
```
3 and 4
```

[53]:
```
4
```

[54]:
```
0 and 4
```

[54]:
```
0
```

[55]:
```
"" and 4
```

[55]:
```
''
```

[56]:
```
0 or "str"
```

[56]:
```
'str'
```

[57]:
```
0 and 4
```

[57]:
```
0
```

Lazy Evaluation

Python 해석기는 and와 or 평가 중간 단계에서 최종 값이 결정되면 더 이상 평가를 진행하지 않는다.

- A1 and A2 and …. An
  - Ai, $1 \leq i \leq n$,이 모두 True이면 An의 값을 반환한다.
  - 그렇지 않으면, 첫번째 False 피연산자를 반환한다.

[58]:
```
1 and 4 and 5 and 7 and "str" and "[1, 2, 3]"
```

[58]:
```
'[1, 2, 3]'
```

[59]:
```
1 and 4 and 5 and [] and "str" and {'a':1, 'b':5}
```

[59]: []

- A1 or A2 or…. An
  - Ai, $1 \leq i \leq n$, 이 모두 False이면 An의 값을 반환한다.
  - 그렇지 않으면, 첫번째 True 피연산자를 반환한다.

```
[60]: 0 or [] or '' or {}
```

[60]: {}

```
[61]: 0 or [] or 'str' or {}
```

[61]: 'str'

- and와 or가 섞여 있으면 and를 먼저 평가한다

```
[62]: 1 and 4 and 8 or 0 and 5
```

[62]: 8

연산자 우선 순위

순위	연산자
1	( )
2	**
3	+x, −x, ~x
4	*, /, //, %
5	+, −
6	≪, ≫
7	&
8	^
9	|
10	==, !=, 〉, 〉=, 〈, 〈=, is, is not, in, not in
11	not
12	and
13	or

- 한 명령 줄에 우선 순위가 동일한 연산자들이 여러 개 있는 경우에는 왼쪽에서 오른쪽 순으로 연산이 실행됨

[63]:
```
print(1 and 3 and 2)
print(1 and 0 and 1)
print(0 or 3 or 2)
print(0.0 or 0 or 1)
print(1 and 2 or 0)
print(0 or 3 and 1)
```

```
2
0
3
1
2
1
```