# 영상처리 프로그래밍을 위한
# Python 문법 기초 (1)

한림대학교 소프트웨어융합대학
박섭형

2022년 1학기

[1]:
```
import sys
sys.version
```

[1]:
```
'3.9.7 (default, Sep 16 2021, 16:59:28) [MSC v.1916 64 bit (AMD64)]'
```

## 0.1 Python 자료 구조

**리터럴**(literal), **상수**(constant), **변수**(variable)

- 리터럴: 소스 코드에서 고정된 값을 표현하는 표기법
- 숫자 리터럴
  - 정수
    - 10진수: 3, -2
    - 2진수 (binary number): 0b101, -0b101
    - 8진수 (octal number): 0o37
    - 16진수 (hexadecimal): 0x3a, -0x3a, 0x1B, -0x1C
  - 실수: 1.2, -3.45, 1.4e3, 1.5e-3
  - 복소수: 1 + 2j, -3 +1j, 3 + 4J, 2 - 5J
- 문자열 리터럴
  - “abc”, ‘abc’, “문자열”, ‘A’, “ ‘spring’ ”, ‘He said “Hello”.’
- 논리값 리터럴
  - True, False
- 특수 리터럴

  - None

- 컬렉션 리터럴
  - 리스트: [1, 3, 1.4, “리스트”], [1, 2, [3, 4]], [1, 2, (3, 4)]
  - 튜플: (1, 3, 1.4, (1, 3))
  - 집합: {1, 3, 5}, {1, 3, “ask”, (1,3)}
  - 사전: {‘name’: “홍길동”, ‘age’: 23}

[2]:
```
0b101
```

[2]:
```
5
```

[3]:
```
-0b101
```

[3]:
```
-5
```

[4]:
```
0o37
```

[4]:
```
31
```

[5]:
```
0x3a
```

[5]:
```
58
```

[6]:
```
0x3A
```

[6]:
```
58
```

[7]:
```
1.2e2, -1.2e-2, 3.5e4
```

[7]:
```
(120.0, -0.012, 35000.0)
```

### 변수

- 메모리에 저장되어 있는 리터럴로 표현된 값(Python에서는 객체)를 잠조하는 이름으로 사용자가 정한다
- 변수를 사용할 때 변수형을 사전에 선언하지 않는다.
- 변수가 참조하는 값들은 프로그램 내에서 여러 방법으로 변경된다
- Python 3의 변수의 이름(식별자)는 Unicode를 기반으로 한다.
- ASCII 코드만을 사용하는 경우 식별자 명명 규칙은 Python 2의 식별자 명명 규칙과 같다.

**Python 2 (ASCII 기반) 식별자 명명 규칙**

- 식별자는 A부터 Z 또는 a부터 z까지의 문자, 또는 밑줄 문자 (_) 다음에 0개 이상의 문자나 밑줄문자 혹은 숫자를 결합하여 만들 수 있다.
- !, @, #, $, %, & 등과 같은 특수 문자를 사용할 수 없다.
- 식별자의 대소문자를 구분한다.
- 식별자의 이름으로 키워드를 사용할 수 없다.
- 식별자의 길이에 제한은 없으나, PEP 8은 Python 프로그램에서 한 줄의 길이를 최대 79 자로 제한할 것을 권장한다.

[8]:
```
import keyword
keyword.kwlist
```

[8]:
```
['False',
 'None',
 'True',
 '__peg_parser__',
 'and',
 'as',
 'assert',
 'async',
 'await',
 'break',
 'class',
 'continue',
 'def',
 'del',
 'elif',
 'else',
 'except',
 'finally',
 'for',
 'from',
 'global',
 'if',
 'import',
 'in',
```

```
 'is',
 'lambda',
 'nonlocal',
 'not',
 'or',
 'pass',
 'raise',
 'return',
 'try',
 'while',
 'with',
 'yield']
```

[9]:
```
a = 3
weight = 3.5
height = 1.5e3
A3_ = 1 + 1J
name = '홍길동'
print(a, type(a))
print(weight, type(weight))
print(height, type(height))
print(A3_, type(A3_))
print(name, type(name))
```

```
3 <class 'int'>
3.5 <class 'float'>
1500.0 <class 'float'>
(1+1j) <class 'complex'>
홍길동 <class 'str'>
```

[10]:
```
print(a, type(a))
print(weight, type(weight))
a = a + weight
print(a, type(a))
```

```
3 <class 'int'>
```

```
3.5 <class 'float'>
6.5 <class 'float'>
```

[11]:
```
print(a, type(a))
a = 'spring'
print(a, type(a))
```

```
6.5 <class 'float'>
spring <class 'str'>
```

상수(constant)

- 항상 같은 값이 저장된 곳
- Python에는 원칙적으로 상수를 정의하는 기능이 없음
- 통상적으로 대문자로 만든 변수를 상수로 사용

[12]:
```
PI = 3.141592
PI = 1
```

**Walrus 연산자** (:=)

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

[13]:
```
x = 3
```

[14]:
```
(x := 3)
```

[14]:
```
3
```

[15]:
```
total = 0
print("이 프로그램은 숫자를 입력받아 더한 결과를 출력한다.")
while True:
    number = input("Enter a number to add, 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 to add, or 'q' to stop: 3
Enter a number to add, or 'q' to stop: 1.5
Enter a number to add, or 'q' to stop: q
Sum =  4.5
```

[16]:
```
total = 0
print("이 프로그램은 숫자를 입력받아 더한 결과를 출력한다.")
while (number := input("Enter a number to add, or 'q' to stop: ")) != 'q':
    total = total + eval(number)
print("Sum = ", total)
```

이 프로그램은 숫자를 입력받아 더한 결과를 출력한다.

```
Enter a number to add, or 'q' to stop: 3
Enter a number to add, or 'q' to stop: 1.5
Enter a number to add, or 'q' to stop: q
Sum =  4.5
```

[17]:
```
a, b = 208, 80
r = a % b
while r:
    a, b = b, r
    r = a % b
print(b)
```

16

[18]:
```
a, b = 208, 80
while r:= a % b:
    a, b = b, r
print(b)
```

16

[19]:
```
b, r = 208, 80
while r:= (a:=b) % (b:=r):
    pass
print(b)
```

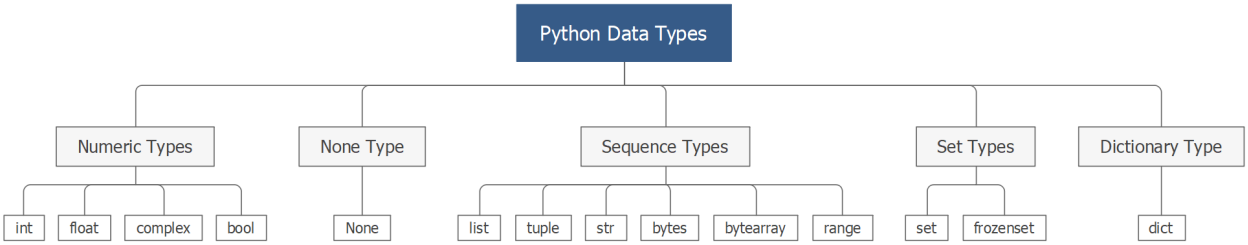
16

[20]:
```
r = 208 % 80
```

[21]:
```
(r := 208 % 80)
```

[21]: 48

### 자료형




- Numeric Types
  - int: 정수형. 메모리가 허락하는 한 무제한의 자리수로 정수를 계산할 수 있음.
  - float: 부동 소수점 형식의 실수형. IEEE 754 배정도(double precision, FP64, float64, binary64) 형식
  - complex: 복소수형
  - bool: int의 subclass로 True, False 두 object만 존재.
- NoneType: None이 유일한 object로, 아무 값도 없는 것을 의미.
- 시퀀스형
  - list: 변경 가능(mutable) 자료형.
  - tuple: 변경 불가능(immutable) 자료형.
  - string: 유니코드 문자열. 변경 불가능(immutable).
    * character: utf-8 인코딩, 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
  - range:range() 함수가 생성하는 변경 불가능 자료형

- 리스트: 변경 가능

[22]:
```
a = [1, 2, 3]
print(a, id(a))
```

```
[1, 2, 3] 2418688419968
```

[23]:
```
print(a[0])
```

```
1
```

[24]:
```
a[0] = 10
print(a, id(a))
```

```
[10, 2, 3] 2418688419968
```

- 튜플: 변경 불가능

[25]:
```
b = (1, 2, 3)
print(b[0])
```

```
1
```

[26]:
```
b[0] = 10
```

```
␣
↪---------------------------------------------------------------------------

TypeError                                 Traceback (most recent call␣
↪last)
```

```
        C:\Users\Public\Documents\ESTsoft\CreatorTemp/ipykernel_18784/1338681980.
↪py in <module>
   ----> 1 b[0] = 10


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

- 문자열: 변경 불가능

[27]:
```
s = "abc한글"
print(s[1], s[3])
```

```
b 한
```

[28]:
```
s[1] = "B"
```

```
      ␣
↪---------------------------------------------------------------------------

        TypeError                                 Traceback (most recent call␣
↪last)

        C:\Users\Public\Documents\ESTsoft\CreatorTemp/ipykernel_18784/55651051.
↪py in <module>
   ----> 1 s[1] = "B"


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

[29]:
```
code_points = (0x31, 0x61, 97, 0x41, 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 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.
```

- 집합형
  - set: 집합형. 중복을 허락하지 않는다. mutable.
  - frozenset: immutable, hashable.
- dict: 사전형.
  - key-value 쌍

[30]:
```
set1 = {1, 4, 5, 4, 6, 5, 4, 4}
set1
```

[30]:
```
{1, 4, 5, 6}
```

[31]:
```
dict1 = {'이름': '홍길동', 'age': 23}
dict1
```

[31]:
```
{'이름': '홍길동', 'age': 23}
```

[32]:
```
dict1['이름']
```

[32]:
```
'홍길동'
```