스마트 모빌리티 프로그래밍

Ch 2. 파이썬 기본 자료형, 기본 연산자와 기본 명령어

영남대학교 정보통신공학과

교수 김 영 탁

(Tel : +82-53-810-2497; E-mail : ytkim@yu.ac.kr)

Outline

- 기본 숫자 자료형
  - bool, int, float, complex
- 기본 시퀀스 (sequence) 자료형 개요
  - list, range
- Indexing, Slicing
- 파이썬 기본 연산자와 기본 명령어
- 문법적 오류와 논리적 오류
- 효과적인 프로그램 개발 요령

파이썬 기본 숫자 자료형과 연산
– bool, int, float, complex

자료형 클래스 (1)

◆ 파이썬 클래스 이름과 자료형

- 모든 파이썬 자료형은 클래스로 구현되어 있음
  (e.g., bool, int, float, complex, str, range, list, tuple, set, dict, etc.)

◆ 객체와 인스탄스

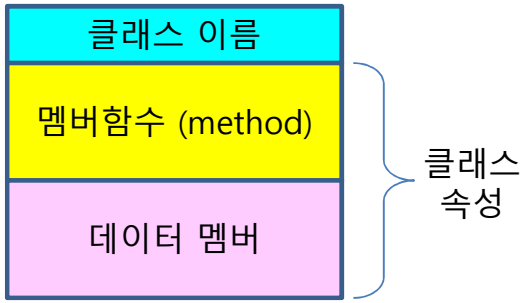
- class : 자료형을 구현하며, 속성 (멤머 데이터, 멤버 함수)를 가짐
- object : 객체 인스탄스 (object instance), 변수 (variable)
- e.g.)

class	Example of Object instance
int	i = 10 # integer
float	pi = 3.141592 # float
str	greeting = "Hello" # str

자료형 클래스 (2)

◆ 클래스 속성 (attribute)

- 클래스는 멤버함수 (메소드)와 멤버 데이터를 속성으로 가짐
- 클래스의 속성은 dot(.) 연산자를 사용하여 소속을 정의
- *e.g.)*
  - *object_name.attribute_identifier*



파이썬 자료형

◆ 파이썬 내장형 자료형 (embedded data type)

type		mutable	iterable	remarks
boolean	bool	no	no	True, False
numeric	int	no	no	
	float	no	yes	
	complex	no	yes	
sequence	str	no	yes	
	bytes	no	yes	
	bytearray	yes	yes	
	memoryview	no/yes	yes	
	list	no	yes	[0, 1, 2, 3],
	tuple	no	yes	(1, 2, 1, 2, 3)
	range	no	yes	
mapping	dict	yes	yes	{1:'A', 2:'B', 3:'C'}
set	set	yes	yes	{1, 2, 3}
	frozenset	no	yes	

정수 데이터의 리스트 생성 및 기본 연산

◆ list와 for-loop

```
# list

L = [1, 3, 5, 7, 9]
print("L = ", L)
print("len(L) = ", len(L))

sum_L = 0
for i in range(len(L)):
    sum_L = sum_L + L[i] #iterable
print("sum_L = ", sum_L)

L[0] = 100 # checking mutability
print("After L[0] = 100, L = ", L)
```

```
L =  [1, 3, 5, 7, 9]
len(L) =  5
sum_L =  25
After L[0] = 100, L =  [100, 3, 5, 7, 9]
```

range() 함수를 사용한 정수 리스트 생성

◆ range() 함수를 사용한 정수형 데이터 리스트 생성

```
# list, range(), append()

L1 = list()
for n in range(10):
    L1.append(n)
print("L1 : ", L1)

L2 = list()
for n in range(1, 10, 2):
    L2.append(n)
print("L2 : ", L2)

L3 = list()
for n in range(0, 15, 3):
    L3.append(n)
print("L3 : ", L3)
```

```
L1 :  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
L2 :  [1, 3, 5, 7, 9]
L3 :  [0, 3, 6, 9, 12]
```

불리언 자료형 (bool)

◆ bool 자료형

- True 또는 False의 값을 가짐

◆ 불리언 자료형 연산자

논리 연산자	의미
x or y	if x is False then y else x
x and y	if x is False the x else y
not x	if x is False then True else False

◆ 불리언 자료형 연산 예

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

불리언 원소와 list

◆ Boolean 원소의 list 구성

```
# Boolean operations wth lists of boolean elements

BL_1 = [False, False, True, True]
BL_2 = [False, True, False, True]

print("BL_1 = ", BL_1)
print("BL_2 = ", BL_2)

BL_and = []
BL_or = []
BL_not = []
for i in range(len(BL_1)):
    da = BL_1[i] and BL_2[i]
    BL_and.append(da)

    do = BL_1[i] or BL_2[i]
    BL_or.append(do)

    dn = not BL_1[i]
    BL_not.append(dn)

print("BL_1 and BL_2 = ", BL_and)
print("BL_1 or BL_2 = ", BL_or)
print("not BL_1 = ", BL_not)
```

```
BL_1 =  [False, False, True, True]
BL_2 =  [False, True, False, True]
BL_1 and BL_2 =  [False, False, False, True]
BL_1 or BL_2 =  [False, True, True, True]
not BL_1 =  [True, True, False, False]
```

숫자 자료형

◆ 파이썬 프로그램의 숫자 자료형

숫자 자료형	의미
int	<class 'int'> 소숫점 이하 값이 사용되지 않는 정수
float	<class 'float'> 소숫점 이하 값이 사용되는 실수
complex	<class 'complex'> 실수부와 허수부가 포함되는 복소수

◆ 숫자 데이터에 대한 기본 산술 연산

산술연산자	의미
c = a + b	덧셈
c = a - b	뺄셈
c = a * b	곱셈
c = a / b	실수 나눗셈
c = a // b	정수 나눗셈
c = a % b	모듈로, 나머지
c = a ** b	거듭제곱

숫자 자료형에 대한 비교 연산자

◆ 숫자 데이터에 대한 비교 연산자

비교 연산자	의미
a < b	less than
a <= b	less than or equal
a > b	greater than
a >= b	greater than or equal
a == b	equal
a != b	not equal
a is b	object identity (two objects are same)
a is not b	negated object identity (two objects are different)

숫자 자료형의 비트단위 연산자

◆ 비트단위 연산자

비트 연산자	의미
x & y	bit-wise and
x | y	bit-wise or
x ^ y	bit-wise exclusive or
~x	bit-wise not
x << n	bit-wise shift left by n bit positions
x >> n	bit-wise shift right by n bit positions

◆ 비트 단위 연산 공식

x	y	x & y	x | y	x ^ y	~x
0	0	0	0	0	1
0	1	0	1	1	1
1	0	0	1	1	0
1	1	1	1	0	0

숫자 자료형에 대한 비트단위 연산

◆ 비트단위 연산 예

변수 및 연산	연산 결과 (비트 단위 표시)	값 (16진수)
x	0000 0111	0x07
y	1010 0011	0xA3
x & y	0000 0011	0x03
x | y	1010 0111	0xA7
x ^ y	1010 0100	0xA4
~x	1111 1000	0xF8
x << 2	0001 1100	0x1C
x >> 2	0000 0001	0x01
y << 2	1000 1100	0x8C
y >> 2	1110 1000 (MSB가 sign 비트일 때)	0xE8

숫자 자료형 관련 파이썬 내장 함수

◆ 숫자 데이터 관련 파이썬 내장 함수

내장 함수	의미
abs(x)	숫자 x의 절대값을 반환
int(n_str)	정수형 문자열 n-str을 정수 데이터로 변환
float(f_str)	실수형 문자열 f_str을 실수 데이터로 변환
complex(c_str)	복소수형 문자열 c_str을 복소수 데이터로 변환
bin(x)	정수 x를 2진 binary digit 문자열로 변환하여 반환
hex(x)	정수 x를 16진 bytes 문자열로 변환하여 반환
oct(x)	수 x를 8진 bytes 문자열로 변환하여 반환
divmod(a, b)	a를 b로 나눈 후 몫과 나머지를 tuple로 반환 a = q * b + a % b 정수: (a//b, a%b) 실수: (q = math.floor(a, b), a%b)
pow(x, y[, z])	z가 없으면 x의 y 거듭제곱 결과를 반환; z가 있으면 x의 y 거듭제곱 결과를 z로 나눈 나머지 (pow(x, y) % z)를 반환
round(number[, ndigits])	실수 number를 소숫점 이하 ndigits에서 반올림;

숫자 데이터관련 내장 함수

◆ abs(), divmode(), pow(), round()

```
# embedded functions for number data
x = -123
print("x      = {:4d}".format(x))
print("abs(x) = {:4d}".format(abs(x)))

a, b = 7, 2
print("a = ", a)
print("b = ", b)

q, r = divmod(a, b)
print("q, r = divmod(a, b)")
print("q = ", q)
print("r = ", r)

a_b = pow(a, b)
print("pow({}, {}) = {}".format(a, b, a_b))

c = 3.1415926535
c_r = round(c, 4)
print("c = ", c)
print("c_r = ", c_r)
```

```
x      = -123
abs(x) =  123
a =  7
b =  2
q, r = divmod(a, b)
q =  3
r =  1
pow(7, 2) = 49
c =  3.1415926535
c_r =  3.1416
```

숫자 데이터관련 내장 함수

◆ bin(), hex(), oct()

```
# embedded functions for number data
x = 255
bin_x = bin(x)
print("bin(x) = {}".format(bin_x))
hex_x = hex(x)
print("hex(x) = {}".format(hex_x))
oct_x = oct(x)
print("oct(x) = {}".format(oct_x))
```

```
bin(x) = 0b11111111
hex(x) = 0xff
oct(x) = 0o377
```

숫자 자료형 클래스의 속성과 메소드

◆ 숫자 자료형 클래스의 주요 속성과 메소드

분류	내장 함수	의미
정수 자료형	int.bit_length(x)	정수 x를 이진수로 표현하는데 필요한 비트수를 반환 (부호와 leading 0은 제외)
	int.to_bytes(length, byteorder, *, signed=False)	정수를 length로 길이가 지정되는 바이트로 표현하는 bytes를 반환. byteorder는 "big" 또는 "little" 중 하나의 값을 가짐. signed=True이면 음수를 위해 2의 보수 (2's complement) 표현을 사용
	int.from_bytes(bytes, byteorder, *, signed=False)	bytes를 byteorder (big 또는 little)를 고려하여 정수로 변환
실수 자료형	float.hex()	실수의 16진수 표현 문자열로 변환하여 반환
	float.fromhex(s)	16진수 문자열 실수 표현 s로부터 실수를 계산하여 반환
	float.as_integer_ratio()	실수에 대한 비율을 갖는 분자와 분모를 tuple로 반환
	float.is_integer()	실수에 대응하는 오차 없는 정수가 있는지 판단
복소수 자료형	.real	복소수의 실수부
	.imag	복소수의 허수부
	complex.conjugate()	켤레복소수를 반환

숫자 자료형 클래스 – pow(), bit_length()

◆ pow(), bit_length()

```
# methods of number data types classes

d = 10
for e in range(0, 10, 1):
    de = pow(d, e)
    bit_len = int.bit_length(de)
    print("bit_length for {:12} = {:3}"\
          .format(de, bit_len))
```

```
bit_length for           1 =   1
bit_length for          10 =   4
bit_length for         100 =   7
bit_length for        1000 =  10
bit_length for       10000 =  14
bit_length for      100000 =  17
bit_length for     1000000 =  20
bit_length for    10000000 =  24
bit_length for   100000000 =  27
bit_length for  1000000000 =  30
```

복소수 (complex)

◆ class complex([real[, imag]])

```
# complex, conjugate

c1 = complex(3, 4)
print("c1 = ", c1)
print("type(c1) = ", type(c1))
print("c1.real = ", c1.real)
print("c1.imag = ", c1.imag)

cj = complex.conjugate(c1)
print("conjugate(c1) = ", cj)
print("cj.real = ", cj.real)
print("cj.imag = ", cj.imag)

c2 = complex(5, 6)
print("c2 = ", c2)
print("c1 + c2 = ", c1 + c2)
print("c1 - c2 = ", c1 - c2)
print("c1 * c2 = ", c1 * c2)
print("c1 / c2 = ", c1 / c2)
```

```
c1 =  (3+4j)
type(c1) =  <class 'complex'>
c1.real =  3.0
c1.imag =  4.0
conjugate(c1) =  (3-4j)
cj.real =  3.0
cj.imag =  -4.0
c2 =  (5+6j)
c1 + c2 =  (8+10j)
c1 - c2 =  (-2-2j)
c1 * c2 =  (-9+38j)
c1 / c2 =  (0.6393442622950819+0.03278688524590165j)
```

복소수 (complex) 입력, Conjugate

```
# complex, input, conjugate, addition, subtraction,
#   multiplication, division, conjugate

cmplx_str = input("input complex number (a+bj) : ")
c1 = complex(cmplx_str)
print("c1 = ", c1)
print("type(c1) = ", type(c1))
print("c1.real = ", c1.real)
print("c1.imag = ", c1.imag)

c2 = complex.conjugate(c1)
print("conjugate(c1) = ", c2)
print("c2.real = ", c2.real)
print("c2.imag = ", c2.imag)

print("c1 + c2 = ", c1 + c2)
print("c1 - c2 = ", c1 - c2)
print("c1 * c2 = ", c1 * c2)
print("c1 / c2 = ", c1 / c2)
```

```
input complex number (a+bj) : 5+7j
c1 =  (5+7j)
type(c1) =  <class 'complex'>
c1.real =  5.0
c1.imag =  7.0
conjugate(c1) =  (5-7j)
c2.real =  5.0
c2.imag =  -7.0
c1 + c2 =  (10+0j)
c1 - c2 =  14j
c1 * c2 =  (74+0j)
c1 / c2 =  (-0.32432432432432432+0.9459459459459459j)
```

숫자 자료형 데이터의 통계 분석

◆ while-loop, find min and max

```
# while-loop, find min and max of input data list

TARGET_NUM_DATA = 10
L = [] # empty list
num_data = 0
print("Input {} integer data."\
      .format(TARGET_NUM_DATA))
while num_data < TARGET_NUM_DATA:
    data = int(input("data = "))
    L.append(data)
    num_data = num_data + 1

L_min = min(L)
L_max = max(L)
L_sum = sum(L)
L_len = len(L)

print("L (num_data = {}) = {}"\
      .format(L_len, L))
print("Min = {}, Max = {}, Avg = {}"\
      .format(L_min, L_max, L_sum/L_len))
```

```
Input 10 integer data.
data = 3
data = 7
data = 1
data = 0
data = 2
data = 8
data = 9
data = 5
data = 4
data = 6
L (num_data = 10) = [3, 7, 1, 0, 2, 8, 9, 5, 4, 6]
Min = 0, Max = 9, Avg = 4.5
```

숫자 자료형에 대한 비교 연산

◆ find min, max

```
# while-loop, find min and max of input data list
TARGET_NUM_DATA = 10
L = [] # empty list
L_len = 0
L_sum = 0
print("Input {} integer data.".format(TARGET_NUM_DATA))
while L_len < TARGET_NUM_DATA:
    data = int(input("data = "))
    L.append(data)
    L_sum = L_sum + data
    L_len = L_len + 1
    if L_len == 1:
        L_min = L_max = data
        continue
    if data > L_max:
        L_max = data
    if data < L_min:
        L_min = data

print("L (num_data = {}) = {}"\
      .format(L_len, L))
print("Min = {}, Max = {}, Avg = {}"\
      .format(L_min, L_max, L_sum/L_len))
```

```
Input 10 integer data.
data = 3
data = 7
data = 1
data = 0
data = 2
data = 8
data = 9
data = 5
data = 4
data = 6
L (num_data = 10) = [3, 7, 1, 0, 2, 8, 9, 5, 4, 6]
Min = 0, Max = 9, Avg = 4.5
```

첨가산술대입 연산 (augmented assignment)

◆ +=, -=, *=, /=, %=, **=, >>=, <<=, &=, ^=, !=

첨가 산술 대입 연산자	예	의미
+=	v1 += v2	v1 = v1 + v2
-=	v1 -= v2	v1 = v1 - v2
*=	v1 *= v2	v1 = v1 * v2
/=	a /= b	a = a / b
//=	a //= b	a = a // b
%=	n %= m	n = n % m
**=	x **= n	x = x ** n
<<=	b <<= n	b = b << n
>>=	b >>= n	b = b >> n
&=	a &= b	a = a & b
^=	a ^= b	a = a^b
|=	a |= b	a = a | b

```
>>> x = 1
>>> x += 10
>>> x
11
>>> x -= 5
>>> x
6
>>> x *= 10
>>> x
60
>>> x %= 7
>>> x
4
>>>
```

파이썬 시퀀스 (Sequence) 자료형 개요
- list, range

파이썬 자료형

◆ 파이썬 내장 자료형 (built-in data type)

자료형		구성 원소 변경 가능 여부 (mutable)	구성 원소의 열거 가능성(iterable)	사용 예
불리언 (boolean)	bool	불가능	불가능	True, False
숫자 (numeric)	int	불가능	불가능	정수형
	float	불가능	가능	실수형
	complex	불가능	가능	복소수
시퀀스 (sequence)	str	불가능	가능	"abcdefg"
	bytes	불가능	가능	b'\0x00\0x01\0x02\0x03'
	bytearray	가능	가능	b'\0x00\0x01\0x02\0x03'
	memoryview	가능/불가능	가능	
	list	가능	가능	[0, 1, 2, 3],
	tuple	불가능	가능	(1, 2, 1, 2, 3)
	range	불가능	가능	range(start, end[, step])
매핑 (mapping)	dict	가능	가능	{1:'A', 2:'B', 3:'C'}
집합 (set)	set	가능	가능	{1, 2, 3, 4, 5}
	frozenset	불가능	가능	{1, 2, 3, 4, 5}

시퀀스 자료형 관련 연산과 함수

◆ 시퀀스 자료형에 사용할 수 있는 연산과 함수

연산, 함수	기능	사용 예
len()	시퀀스 자료형 객체의 길이	len([0, 1, 2, 3, 4])
+	2개의 시퀀스를 연결 (concatenation)	[1, 2, 3, 4] + [5, 6, 7, 8, 9]
*	지정된 횟수만큼 시퀀스를 반복 (repetition)	['ABC'] * 3
in	시퀀스에 포함되어 있는지 확인	strList['Mon', 'Tue', Wed'] if 'Mon' in strList: print("Mon is included in strList")
not in	시퀀스에 포함되어 있지 않는지 확인	5 not in [0, 1, 2, 3, 4]
[ ]	인덱싱	strList[0]
min()	시퀀스에서 제일 작은 요소	min([3, 1, 5, 7])
max()	시퀀스에서 제일 큰 요소	max([3, 1, 5, 7])
sum()	시퀀스에 포함된 원소들의 합	sum([3, 1, 5, 7])
sorted()	시퀀스에 포함된 원소들을 정렬한 결과를 반환	L = [3, 1, 5, 7, 0, 4, 2, 6] L_sorted = sorted(L)
for x in 시퀀스 자료형:	시퀀스 자료형의 원소를 차례로 열거하며 for 반복문 실행	for n in [3, 1, 5, 7, 0, 4, 2, 6] : print(n)

파이썬 리스트 (list) 자료형

◆ 리스트 (list) 기본 함수

구분	list 자료형의 기본 함수	설명
list 생성	L = [] L = list()	포함된 항목이 없는 빈 (empty) 리스트 자료형 객체 생성
	L = [1, 2, 3, 4]	콤마로 나누어진 원소들을 차례대로 포함하는 리스트 객체 생성
	L = list(range(10))	range() 함수가 생성하는 정수 데이터를 원소로 하는 리스트 객체 생성
list 기본 연산	len(L)	리스트의 원소 개수 (길이)를 반환
	L.append(7)	리스트 L의 맨 뒤에 7을 새로운 원소로 추가
	L.expand([5, 6, 7])	리스트 L 뒤에 인수로 주어진 리스트 [5, 6, 7]의 원소들을 차례로 추가
인덱싱	L[i]	리스트에 포함된 원소를 인덱스 i로 접근하며 그 원소 값을 읽거나 변경
	L[i][j]	2차원 리스트에 포함된 원소를 인덱스 i, j로 접근하며 그 원소 값을 읽거나 변경
	del L[i]	리스트 L의 i 번째 원소를 삭제
슬라이싱	L[i:j] = [4, 5, 6, 7]	리스트 L의 i ~ j-1번째 원소를 대입식 오른쪽에서 제공하는 리스트의 원소들로 변경
	L2 = L1[i:j]	리스트 L1 i ~ j-1번째 원소를 읽어 새로운 리스트 L2를 생성
	L[i:j] = []	리스트 L1 i ~ j-1번째 원소를 삭제
	del L[i:j]	리스트 L1 i ~ j-1번째 원소를 삭제

시퀀스 자료형에 대한 기본 연산

◆ 시퀀스 자료형 list에 대한 기본 연산

```
# Application of sequence type

L = [4, 5, 6, 1, 3, 8, 9, 2, 0, 7]
sum_L = sum(L)
max_L = max(L)
min_L = min(L)
len_L = len(L)
avg_L = sum_L / len_L
print("L (size: {}) : {}".format(len_L, L))
print("Statistics of L : Max ({}), Min ({}), Avg({:7.2f})"\
      .format(max_L, min_L, avg_L))

L_sorted = sorted(L)
print("Sorted L : ", L_sorted)
print("Original L : ", L)
```

```
L (size: 10) : [4, 5, 6, 1, 3, 8, 9, 2, 0, 7]
Statistics of L : Max (9), Min (0), Avg(   4.50)
Sorted L :  [0, 1, 2, 3, 4, 5, 6, 7, 8, 9]
Original L :  [4, 5, 6, 1, 3, 8, 9, 2, 0, 7]
```

시퀀스 자료형의 접합 (concatenation), 반복 (repetition)

◆ 시퀀스 자료형의 concatenation, repetition

```
# Applications of sequence types - concatenation, repeat

Str_A = "ABCD_" # str
Str_B = "1234"
print("Str_A = ", Str_A)
print("Str_B = ", Str_B)
print("Str_A + Str_B = ", Str_A + Str_B)
print("Str_A * 3 = ", Str_A * 3)

L_A = [1, 3, 5, 7, 9] # list
L_B = [0, 2, 4, 6, 8]
print("L_A = ", L_A)
print("L_B = ", L_B)
print("L_A + L_B = ", L_A + L_B)
print("L_A * 3 = ", L_A * 3)

T_A = (1, 3, 5, 7, 9) # tuple
T_B = (0, 2, 4, 6, 8)
print("T_A = ", T_A)
print("T_B = ", T_B)
print("T_A + T_B = ", T_A + T_B)
print("T_A * 3 = ", T_A * 3)
```

```
Str_A =  ABCD_
Str_B =  1234
Str_A + Str_B =  ABCD_1234
Str_A * 3 =  ABCD_ABCD_ABCD_
L_A =  [1, 3, 5, 7, 9]
L_B =  [0, 2, 4, 6, 8]
L_A + L_B =  [1, 3, 5, 7, 9, 0, 2, 4, 6, 8]
L_A * 3 =  [1, 3, 5, 7, 9, 1, 3, 5, 7, 9, 1, 3, 5, 7, 9]
T_A =  (1, 3, 5, 7, 9)
T_B =  (0, 2, 4, 6, 8)
T_A + T_B =  (1, 3, 5, 7, 9, 0, 2, 4, 6, 8)
T_A * 3 =  (1, 3, 5, 7, 9, 1, 3, 5, 7, 9, 1, 3, 5, 7, 9)
```

시퀀스 자료형의 in, not in

◆ Sequence data type with in, not in

```
# sequence type - in, not in

L = ["black", "white", "yellow", "red", "green",\
     "blue", "orange", "purple"]
Color_to_Search = ["red", "blue", "cyan", "violet"]

print("L = ", L)
print("Color_to_search = ", Color_to_Search)

for color in Color_to_Search:
    if color in L:
        print("{} is in L".format(color))
    elif color not in L:
        print("{} is not in L".format(color))
    else:
        print("{} is not determined in L"\
              .format(color))
```

```
L =  ['black', 'white', 'yellow', 'red', 'green', 'blue', 'orange', 'purple']
Color_to_search =  ['red', 'blue', 'cyan', 'violet']
red is in L
blue is in L
cyan is not in L
violet is not in L
```

리스트 (List) 생성

◆ class list([iterable])

```
# class list

A = []
print("A = ", A)
print("type(A) = ", type(A))
A = [1, 2, 3]
print("A = ", A)

B = [x for x in range(5)]
print("B = ", B)

C = [x for x in 'abcd']
print("C = ", C)

D = [(x, x+2) for x in range(5)]
print("D = ", D)
```

```
A =  []
type(A) =  <class 'list'>
A =  [1, 2, 3]
B =  [0, 1, 2, 3, 4]
C =  ['a', 'b', 'c', 'd']
D =  [(0, 2), (1, 3), (2, 4), (3, 5), (4, 6)]
```

2차원 리스트 생성

◆ 2차원 리스트

```
# 2-D list
ROWS = 2
COLS = 3
A = [[i*3+j for j in range(COLS)] for i in range(ROWS)] # 2D array, matrix
print("A = ", A)
print("A = ")
for i in range(ROWS):
    for j in range(COLS):
        print(A[i][j], end = " ")
    print()

B = list() # B = []
print("B = ", B)
B = list('abc')
print("B = ", B)

C = list(range(5))
print("C = ", C)

D = list((1, 2, 3))
print("D = ", D)
```

```
A =  [[0, 1, 2], [3, 4, 5]]
A =
0 1 2
3 4 5
B =  []
B =  ['a', 'b', 'c']
C =  [0, 1, 2, 3, 4]
D =  [1, 2, 3]
```

range()

◆ range(start, stop, stride)

range() 사용 예	설 명
range(10)	0부터 9까지 1씩 증가하며 정수를 생성
range(1, 10)	1부터 9까지 1씩 증가하며 정수를 생성
range(1, 10, 2)	1부터 9까지 2씩 증가하며 정수를 생성
range(10, 100, 5)	10부터 99까지 5씩 증가하며 정수를 생성
range(100, 0, -1)	100부터 1까지 1씩 감소하며 정수를 생성
range(**0.0**, 10.0, **0.1**)	range()에는 정수만 사용 가능 NumPy 패키지의 **arange()** 기능에서는 실수 (float)형 지원

Creation of List with range(start, stop, step)

◆ Creation of List with range()

```
# sequency type - list and range()

print("range(10)         : ", end='')
for i in range(10):
    print("{:2}".format(i), end=' ')
print()

print("range(0, 10)      : ", end='')
for i in range(0, 10):
    print("{:2}".format(i), end=' ')
print()

print("range(0, 10, 2)   : ", end='')
for i in range(0, 10, 2):
    print("{:2}".format(i), end=' ')
print()

print("range(1, 10, 2)   : ", end='')
for i in range(1, 10, 2):
    print("{:2}".format(i), end=' ')
print()

print("range(10, -1, -1): ", end='')
for i in range(10, -1, -1):
    print("{:2}".format(i), end=' ')
print()
```

```
range(10)         :  0  1  2  3  4  5  6  7  8  9
range(0, 10)      :  0  1  2  3  4  5  6  7  8  9
range(0, 10, 2)   :  0  2  4  6  8
range(1, 10, 2)   :  1  3  5  7  9
range(10, -1, -1): 10  9  8  7  6  5  4  3  2  1  0
```

range(start, stop, stride) and List

◆ range() and list

```
# list and range
L = range(11, 20)
print("L = ", L)
print("len(L) = ", len(L))
print("min(L) = ", min(L))
print("max(L) = ", max(L))
print("11 in L = ", 11 in L)
print("20 in L = ", 20 in L)
print("L.start = ", L.start)
print("L.stop = ", L.stop)
print("L.step = ", L.step)
print("L.count(11) = ", L.count(11))
print("L.index(11) = ", L.index(11))
print("L[0] = ", L[0])
print("L[-1] = ", L[-1])
```

```
L =  range(11, 20)
len(L) =  9
min(L) =  11
max(L) =  19
11 in L =  True
20 in L =  False
L.start =  11
L.stop =  20
L.step =  1
L.count(11) =  1
L.index(11) =  0
L[0] =  11
L[-1] =  19
```

다양한 자료형의 원소를 포함하는 리스트

◆ list with different types, different length

```
# List with heterogeneous elements

A = [0, 1.0, '1', b'bcd']
print("A = ", A)

B = [[1, 2], [3, 4, 5], [6, 7, 8, 9, 10]]
print("B = ", B)

C = [1, 2, 3]
print("C = ", C)

D = [1, 2, 3]
print("D = ", D)
print("C == D : ", C==D)

C[0] = 10
print("C = ", C)
print("C == D : ", C==D)

E = D
print("E = ", E)
print("E is D : ", E is D)

print("id(D), id(E) = ", id(D), id(E))
```

```
A =  [0, 1.0, '1', b'bcd']
B =  [[1, 2], [3, 4, 5], [6, 7, 8, 9, 10]]
C =  [1, 2, 3]
D =  [1, 2, 3]
C == D :  True
C =  [10, 2, 3]
C == D :  False
E =  [1, 2, 3]
E is D :  True
id(D), id(E) =  19345936 19345936
```

리스트 연산자 – in, not in, +, 인덱싱, 슬라이싱

◆ sequence-type operations for list

```
# List with heterogeneous elements

A = [10, 20, 30]
print("A = ", A)
print("10 in A : ", 10 in A)
print("10 not in A : ", 10 not in A)

B = [1, 2, 3] * 3
print("B = ", B)

C = [1, 2, 3] + [4, 5, 6]
print("C = ", C)

print("C[0] = {}, C[-1] = {}"\
      .format(C[0], C[-1])) # indexing
print("C[1:4] = ", C[1:4]) # slicing
print("C[::2] = ", C[::2])
print("len(C) = {}, min(C) = {}, max(C) = {}"\
      .format(len(C), min(C), max(C)))
print("C.index(4) = ", C.index(4))
print("C.count(5) = ", C.count(5))
```

```
A =  [10, 20, 30]
10 in A :  True
10 not in A :  False
B =  [1, 2, 3, 1, 2, 3, 1, 2, 3]
C =  [1, 2, 3, 4, 5, 6]
C[0] = 1, C[-1] = 6
C[1:4] =  [2, 3, 4]
C[::2] =  [1, 3, 5]
len(C) = 6, min(C) = 1, max(C) = 6
C.index(4) =  3
C.count(5) =  1
```

리스트 슬라이싱, append(), clear()

◆ sequence-type operations for list

```
# List with heterogeneous elements

A = [1, 2, 3, 4, 5, 6]
print("A = ", A)
A[0] = 10 # indexing
print("A = ", A)

A[1:4] = [15, 25, 35] # slicing
print("A = ", A)
A[1:4] = [] # delete, del A[1:4]
print("A = ", A)

A.append(7)
print("A = ", A)

A.append([8, 9, 10])
print("A = ", A)

A.clear()
print("A = ", A)
```

```
A =  [1, 2, 3, 4, 5, 6]
A =  [10, 2, 3, 4, 5, 6]
A =  [10, 15, 25, 35, 5, 6]
A =  [10, 5, 6]
A =  [10, 5, 6, 7]
A =  [10, 5, 6, 7, [8, 9, 10]]
A =  []
```

리스트 복사 - copy

```
# Testing copy of List
import copy

A = [0, [1, 2]]
print("A (id = {}) = {}".format(id(A), A))

B = A
print("B = ", B)
B[0] = 100
print("B (id = {}) = {}".format(id(B), B))
print("A (id = {}) = {}".format(id(A), A))

C = copy.copy(A) # shallow copy on mutable item
print("\nC = copy.copy(A) = {} (id = {}) "\
   .format(C, id(C)))
print("C[0] is A[0] : ", C[0] is A[0])
C[0] = 300
print("After C[0] = 300 :")
print("C (id = {}) = {}".format(id(C), C))
print("A (id = {}) = {}".format(id(A), A))
print("C[1] is A[1] : ", C[1] is A[1])
C[1][0] = 5
print("After C[1][0] = 5")
print("C = ", C)
print("A = ", A)

D = copy.deepcopy(A) # deep copy
print("\nD = copy.deepcopy(A) = {} (id = {})".format(D, id(D)))
print("D[1] is A[1] : ", D[1] is A[1])
A[1][0] = 500
print("A (id = {}) = {}".format(id(A), A))
print("D (id = {}) = {}".format(id(D), D))
```

```
A (id = 66223560) = [0, [1, 2]]
B =  [0, [1, 2]]
B (id = 66223560) = [100, [1, 2]]
A (id = 66223560) = [100, [1, 2]]

C = copy.copy(A) = [100, [1, 2]] (id = 59714184)
C[0] is A[0] :  True
After C[0] = 300 :
C (id = 59714184) = [300, [1, 2]]
A (id = 66223560) = [100, [1, 2]]
C[1] is A[1] :  True
After C[1][0] = 5
C =  [300, [5, 2]]
A =  [100, [5, 2]]

D = copy.deepcopy(A) = [100, [5, 2]] (id = 66223112)
D[1] is A[1] :  False
A (id = 66223560) = [100, [500, 2]]
D (id = 66223112) = [100, [5, 2]]
```

리스트 – append(), extend()

◆ append() vs extend()

- L1, L2: list
- L1.append(L2) inserts the whole L2 as an element in L1
- L1.extend(L2) inserts each element in L2 as elements in L1

```
# List with append() and extend()

A = [0, 1, 2, 3, 4]
print("A = ", A)
B = [5, 6, 7, 8, 9]
print("B = ", B)
C = [10, 11, 12, 13, 14]
print("C = ", C)

A.append(C)
print("A.append(C) = ", A)
B.extend(C)
print("B.extend(C) = ", B)
```

```
A =  [0, 1, 2, 3, 4]
B =  [5, 6, 7, 8, 9]
C =  [10, 11, 12, 13, 14]
A.append(C) =  [0, 1, 2, 3, 4, [10, 11, 12, 13, 14]]
B.extend(C) =  [5, 6, 7, 8, 9, 10, 11, 12, 13, 14]
```

리스트 연산 - insert(), pop(), remove(), reverse()

◆ insert(), pop(), remove(), reverse()

```
# List with insert(), pop(), remove(), reverse()

A = [0, 1, 2]
print("A = ", A)
A.extend([3, 4, 5])
print("After A.extend([3, 4, 5]) : A = ", A)
A.insert(1, 10)
print("After A.insert(1, 10): A = ", A)
print("A.pop() : ", A.pop()) # return the last item and delete it
print("A = ", A)
A.pop(1)
print("After A.pop(1) : A = ", A)

A.remove(3)
print("After A.remove(3): A = ", A)
A.reverse()
print("After A.reverse() : A = ", A)
```

```
A =  [0, 1, 2]
After A.extend([3, 4, 5]) : A =  [0, 1, 2, 3, 4, 5]
After A.insert(1, 10): A =  [0, 10, 1, 2, 3, 4, 5]
A.pop() :  5
A =  [0, 10, 1, 2, 3, 4]
After A.pop(1) : A =  [0, 1, 2, 3, 4]
After A.remove(3): A =  [0, 1, 2, 4]
After A.reverse() : A =  [4, 2, 1, 0]
```

리스트와 all() 및 any()

◆ all(), any()

함수	의미
all(iterable)	반복 가능한 자료형 (iterable)의 모든 항목 (item) 들이 True이면 함수의 결과는 True; 어떤 항목 하나라도 False이면 결과는 False
any(iterable)	반복 가능한 자료형 (iterable)의 어떤 항목 (item) 이 True이면 함수의 결과는 True; 모든 항목이 False이면 결과는 False

```
# Application of sequence type - all, any

L_A = [True, True, False, True] # list of boolean
L_B = [False, False, True, True]
print("L_A = ", L_A)
print("L_B = ", L_B)

print("any(L_A) = ", any(L_A))
print("all(L_B) = ", all(L_B))

L_C = []
for i in range(len(L_A)):
    L_C.append(L_A[i] and L_B[i])
print("L_A and L_B = ", L_C)

L_D = []
for i in range(len(L_A)):
    L_D.append(L_A[i] or L_B[i])
print("L_A or L_B = ", L_D)
```

```
L_A =  [True, True, False, True]
L_B =  [False, False, True, True]
any(L_A) =  True
all(L_B) =  False
L_A and L_B =  [False, False, False, True]
L_A or L_B =  [True, True, True, True]
```

list와 sorted()

◆ sorted()

```
# list and sorted()

A = [2, 3, 5, 7, 1, 4]
print("A = ", A)
A_sorted = sorted(A)
print("A_sorted = ", A_sorted)

Colors = ['blue', 'green', 'orange', 'red', 'yellow', 'purple']
Colors_sorted = sorted(Colors)
print("Colors = ", Colors)
print("Colors_sorted = ", Colors_sorted)
```

```
A =  [2, 3, 5, 7, 1, 4]
A_sorted =  [1, 2, 3, 4, 5, 7]
Colors =  ['blue', 'green', 'orange', 'red', 'yellow', 'purple']
Colors_sorted =  ['blue', 'green', 'orange', 'purple', 'red', 'yellow']
```

파이썬의 기본 연산자와 기본 명령어

파이썬 기본 연산자

◆ 파이썬 기본 연산자

자료형	구분	사용가능 연산자
불리언	불리언 연산	and, or, not
숫자형	산술 계산	+ (덧셈), - (뺄셈), *(곱셈), / (실수 나눗셈), // (정수 나눗셈), % (모듈로)
	비교 연산	>, >=, <, <=, ==
	비트 단위 연산	& (bit-wise and), | (bit-wise or), ^(bit-wise exclusive or), ~ (bit-wise not), << (bit-wise left shift), >> (bit-wise right shift)
시퀀스 (str, bytes, bytearray, memoryview, list, tuple)	접합, 반복	+ : 두 개가 시퀀스 객체를 순차적으로 접합 (concatenation) * : 시퀀스 객체를 지정된 회수만큼 반복
	인덱싱	[i] : 시퀀스 자료형 객체의 i번째 항목
	슬라이싱	[i:j:k] : 시퀀스 자료형 객체의 i번째 항목부터 k씩 인덱스를 증가하면서 j-1번째 까지의 항목
매핑(dict)	인덱싱	[keyword] : dict 자료형 객체에서 키워드 (keyword)로 지정된 항목
집합(set)	집합 관계	<= (부분집합), < (진부분 집합), > (진부분 집합), >= (부분집합), | (합집합), & (교집합), - (차집합), ^ (대칭 차집합)

List, Tuple, Set, Dict

◆ Comparison of list, tuple, set, dict

type		mutable	example, methods, remarks
sequence	list	no	L = [0, 1, 2, 3] L = [0, 1.0, 'a', b'xyz'] len(), min(), max(), L.index(), L.count() list with different types and different length are possible
	tuple	no	T = (1, 2, 1, 2, 3) in, not-in duplication is possible,
set	set	yes	S = {1, 2, 3} S1 |= S2, S1 &= S2, S1 -= S2, S1 ^= S2 unordered, no duplication, collection indexing and slicing are not used
	frozenset	no	immutable collection

mapping	dict	yes	D = {1:'A', 2:'B', 2:'X', 3:'C'} //key:value
			D.items(), D.keys(), D.values(), D.update(key=value)

파이썬 연산자 우선 순위 (1)

◆ 파이썬 연산자 우선 순위 (precedence)

precedence	operator	Meanings
highest	(expressions ...), [expressions...], {key:value...}, {expressions ...}	parenthesis (binding, tuple), list, dictionary, set
	x[index], x[index:index], x(arguments...), x.attribute	indexing, slicing function call, referring attribute
	**	power (exponentiation)
	+x, -x, ~x	unary operator, negation,
	*, @, /, //, %	multiplication, division, remainder
	+, -	binary addition, subtraction
	<<, >>	shift bits left, right
	&	bit-wise AND
	^	bit-wise XOR
	|	bit-wise OR

파이썬 연산자 우선 순위 (2)

◆ 파이썬 연산자 우선 순위 (precedence)

precedence	operator	|Meanings
	in, not in, is, is not, <, <=, >, >=, !=, ==	comparison, membership, identity
	not x	logical NOT
	and	logical AND
	or	logical OR
	if – else	conditional expression
lowest	lambda	lambda expression

파이썬 기본 명령어 (1)

◆ 파이썬 기본 명령어

기본 명령어	구분	상세설명, 예
대입 (assignment), =	creating references	a, b = 'good', 'bad' a, b = b, a # swap a, b
func(args)	invoke function with arguments	log.write("spam, ham")
print()	printing objects	print('Welcome to Python")
if, elif, else	selecting actions	조건문
for	iteration	for-반복문
while	general loops	while-반복문
pass	entry placeholder	블록의 항목
break	loop exit	반복문을 탈출
continue	loop continue	반복문의 나머지 구간을 생략
def	functions and methods	함수의 정의
return	functions results	함수의 결과값 반환
yield	generator functions	제네레이터 함수의 값 전달
global	namespaces	전역 네임스페이스 지정
nonlocal	namespaces	지역 네임스페이스에 해당되지 않음을 지정

파이썬 기본 명령어 (1)

◆ 파이썬 기본 명령어

기본 명령어	구분	상세설명, 예
import	module access	모듈을 추가 import turtle
with / as	context managers	모듈의 이름 간략화 import numpy as np
from	attribute access	모듈에서의 특정 함수/속성 추가 from tkinter import *
class	building objects	클래스
try / except / else/ finally	catching exceptions	예외상황 처리
raise	triggering exceptions	예외상황 발생
assert	debugging checks	실행 조건 지정 및 검사
del	deleting references	지정된 객체를 삭제

한 줄에 여러 정수 데이터 입력 - split(), map()

◆ split()과 map()을 사용하여 한 줄에 여러 정수 데이터 입력 방법

```
# input multiple numbers in one line using split() and map()

input_data_str = input("input multiple data (a b c) (separated in space) = ")
decimal_strings = input_data_str.split(sep= ' ')
print("decimal_strings = ", decimal_strings)
a, b, c = map(int, decimal_strings)
print("Input a = {}, b = {}, c = {}".format(a, b, c))
```

```
input multiple data (a b c) (separated in space) = 10 20 30
decimal_strings =  ['10', '20', '30']
Input a = 10, b = 20, c = 30
```

한 줄에 입력된 정수 데이터들을 사용한 list 생성

◆ split()과 map()을 사용하여 한 줄에 여러 정수 데이터 입력 방법

```
# input multiple numbers from one line using split() and map()

input_data_str = input("input data : ")
decimal_strings = input_data_str.split(sep=' ')
print("Input decimal_strings : ", decimal_strings)
L = list(map(int, decimal_strings))
print("Input integers : ", L)
```

```
input data : 1 2 3 4 5 6 7 8 9 10
Input decimal_strings :  ['1', '2', '3', '4', '5', '6', '7', '8', '9', '10']
Input integers :  [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

```
input data : 1 1234 6543 9876
Input decimal_strings :  ['1', '1234', '6543', '9876']
Input integers :  [1, 1234, 6543, 9876]
```

한 줄에 여러 실수 (float) 데이터 입력 - split(), map()

```
# Getting multiple float numbers in one line using split() and map()

input_data_str = input("input float data : ")
float_strings = input_data_str.split(sep=' ')
print("Input decimal_strings : ", float_strings)
L = list(map(float, float_strings))
print("Input float data : ", L)
```

```
input float data : 1.23 4.56 7.78 10.12345
Input decimal_strings :  ['1.23', '4.56', '7.78', '10.12345']
Input float data :  [1.23, 4.56, 7.78, 10.12345]
```

문법적 오류와 논리적 오류

문법적 오류와 오류메시지

◆ 문법적 오류 (Syntax Error)

```
# python program - syntax error & debugging

a = 10
c = a + b
```

```
Traceback (most recent call last):
  File "C:\MyPyPackage\TextBook - 2019\ch 2
 in <module>
    c = a + b
NameError: name 'b' is not defined
```

```
# python program - syntax error & debugging

a = input("input integer data = ")
b = 20
c = a + b
print("a = ", a)
print("b = ", b)
print("a + b = ", c)
```

```
input integer data = 10
Traceback (most recent call last):
  File "C:\MyPyPackage\TextBook - 2019\ch 2 Python Overvi
 function.py", line 5, in <module>
    c = a + b
TypeError: can only concatenate str (not "int") to str
```

논리적 오류 (logical error)

◆ 알고리즘이 실행 순서상에 논리적인 오류가 있는 경우

/* 올바른 제빵 알고리즘 */

1: 재료 준비: 밀가루, 우유, 계란
2: 그릇 준비: 반죽용 그릇
3: **밀가루, 우유, 계란 반죽하여 그릇에 담기**
4: **그릇을 오븐에 넣기**
5: **오븐을 1시간 동안 가열**
6: 오븐을 끄기
7: 그릇을 꺼내기

(a) 올바른 제빵 알고리즘

/* 논리적 오류가 포함된 제빵 알고리즘 */

1: 재료 준비: 밀가루, 우유, 계란
2: 그릇 준비: 반죽용 그릇
3: **그릇을 오븐에 넣기**
4: **오븐을 1시간 동안 가열**
5: 오븐을 끄기
6: 그릇을 꺼내기
7: **밀가루, 우유, 계란 반죽하여 그릇에 담기**

(b) 논리적 오류가 있는 제빵 알고리즘

논리적 오류의 디버깅 – 중간 단계의 결과 출력

◆ 논리적 오류의 예

```
# python program - logical error in while-loop & debugging

count = 0
total_sum = 0
while count < 10:
    total_sum = total_sum + count
    print("At count {:3}, total_sum = {:3}".format(count, total_sum))
```

```
At count   0, total_sum =   0
At count   0, total_sum =   0
At count   0, total_sum =   0
At count   0, total_sum =   0
At count   0, total_sum =   0
At count   0, total_sum =   0
At count   0, total_sum =   0
At count   0, total_sum =   0
At count   0, total_sum =   0
At count   0, total_sum =   0
At count   0, total_sum =   0
```

파이썬 키워드를 임의로 재지정한 경우

◆ 키워드의 잘못된 재지정

```
# Python program - debugging

n_str = "100"
print("n_str= {}, type(n_str) = {}".format(n_str, type(n_str)))
n = int(n_str)
print("n = {}, type(n) = {}".format(n, type(n)))

int = 123
m_str = "200"
m = int(m_str)
print("m = {}, type(m) = {}".format(m, type(m)))
```

```
n_str= 100, type(n_str) = <class 'str'>
n = 100, type(n) = <class 'int'>
Traceback (most recent call last):
  File "C:\MyPyPackage\TextBook - 2019\ch 2
, line 10, in <module>
    m = int(m_str)
TypeError: 'int' object is not callable
```

효과적인 프로그램 개발 방법 Tip 1

◆ 프로그램 설계 (알고리즘 및 자료구조 설계)에 더 많은 시간을 투자할 것

- 주어진 문제를 해결하기 위한 효율적인 알고리즘을 먼저 구성 할 것
- 구성된 알고리즘에 적합한 자료구조를 선택할 것
- 예) 100,000개의 학생 데이터로 부터 0.01초의 데이터 처리 시간 내에 특정 조건을 만족하는 학생을 탐색 (search)하여야 하는 경우, 어떤 자료구조를 사용하며, 어떤 방식으로 탐색할 것인가 ?
  - array vs. linked list vs. binary tree
  - sequential search vs. binary search

효과적인 프로그램 개발 방법 Tip 2

◆ 프로그램 오류 수정을 위한 debugging 방법을 먼저 숙달 할 것

- 프로그램 소스코드 상에 존재하는 문법적 에러 및 논리적인 에러를 빨리 찾아내는 방법을 먼저 숙달
- VS Code의 debugging 기능을 먼저 확인하고, 숙달할 것
  - break point 기능
  - trace 기능: step-over (F10), step-into(F11)
  - 각 단계에서의 local variable 값 확인

Homework 2

Homework 2

2.1 0 ~ 255의 값을 10진수, 2진수, 8진수, 16진수로 각각 출력하는 파이썬 프로그램을 작성하고, 실행결과를 제출하라. 2진수는 총 8자리, 8진수는 접두어를 포함하여 5자리, 16진수는 접두어를 포함하여 4자리로 출력하며, 앞부분에 빈자리가 있는 경우 0으로 채울 것.
(실행 예제)

```
===================================
Decimal   Bit  Octal  Hexadecimal
-----------------------------------
      0 00000000 0o000 0X00
      1 00000001 0o001 0X01
      2 00000010 0o002 0X02
      3 00000011 0o003 0X03
      4 00000100 0o004 0X04
      5 00000101 0o005 0X05
      6 00000110 0o006 0X06
      7 00000111 0o007 0X07
      8 00001000 0o010 0X08
      9 00001001 0o011 0X09
     10 00001010 0o012 0X0A
     11 00001011 0o013 0X0B
     12 00001100 0o014 0X0C
     13 00001101 0o015 0X0D
     14 00001110 0o016 0X0E
     15 00001111 0o017 0X0F
     16 00010000 0o020 0X10
```

```
    240 11110000 0o360 0XF0
    241 11110001 0o361 0XF1
    242 11110010 0o362 0XF2
    243 11110011 0o363 0XF3
    244 11110100 0o364 0XF4
    245 11110101 0o365 0XF5
    246 11110110 0o366 0XF6
    247 11110111 0o367 0XF7
    248 11111000 0o370 0XF8
    249 11111001 0o371 0XF9
    250 11111010 0o372 0XFA
    251 11111011 0o373 0XFB
    252 11111100 0o374 0XFC
    253 11111101 0o375 0XFD
    254 11111110 0o376 0XFE
    255 11111111 0o377 0XFF
===================================
```

Homework 2

2.2 10개의 실수 (float) 데이터를 한 줄에 입력 받아 리스트에 저장하고, 이 리스트에 포함된 데이터 중 최솟값, 최댓값, 평균값을 계산하여 출력하라. 평균값은 소수점 이하 2자리까지 출력할 것.
(실행 예제)

```
input 10 float data in one line (separated in space) = 1.1 2.2 3.3 4.4 5.5 6.6 7.7 8.8 9.9 10.1
float_strings =  ['1.1', '2.2', '3.3', '4.4', '5.5', '6.6', '7.7', '8.8', '9.9', '10.1']
L_float_data =  [1.1, 2.2, 3.3, 4.4, 5.5, 6.6, 7.7, 8.8, 9.9, 10.1]
L_max = (10.1), L_min = (1.1), L_avg = (5.960)
```

Homework 2

2.3 3개의 16진수 x, y, z를 각각 입력 받고, 이 16진수 x와 y의 bit-wise and, bit-wise or, bit-wise exclusive or 값을 각각 계산하여 2진수 및 16진수로 출력하며, x의 bit-wise not과 bit-wise left shift 2, y와 z의 bitwise right shift 2를 각각 출력하는 파이썬 프로그램을 작성하고, 실행 결과를 제출하라. 각 출력 항목들은 오른쪽으로 줄 맞춤 할 것.
(실행 예제)

```
hexadecimal x = 0x07
hexadecimal y = 0xA3
hexadecimal z = -0x07
 x =    7 in decimal,              0b111 in bits
 y =  163 in decimal,         0b10100011 in bits
 z =   -7 in decimal,             -0b111 in bits
 x & y  = in hex(     0x3), in bin(              0b11)
 x | y  = in hex(    0xa7), in bin(        0b10100111)
 x ^ y  = in hex(    0xa4), in bin(        0b10100100)
 ~x     = in hex(    -0x8), in bin(           -0b1000)
 x << 2 = in hex(    0x1c), in bin(           0b11100)
 y >> 2 = in hex(    0x28), in bin(          0b101000)
 z >> 2 = in hex(    -0x2), in bin(             -0b10)
```

Homework 2

2.4 복소수 (complex number) c1을 입력 받고, 이 c1의 켤레 복소수 c2를 계산하여 출력하라. c1과 c2의 덧셈, 뺄셈, 곱셈, 나눗셈을 계산하여 결과를 출력하는 파이썬 프로그램을 작성하고, 실행 결과를 제출하라.
(실행 예제)

```
input c1 (in complex, re + imj): (3+4j)
c1 =  (3+4j)
c2 = conjugate of c1 =  (3-4j)
c1 + c2 = (6+0j)
c1 - c2 = 8j
c1 * c2 = (25+0j)
c1 / c2 = (-0.28+0.96j)
```

Homework 2

2.5 다각형 꼭지점 개수 n과 한 변의 길이 (length), 그 다각형의 중심 좌표 x0, y0를 한 줄로 입력 받고, 지정된 위치 (x0, y0)를 중심으로 다각형을 그리는 파이썬 프로그램을 작성하라. 다각형의 좌측 하단 꼭지점의 좌표를 정확하게 계산하여 출력하며, 터틀 그래픽의 중앙과 다각형의 각 꼭지점 좌표를 터틀 객체의 write() 함수를 사용하여 출력할 것.
(실행 예제)

```
Input number of vertices polygon, side_length, center position x0 and y0 in one
line (e.g., 7 100 100 100) = 7 100 100 100
Drawing a polygon with 7 vertices and length (100) at (100, 100) ...
```