기초 컴퓨터 프로그래밍

- Python 기본: 자료형 및 연산자 -

충남대학교 의과대학 의공학교실

구 윤 서 교수

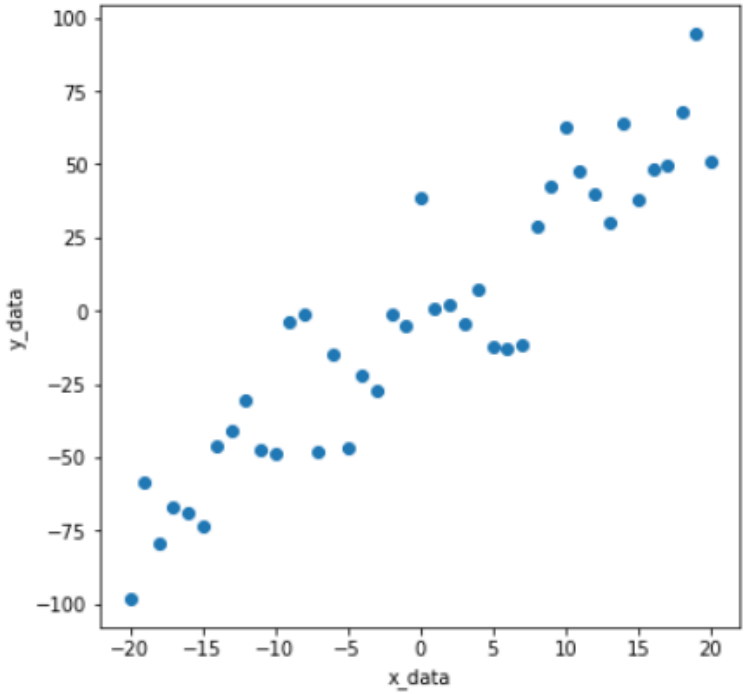
This Class

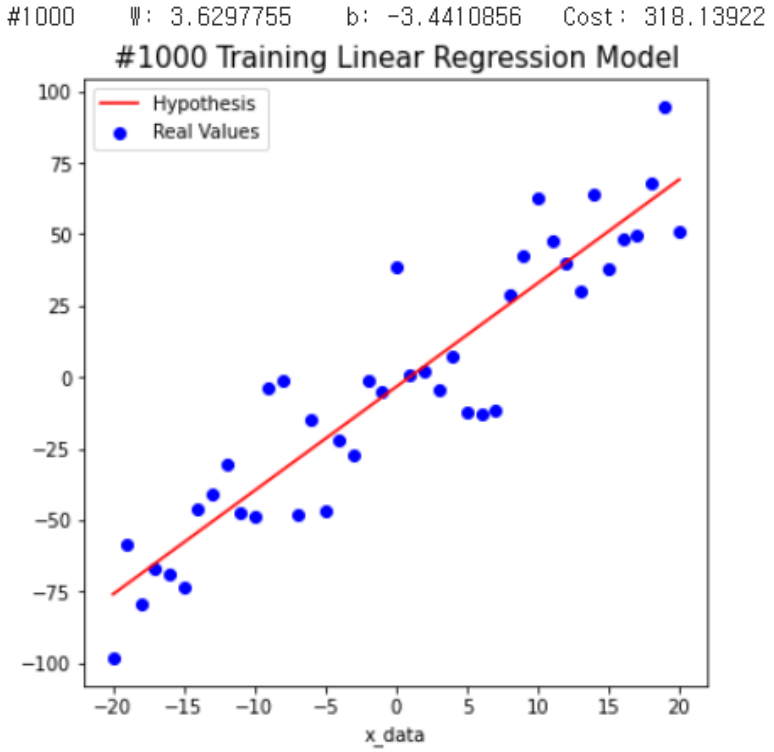
- Python 기본
  - 변수 (Variable) & 메모리 (Memory)
  - 자료형 (Data Type)
  - 연산자 (Operator)

A python code example

Let's see a simple linear regression model in TensorFlow 2.0!

$f(x) = 3x - 3$ + Random noise





A python code example

```python
# Import modules
import tensorflow as tf
import numpy as np
import seaborn as sns
import matplotlib.pyplot as plt

# X data
x_data = list(range(-20,21))

# Y data
np.random.seed(2020)

mu = 0
sigma = 20
n = len(x_data)

noises = np.random.normal(mu, sigma, n)

W_answer = 3
b_answer = -3

y_temp = list(np.array(x_data)*W_answer + b_answer)
y_data = list(np.array(y_temp) + np.array(noises))
```

모듈

변수

함수

값

연산자

A python code example

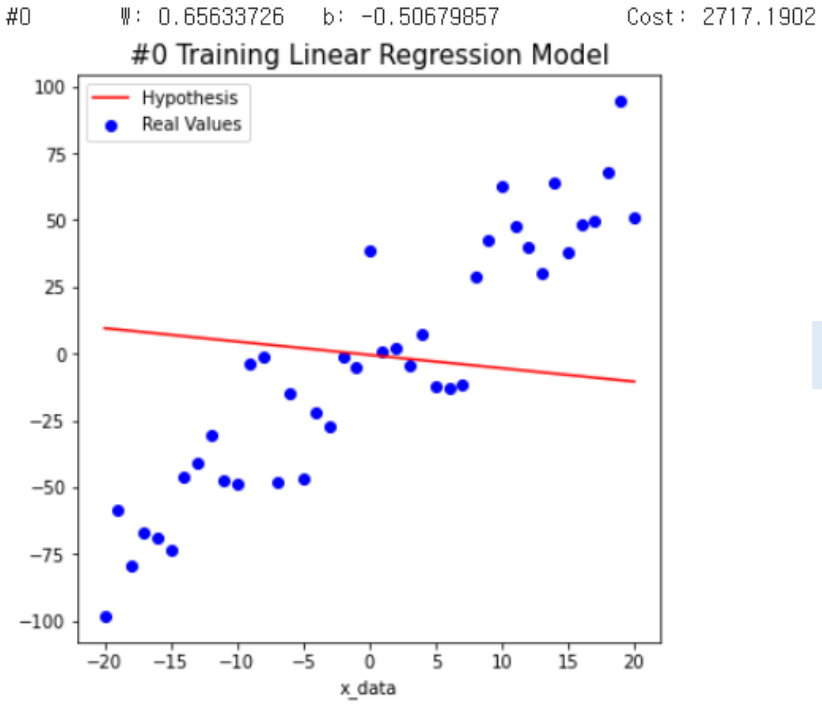
제어문 (for문)

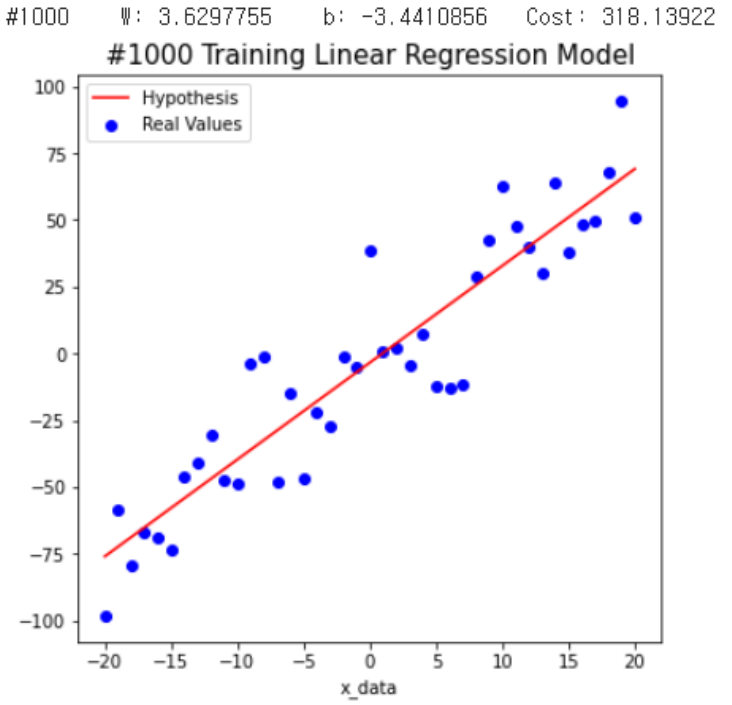
제어문 (if문)

```
24
25 # Model Initialization
26 W = tf.Variable(-0.5)
27 b = tf.Variable(-0.5)
28 learning_rate = 0.001
29
30 # Regression Training
31 cost_list=[]
32
33 # Train
34 for i in range(1000+1):
35
36     # Gradient Descent
37     with tf.GradientTape() as tape:
38
39         hypothesis = W * x_data + b
40         cost = tf.reduce_mean(tf.square(hypothesis - y_data))
41
42         # Update
43         W_grad, b_grad = tape.gradient(cost, [W, b])
44         W.assign_sub(learning_rate * W_grad)
45         b.assign_sub(learning_rate * b_grad)
46         cost_list.append(cost.numpy())
47
48     # Output
49     if i % 200 == 0:
50
51         print("#%s ₩t W: %s ₩t b: %s ₩t Cost: %s" % (i, W.numpy(), b.numpy(), cost.numpy()))
52
53         plt.figure(figsize=(6,6))
54         plt.title('#%s Training Linear Regression Model' % i, size=15)
55         plt.scatter(x_data, y_data, color='blue', label='Real Values')
56         plt.plot(x_data, hypothesis, color='red', label='Hypothesis')
57         plt.xlabel('x_data')
58         plt.legend(loc='upper left')
59         plt.show()
60
61         print('₩n')
```

A python code example

After a total of 1000 learning, we predict that W is about 3.6 and b is about -3.4. This is close to the actual value.





변수(Variable)

- 변수(Variable): 숫자, 문자 등의 값을 저장하는 공간의 상징

1. 문자, 숫자, Underscore(_) 사용 가능
   -문자로 시작하는 것을 권장
   Ex) time, time2, time_abp

```
In [16]: time=1
         time2=60
         time_abp=10
         print(time,time2,time_abp)

1 60 10
```

2. 대소문자 구분
   -소문자로 시작하는 것을 권장
   Ex) time, Time은 서로 다른 변수

```
In [17]: time = 60
         Time = 70
         print(time,Time)

60 70
```

3. 예약어 사용불가
   Ex) True, False, None

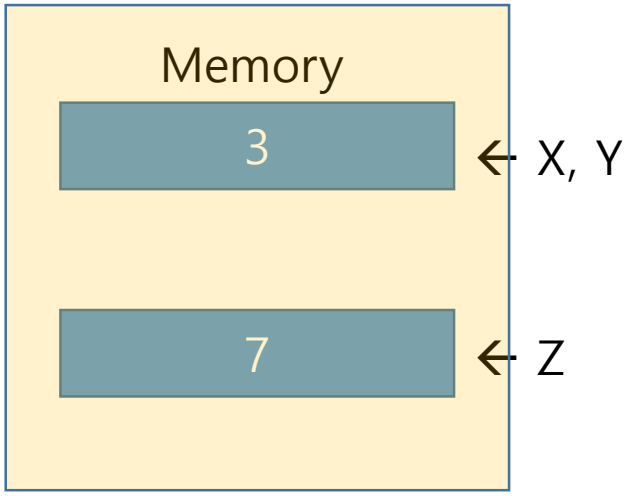
```
True = 1
False = 2
None =

  File "<ipython-input-18-1598b51f0f76>", line 1
    True = 1
         ^
SyntaxError: can't assign to keyword
```

메모리(Memory)

- 메모리(Memory): 값을 실제로 저장하는 물리적인 공간

```
In [9]: X = 3
        Y = 3
        Z = 7
        print(X,Y,Z)
        print(id(X),id(Y),id(Z))

        3 3 7
        1927966240 1927966240 1927966368
```

메모리의 물리적 주소



- Python의 모든 자료형은 객체 (cf. 같은 값을 가진 변수가 위와 같이 동일한 메모리 주소를 가질 수 있으나 항상 동일한 object를 가리키는 것은 아님. 즉, 메모리 주소가 달라질 수 있음.)

자료형

- 자료형이란 프로그래밍을 할 때 쓰이는 숫자, 문자열 등 자료 형태로 사용하는 모든 것
- 프로그램의 기본이자 핵심 단위
- 프로그래밍 언어의 자료형을 이해하고 있다면, 언어의 절반을 터득한 것

https://wikidocs.net/book/1

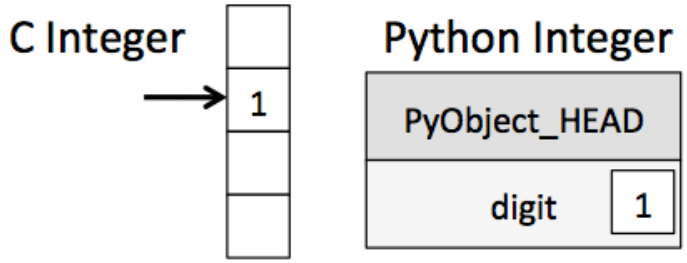
자료형

- 숫자 자료형 (Number)
- 문자열 자료형 (String)
- 불 자료형 (Bool)

- 리스트 자료형 (List)
- 튜플 자료형 (Tuple)
- 딕셔너리 자료형 (Dictionary)
- 집합 자료형 (Set)

Python 자료형 관련 주요 특징

- **Dynamic typing**
  - There is no declaration of a variable, just an assignment statement.
  - It doesn't know about the type of the variable **until the code is run**.




```c
/* C code */
int a = 1;
int b = 2;
int c = a + b;
```

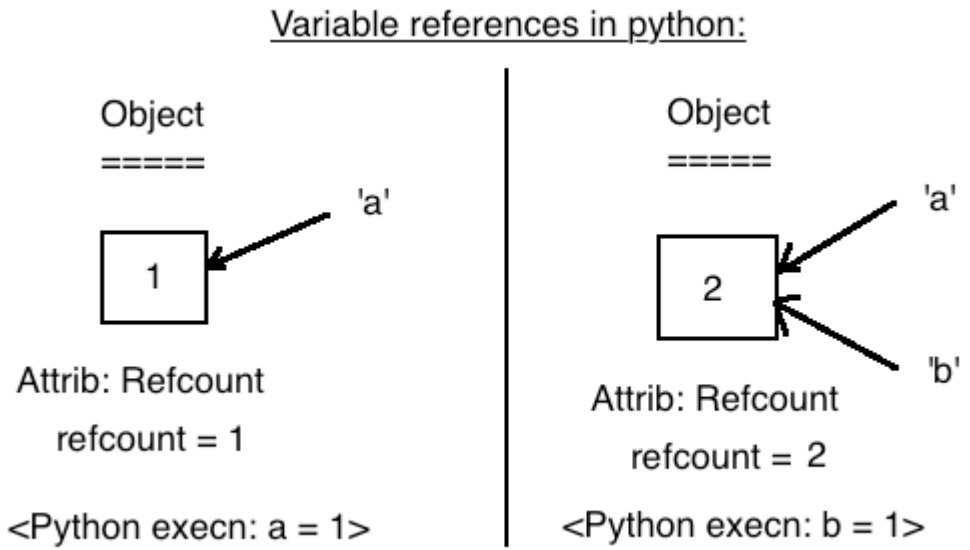
```python
# python code
a = 1
b = 2
c = a + b
```

https://jakevdp.github.io/blog/2014/05/09/why-python-is-slow/

Python memory management process

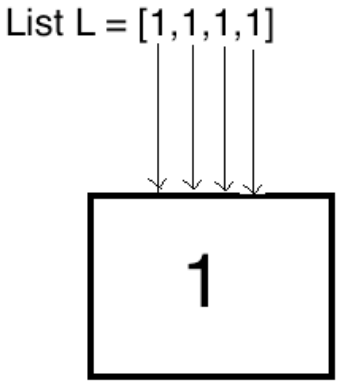
- Python memory management process

```
a=1
b=1
c=2
```



https://www.codementor.io/arch/variable-references-in-python-u9z8j2gk0

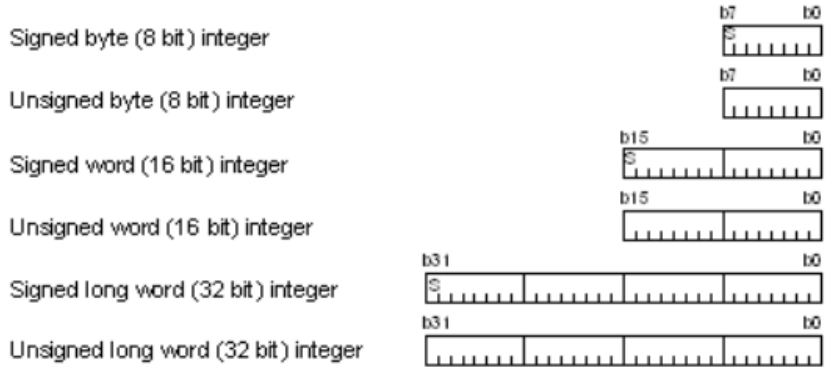
Python memory management process



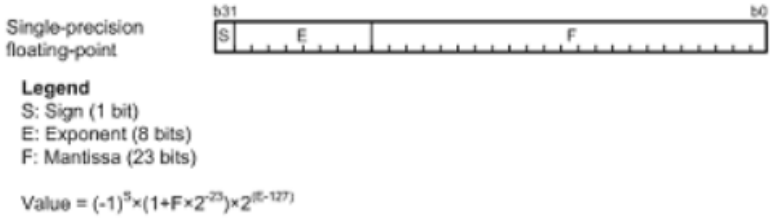

```
L = [1,1,1,1]
41596856

>>>id(L[0])
41596856

>>>id(L[1])
41596856

>>>id(L[2])
41596856

>>>id(L[3])
41596856
```

숫자형

- 정수형 (integer)
  - 예) 10, 100, -1, 0
  - Signed vs Unsigned
  - 32 bit(4 byte), 16 bit(2 byte)



- 실수형 (float) → 소수점이 있는 숫자
  - 예) 1.9, 4.41, 5.0



- 복소수형 (complex) → c.real, c.imag
  - 예) 2-3j

숫자형

```
x = 10
print(type(x))


y = 10.
print(type(y))


print(10/2)
```

숫자형

```
x = 10
print(type(x))
<class 'int'>

y = 10.
print(type(y))
<class 'float'>

print(10/2) # int/int → float
5.0
```

숫자형

```
x = 3.141592e2
print(x)



x = 10
y = 10.0
print(x == y)
```

숫자형

```
x = 3.141592e2
print(x)
```

실수 뒤의 e는 10의 제곱을 의미

```
314.1592
```

```
x = 10
y = 10.0
print(x == y)
```

```
True
```

수치 형 변환(Type Conversion)

```
x = 3.14
print(int(x))


x = 3.99
print(int(x))


x = 3
print(float(x))
```

수치 형 변환(Type Conversion)

```
x = 3.14
print(int(x))
3


x = 3.99                    소수점 이하는 버림
print(int(x))
3


x = 3
print(float(x))
3.0
```

연산자(Operator)

사칙연산

```
print(1+2)
print(2-1)
print(2*3)
print(10/2)
```

```
3
1
6
5.0
```

정수 간의 나눗셈은 실수를 반환

```
print(10//2)
```

```
5
```

// 를 이용하여 나눗셈을 하게 되면 정수를 반환

연산자(Operator)

1. 제곱

```
print(2*2*2*2) # 2의 4제곱
print(2**4) # 2의 4제곱
print(2**2.5) # 2의 2.5제곱
```

```
16
16
5.656854249492381
```

3**3은 3의 3제곱, 3e3은 3*10의 3제곱

2. 나머지

```
print(5/2) # 몫은 2 나머지는 1
print(5//2) # 몫을 반환
print(5%2) # 나머지를 반환
```

```
2.5
2
1
```

3. 증가와 감소

```
number = 3
print(number)
number = number+1
print(number)
```

```
3
4
```

```
number =3
print(number)
number += 1
print(number)
number += 3
print(number)
number -=1
print(number)
number -=3
print(number)
```

```
3
4
7
6
3
```

연산자(Operator)

bitwise operation

연산자	설명	예시
<<	왼쪽 시프트 연산자	2<<2 = 8
>>	오른쪽 시프트 연산자	11>>1 = 5
|	OR 연산자	5 | 3 = 7
&	AND 연산자	5 & 3 = 1
^	XOR 연산자	5^3 = 6
~	비트 반전 연산자	~5 = -6
< , >, <=, =>, ==, !=	작음, 큼, 작거나 같음, 크거나 같음, 같음, 같지 않음	5<3 인 경우 False 반환 5>3 인 경우 True 반환 5==5인 경우 True 반환

이진법 10 → 1000

1011→ 101

101 = 5
011 = 3

101 = 5
011 = 3

101 = 5
011 = 3

0 0101 =5 → 1 1010 = -6

연산자(Operator))

```
print(2 << 2)

print(11 >> 1)

print(5 | 3)

print(5 & 3)

print(5 ^ 3)

print(~5)
```

연산자(Operator))

```
print(2 << 2)
8
print(11 >> 1)
5
print(5 | 3)
7
print(5 & 3)
1
print(5 ^ 3)
6
print(~5)
-6
```

연산자(Operator)

연산자 우선순위

위에서부터 아래로 연산자의 우선순위	설명
**	제곱 연산자
*, /, //, %	곱셈,나눗셈,몫,나머지
+, -	덧셈, 뺄셈
<< , >>	시프트 연산자
&	AND 연산자(논리곱)
^	XOR 연산자(배타적 논리합)
|	OR 연산자 (논리합)
<,>,<=,>=	비교 연산자
==,!=	비교 연산자

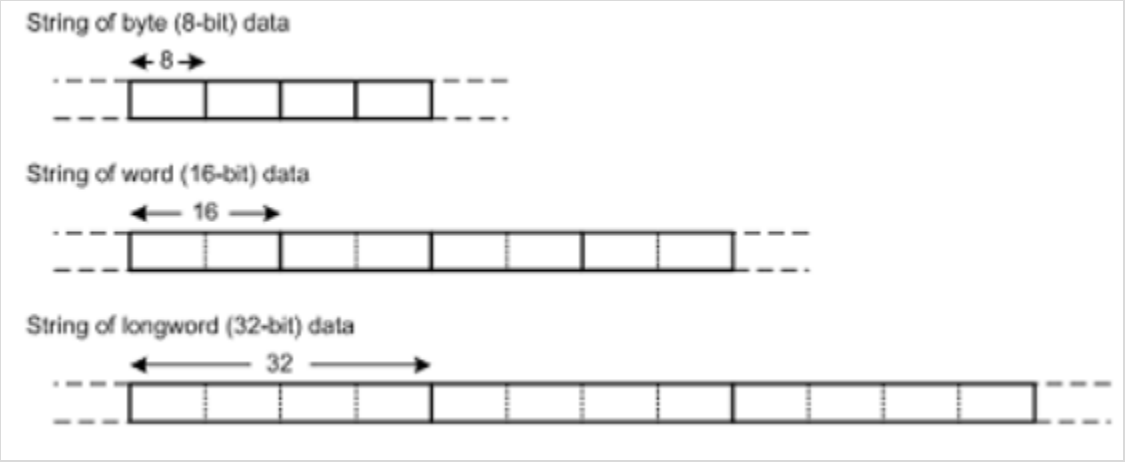
Module Import

복잡한 연산을 간단히 하기 위해서 기존에 만들어진 math 모듈을 사용

```
In [39]: import math
         print(math.pow(2,10)) # 2의 10제곱
         print(math.sqrt(144)) # 제곱근 연산 실수를 반환합니다.
         print(math.log(15)) # 자연로그 연산
         print(math.log10(100)) # 사용로그 연산

         1024.0
         12.0
         2.70805020110221
         2.0
```

문자열(String)

```
x = 'cnu' #작은 따옴표
y = "medicne"  #큰 따옴표
z = "'sophomore'" #큰 따옴표 & 작은 따옴표
```




```
print(type(x))
print(x)
print(y)
print(z)
<class 'str'>
cnu
medicine
'sophomore'
```

문자열(String)

```
x = 'cnu' #작은 따옴표
y = "medicne"  #큰 따옴표
z = "'sophomore'" #큰 따옴표 & 작은 따옴표

print(type(x))
print(x)
print(y)
print(z)
```

```
<class 'str'>
cnu
medicine
'sophomore'
```

작은 따옴표와 큰 따옴표는 같은 기능

문자열(String)

```
print(x+y+z)



x = ‘cnu'
y = ‘’ medicine"   #space 시작
z = ‘’ 'sophomore'" #space 시작
print(x+y+z)



x = 'cnu'
y = "₩nmedicine"   #backslash n
z = "₩n'sophomore'" #backslash n
print(x+y+z)
```

[참고] Escape codes

[이스케이프 코드란?]

문자열 예제에서 여러 줄의 문장을 처리할 때 백슬래시 문자와 소문자 n을 조합한 `\n` 이스케이프 코드를 사용했다. 이스케이프 코드란 프로그래밍할 때 사용할 수 있도록 미리 정의해 둔 "문자 조합"이다. 주로 출력물을 보기 좋게 정렬하는 용도로 사용한다. 몇 가지 이스케이프 코드를 정리하면 다음과 같다.

코드	설명
`\n`	문자열 안에서 줄을 바꿀 때 사용
`\t`	문자열 사이에 탭 간격을 줄 때 사용
`\\`	문자 `\`를 그대로 표현할 때 사용
`\'`	작은따옴표(`'`)를 그대로 표현할 때 사용
`\"`	큰따옴표(`"`)를 그대로 표현할 때 사용
`\r`	캐리지 리턴(줄 바꿈 문자, 현재 커서를 가장 앞으로 이동)
`\f`	폼 피드(줄 바꿈 문자, 현재 커서를 다음 줄로 이동)
`\a`	벨 소리(출력할 때 PC 스피커에서 '삑' 소리가 난다)
`\b`	백 스페이스
`\000`	널 문자

이중에서 활용빈도가 높은 것은 `\n`, `\t`, `\\`, `\'`, `\"`이다. 나머지는 프로그램에서 잘 사용하지 않는다.

https://wikidocs.net/13

문자열(String)

```
print(x+y+z) #'+'을 이용하여 문자열을 붙일 수 있음.
cnumedicne'sophomore'

x = 'cnu'
y = " medicine"  #띄어쓰기: space로 시작
z = " 'sophomore'" #띄어쓰기: space로 시작
print(x+y+z)
cnu medicine 'sophomore'

x = 'cnu'
y = "₩nmedicine"  #backslash n → 줄바꾸기
z = "₩n'sophomore'" #backslash n → 줄바꾸기
print(x+y+z)
cnu
medicine
'sophomore'
```

문자열(String)

```
x = 3
y = 9
print(x+y)



print('x+y')



print('x'+'y')
```

문자열(String)

```
x = 3
y = 9
print(x+y)
12

print('x+y')
x+y

print('x'+'y')
xy
```

불(Bool) 자료형

- 참(True)과 거짓(False)을 나타내는 자료형
  - True - 참
  - False - 거짓

- True나 False는 파이썬의 예약어로 true, false와 같이 사용하지 말고 첫 문자를 항상 대문자로 사용

불(Bool) 자료형

```
a = True
b = False
print(type(a))
print(a==b)



True == 1

False == 0

2 < 1

1 == 1.0
```

불(Bool) 자료형

```
a = True
b = False
print(type(a))
print(a==b)
<class 'bool'>
False
True == 1
True
False == 0
True
2 < 1
False
1 == 1.0
True
```

리스트(List)

여러 데이터들의 집합

List = [element1,element2,element3]

문자 데이터

```
color = ['R','G','B']
print(color)
print(color[0])
print(color[1])
print(color[2])
print(len(color)) # len은 List의 길이를 의미합니다.
```

```
['R', 'G', 'B']
R
G
B
3
```

숫자 & 숫자,문자 섞어서도 가능

```
A = [3, 7, 9, 13, 2]
print(A[0])
print(A[1])
print(len(A))
```

```
3
7
5
```

```
A = ['B',3,'C']
print(A[0])
print(A[1])
print(len(A))
```

```
B
3
3
```

리스트(List)

리스트의 활용

```
A = ['B',3,'C']
B = [7,6,5]
print(2*A) # 2회 반복
print(A+B) # 두 리스트 합치기
print('B' in A) # A 리스트 안에 B가 있는지 T/F로 반환
print('D' in A)
A.append('D')# 리스트에 새로운 요소 추가
print(A)
A.extend(['E','F']) # 리스트에 새로운 리스트를 추가
print(A)
A.insert(0,'Z') # 0번째 주소에 새로운 요소를 추가
print(A)
A.remove('Z') # 리스트에서 특정 요소 삭제
print(A)
del A[0] # 리스트에서 0번째 주소의 요소 삭제
print(A)
```

```
['B', 3, 'C', 'B', 3, 'C']
['B', 3, 'C', 7, 6, 5]
True
False
['B', 3, 'C', 'D']
['B', 3, 'C', 'D', 'E', 'F']
['Z', 'B', 3, 'C', 'D', 'E', 'F']
['B', 3, 'C', 'D', 'E', 'F']
[3, 'C', 'D', 'E', 'F']
```

리스트의 특징

인덱싱, 연산을 동일하게 사용

다양한 데이터 타입 입력 가능

리스트 안에 리스트도 입력 가능

```
A = ["B",0,1]
B = [1,2,3]
A[0] = B
print(A)
```

```
[[1, 2, 3], 0, 1]
```

리스트(List)

리스트의 함수

```
A = [4,7,9,3,2,11,4]
print(A)
print(A.index(4)) # 정수 4가 있는 주소값을 출력
print(A.count(4)) # 리스트안의 4의 갯수
A.sort() # 리스트 정렬
print(A)
A.reverse() # 리스트 역정렬
print(A)
C = sorted(A) # 정렬된 A를 C에 할당
print(C)
```

```
[4, 7, 9, 3, 2, 11, 4]
0
2
[2, 3, 4, 4, 7, 9, 11]
[11, 9, 7, 4, 4, 3, 2]
[2, 3, 4, 4, 7, 9, 11]
```

정수 4가 있는 첫번째 주소

리스트(List)

패킹(packing): 한 변수에 여러 데이터를 입력하는 것 (List)

언패킹(unpacking): 여러 데이터가 입력된 한 변수로부터 각각의 변수로 반환하는 것

```
A = [3, 4, 5]
a, b, c = A
print(a,b,c)
```

```
3 4 5
```

A에 있는 데이터 3, 4, 5를 각각의 변수 a, b, c에 할당

리스트(List)

```
color = ['R', 'G', 'B', 'R']
print(type(color))
print(color)
print(color[0])
print(color[1])
print(color[2])
print(len(color))  # len은 list의 길이
print(color.index('R'))
```

리스트(List)

```
color = ['R', 'G', 'B', 'R']
print(type(color))
print(color)
print(color[0])
print(color[1])
print(color[2])
print(len(color))  # len은 list의 길이
print(color.index('R'))
<class 'list'>
['R', 'G', 'B', 'R']
R
G
B
4
0
```

튜플(Tuple)

- 리스트와 유사한 자료형
- 차이점
  - ( )을 사용 (생략도 가능)
  - 읽기 전용 → 프로그램이 실행되는 동안 값이 항상 동일하길 원하는 경우 사용

```
t1 = ()
t2 = (1,)
t3 = (1, 2, 3)
t4 = 1, 2, 3
t5 = ('a', 'b', ('ab', 'cd'))
```

행렬 (Matrix)

리스트 안에 리스트를 만들어 행렬을 생성 가능합니다.

```
A = [3,4,5,6,7]
B = [8,9,10,11,12]
C = [1,2,3,4,5]
D = [A,B,C] # 3x5 행렬을 생성

print(D)
print(D[0][2])
print(D[1][0])
```

```
[[3, 4, 5, 6, 7], [8, 9, 10, 11, 12], [1, 2, 3, 4, 5]]
5
8
```

각각의 행렬의 요소를 호출 가능

딕셔너리(Dictionary)

- {Key1:Value1, Key2:Value2, Key3:Value3 ...}
- Key에는 변하지 않는 값을 사용하고, Value에는 변하는 값과 변하지 않는 값 모두 사용 가능

```
>>> dic = {'name':'pey', 'phone':'0119993323', 'birth': '1118'}
```

딕셔너리 dic의 정보

key	value
name	pey
phone	01199993323
birth	1118

딕셔너리(Dictionary)

```
grade = {'pey': 10, 'julliet': 99}
print(type(grade))

grade['pey']

grade['julliet']

dic = {'name':'pey', 'phone':'0119993323', 'birth': '1118'}
dic['name']

dic['phone']

dic['birth']
```

딕셔너리(Dictionary)

```
grade = {'pey': 10, 'julliet': 99}
print(type(grade))
<class 'dict'>
grade['pey']
10
grade['julliet']
99
dic = {'name':'pey', 'phone':'0119993323', 'birth': '1118'}
dic['name']
'pey'
dic['phone']
'0119993323'
dic['birth']
'1118'
```

기타 자료형

- 집합(Set)

Next Class

- Python 기본: 제어문 & 함수

제어문 (for문)

제어문 (if문)

```
24
25 # Model Initialization
26 W = tf.Variable(-0.5)
27 b = tf.Variable(-0.5)
28 learning_rate = 0.001
29
30 # Regression Training
31 cost_list=[]
32
33 # Train
34 for i in range(1000+1):
35
36     # Gradient Descent
37     with tf.GradientTape() as tape:
38
39         hypothesis = W * x_data + b
40         cost = tf.reduce_mean(tf.square(hypothesis - y_data))
41
42         # Update
43         W_grad, b_grad = tape.gradient(cost, [W, b])
44         W.assign_sub(learning_rate * W_grad)
45         b.assign_sub(learning_rate * b_grad)
46         cost_list.append(cost.numpy())
47
48     # Output
49     if i % 200 == 0:
50
51         print("#%s \t W: %s \t b: %s \t Cost: %s" % (i, W.numpy(), b.numpy(), cost.numpy()))
52
53         plt.figure(figsize=(6,6))
54         plt.title('#%s Training Linear Regression Model' % i, size=15)
55         plt.scatter(x_data, y_data, color='blue', label='Real Values')
56         plt.plot(x_data, hypothesis, color='red', label='Hypothesis')
57         plt.xlabel('x_data')
58         plt.legend(loc='upper left')
59         plt.show()
60
61         print('\n')
```

Appendix

연산자(Operator)

bitwise operation

연산자	설명	예시
<<	왼쪽 시프트 연산자	2<<2 = 8
>>	오른쪽 시프트 연산자	11>>1 = 5
|	OR 연산자	5 | 3 = 7
&	AND 연산자	5 & 3 = 1
^	XOR 연산자	5^3 = 6
~	비트 반전 연산자	~5 = -6
< , >, <=, =>, ==, !=	작음, 큼, 작거나 같음, 크거나 같음, 같음, 같지 않음	5<3 인 경우 False 반환 5>3 인 경우 True 반환 5==5인 경우 True 반환

이진법 10 → 1000

1011→ 101

101 = 5
011 = 3

101 = 5
011 = 3

101 = 5
011 = 3

0 0101 =5 → 1 1010 = -6

Complement

- Complement
  - A mathematical operation on binary number
  - Used in computing as a method of signed number representation
- 1’s complement
- 2’s complement
  - The most common method of representing signed integers on computers

2’s Complement

- The two's complement of an N-bit number is defined as its complement with respect to 2^N.
- For instance, for the three-bit number 010, the two's complement is 110, because 010 + 110 = 1000.
- The two's complement is calculated by inverting the digits and adding one.
  - 2's complement= 1's complement + 1

https://en.wikipedia.org/wiki/Two%27s_complement

2’s Complement

- Compared to other systems for representing signed numbers (e.g., ones' complement), two's complement has the advantage that the fundamental arithmetic operations of addition, subtraction, and multiplication are identical to those for unsigned binary numbers
  - (as long as the inputs are represented in the same number of bits - as the output, and any overflow beyond those bits is discarded from the result).

- This property makes the system simpler to implement, especially for higher-precision arithmetic.

- Unlike ones' complement systems, two's complement has no representation for negative zero, and thus does not suffer from its associated difficulties.

2's Complement

Example #1

```
 5 = 00000101
     ↓↓↓↓↓↓↓↓   Complement Digits
     11111010
           +1   Add 1
    ---------
-5 = 11111011
```

Example #2

```
-13 = 11110011
      ↓↓↓↓↓↓↓↓   Complement Digits
      00001100
            +1   Add 1
     ---------
 13 = 00001101
```

http://enjoytheelectronics.blogspot.com/2016/12/1s-complement-and-2s-complement-of.html

2’s Complement

Three-bit signed integers

Decimal value	Binary (two's-complement representation)	Two's complement $(2^3 - n)_2$
0	000	000
1	001	111
2	010	110
3	011	101
−4	100	100
−3	101	011
−2	110	010
−1	111	001

Eight-bit signed integers

Decimal value	Binary (two's-complement representation)	Two's complement $(2^8 - n)_2$
0	0000 0000	0000 0000
1	0000 0001	1111 1111
2	0000 0010	1111 1110
126	0111 1110	1000 0010
127	0111 1111	1000 0001
−128	1000 0000	1000 0000
−127	1000 0001	0111 1111
−126	1000 0010	0111 1110
−2	1111 1110	0000 0010
−1	1111 1111	0000 0001

Our example

^	XOR 연산자	5^3 = 6
~	비트 반전 연산자	~5 = -6
< , >, <=, =>, ==, !=	작음, 큼, 작거나 같음, 크거나 같음, 같음, 같지 않음	5<3 인 경우 False 반환 5>3 인 경우 True 반환 5==5인 경우 True 반환

101 = 5
011 = 3

0 0101 =5 → 1 1010 = -6

5 = 0 0101
1 1010
+1
-5 = 1 1011

==========

6 = 0 0110
1 1001
+1
-6 = 1 1010

[5-bit signed integer]

5 = 0 000 0101
1 111 1010
+1
-5 = 1 111 1011

==========

6 = 0 000 0110
1 111 1001
+1
-6 = 1 111 1010

[8-bit signed integer]