

# 기초 컴퓨터 프로그래밍

- Python 기본: 제어문 -

충남대학교 의과대학 의공학교실


구 윤 서 교수

# 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 W: %s b: %s 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')
```



# 제어문

- 조건문: if문
- 반복문: while문, for문

# 조건문

- 주어진 조건을 판단한 후 상황에 맞게 처리해야 할 경우

예) 시험 점수에 따른 학점 부여

점수	학점
90점 이상	A
90점 미만 80점 이상	B
80점 미만 70점 이상	C
70점 미만 60점 이상	D
60점 미만	F

# if문의 기본 구조

**if** 조건문: # 조건문 다음에 콜론(:)을 잊지 말자!

수행할 문장1

수행할 문장2

...

**else:**

수행할 문장A

수행할 문장B

...

- if문은 순차적으로 수행
  - 조건문이 참이면 if문 바로 다음의 문장(if 블록)들을 수행
  - 조건문이 거짓이면 else문 다음의 문장(else 블록)들을 수행
- else문은 if문 없이 독립적으로 사용할 수 없으며 반드시 필요한 것은 아님.

# 들여쓰기(Indentation)

if 조건문 이하 if문에 속하는 모든 문장은 **들여쓰기(indentation)**

if 조건문:

수행할 문장1

수행할 문장2



들여쓰기: spacebar 4번 or tab 1번 → spacebar 4번 권장

if 조건문:

수행할 문장1

수행할 문장2



에러 발생

# 조건문

조건문: **참과 거짓을 판단**하는 문장

if 조건문:

수행할 문장1

수행할 문장2

자료형	참	거짓
숫자	0이 아닌 숫자	0
문자열	"abc"	""
리스트	[1,2,3]	[]
튜플	(1,2,3)	()
딕셔너리	{"a":"b"}	{}

# 비교 연산자

자료형보다는 주로 **비교 연산자**(<, >, ==, !=, >=, <=)를 사용

비교연산자	설명
$x < y$	x가 y보다 작다
$x > y$	x가 y보다 크다
$x == y$	x와 y가 같다
$x != y$	x와 y가 같지 않다
$x >= y$	x가 y보다 크거나 같다
$x <= y$	x가 y보다 작거나 같다



# and, or, not

조건을 판단하기 위해 사용하는 연산자: **and, or, not**

연산자	설명
x or y	x와 y 둘 중에 하나만 참이면 참이다
x and y	x와 y 모두 참이어야 참이다
not x	x가 거짓이면 참이다

# x in s, x not in s

```
>>> 1 in [1, 2, 3]
```

True

```
>>> 1 not in [1, 2, 3]
```

False

in	not in
x in 리스트	x not in 리스트
x in 튜플	x not in 튜플
x in 문자열	x not in 문자열

# 다양한 조건을 판단하는 elif

- elif는 개수에 제한없이 사용
- else에는 조건문을 붙이지 않음.

if 조건문:

수행할 문장1

수행할 문장2

elif 조건문: *# 위의 if 조건문을 만족하지 않으면 수행*

수행할 문장A

수행할 문장B

elif 조건문: *# 위의 조건문들을 만족하지 않으면 수행*

수행할 문장A

수행할 문장B

else: *# 모든 조건문 조건을 만족하지 않으면 수행*

수행할 문장A

수행할 문장B

# 다중 조건 판단문

점수	학점
90점 이상	A
90점 미만 80점 이상	B
80점 미만 70점 이상	C
70점 미만 60점 이상	D
60점 미만	F

**[Q.1]** 아래와 같이 서로 다른 형태로 구성된 자동 학점 계산기 프로그램을 사용 시 95점을 받은 학생의 학점은?

```
print("Enter an exam score")
score = int(input()) # 숫자 입력을 받는 함수: input(), int로 type casting
print("The exam score is", score)

grade = ""
if score >= 90: grade = "A"
if score >= 80: grade = "B"
if score >= 70: grade = "C"
if score >= 60: grade = "D"
if score < 60: grade = "F"
print("The grade is", grade)
```

```
print("Enter an exam score")
score = int(input()) # 숫자 입력을 받는 함수: input(), int로 type casting
print("The exam score is", score)

grade = ''
if score >= 90: grade = 'A'
elif score >= 80: grade = 'B'
elif score >= 70: grade = 'C'
elif score >= 60: grade = 'D'
else: grade = 'F'
print("The grade is", grade)
```

(참고) 수행할 문장이 한 문장인 경우에는 붙여쓰기 가능

## [참고] Data Console

- **input()**: 사용자에게 console window에서 string (문자열)을 입력받는 함수
- 항상 string으로 입력되므로, 숫자로 사용하기 위해서는 integer, float 등으로 type casting (=type conversion) 해야함.

```
print("Enter a exam core")
```

```
score = int(input()) # 숫자 입력을 받는 함수: input(), int로 type casting
```

```
print("A exam core is", score)
```

# 다중 조건 판단문 (Solution)

95점을 받은 학생의 학점

점수	학점
90점 이상	A
90점 미만 80점 이상	B
80점 미만 70점 이상	C
70점 미만 60점 이상	D
60점 미만	F

```
print("Enter an exam score")
score = int(input()) # 숫자 입력을 받는 함수: input(), int로 type casting
print("The exam score is", score)

grade = ""
if score >= 90: grade = "A"
if score >= 80: grade = "B"
if score >= 70: grade = "C"
if score >= 60: grade = "D"
if score < 60: grade = "F"
print("The grade is", grade)
```

Enter an exam score

95

The exam score is 95

The grade is D

```
print("Enter an exam score")
score = int(input()) # 숫자 입력을 받는 함수: input(), int로 type casting
print("The exam score is", score)

grade = ''
if score >= 90: grade = 'A'
elif score >= 80: grade = 'B'
elif score >= 70: grade = 'C'
elif score >= 60: grade = 'D'
else: grade = 'F'
print("The grade is", grade)
```

Enter an exam score

95

The exam score is 95

The grade is A

- 순차적으로 if문이 수행되어 D와 F 학점만 출력됨.

- 다중 조건 판단을 가능하게 하는 elif으로 인해 점수 구간에 맞는 학점이 출력됨.

# 다중 조건 판단문 (Solution)



The screenshot shows a Jupyter Notebook interface with a file named 'if\_while\_for.ipynb'. The code is written in Python and demonstrates a solution for a multiple condition judgment problem. The code is as follows:

```
1 print("Enter an exam score")
2 score = int(input()) # 숫자 입력을 받는 함수: input(), int로 type casting
3 print("The exam score is", score)
4
5 grade = ""
6 if score >= 90: grade = "A"
7 elif score >= 80: grade = "B"
8 elif score >= 70: grade = "C"
9 elif score >= 60: grade = "D"
10 else: grade = "F"
11 print("The grade is", grade)
```

The output of the code is shown in the cell below, indicating that the code was executed successfully and the result is displayed.

```
[3] 1 print("Enter an exam score")
2 score = int(input()) # 숫자 입력을 받는 함수: input(), int로 type casting
3 print("The exam score is", score)
4
5 grade = ""
6 if score >= 90: grade = "A"
7 elif score >= 80: grade = "B"
8 elif score >= 70: grade = "C"
9 elif score >= 60: grade = "D"
10 else: grade = "F"
11 print("The grade is", grade)
```

# While 문

- 반복해서 문장을 수행해야 할 경우 while문을 사용

**while** 조건문: *# 조건문 다음에 콜론(:)을 잊지 말자!*

수행할 문장1

수행할 문장2

수행할 문장3

...

- while문은 조건문이 참인 동안에 while문 아래에 속하는 문장들이 반복해서 수행



# While 문

```
numStamp = 0
while numStamp < 10:
    numStamp = numStamp + 1
    print("커피 쿠폰에 도장을 %d번 찍었습니다." % numStamp)
    if numStamp == 10:
        print("무료 Iced Americano 1잔!")
```

*#while 문 끝에 : (colon)*

*#if 문 끝에 : (colon)*

*#마지막 print문  
indentation 주의*

커피 쿠폰에 도장을 1번 찍었습니다.  
커피 쿠폰에 도장을 2번 찍었습니다.  
커피 쿠폰에 도장을 3번 찍었습니다.  
커피 쿠폰에 도장을 4번 찍었습니다.  
커피 쿠폰에 도장을 5번 찍었습니다.  
커피 쿠폰에 도장을 6번 찍었습니다.  
커피 쿠폰에 도장을 7번 찍었습니다.  
커피 쿠폰에 도장을 8번 찍었습니다.  
커피 쿠폰에 도장을 9번 찍었습니다.  
커피 쿠폰에 도장을 10번 찍었습니다.  
무료 Iced Americano 1잔!

[formatting]

- %s is used as a placeholder for string values.
- %d is used as a placeholder for numeric or decimal values.

# While 문

```
numStamp = 0
while numStamp < 10:
    numStamp = numStamp + 1
    print("커피 쿠폰에 도장을 %d번 찍었습니다." % numStamp)
    if numStamp == 10:
        print("무료 Iced Americano 1잔!")
```

numStamp	조건문	조건판단	수행하는 문장	while문
0	0 < 10	참	커피 쿠폰에 도장을 1번 찍었습니다.	반복
1	1 < 10	참	커피 쿠폰에 도장을 2번 찍었습니다.	반복
2	2 < 10	참	커피 쿠폰에 도장을 3번 찍었습니다.	반복
3	3 < 10	참	커피 쿠폰에 도장을 4번 찍었습니다.	반복
4	4 < 10	참	커피 쿠폰에 도장을 5번 찍었습니다.	반복
5	5 < 10	참	커피 쿠폰에 도장을 6번 찍었습니다.	반복
6	6 < 10	참	커피 쿠폰에 도장을 7번 찍었습니다.	반복
7	7 < 10	참	커피 쿠폰에 도장을 8번 찍었습니다.	반복
8	8 < 10	참	커피 쿠폰에 도장을 9번 찍었습니다.	반복
9	9 < 10	참	커피 쿠폰에 도장을 10번 찍었습니다. 무료 Iced Americano 1잔!	반복
10	10 < 10	거짓		종료

# 무한 Loop

```
while True:
```

```
    수행할 문장1
```

```
    수행할 문장2
```

```
    ...
```

- 무한 loop란 **무한히 반복**한다는 의미
- while문의 조건문이 True이므로 항상 참 → 따라서 while 문 안에 있는 문장들은 무한하게 수행
- 일반적인 프로그램 중에서 무한 loop의 개념을 사용하지 않는 프로그램은 거의 없음 → 그만큼 자주 사용!
  - 일단 무한 loop를 만들어놓고 어떤 특정 입력이나 조건이 들어왔을 때 무한 loop를 빠져나오게 하는 방식

# while문 강제로 빠져나가기

- **break문**: 호출되는 경우, while문을 빠져나감

```
item = 10
money = 500
while money:
    print("돈을 받았으니 물건을 줍니다.")
    item = item - 1
    print("재고는 %d개입니다." %item)
    if not item:
        print("Out of Stock!")
        break
```

# while문 강제로 빠져나가기

```
item = 10
money = 500
while money:
    print("돈을 받았으니 물건을 줍니다.")
    item = item - 1
    print("재고는 %d개입니다." % item)
    if not item:
        print("Out of Stock!")
        break
```

돈을 받았으니 물건을 줍니다.  
재고는 9개입니다.  
돈을 받았으니 물건을 줍니다.  
재고는 8개입니다.  
돈을 받았으니 물건을 줍니다.  
재고는 7개입니다.  
돈을 받았으니 물건을 줍니다.  
재고는 6개입니다.  
돈을 받았으니 물건을 줍니다.  
재고는 5개입니다.  
돈을 받았으니 물건을 줍니다.  
재고는 4개입니다.  
돈을 받았으니 물건을 줍니다.  
재고는 3개입니다.  
돈을 받았으니 물건을 줍니다.  
재고는 2개입니다.  
돈을 받았으니 물건을 줍니다.  
재고는 1개입니다.  
돈을 받았으니 물건을 줍니다.  
재고는 0개입니다.  
Out of Stock!

# while문 강제로 빠져나가기



if\_while\_for.ipynb ☆ [www.BANDICAM.com](http://www.BANDICAM.com) 댓글

파일 수정 보기 삽입 런타임 도구 도움말 모든 변경사항이 저장됨

+ 코드 + 텍스트

[5] 

```
8 elif score >= 70: grade = "C"
9 elif score >= 60: grade = "D"
10 else: grade = "F"
11 print("The grade is", grade)
```

Enter an exam score  
95  
The exam score is 95  
The grade is A

```
1 item = 10
2 money = 500
3 while money:
4     print("돈을 받았으니 물건을 줍니다.")
5     item = item - 1
6     print("재고는 %d개입니다." % item)
7     if not item:
8         print("Out of Stock!")
9         break
```

# while문의 처음으로 돌아가기

- Continue 문
- while문 안의 문장을 수행할 때 입력된 조건을 검사해서 조건에 맞지 않으면 while문을 빠져 나감.
- while문을 빠져나가지 않고 while문의 맨 처음(=조건문)으로 다시 돌아가게 만들고 싶은 경우가 발생

(예제) 1부터 10까지의 숫자 중 홀수만 출력

```
a = 0
while a < 10:
    a = a+1
    if a % 2 == 0: continue #한 문장 수행 시 붙여쓰기 가능
    print(a)
```

# while문의 처음으로 돌아가기

- **1부터 10까지의 숫자 중 홀수만 출력**

- a가 10보다 작은 동안 a는 1만큼씩 계속 증가
- if a % 2 == 0(a를 2로 나누었을 때 나머지가 0인 경우)이 참이 되는 경우 continue 문장 수행
- continue문은 while문의 맨 처음(조건문: a<10)으로 돌아가게 하는 명령어
- a가 짝수이면 print(a)는 수행되지 않음.

**[Q.2]** 아래와 같이 서로 다른 형태로 구성된 프로그램의 결과는 동일한가?

```
a = 0
while a < 10:
    a = a+1
    if a % 2 == 0: continue
    print(a)
```

```
a = 0
while a < 10:
    a = a+1
    if a % 2 != 0: print(a)
```



# while문의 처음으로 돌아가기 (Solution)

- 동일한 결과를 서로 다른 구조의 프로그램으로 작성 가능!
- 따라서 동일 결과를 출력하더라도 프로그래머의 실력에 따라 코드의 간결성, 처리 속도, 사용하는 메모리 크기가 달라짐.

```
a = 0
while a < 10:
    a = a+1
    if a % 2 == 0: continue
    print(a)
```

1  
3  
5  
7  
9

```
a = 0
while a < 10:
    a = a+1
    if a % 2 != 0: print(a)
```

1  
3  
5  
7  
9

# for문

- Python의 직관적인 특징을 가장 잘 표현
- While문과 비슷한 반복문인 for문은 매우 유용하고 문장 구조 파악이 용이
- **for**문은 반복 횟수를 명확히 알고 있을 때 주로 사용하고 **while**문은 조건에 따라 반복 횟수를 결정해야 할 때 주로 사용

**for** 변수 **in** 리스트(또는 튜플, 문자열):

수행할 문장1

수행할 문장2

...

# for문

```
test_list = ['one', 'two', 'three']  
for i in test_list:  
    print(i)
```

```
a = [(1,2), (3,4), (5,6)]  
for (first, last) in a:  
    print(first + last)
```

# for문

```
test_list = ['one', 'two', 'three']  
for i in test_list:  
    print(i)
```

```
test_list = ['one', 'two', 'three']  
for i in test_list:  
    print(i)
```

```
one  
two  
three
```

```
a = [(1,2), (3,4), (5,6)]  
for (first, last) in a:  
    print(first + last)
```

```
a = [(1,2), (3,4), (5,6)]  
for (first, last) in a:  
    print(first + last)
```

```
3  
7  
11
```

## for문 응용

(예제) 총 5명의 학생이 시험을 보았는데 시험 점수가 60점이 넘으면 pass이고 그렇지 않으면 fail이다.

학생의 점수가 아래와 같을 때 각 학생이 pass인지 fail인지 결과를 출력하시오.

```
marks = [90, 25, 67, 45, 80]
```

1번 학생은 90점이고 5번 학생은 80점이다.

# for문 응용

```
marks = [90, 25, 67, 45, 80]
```

```
number = 0
```

```
for mark in marks:
```

```
    number = number + 1
```

```
    if mark >= 60:
```

```
        print("%d번 학생은 pass입니다." % number)
```

```
    else:
```

```
        print("%d번 학생은 fail입니다." % number)
```

# for문 응용

```
marks = [90, 25, 67, 45, 80]
number = 0

for mark in marks:
    number += 1
    if mark >= 60:
        print("%d번 학생은 pass입니다." % number)
    else:
        print("%d번 학생은 fail입니다." % number)
```

1번 학생은 pass입니다.  
2번 학생은 fail입니다.  
3번 학생은 pass입니다.  
4번 학생은 fail입니다.  
5번 학생은 pass입니다.

# for와 함께 자주 사용하는 range함수

- for문은 숫자 리스트를 자동으로 만들어 주는 range라는 함수와 함께 사용되는 경우가 많음.

```
a = range(10)
print(a)
```

```
range(0, 10)
```

- range(10)은 0부터 10 미만의 숫자를 포함하는 range 객체를 생성
- 시작 숫자와 끝 숫자를 지정하려면 range(시작 숫자, 끝 숫자) 형태를 사용 (끝 숫자는 포함되지 않음.)



# range 응용

- for와 range 함수를 이용하면 1부터 10까지 더하는 것을 다음과 같이 쉽게 구현 가능

```
totalSum = 0
for i in range(1, 11):
    totalSum += i
print(totalSum)
```

# range 응용

- for와 range 함수를 이용하면 1부터 10까지 더하는 것을 다음과 같이 쉽게 구현 가능

```
totalSum = 0
for i in range(1, 11):
    totalSum += i
print(totalSum)
```

```
totalSum = 0
for i in range(1, 11):
    totalSum += i
print(totalSum)
```

# range 응용

- for와 range 함수를 이용하여 간결한 코드 작성 가능

```
marks = [90, 25, 67, 45, 80]
number = 0

for mark in marks:
    number += 1
    if mark >= 60:
        print("%d번 학생은 pass입니다." % number)
    else:
        print("%d번 학생은 fail입니다." % number)
```



```
marks = [90, 25, 67, 45, 80]

for i in range(len(marks)): #len 함수는 list의 길이를 출력
    if marks[i] >= 60:
        print("%d번 학생은 pass입니다." % (i+1))
    else:
        print("%d번 학생은 fail입니다." % (i+1))
```

1번 학생은 pass입니다.  
2번 학생은 fail입니다.  
3번 학생은 pass입니다.  
4번 학생은 fail입니다.  
5번 학생은 pass입니다.

1번 학생은 pass입니다.  
2번 학생은 fail입니다.  
3번 학생은 pass입니다.  
4번 학생은 fail입니다.  
5번 학생은 pass입니다.

# 이중 for 문

- 이중 for문 사용 가능
- 이중 for문과 range 함수를 이용하면 소스 코드 단 4줄만으로 구구단 출력 가능

```
for i in range(2,10):  
    for j in range(1, 10): #각 i마다 j를 증가시키며 print문을 수행  
        print(i*j, end=" ")  
    print("")
```

[입력 인수 end]

- `print(i*j, end=" ")` 마지막에 입력 인수 end를 끝문자에 지정해주면, 줄바꿈되지 않고 한 줄에 연속으로 출력 가능
- 그 다음에 이어지는 `print("")`는 2단, 3단 등을 구분하기 위해 두 번째 for문이 끝나면 결과값을 다음 줄부터 출력하게 함.

## 이중 for 문

```
for i in range(2,10):  
    for j in range(1, 10):  
        print(i*j, end=" ")  
    print('')
```

```
2 4 6 8 10 12 14 16 18  
3 6 9 12 15 18 21 24 27  
4 8 12 16 20 24 28 32 36  
5 10 15 20 25 30 35 40 45  
6 12 18 24 30 36 42 48 54  
7 14 21 28 35 42 49 56 63  
8 16 24 32 40 48 56 64 72  
9 18 27 36 45 54 63 72 81
```

# 리스트 안에 for문 포함하기

- 리스트 안에 for문을 포함하는 **리스트 내포(List comprehension)**를 이용하면 좀 더 편리하고 직관적인 프로그램을 만들 수 있다. (**Pythonic code!**)

```
a = [1,2,3,4]
result = []
for num in a:
    result.append(num*3)
print(result)
```

[3, 6, 9, 12]



```
a = [1,2,3,4]
result = [num * 3 for num in a]
print(result)
```

[3, 6, 9, 12]

# 리스트 안에 for문 포함하기

- "if 조건"을 사용 가능

```
[표현식 for 항목 in 반복가능객체 if 조건]
```

- for문을 2개 이상 사용하는 것도 가능

```
[표현식 for 항목1 in 반복가능객체1 if 조건1  
    for 항목2 in 반복가능객체2 if 조건2  
    ...  
    for 항목n in 반복가능객체n if 조건n]
```

## 리스트 안에 for문 포함하기

- 짝수에만 3을 곱하여 새로운 리스트를 만들고 싶다면 다음과 같이 "if 조건"을 사용 가능

```
a = [1,2,3,4]
result = [num * 3 for num in a if num % 2 == 0]
print(result)
```

[6, 12]



# 리스트 안에 for문 포함하기

- 동일한 결과를 서로 다른 구조의 프로그램으로 작성 가능!

```
for i in range(2,10):  
    for j in range(1, 10):  
        print(i*j, end=" ")  
    print('')
```

```
2 4 6 8 10 12 14 16 18  
3 6 9 12 15 18 21 24 27  
4 8 12 16 20 24 28 32 36  
5 10 15 20 25 30 35 40 45  
6 12 18 24 30 36 42 48 54  
7 14 21 28 35 42 49 56 63  
8 16 24 32 40 48 56 64 72  
9 18 27 36 45 54 63 72 81
```



```
result = [x*y for x in range(2,10)  
          for y in range(1,10)]  
print(result)
```

```
[2, 4, 6, 8, 10, 12, 14, 16, 18, 3, 6, 9, 12, 15, 18, 21, 24, 27, 4, 8, 12, 16, 20, 24, 28, 32, 36, 5, 10, 15, 20, 25, 30, 35, 40, 45, 6, 12, 18, 24, 30, 36, 42, 48, 54, 7, 14, 21, 28, 35, 42, 49, 56, 63, 8, 16, 24, 32, 40, 48, 56, 64, 72, 9, 18, 27, 36, 45, 54, 63, 72, 81]
```



# Next Class

- 함수 (Function)