인공지능 기초 프로그래밍

Python 기초 (제어문)

파이썬 프로그램 기초-제어문

- if 문 (if statement)

  돈이 10,000원 이상 있으면 택시를 타고, 10,000원 미만이면 버스를 탄다.

```
if have_money >= 10000:
    take_taxi()
else:
    take_bus()
```

  - 프로그래밍에서 조건을 판단하여 해당 조건에 맞는 상황을 수행하는데 사용

파이썬 프로그램 기초-제어문

- if 문 (if statement)

▪ if문 기본 구조

- if와 else를 사용한 조건문의 기본 구조

```
if 조건문:
    수행할 문장1
    수행할 문장2
    ...
else:
    수행할 문장A
    수행할 문장B
    ...
```

주의 사항

- 조건문 뒤에 콜론 (:)
- 들여쓰기 (indentation)

- 조건문이 참이면 if 블록 수행
- 조건문이 거짓이면 else 블록 수행

파이썬 프로그램 기초-제어문

- if 문 (if statement)

▪ 비교 연산자

비교 연산자	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

▪ 비교 연산자 사용 예시

```
>>> x = 3
>>> y = 2
>>> x > y  ◀— 3 > 2
True
```

```
>>> x == y  ◀— 3 == 2
False
```

연산자	설명
x or y	x와 y 둘 중에 하나만 참 이어도 참
x and y	x와 y 둘 모두 참이어야 참
not x	x가 거짓이면 참

in	not in
x in list	x not in list
x in tuple	x not in tuple
x in string(문자열)	x not in string

파이썬 프로그램 기초-제어문

- if 문 (if statement)
  - **비교 연산자**
    - if 조건문에 비교 연산자를 사용하는 예시

만약 3000원 이상의 돈을 가지고 있으면 택시를 타고 그렇지 않으면 걸어 가라.

```
>>> money = 2000 ◀── 2000원을 가지고 있다.
>>> if money >= 3000:
...     print("택시를 타고 가라")
... else:
...     print("걸어 가라")
...
걸어 가라
```

파이썬 프로그램 기초-제어문

- if 문 (if statement)

연산자	설명	in	not in
x or y	x와 y 둘 중에 하나만 참 이어도 참	x in list	x not in list
x and y	x와 y 둘 모두 참이어야 참	x in tuple	x not in tuple
not x	x가 거짓이면 참	x in string(문자열)	x not in string

```
sum_var = 0
for i in range(10):
  sum_var += i
print(sum_var)

if(not sum_var):
  print(sum_var)

card = True
if money >= 3000 or card:
  print("Take Taxi")
else:
  pass
```

```
list_var = [1, 2, 3, 4, 5]
if 8 in list_var:
  print("1 is in the list")
else:
  print("Can't find input number")
```

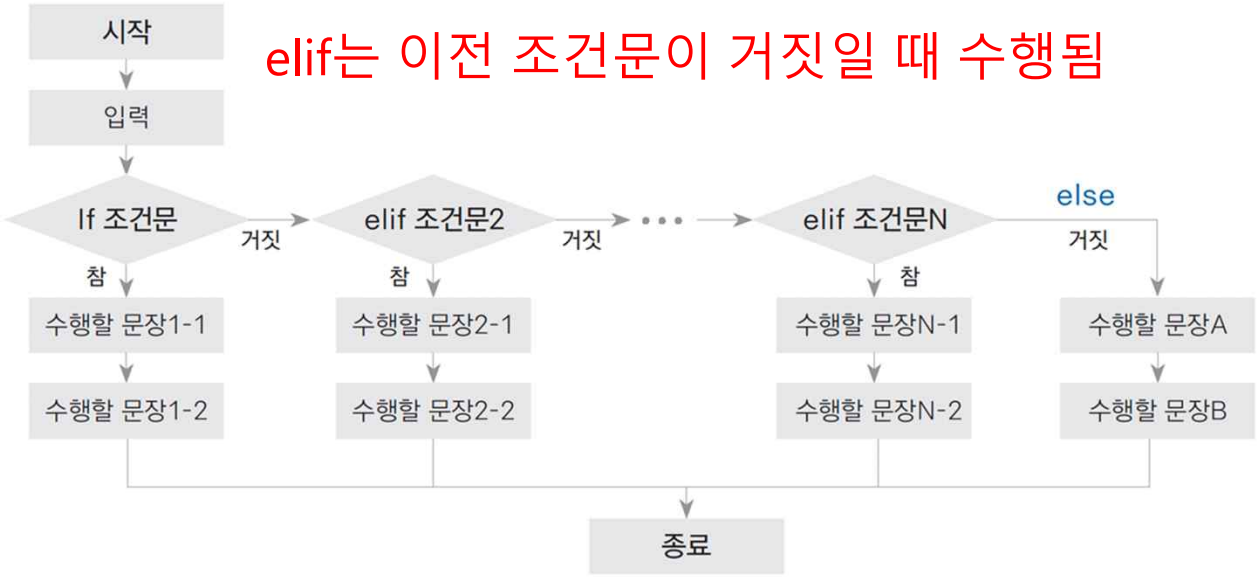
```
#include <stdio.h>

#define TRUE   (1)
#define FALSE  (0)

int main(void) {
    const int find_num = 2;
    int i = 0;
    _Bool in_array = FALSE;
    int array[5] = {1, 2, 3, 4, 5};

    for(i=0; i<5; i++)
    {
        if(find_num == array[i])
        {
            in_array = TRUE;
        }
    }

    if(in_array)
    {
        printf("the find number is in the array\n");
    }
    else
    {

    }

    return 0;
}
```

파이썬 프로그램 기초-제어문

- if 문 (if statement)

▪ elif 사용법

▪ elif는 개수에 제한 없이 사용 가능



elif는 이전 조건문이 거짓일 때 수행됨

```
if 조건문:
    수행할 문장1-1
    수행할 문장1-2
    ...
elif 조건문2:
    수행할 문장2-1
    수행할 문장2-2
    ...
...
elif 조건문N:
    수행할 문장N-1
    수행할 문장N-2
    ...
...
else:
    수행할 문장A
    수행할 문장B
    ...
```

파이썬 프로그램 기초-제어문

- if 문 (if statement)
  - elif

▪ elif 사용법

- if와 else만으로는
  조건 판단에 어려움이 있음

> 주머니에 돈이 있으면 택시를 타고,
> 주머니에 돈은 없지만 카드가 있으면 택시를 타고,
> 돈도 없고 카드도 없으면 걸어 가라.

- 조건 판단하는 부분
  1) 주머니에 돈이 있는지 판단
  2) 주머니에 돈이 없으면,
     주머니에 카드가 있는지 판단

```
>>> pocket = ['paper', 'cellphone'] ◀── 주머니 안에 종이, 휴대전화가 있다.
>>> card = True ◀── 카드를 가지고 있다.
>>> if 'money' in pocket:
...     print("택시를 타고 가라")
... else:
...     if card:
...         print("택시를 타고 가라")
...     else:
...         print("걸어 가라")
...
택시를 타고 가라
```

```
>>> pocket = ['paper', 'cellphone']
>>> card = True
>>> if 'money' in pocket: ◀── 주머니에 돈이 있으면
...     print("택시를 타고 가라")
... elif card: ◀── 주머니에 돈이 없고 카드가 있으면
...     print("택시를 타고 가라")
... else: ◀── 주머니에 돈도 없고 카드도 없으면
...     print("걸어 가라")
...
택시를 타고 가라
```

- if와 else만으로는
  이해하기 어렵고 산만한 느낌

파이썬 프로그램 기초-제어문

- if 문 (if statement)
  - Append
    - If statement 한 줄로 작성
    - 조건부 표현식

```
#if statement 한 줄로 표현
if money >= 3000: print("Take Taxi")
else: print("Take Bus")

#조건부 표현식
score = 20
if score >= 60: message="Success"
else: message="Failure"
print(message)

message = "Success" if score >= 60 else "Failure"
print(message)
```

"조건문이 참인 경우" if 조건문 else "조건문이 거짓인 경우"

Practice #1 (if statement)

```
money = 2000
if money >= 3000:
  print("Take Taxi")
else:
  print("Take Bus")

sum_var = 0
for i in range(10):
  sum_var += i
print(sum_var)

if(not sum_var):
  print(sum_var)

card = True
if money >= 3000 or card:
  print("Take Taxi")
else:
  pass

list_var = [1, 2, 3, 4, 5]
if 8 in list_var:
  print("1 is in the list")
else:
  print("Can't find input number")
```

```
#if statement 한 줄로 표현
if money >= 3000: print("Take Taxi")
else: print("Take Bus")

#조건부 표현식
score = 20
if score >= 60: message="Success"
else: message="Failure"
print(message)

message = "Success" if score >= 60 else "Failure"
print(message)
```

```
Take Bus
45
Take Taxi
Can't find input number
Take Bus
Failure
Failure
```

파이썬 프로그램 기초-제어문

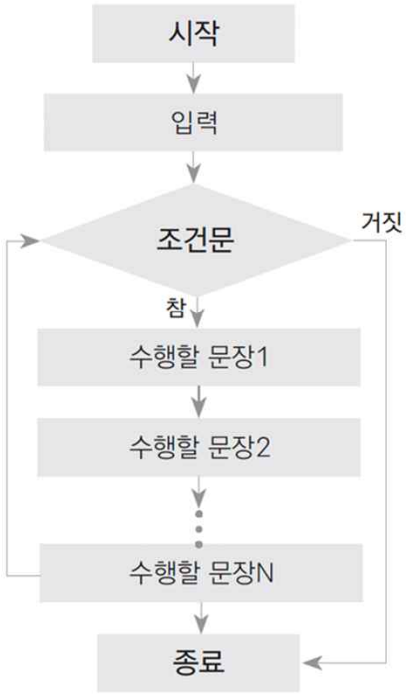
- while 문 (while statement)

▪ while문 기본 구조

- 반복해서 문장을 수행해야 할 경우 while문 사용
- **반복문**이라고도 부름

```
while 조건문:
    수행할 문장1
    수행할 문장2
    수행할 문장3
    ...
```

- while문은 **조건문이 참인 동안**에 while문 아래 문장이 **반복해서 수행**됨



파이썬 프로그램 기초-제어문

- while 문 (while statement)

while문 기본 구조

예제) '열 번 찍어 안 넘어가는 나무 없다'는 속담 구현

while문의 조건문은 **treeHit < 10**

treeHit이 10보다 작은 동안에 while문 안의 문장 반복 수행

```
>>> treeHit = 0 ◀── 나무를 찍은 횟수
>>> while treeHit < 10: ◀── 나무를 찍은 횟수가 10보다 작은 동안 반복
...     treeHit = treeHit + 1 ◀── 나무를 찍은 횟수 1씩 증가
...     print("나무를 %d번 찍었습니다." % treeHit)
...     if treeHit == 10: ◀── 나무를 열 번 찍으면
...         print("나무 넘어갑니다.")
```

```
나무를 1번 찍었습니다.
나무를 2번 찍었습니다.
나무를 3번 찍었습니다.
나무를 4번 찍었습니다.
나무를 5번 찍었습니다.
나무를 6번 찍었습니다.
나무를 7번 찍었습니다.
나무를 8번 찍었습니다.
나무를 9번 찍었습니다.
나무를 10번 찍었습니다.
나무 넘어갑니다.
```

treeHit	조건문	조건 판단	수행하는 문장	while문
0	0 < 10	참	나무를 1번 찍었습니다	반복
1	1 < 10	참	나무를 2번 찍었습니다	반복
2	2 < 10	참	나무를 3번 찍었습니다	반복
3	3 < 10	참	나무를 4번 찍었습니다	반복
4	4 < 10	참	나무를 5번 찍었습니다	반복
5	5 < 10	참	나무를 6번 찍었습니다	반복
6	6 < 10	참	나무를 7번 찍었습니다	반복
7	7 < 10	참	나무를 8번 찍었습니다	반복
8	8 < 10	참	나무를 9번 찍었습니다	반복
9	9 < 10	참	나무를 10번 찍었습니다 나무 넘어갑니다	반복
10	10 < 10	거짓		종료

파이썬 프로그램 기초-제어문

- while 문 (while statement)

▪ break문

▪ 강제로 while문을 빠져나가야 할 때 사용


```
>>> coffee = 10 ◀— 자판기에 커피가 10개 있다.
>>> money = 300 ◀— 자판기에 넣을 돈은 300원이다.
>>> while money:
...     print("돈을 받았으니 커피를 줍니다.")
...     coffee = coffee - 1 ◀— while문을 한 번 돌 때 커피가 하나 줄어든다.
...     print("남은 커피의 양은 %d개입니다." % coffee)
...     if coffee == 0:
...         print("커피가 다 떨어졌습니다. 판매를 중지합니다.")
...         break
...
```

- money가 300으로 고정되어 있어, while문의 조건문은 항상 참 → 무한 루프
- break문 호출 시 while문 종료

파이썬 프로그램 기초-제어문

- while 문 (while statement)

▪ continue문

- while문을 빠져나가지 않고 while문의 맨 처음(조건문)으로 다시 돌아가야 할 때 사용

- 1부터 10까지의 숫자 중 홀수만 출력하는 예시
  - 조건문이 참이 되는 경우 → a가 짝수
  - continue 문장 수행 시 while문의 맨 처음, 즉 조건문 a < 10으로 돌아감
  - 따라서 a가 짝수이면 print(a)는 수행되지 않음

```
>>> a = 0
>>> while a < 10:
...     a = a + 1
...     if a % 2 == 0: continue
...     print(a)
...
1
3
5
7
9
```

a를 2로 나누었을 때 나머지가 0이면 맨 처음으로 돌아간다.

파이썬 프로그램 기초-제어문

- while 문 (while statement)

▪ continue문

- while문을 빠져나가지 않고 while문의 맨 처음(조건문)으로 다시 돌아가야 할 때 사용

- 1부터 10까지의 숫자 중 홀수만 출력하는 예시
  - 조건문이 참이 되는 경우 → a가 짝수
  - continue 문장 수행 시 while문의 맨 처음, 즉 조건문 a < 10으로 돌아감
  - 따라서 a가 짝수이면 print(a)는 수행되지 않음

```
>>> a = 0
>>> while a < 10:
...     a = a + 1
...     if a % 2 == 0: continue
...     print(a)
...
1
3
5
7
9
```

a를 2로 나누었을 때 나머지가 0이면 맨 처음으로 돌아간다.

Practice #2 (while statement)

```
heat_tree = 0
while heat_tree < 10:
  heat_tree += 1
  print(f'나무를 {heat_tree}번 찍었습니다')
  if heat_tree == 10:
    print("나무 넘어갑니다")

prompt = """
    1.ADD
    2.DEL
    3.LIST
    4.QUIT
    Enter number:
"""

number = 0
# while number != 4:
while number < 4:
  print(prompt)
  number = int(input())
```

```
coffee = 3
while True:  #무한 루프 (Break statement로 빠져나가야함)
  money = int(input("Insert Money: "))
  if not coffee:
    print("커피가 없습니다")
    break

  if money == 300:
    print("커피")
    coffee -= 1
  elif money > 300:
    print(f'{money-300}원 반환 + 커피')
    coffee -= 1
  else:
    print(f'돈이 모자랍니다.')

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

파이썬 프로그램 기초-제어문

- For 문 (for statement)

▪ for문의 기본 구조

▪ for문

- while문과 비슷한 반복문

```
for 변수 in 리스트(또는 튜플, 문자열):
    수행할 문장1
    수행할 문장2
    ...
```

- 리스트나 튜플, 문자열의 첫 번째 요소부터 마지막 요소까지 차례로 변수에 대입되어 '수행할 문장1', '수행할 문장2' 등이 수행됨

```
>>> test_list = ['one', 'two', 'three']
>>> for i in test_list:  ◀— one, two, three를 순서대로 i에 대입
...     print(i)
...
one
two
three
```

파이썬 프로그램 기초-제어문

- For 문 (for statement)

▪ range 함수 사용법

▪ range( )

- 숫자 리스트를 자동으로 만들어주는 함수

```
>>> a = range(10)
>>> a
range(0, 10) ◀── 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
```

- range(10)은 0부터 10 미만의 숫자를 포함하는 range 객체를 만들어 준다.

- range(a, b)
  - a: 시작 숫자
  - b: 끝 숫자 (반환 범위에 포함되지 않음)

```
>>> a = range(1, 11)
>>> a
range(1, 11) ◀── 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
```

파이썬 프로그램 기초-제어문

- For 문 (for statement)

▪ 구구단 만들기

①번 for문

- 2부터 9까지의 숫자(range(2, 10))가 차례로 i에 대입됨

②번 for문

- 1부터 9까지의 숫자(range(1, 10))가 차례로 j에 대입됨
- print(i*j) 수행

```
>>> for i in range(2,10): ◀── ①번 for문
...     for j in range(1, 10): ◀── ②번 for문
...         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
```

파이썬 프로그램 기초-제어문

- For 문 (for statement)
  - **리스트 내포 사용법**
    - **리스트 내포(List comprehension)**
      - 리스트 안에 for문 포함하기
      - 예제 1
        - a 리스트의 각 항목에 3을 곱한 결과를 result 리스트에 담는 예제

```
>>> 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]
```

파이썬 프로그램 기초-제어문

- For 문 (for statement)

▪ 리스트 내포 사용법

▪ 리스트 내포(List comprehension)

- 리스트 안에 for문 포함하기

- 예제 3
  - 리스트 내포 안에 'if 조건' 사용 가능
  - [1, 2, 3, 4] 중에서 짝수에만 3을 곱하여 담도록 수정

```
>>> a = [1,2,3,4]
>>> result = [num * 3 for num in a if num % 2 == 0]
>>> print(result)
[6, 12]
```

파이썬 프로그램 기초-제어문

- For 문 (for statement)

▪ 리스트 내포 사용법

▪ 리스트 내포의 일반 문법

- 'if 조건' 부분은 생략 가능

```
[표현식 for 항목 in 반복 가능 객체 if 조건]
```

- for문 2개 이상 사용 가능

```
[표현식 for 항목1 in 반복 가능 객체1 if 조건1
       for 항목2 in 반복 가능 객체2 if 조건2
       ...
       for 항목n in 반복 가능 객체n if 조건n]
```

- 구구단의 모든 결과를 리스트로 담는 리스트 내포 예제

```
>>> 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]
```

Practice #3 (for statement)

```
a = [(1, 3), (2, 4), (3, 5)]
for (f, r) in a:
  print(f + r)

marks = [90, 25, 67, 45, 80]
number = 0 # print (student num)
for points in marks:
  number += 1
  if points < 60:
    print(f'{number}인 학생은 불합격입니다.')
  else:
    print(f'{number}인 학생은 합격입니다')

#Range
# (start, end, step)
for i in range(0, 10):
  print(i, end="")
print(" ")
for i in range(0, 10, 2):
  print(i, end="")
print()
# list comprehension
list_a = [1, 2, 3, 4]
result = [num*3 for num in list_a]
print(result)

result = [num*3 for num in list_a if num % 2 == 0]
print(result)

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

Thank you

Q&A

www.kopo.ac.kr
jsshin7@kopo.ac.kr

