제어문

정수를 입력받아서 홀수인지 짝수인지 판단하는 프로그램을 작성하자.
홀수이면 ‘odd’ 짝수이면 ‘even’을 출력하자.

03-1 if문

if문 기본 구조

- 조건의 결과에 따라서 실행되는 문장이 결정

```
if 조건식:
   문장
```

- 조건식이 참(True)과 거짓(False)으로 나뉘는데 주로 논리 연산자나 산술 연산자 등이 많이 사용됨

```
만약에 변수 a의 값이 5보다 클 경우 ~ 해라
if a > 5:
 ~해라
```

```
만약에 평균 점수가 90보다 클 경우 장학금 지급을 나타내시오
if 평균 > 90:
    print("장학금 지급")
```

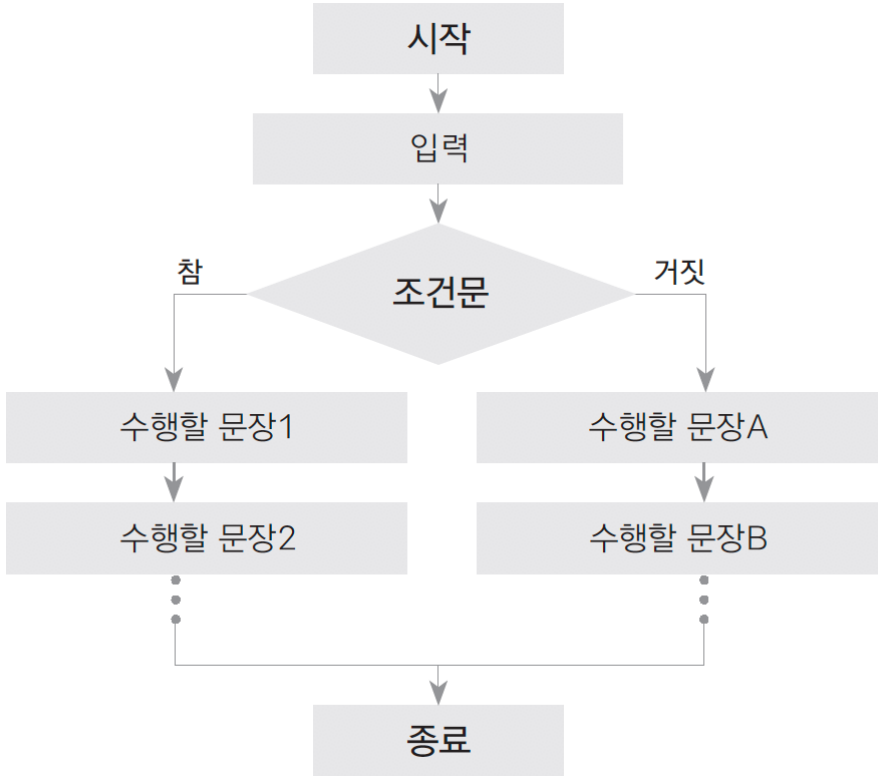
03-1 if문

▪ if문 기본 구조

- if와 else를 사용한 조건문의 기본 구조

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

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



03-1 if문

▪ 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
```

03-1 if문

▪ 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
```

03-1 if문

▪ 들여쓰기

```
if 조건문:
    수행할 문장1
    수행할 문장2
    수행할 문장3
```

▪ 들여쓰기 오류

```
if 조건문:
    수행할 문장1
수행할 문장2
    수행할 문장3
```

```
>>> if money:
...     print("택시를")
... print("타고")
...     print("가자")
```

03-1 if문

▪ and, or, not 연산자

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

▪ 돈이 3000원 이상 있거나 카드가 있다면 택시를 타고 그렇지 않으면 걸어 가라

```
>>> money = 2000
>>> card = 1
>>> if money >= 3000 or card:
...     print("택시를 타고 가라")
... else:
...     print("걸어가라")
...
택시를 타고 가라
```

03-1 if문

x in s, x not in s

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

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

```
>>> 'a' in ('a', 'b', 'c')
True
>>> 'j' not in 'python'
True
```

03-1 if문

▪ 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
```

- if와 else만으로는
  이해하기 어렵고 산만한 느낌

03-1 if문

▪ elif 사용법

▪ elif를 사용한다면?

```
>>> 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
```

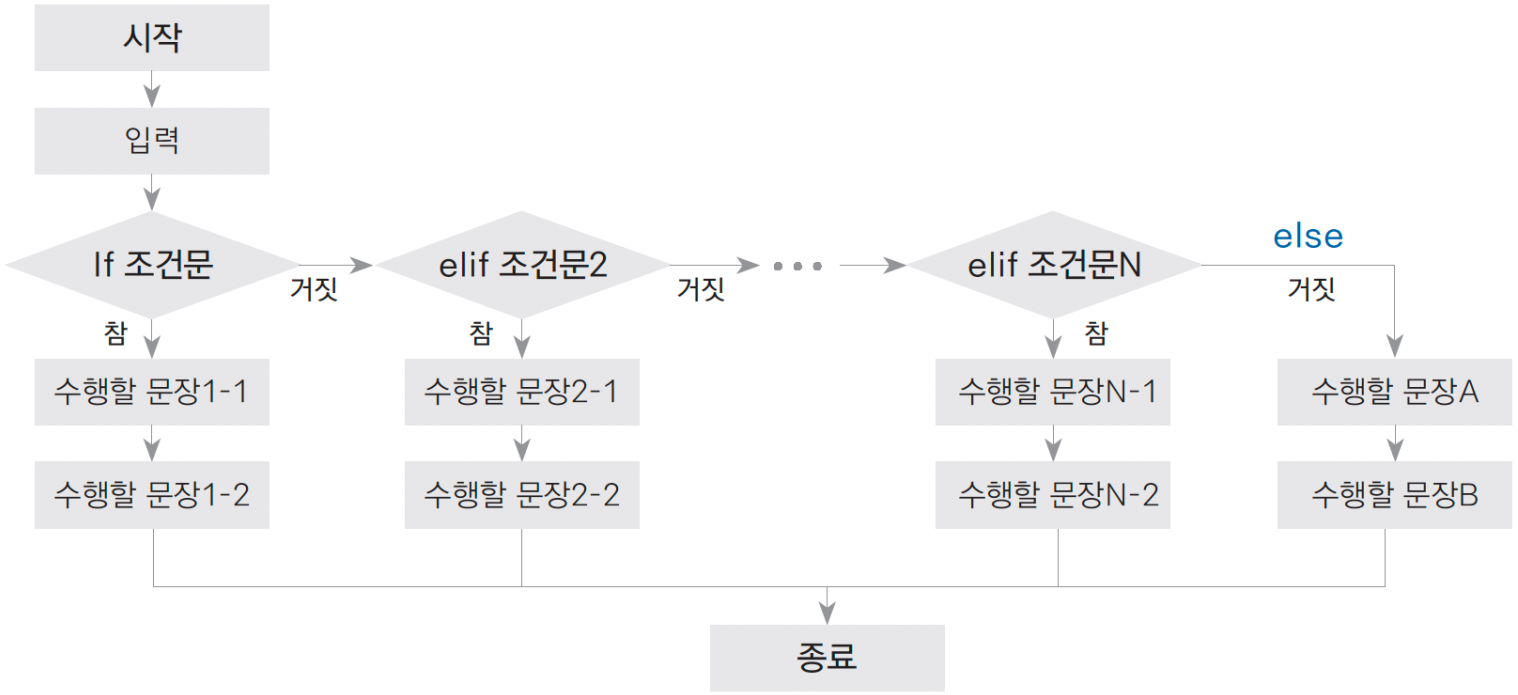
- elif는 이전 조건문이 거짓일 때 수행됨

```
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
    ...
```

03-1 if문

▪ elif 사용법

- elif는 개수에 제한 없이 사용 가능



03-1 if문

- 조건문에서 아무 일도 하지 않게 설정하고 싶다면?

```
>>> pocket = ['paper', 'money', 'cellphone']
>>> if 'money' in pocket:
...     pass
... else:
...     print("카드를 꺼내라")
...
```

- 조건부 표현식(Conditional expression)

```
if score >= 60:
    message = "success"
else:
    message = "failure"
```

```
message = "success" if score >= 60 else "failure"
```

조건문이 참인 경우 if 조건문 else 조건문이 거짓인 경우

03-1 if문

- **실습문제**
  - 점수(score)를 입력 받아서 90점 이상이면 'A', 80점 이상이면 'B', 70점 이상이면 'C'를 그 외 점수는 'F'를 출력하는 프로그램을 작성하시오.

감사합니다

03-2 while문

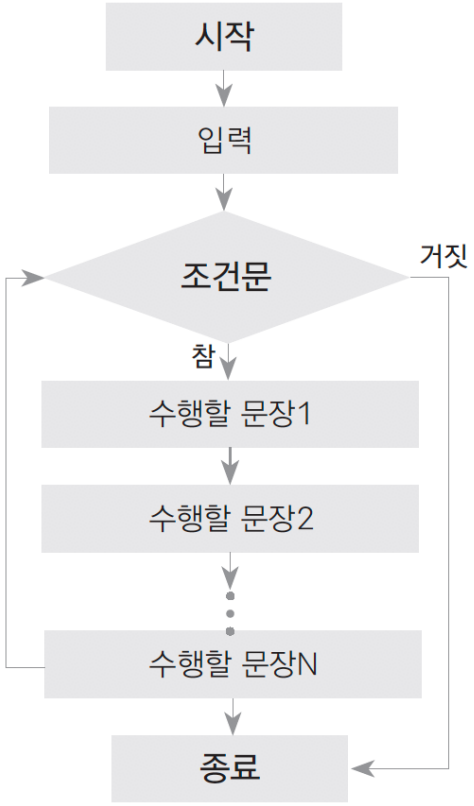
while문 기본 구조

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

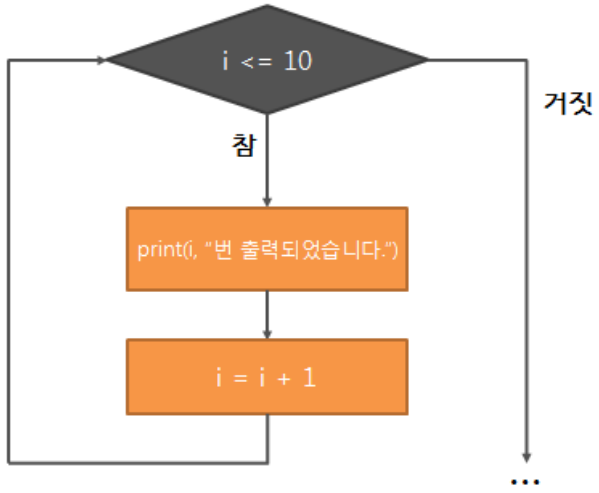
```
while 조건문:
    수행할 문장1
    수행할 문장2
    수행할 문장3
    ...
```

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

03-2 while문




```
>>> i = 1
    while i <= 10:
        print(i, "번 출력되었습니다.")
        i = i + 1
```



03-2 while문

▪ 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.
```

03-2 while문

▪ while문 기본 구조

- 예제) '열 번 찍어 안 넘어가는 나무 없다'는 속담 구현
  - while문이 반복되는 과정

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

03-2 while문

▪ break문

- 강제로 while문을 빠져나가야 할 때 사용
- 일반적으로 if 조건문 안에 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

03-2 while문

continue문

- continue문을 만나면 아래 실행문 수행하지 않고 시작 지점(조건 검사)으로 돌아가기
- while문을 빠져나가지 않고 while문의 맨 처음(조건문)으로 다시 돌아가야 할 때 사용
- 일반적으로 if 조건문 안에 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.

03-2 while문

▪ 무한 루프

- 반복문의 조건이 무조건 참이 되어 반복이 종료 되지 않음

```
while True:
    수행할 문장1
    수행할 문장2
    ...
```

```
>>> while True:
...     print("Ctrl+C를 눌러야 while문을 빠져나갈 수 있습니다.")
...
Ctrl+C를 눌러야 while문을 빠져나갈 수 있습니다.
Ctrl+C를 눌러야 while문을 빠져나갈 수 있습니다.
Ctrl+C를 눌러야 while문을 빠져나갈 수 있습니다.
....
```

03-2 while문

- **실습문제**
  - while문을 사용해 1부터 1000까지의 자연수 중 3의 배수의 합을 출력하는 프로그램을 작성하시오.

감사합니다

03-3 for문

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
```

```
>>> a = [(1,2), (3,4), (5,6)]
>>> for (first, last) in a:
...     print(first + last)
...
3
7
11
```

03-3 for문

for문 활용

```
>>> a = [(1,2), (3,4), (5,6)]
>>> for (first, last) in a:
...     print(first + last)
...
3
7
11
```

- 5명의 학생 점수가 60점이 넘으면 합격이고 그렇지 않으면 불합격

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

number = 0
for mark in marks:
    number = number +1
    if mark >= 60:
        print("%d번 학생은 합격입니다." % number)
    else:
        print("%d번 학생은 불합격입니다." % number)
```

03-3 for문

▪ for문과 continue문

- for문 안의 문장을 수행하는 도중에 continue문을 만나면 for문의 처음으로 돌아감.

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

number = 0
for mark in marks:
    number = number +1
    if mark < 60: continue
    print("%d번 학생 축하합니다. 합격입니다. " % number)
```

03-3 for문

▪ 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
```

03-3 for문

▪ range 함수 활용법

```
>>> for i in range(10):
...     print(i, end=" ")
```

```
>>> add = 0
>>> for i in range(1, 11):
...     add = add + i
...
>>> print(add)
55
```

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

```
>>> for i in range(len(test_list)):
...     print(test_list[i])
```

03-3 for문

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

number = 0
for mark in marks:
    number = number +1
    if mark < 60: continue
    print("%d번 학생 축하합니다. 합격입니다. " % number)
```

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

for number in range(len(marks)):
    if marks[number] < 60: continue
    print("%d번 학생 축하합니다. 합격입니다. " % (number+1))
```

03-3 for문

▪ 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
```

03-3 for문

▪ 구구단 만들기

i가 2일 때

i	j	i*j
2	1	2
	2	4
	3	8
	4	8
	5	10
	6	12
	7	14
	8	16
	9	18
②번 for문 종료		

i가 3일 때

i	j	i*j
3	1	3
	2	6
	3	9
	4	12
	5	15
	6	18
	7	21
	8	24
	9	27
②번 for문 종료		

i가 4일 때

i	j	i*j
4	1	4
	2	8
	3	12
	4	16
	5	20
	6	24
	7	28
	8	32
	9	36
②번 for문 종료		

• • •

i가 9일 때

i	j	i*j
9	1	9
	2	18
	3	27
	4	36
	5	45
	6	54
	7	63
	8	72
	9	81
전체 for문 종료		

03-3 for문

▪ 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]
```

03-3 for문

▪ 리스트 내포 사용법

- **리스트 내포(List comprehension)**
  - 리스트 안에 for문 포함하기

  - 예제 2
    - 예제 1을 리스트 내포를 사용하도록 수정

```
>>> a = [1,2,3,4]
>>> result = [num * 3 for num in a]
>>> print(result)
[3, 6, 9, 12]
```

03-3 for문

▪ 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]
```

03-3 for문

- **실습문제**
  - A 학급에 총 10명의 학생이 있다. 이 학생들의 중간고사 점수는 다음과 같다.
    [70, 60, 55, 75, 95, 90, 80, 80, 85, 100]
    for문을 사용하여 A 학급의 평균 점수를 구하시오.

감사합니다