단국대학교
SW중심대학

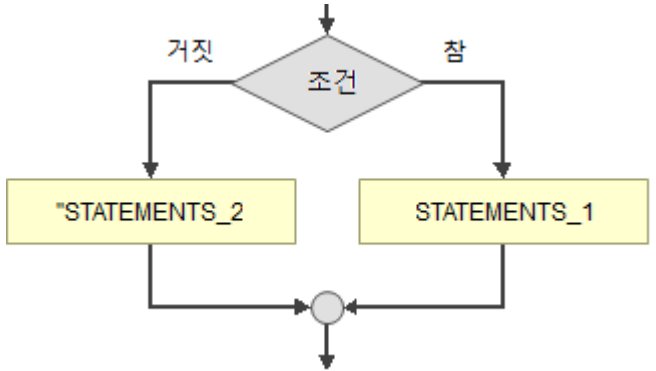
창의적 사고와 코딩

Lecture 3-2. 선택

2. IF-ELSE 문

if – else 문

- 파이썬에서 선택 구조를 위해 if문 사용
- 들여쓰기(indent)를 주의하여 작성해야 올바르게 실행
  - 들여쓰기: tab 1번 또는 space 4번
  - 내어쓰기: backspace 키 이용




```
if  BOOLEAN EXPRESSION: # BOOLEAN EXPRESSION is called the condition
                        #  불리언 식의 결과는 부울형 반환
        STATEMENTS_1 # executed if condition evaluates to True
else:
        STATEMENTS_2 # executed if condition evaluates to False
```

동전 던지기 게임

- 동전을 던지기 게임을 작성해 보자.
- import random한 후에 random.randrange(2) 하면 0이나 1을 랜덤하게 생성할 수 있다.

동전 던지기 게임

- **random.randrange(start, stop[, step=1]) 함수**
  - 'start'부터 'stop-1'까지 정수 중에서 임의의 숫자를 하나 반환
  - step: 생략하면 1, 숫자들의 단계
    - random.randrange(0,10)은 0부터 9까지 숫자 중에서 임의의 숫자를 반환
    - random.randrange(0,10,2) ➔ 0, 2, 4, 6, 8 중에서 임의의 숫자 반환
    - random.randrange(0,-10,-2) ➔ 0, -2, -4, -6, -8 중에서 임의의 숫자 반환
- **random.randrange(stop)함수**
  - 0 부터 'stop-1'까지의 정수 중에서 임의의 숫자를 하나 반환
    - random.randrange(5) ➔ 0, 1, 2, 3, 4 중에서 임의의 숫자 반환

```
import random
random.randrange(0,2)
```

```
import random
random.randrange(2)
```

Solution : 파일명 coin.py

```
import random

print("동전 던지기 게임을 시작합니다.")
coin = random.randrange(2)

if coin == 0 :
    print("앞면입니다.")
else :
    print("뒷면입니다.")

print("게임이 종료되었습니다.")
```

축구게임

- 사용자가 컴퓨터를 상대로 패널티 킥을 하는 축구게임을 만든다. 사용자는 3가지 영역 중 하나를 선택하여 패널티 킥을 하고, 컴퓨터도 난수를 생성하여 3개의 영역 중 하나를 수비한다.

어디를 공격하시겠어요?(왼쪽, 중앙, 오른쪽) 중앙
패널티 킥을 성공하였습니다.

함수	설명	사용 예
random.choice(seq)	seq (데이터 집합에서 일부를 무작위로 선택하는 것 seq : 데이터 집합	import random menu = ['쫄면', '육계장', '비빔밥'] random.choice(menu) # menu 중 하나를 반환해줌

Solution : 파일명 football.py

논리 연산자

- 논리 연산자(logical operator)는 여러 개의 조건을 조합하여 참인지 거짓인지를 따질 때 사용한다.
- 종류 : AND(논리곱), OR(논리합), NOT(논리부정)

연산	의미	사용 예
x and y	AND 연산, x와 y가 모두 참이면 참, 그렇지 않으면 거짓	(a > 100) and (a < 200)
x or y	OR 연산, x나 y중에서 하나만 참이면 참, 모두 거짓이면 거짓	(a == 100) or (a == 200)
not x	NOT 연산, x가 참이면 거짓, x가 거짓이면 참	not(a < 100)

논리 연산자

- 불리언 식의 결과는 집합의 진리표(Truth Table)와 같음

p	*q*	*p* and *q*	*p* or *q*	*not p*	*not q*
False	False	False	False	True	True
False	True	False	True	True	False
True	False	False	True	False	True
True	True	True	True	False	False

```
>>> a = True
>>> b = False
>>> a and b
False
>>> a or b
True
>>> not b
True
```

논리 연산자

- 관계연산자와 논리 연산자를 포함한 불리언 식
- 변수 a는 b보다 크고 c보다 작은가?

```
>>> a=26
>>> b=31
>>> c=54
>>> a>b and a<c
False
>>> b<a<c
False
```

```
>>> a<b<c  # b>a and b<c
True
>>> b>a and b<c
True
```

논리 연산자

- 놀이공원에서 놀이기구를 탈 수 있는 조건을 논리 수식으로 작성하 여 보면 다음과 같다.

조건 1 조건 2

나이가 10살 이상이고, 그리고 키가 165cm 이상이면
→ 놀이기구를 탈 수 있다.

(나이가 10살 이상이다) and (키가 165cm 이상이다)
→ 놀이기구를 탈 수 있다.

그리고

전체적인 구조

```
age >= 10  and  height >= 165
```

나이가 10살 이상

키가 165cm 이상

예제

```
age = 20
height  = 180
if age>=10 and  height>=165 :
    print("놀이 기구를 탈 수 있습니다.")
else :
    print("놀이 기구를 탈 수 없습니다.")
```

```
놀이 기구를 탈 수 있습니다.
```

부울 식의 활용

요구사항 및 문제 해결 알고리즘

어느 회사의 입사 조건은 나이가 30세 이하이거나 토익 점수가 700점 이상이 만족되어야 지원 가능하다.

단국이는 나이가 31세이고 토익이 900점이다. 단국이가 이 회사에 지원이 가능한지 부울 식을 사용하여 확인해보자.

단국의 나이 = 31
단국이의 토익 점수 = 900
만약 (단국이의 나이가 <= 30) or (단국이의 영어 점수 >= 700) 길동은 회사에 지원할 수 있다.

프로그램

```
>>> age = 31
>>> toeic = 900
>>> age <= 30 or toeic >= 700
True
```

Lab: 졸업 학점 검사하기

- 어떤 대학교를 졸업하려면 적어도 140학점을 이수해야 하고 평점이 2.0은 되어야 한 다고 하자. 이것을 파이썬 프로그램으로 검사해보자. 사용자에게 이수학점수와 평점 을 물어보고 졸업 가능 여부를 출력하는 프로그램을 작성해보자.

```
이수한 학점수를 입력하시오: 120
평점을 입력하시오: 2.3
졸업이 힘듭니다!
```

```
이수한 학점수를 입력하시오: 140
평점을 입력하시오: 2.5
졸업 가능합니다!
```

Solution : 파일명 grad.py

Lab: 윤년 판단

- **입력된 연도가 윤년인지 아닌지를 판단하는 프로그램을 만들어보자. 윤년은 다음의 조건 을 만족해야 한다.**
  - 연도가 4로 나누어 떨어지고, 100으로 나누어 떨어지는 연도를 제외하면 윤년이다.
  - 400으로 나누어 떨어지는 연도는 윤년이다.

```
연도를 입력하시오: 2020
2020 년은 윤년입니다.
```

Solution : 파일명 cal_leapyear.py

종달새가 노래할까?

- **동물원에 있는 종달새가 다음과 같은 2가지 조건이 충족될 때 노래를 한다고 하자.**
  - 오전 6시부터 오전 9시 사이 이면서 날씨가 화창하면 종달새가 노래를 한다.

난수 이용

- 현재 시각을 난수로 생성하고 날씨도 [True, False] 중에서 랜덤하게 선택하자. 종달새가 노래를 부를 것인지, 조용히 있을 것인지를 판단해보자.

```
import random
time = random.randint(1, 24)
sunny = random.choice([True, False])
```

1 ≤ ≤ 24

randint

리스트

```
좋은 아침입니다. 지금 시각은 1시 입니다.
현재 날씨가 화창하지 않습니다.
종달새가 노래를 하지 않는다.
```

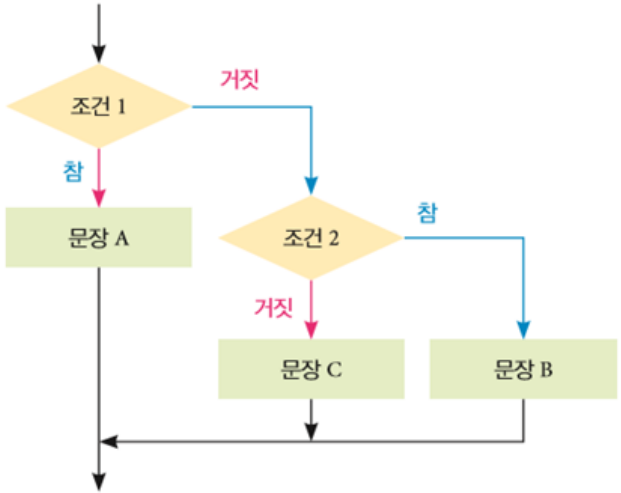
Solution: 파일명 bird.py

중첩 if-else 문

- 우리는 if-else 문장 안에 다른 if-else 문장을 넣을 수 있다. 이것을 중첩(nesting)된 if-else 문이라고 부른다.

전체적인 구조

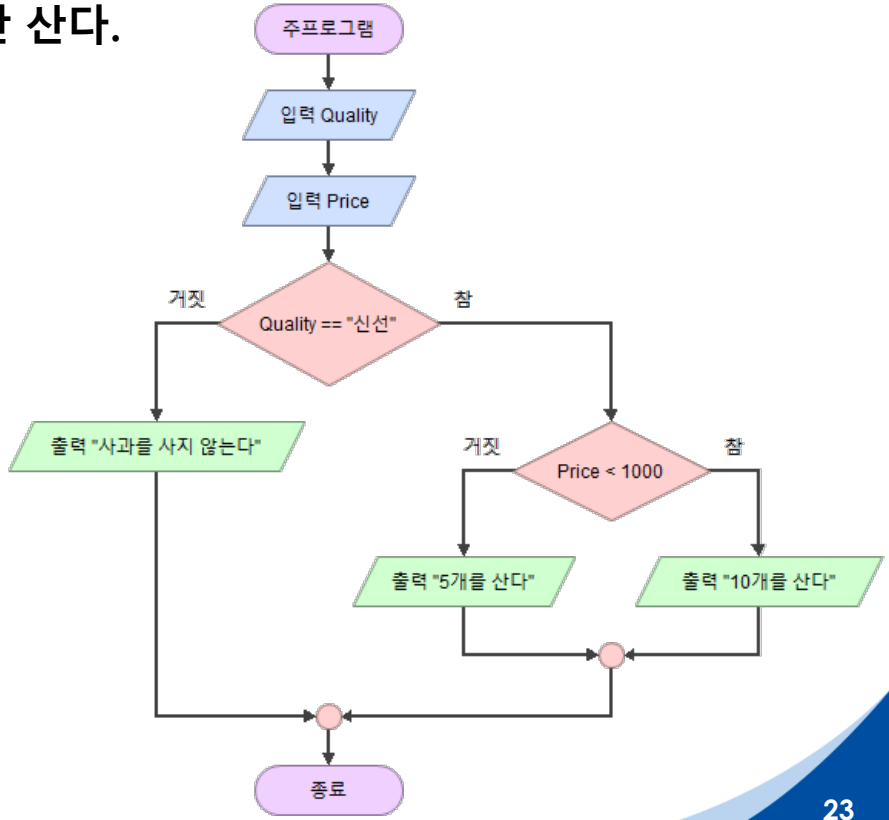
```
if 조건1 :
    문장_A
else:
    if 조건2 :
        문장_B
    else:
        문장_C
```



예제: 파일명 apple.py

- 마트에서 사과가 신선하면 사과를 사기로 한다. 만약 사과가 개당 1000원 미만이면 10개를 산다. 하지만 사과가 개당 1000원 이상이면 5개만 산다.

```
Quality = input("사과의 상태를  입력하시오: ")
Price = int(input("사과 1개의 가격을 입력하시오: "))
if Quality == "신선":
    if Price < 1000:
        print("10개를 산다")
    else:
        print("5개를 산다")
else:
    print("사과를 사지 않는다.")
```



Lab: 음수, 0, 양수 구별하기

- 앞에서 사용자가 입력한 수가 양수인지 0인지 음수인지를 구별하여 출력하는 프로그램을 작성해보았다. 여기서는 중첩 if-else문을 사용하여 동일한 기능을 하는 프로그램을 작성해보자.

```
정수를 입력하시오: 0
0입니다.
```

```
정수를 입력하시오: -10
음수입니다.
```

```
정수를 입력하시오: 10
양수입니다.
```

Solution : 파일명 pos_neg.py

```
num = int(input("정수를 입력하시오: "))
if num >= 0:
    if num == 0:
        print("0입니다.")
    else:
        print("양수입니다.")
else:
    print("음수입니다.")
```

in 과 not in 연산자 : 멤버쉽 연산자

- x가 문자열이나 리스트에서 존재하는지 여부를 알 수 있다.

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

```
>>> 'a' in ['a', 'b', 'c', 'd']
True
>>> 9 in [3, 2, 9, 10, 9.0]
True
>>> "a" in ["apple", "absolutely", "application", "nope"]
False
>>> "apple" in ["apple", "absolutely", "application", "nope"]
True
>>> "a" not in ["apple", "absolutely", "application", "nope"]
True
```

```
>>> 'p' in 'apple'
True
>>> 'x' in 'apple'
False
>>> 'x' not in 'apple'
True
>>> 'ap' not in 'apple'
False
>>> 'apple' in 'apple'
True
>>> '' in 'apple'
True
```

연산자 우선순위

Level	Category	Operators
7(high)	exponent	**
6	multiplication	*,/,//,%
5	addition	+,-
4	relational	==,!=,<=,>=,>,<
3	logical	not
2	logical	and
1(low)	logical	or

Lab: 아이디 검사

- 아이디를 입력받아서 등록된 아이디인지를 검사하는 프로그램을 작성해보자. 등록된 아이디를 리스트(list)에 저장하도록 한다. 아이디가 일치하면 패스워드 물어본다.

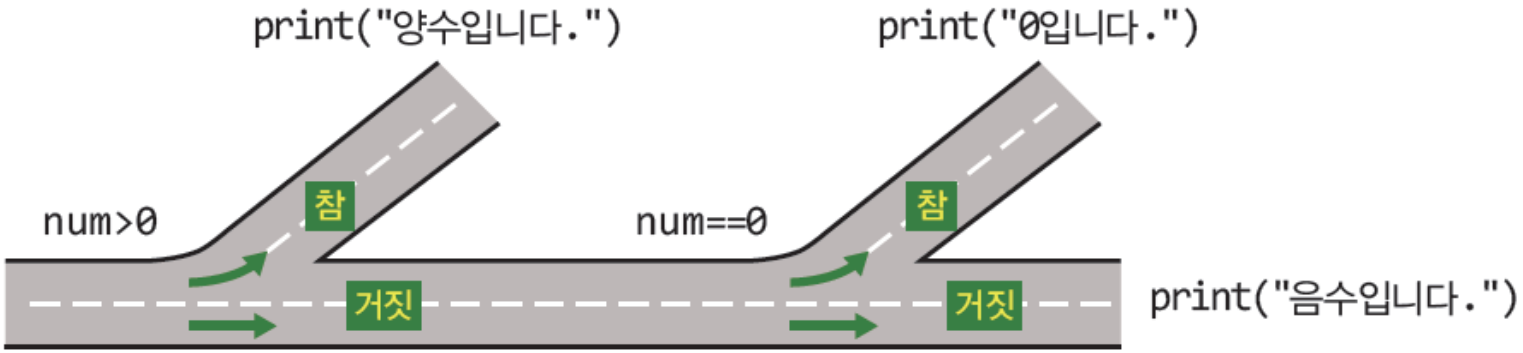
```
아이디: hong
알 수 없는 사용자입니다!
```

```
아이디: 김철수
패스워드를 입력하시오: 12345678
환영합니다.
```

Solution

연속적인 if-else 문 (if-elif-else 문)

- **종종 우리는 조건에 따라서 다중으로 분기되는 결정을 내려야 하는 경우가 있다.**
  - 중첩 if문은 조건이 많아지면 가독성이 떨어져 프로그램의 흐름을 이해하기 어려운 단점이 있다.
  - if ~ elif ~ else 문은 보다 직관적인 표현으로 프로그램을 작성할 수 있기 때문에 프로그램의 흐름을 이해하는데 보다 편리하다.



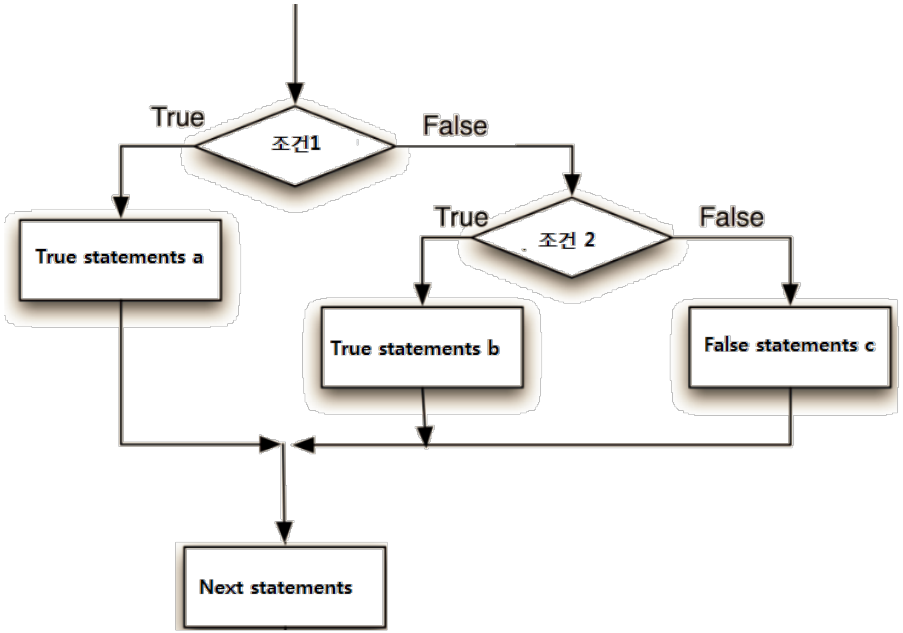
연속적인 if-else 문 (if-elif-else 문)

```
if 조건1 :
    True_statements_a
elif 조건2 :
    True_statements_b
else :
    False_statements
Next_statements
```

"조건1"이 True 이면 True_statements_a를 실행한 후, Next_statements를 실행

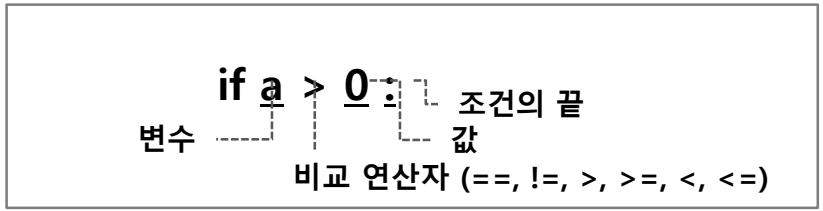
"조건1"이 False 이고 "조건2"가 True 이면 True_statements_b를 실행,

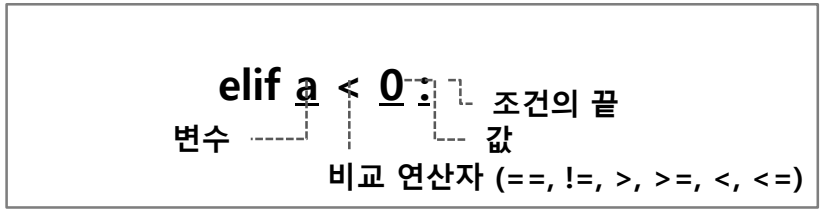
"조건1"이 False 이고 "조건2"도 False 이면 False_statements를 실행한 후 Next_statements를 실행

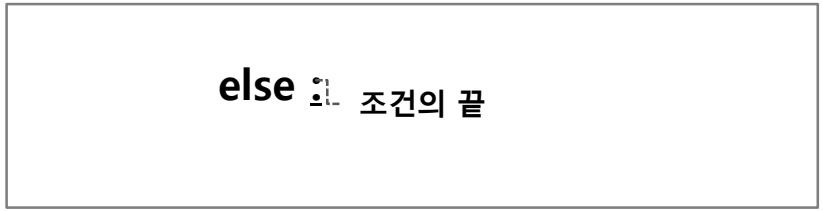


연속적인 if-else 문 (if-elif-else 문)

- if - else문의 header 구성







연속적인 if-else 문 (if-elif-else 문)

- 1~8까지 입력 받은 숫자를 판별하는 프로그램을 작성해보자 (비교)

```
num = int(input('숫자: '))
if num < 5 :
  if num < 3 :
    if num < 2 :
        print('1')
    else:
        print('2')
  else :
    if  num < 4 :
        print('3')
    else :
        print('4')

else :
  if num < 7 :
    if num < 6 :
        print('5')
    else :
        print('6')
  else :
    if num < 8 :
        print('7')
    else :
        print('8')
```

3번의 비교연산

조건의 수가 많아질 경우
프로그램 실행 시간이
→ 짧아진다

But 가독성이 낮다

```
num = int(input('숫자: '))

if num == 1 :
    print('1')

elif num == 2 :
    print('2')

elif num == 3 :
    print('3')

elif num == 4 :
    print('4')

elif num == 5 :
    print('5')

elif num == 6 :
    print('6')

elif num == 7 :
    print('7')

elif num == 8 :
    print('8')
```

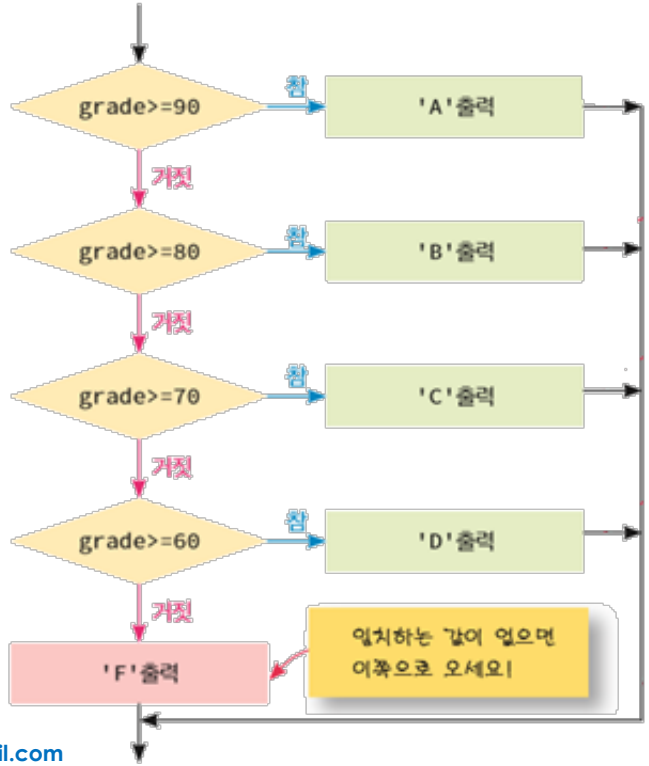
8번의 비교연산

조건의 수가 많아질 경우
프로그램 실행시간이
→ 길어진다

But 가독성이 높다

연속적인 if-else 문 (if-elif-else 문)

- 학생들의 성적을 받아서 학점을 출력하는 프로그램을 작성하여 실행하여 보자. 성적이 90점 이상이면 A 학점, 80점 이상이고 90점 미만이면 B학점, 70점 이상이고 80점 미만이면 C학점과 같이 결정하는 것이다.




```
score = int(input('Input Score: '))

if  score >= 90:
    grade="A"
elif  score >= 80:
    grade="B"
elif  score >= 70:
    grade="C"
elif  score >= 60:
    grade="D"
else:
    grade="F"

print("Grade is ", grade)
```

Lab: 음수, 0, 양수 구별하기

- 사용자로부터 정수를 받아서 음수, 0, 양수 중의 하나로 분류하여 보자.

```
정수를 입력하시오: -10
입력된 정수는 음수입니다.
```

```
정수를 입력하시오: 0
입력된 정수는 0입니다.
```

```
정수를 입력하시오: 10
입력된 정수는 양수입니다.
```

Solution : 파일명 pos_neg2.py

```
number=int(input("정수를 입력하시오: "))
if number<0:
    print("입력된 정수는 음수입니다.")
elif number ==0 :
    print("입력된 정수는 0입니다.")
else :
    print("입력된 정수는 양수입니다.")
```

```
num = int(input("정수를 입력하시오: "))
if num >= 0:
    if num == 0:
        print("0입니다.")
    else:
        print("양수입니다.")
else:
    print("음수입니다.")
```

중첩if문

Lab: 달의 일수 출력

- 1년의 각 달의 일수를 출력하는 프로그램을 작성하여보자. 즉 특정 달이 입력되면 그 달의 일수를 출력한다. 여러 가지 방법으로 작성할 수 있겠으나 여기서는 if-else 문을 사용하여 보자.

```
월을 입력하시오: 2
월의 날수는 29일
```

```
월을 입력하시오: 7
월의 날수는 31일
```

Solution : 파일명 days_of_month.py

```
month =int(input("월을 입력하시오: "))
if (month==2):
        print("월의 날수는 29일")
elif (month==4 or month==6 or month==10):
        print("월의 날수는 30일")
else :
        print("월의 날수는 31일")
```

Lab: 인사말 출력하기

- 시스템으로부터 현재 시각을 받아서 적절한 인사를 출력하는 프로그램을 작성하여 보자. 현재 시각에 따라서 연속적인 if문을 사용하여서 프로그램을 작성한다.
- 현재 시간이 11시 미만이면 'Good morning', 11시 이상 15시 미만이면 'Good afternoon', 15시 이상 20시 미만이면 'Good evening', 20시 이상이면 'Good night' 을 인사말로 출력한다.

```
현재시간은  Wed Apr 20 23:35:54 2020
Good night
```

```
현재시간은  Wed Apr 20 10:05:14 2020
Good morning
```

Lab: 인사말 출력하기 – time 모듈(참고: Python Documentation)

- **컴퓨터 시간 표현**
  - 타임스탬프(TimeStamp) : 1970년 1월1일 자정이후로 초 단위로 누적하여 측정한 절대시간
  - 협정세계시(UTC, Universal Time Coordinated): 1972부터 시행된 국제 표준시
  - 지방표준시(LST, Local Standard Time): UTC를 기준으로 경도 15도마다 1시간 차이가 발생하는 시간
    (한국은 동경 135도를 기준으로 UTC보다 9시간 빠름)
- **struct_time 시퀀스 객체**

속성	의미	속성	의미	속성	의미
tm_year	년(ex 2020)	tm_hour	시(0~23)	tm_wday	각 요일을 숫자로 표시(월요일 0 ~ 일요일 6)
tm_mon	월(1~12)	tm_min	분(0~59)	tm_yday	1월 1일부터 오늘까지 누적된 날짜를 반환 (1~366)
tm_mday	일(1~31)	tm_sec	초(0~61) , 61초: 윤초일때	tm_isdst	서머타임일 경우 1, 아닐 경우 0, 모를 경우 -1

- **time 모듈 함수**
  - time.time() : 현재까지의 타임스탬프를 실수형태로 반환
  - time.localtime([secs]) : 입력된 초를 LST기준 struct_time시퀀스 객체 반환, 초가 입력되지 않은 경우 time()을 이용하여 현재 시간 변환
  - time.gmtime([secs]) : 입력된 초를 UTC기준의 struct_time 시퀀스 객체로 반환, 초가 입력되지 않은 경우 time()을 이용하여 현재 시간 변환
  - time.asctime([t]) : struct_time 시퀀스 객체를 인자로 받아서 'Sun Mar 15 09:45:30 2020'과 같은 형태로 반환, struct_time 시퀀스 객체가 입력되지 않은 경우 localtime()을 이용하여 시간을 나타냄

Lab: 인사말 출력하기 – time 모듈

```
>>> import time
>>> time.time()  #1970년 1월 1일 0시 0분 0초 이후 경과한 시간을 초단위로 반환
1587365014.7389512
>>> time.gmtime() #UTC 기준의 현재 시간(타임스탬프를 struct_time형태로 변경)
time.struct_time(tm_year=2020, tm_mon=4, tm_mday=20, tm_hour=6, tm_min=43, tm_sec=34, tm_wday=0, tm_yday=111, tm_isdst=0)
>>> time.localtime()  #LST 기준의 현재시간(타임스탬프를 struct_time형태로 변경)
time.struct_time(tm_year=2020, tm_mon=4, tm_mday=20, tm_hour=15, tm_min=43, tm_sec=34, tm_wday=0, tm_yday=111, tm_isdst=0)
>>> time.localtime(1587365014)
time.struct_time(tm_year=2020, tm_mon=4, tm_mday=20, tm_hour=15, tm_min=43, tm_sec=34, tm_wday=0, tm_yday=111, tm_isdst=0)
>>> time.asctime()   # time.localtime() 을 알아보기 쉬운 형태로 변경
'Mon Apr 20 15:43:34 2020'
>>> time.asctime(time.gmtime())   # time.gmtime() 을 알아보기 쉬운 형태로 변경
'Mon Apr 20 06:43:34 2020'
>>> now = time.time()
>>> t = time.localtime(now)
>>> print('%s년 %s월 %s일'%(t.tm_year, t.tm_mon, t.tm_mday))  # print('%s년 %s월 %s일'%(t[0], t[1], t[2]))
2020년 4월 20일
>>> print(time.strftime('%Y-%m-%d %H:%M:%S', t))   # time.strtime(format, t) 함수를 사용하여 출력 형식 지정 가능
2020-04-20 15:43:38
```

Solution: 파일명 get_time.py

현재 시간이 11시 미만이면 'Good morning',
11시 이상 15시 미만이면 'Good afternoon',
15시 이상 20시 미만이면 'Good evening',
20시 이상이면 'Good night' 을 인사말로 출력

Lab: 이차 방정식

- 다음 이차 방정식의 근을 계산하는 프로그램을 작성하여 보자.

$$ax^2 + bx + c = 0 \quad \Rightarrow \quad \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Solution : 파일명 quad_eq.py (참고: math module)

$$ax^2 + bx + c = 0 \quad \Rightarrow \quad \frac{-b \pm \sqrt{b^2 - 4ac}}{2a}$$

Lab: 숫자 맞추기 게임

- 이 예제는 프로그램이 가지고 있는 정수를 사용자가 알아맞히는 게임이다. 사용자가 답을 제시하면 프로그램은 자신이 저장한 정수와 비교하여 제시된 정수가 더 높은지 낮은지 만을 알려준다. 정수의 범위를 1부터 100까지로 한정하도록 하자. 그리고 사용자는 단 한 번의 기회만 가진다.

```
숫자 게임에 오신 것을 환영합니다.
숫자를 맞춰 보세요: 9
너무 큼!
게임 종료
```

```
숫자 게임에 오신 것을 환영합니다.
숫자를 맞춰 보세요: 5
너무 작음!
게임 종료
```

Solution: 파일명 guess_number.py

Lab: 가위 바위 보

- 사용자가 가위, 바위, 보 중에서 하나를 선택하고 컴퓨터도 난수로 가위, 바위, 보 중에서 하나를 선택한다. 사용자의 선택과 컴퓨터의 선택을 비교하여서 승패를 화면에 출력한다.

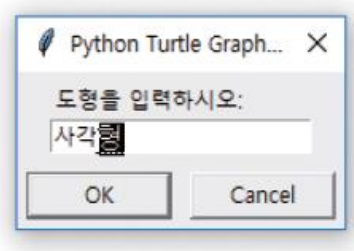
```
(가위, 바위, 보) 중에서 하나를 선택하세요: 가위
사용자:  가위 컴퓨터:  바위
컴퓨터가 이겼음!
```

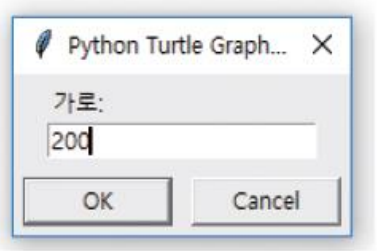
```
(가위, 바위, 보) 중에서 하나를 선택하세요: 보
사용자:  보 컴퓨터:  바위
사용자가 이겼음!
```

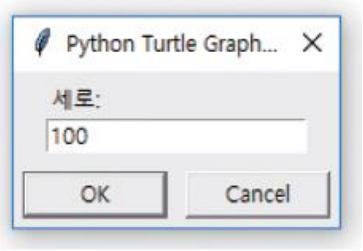
Solution: 파일명 rock_paper.py

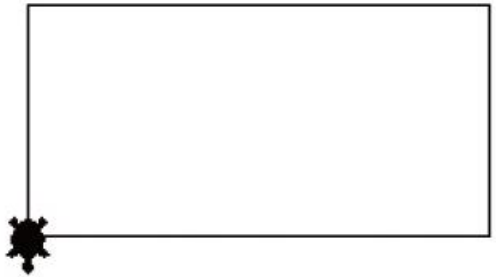
Lab 도형그리기

- **터틀 그래픽을 이용하여 사용자가 선택하는 도형을 화면에 그리는 프로그램을 작성해보자. 도형은 "사각형", "삼각형", "원" 중의 하나이다. 각 도형의 치수는 사용자에게 물어본 후 그림을 그린다.**
  - 사각형 ➔ 가로, 세로 입력받음
  - 삼각형 ➔ 한 변의 길이 입력받음
  - 원 ➔ 반지름 입력받음









Solution: 파일명 turtle_shape.py