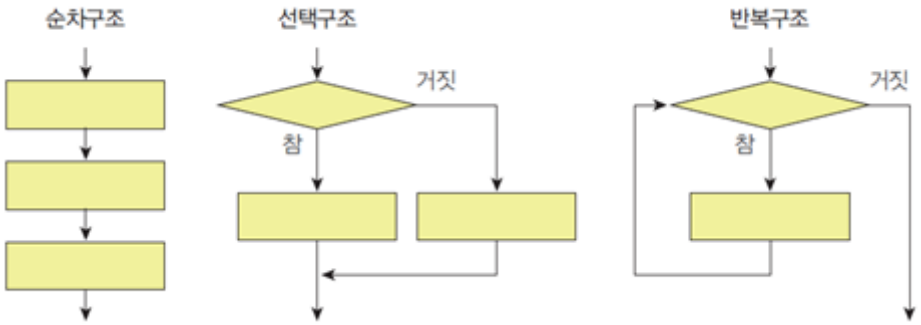
[기본 제어 구조 (3가지)]

- 순차 구조(sequence) - 명령들이 순차적으로 실행되는 구조

- 선택 구조(selection) - 둘 중의 하나의 명령을 선택하여 실행 구조 (if 문)

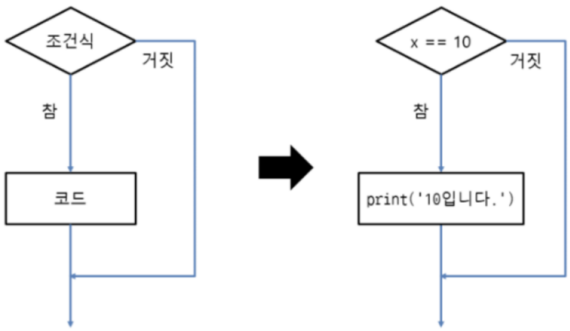
- 반복 구조(iteration) - 동일한 명령이 반복되면서 실행되는 구조

- **(지정된 범위)만큼 반복하는 for 문**
- **(조건)에 따라 반복하는 while 문**
- **반복문의 흐름 변경 : break, continue**



[if 조건문]

- 조건에 따라 코드 실행하거나 / 실행하지 않도록 만드는 구문
- 코드의 실행 흐름 변경 (조건 분기)



if 문에서 조건식이란 ?

- if (조건문: 참과 거짓을 판단하는 문장)

- 논리값이 True 일 때 실행 (논리식이 True)

In [ ]:
```
True
```

In [ ]:
```
2>1
```

In [ ]:
```
x = 3
y = 2
x > y, x < y, x == y, x != y
```

In [ ]:
```
print (x > 5 and x <= 20)
print (5 < x <= 20)  # 파이썬에서 더 직관적으로 표현 가능

print (x > 5 and y > 10)
print (x < 5 or y < 5 )
print (not x < 5)
```

- 두가지 이상 조건절 한번에 쓰기

놀이기구 입장조건 (나이가 10살 이상이고, 그리고 키가 140cm 이상)

In [ ]:
```
# 어린이 놀이 기구 탈 수 있는지 ?  ( 안전을 위한 수칙 : 나이10살 이상이고, 키 140 cm)
age = 13   # 나이
height = 135  # 키

age >=10 and height >= 140 # 안전수칙 조건
```

- 세 종류의 연산자가 섞여 있다면, 산술 --> 관계 --> 논리 연산자의 순서 계산

In [ ]:
```
a = 3; b = 5; c = 2
print( a + b > c and a * b == 15 )
```

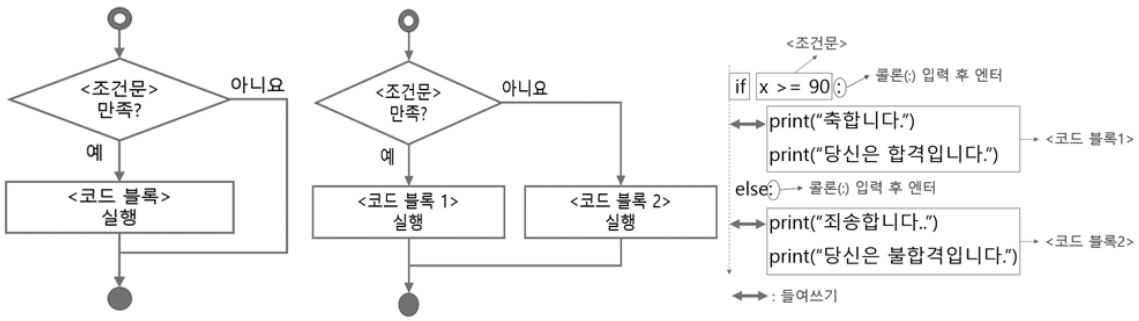
[if 조건문 종류]

1. 조건이 참인경우만 처리 (if ~)

2. 조건이 참인경우와, 거짓인 경우만 처리 (if~ else)

3. 여러 개의 조건에 따라 처리해야 하는 과정이 다른 경우(if~ elif~ else)

if / if ~ else 형태 기본 구조



< 파이썬 if, while, for, def, class 문 > (문장끝 콜론(:), 들여쓰기(indentation))

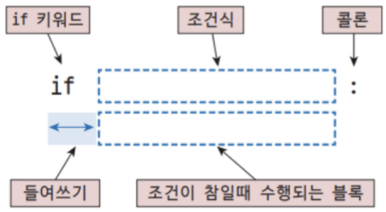
- 타 언어 { } 사용 (비교)

- 들여쓰기를 할 때 공백(Spacebar) 4개를 사용하는 것을 권장

- 파이썬이 타언어보다 보기 쉽고, 소스 코드 간결한 이유 : (콜론(:) 과 들여쓰기(indentation) 사용)

- else 뒤에는 조건문이 없다 (if문 조건식 거짓 -- 나머지 전부)

[if ~ 기본형태]



In [ ]:
```
x = 10

if x == 10:
    print('10입니다.')
```

In [ ]:
```
# 들여쓰기 예:
money = False
#money = True

if money:
    print("택시")

print("버스타고가기")
print("걸어가기")
```

In [ ]:
```
a = 10; b = 15; c = 20

if a < b:
    print('a is smaller than b')
#print('between if and else')  # 에러 발생 라인, if와 else의 연결을 끊습니다.

else: # if ~ else만 있는 경우에는 else는 반드시 if 블록이 끝나면 바로 나와야 합니다.
    print('a is larger than b')
```

In [ ]:
```
# 입력을 받습니다.

n = input("정수 입력> ")
number = int(n) # 문자열 --> 정수 변화

# 양수 조건
if number > 0:
    print("양수입니다")
```

```
# 음수 조건
if number < 0:
    print("음수입니다")

# 0 조건
if number == 0:
    print("0입니다")

# 첫번째 조건을 만족해서 수행했지만, 나머지 if문도 수행
```

[if ~ else 형태 기본 구조]

- if 조건문만 사용할 경우와 비교하면 불필요한 구분 절차 제거 (if ~ else)

- 조건에 해당하는 수행문만 실행 (한번만 실행)

In [ ]:
```
# [비교] if 문을 이용한 '오전' 혹은 '오후'의 출력기능
hour = 10
if hour < 12:
    print('오전입니다.')
if hour >= 12:
    print('오후입니다.')
```

In [ ]:
```
# [비교] if-else 문을 이용한 '오전' 혹은 '오후'의 출력 기능
hour = 10
if hour < 12:      # 배타적관계
    print('오전입니다.')
else:
    print('오후입니다.')
```

In [ ]:
```
## 입력을 받습니다.

n = input("정수 입력> ")
number = int(n)

# 짝수 조건
if number % 2 == 0:
    print("짝수입니다")

# 홀수 조건
if number % 2 != 0:
    print("홀수입니다")

# 짝수, 홀수 조건 각각 조건검사 --> 낭비
```

In [ ]:
```
# if 문과 비교해서 효율적 (if ~ else)

# 입력을 받습니다.
number = int(input("정수 입력> "))

# 조건문을 사용합니다.

if number % 2 == 0:
    # 짝수 조건
    print("짝수입니다")
else:
    # 짝수가 아닐 때의 조건 → 홀수 조건
```

```
    print("홀수입니다")
```

In [ ]:
```
# if 구조 : 점수 90보다 크거나 같으면 '합격'

x = int(input("정수 입력> "))
if x >= 90:
    print("축하합니다.")
    print("합격입니다.")
if x < 90:
    print("죄송합니다.")
    print("당신은 불합격 입니다")
```

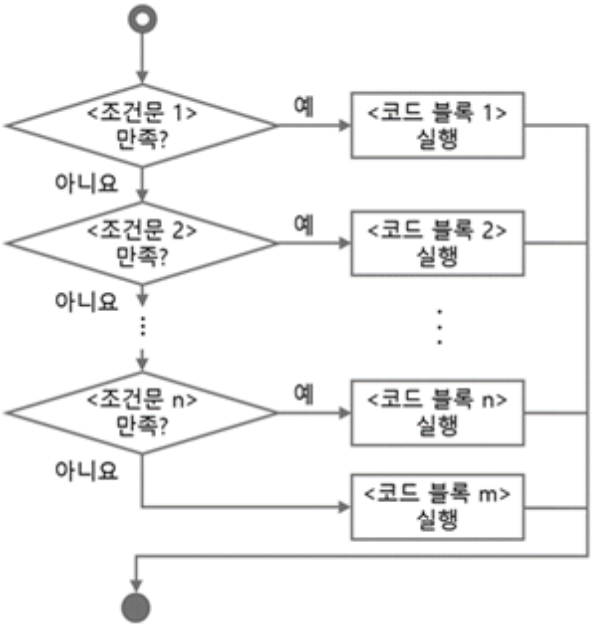
In [ ]:
```
#'if ~ else' 구조 : 점수 90 이상 '합격', 미만 불합격

x = int(input("정수 입력> "))
if x >= 90:
    print("축하합니다.")
    print("당신은 합격입니다.")
else:
    print("죄송합니다.")
    print("당신은 불합격 입니다.")
```

[if ~ elif ~ else 형태]

[중요] 여러 개의 조건식 중 [오직 하나의 참 조건식 실행] 할 때 사용

- 조건이 두개 이상의 그룹으로 나눌 수 있는 경우에 사용한다

- 한 조건문만 만족하게 조건문 만들어야 함



In [ ]:
```
### if 문 에러 코드 : 오류찾기

score = int(input('점수를 입력하세요 : '))

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('당신의 등급은 :', grade)
```

- 실행하면 모든 값이 'D'나 'F'로 나온다.

- 문제를 해결하기 위해서는 조건구간을 정확히 작성

- 또는 여러 개의 조건을 하나의 if문에서 검토할 수 있도록 elif를 사용한 if-elif-else문으로 작성

- elif는 else if의 줄임말로, if문과 같은 방법으로 조건문을 표현할 수 있다

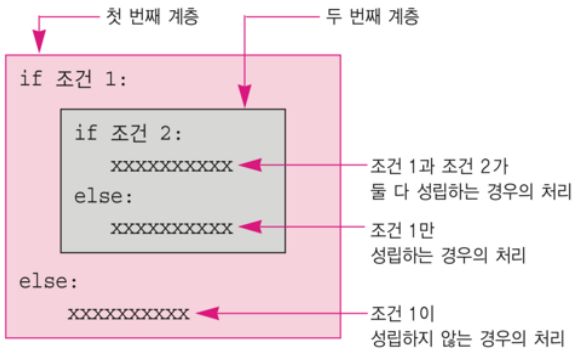
In [ ]:
```
# if문 만 사용해서 오류 수정
# 그러나 각각의 if문의 의미를 하나하나 파악해야하기 때문에 오류의 가능성이 높아짐

score = int(input('점수를 입력하세요 : '))

if score >= 90:    # 성적이 90점 이상인 경우
    grade = 'A'
# if score < 90 and score >= 80:    # score < 90 and score >= 80
if 80 <= score < 90 :    # 파이썬에서 가능
    grade = 'B'
# if score < 80 and score >= 70 :  # score < 80 and score >= 70
if 70 <= score < 80 :    # 파이썬에서 가능
    grade = 'C'
if 60 <= score < 70 :    # 파이썬에서 가능
    grade = 'D'
if score < 60:    # score < 60
    grade = 'F'

print('당신의 등급은 :', grade)
```

In [ ]:
```
# if elif else 을 사용해서 오류수정
# if 문만 사용해서 작성코드와 비교 장점
# 1. 코드 Simple  2. 윗 코드 if문 각각이 독립적이라서 전부 수행
# if-elif 는 조건만족한 것만 수행하고 빠져나와서 효율적

score = int(input('점수를 입력하세요 : ')) # 성적을 입력받습니다.

if score >= 90:       # 성적이 90점 이상인 경우
    grade = 'A'
elif score >= 80:     # score < 90 and score >= 80
    grade = 'B'
elif score >= 70:      # score < 80 and score >= 70
    grade = 'C'
elif score >= 60:      # score < 70 and score >= 60
    grade = 'D'
else:                 # score < 60
    grade = 'F'

print('당신의 등급은 :', grade)  # if 조건 만족 한번 수행후 print()
```

In [ ]:
```
## 조건문 실행문이 한줄이면 붙여서 사용 가능

score = int(input('점수를 입력하세요 : '))

if score >= 90: grade = 'A'          # 90 이상일 경우 A
elif score >= 80: grade = 'B'        # 80 이상일 경우 B
elif score >= 70: grade = 'C'        # 70 이상일 경우 C
elif score >= 60: grade = 'D'        # 60 이상일 경우 D
else: grade = 'F'                    # 모든 조건에 만족하지 못할 경우 F

print('당신의 등급은 :', grade)
```

[중첩조건문] : 중첩조건에 따른 분기

조건문 안에 또 다른 조건문을 사용한 구조



In [ ]:
```
# 중첩 조건문 : 조건문 안에 또 다른 조건문을 사용한 구조

#x = int(input('성적을 입력하세요 : '))
#x = 100
if x >= 90:
    if x==100 :
        print("Perfect")
    else:
        print("Very Good")
elif x >= 80:
    print("Good")
else:
    print("Bad")
```

비교

In [ ]:
```
# 3000원 이상의 돈있음 택시를 타고, 그렇지 않으면 걸어 가라

money = 2000
if money >= 3000:
    print("택시이용")
else:
    print("도보")
```

In [ ]:
```
# 3000원 이상 있거나 카드가 있다면 택시를 타고 그렇지 않으면 걸어 가라

money = 2000
```

```
card = True
if money >= 3000 or card:
    print("택시이용")
else:
    print("도보")
```

In [ ]:
```
# 돈이 있으면 택시를 타고, 돈은 없지만 카드가 있으면 택시를 타고,

money = 2000
card = True
if money > 2000:
    print("버스이용")
elif card:
    print("택시이용")
else:
    print("도보")
```

- 리스트에 특정 값이 있는지 체크하기

In [ ]:
```
# 만약 주머니에 돈이 있으면 택시를 타고, 없으면 걸기
pocket = ['paper', 'cellphone', 'money']
if 'money' in pocket:
    print("택시")
else:
    print("도보")
```

In [ ]:
```
pocket = ['paper', 'cellphone', 'money']
if 'money' in pocket:print("택시")
else:print("도보")
```

- 리스트에 특정 값이 없는지 체크하기

In [ ]:
```
# 만약 주머니에 돈이 있으면 택시를 타고, 없으면 걷기
pocket = ['paper', 'cellphone', 'money']
if 'money' in pocket:
    print("택시")
else:
    print("도보")
```

---------- End ----------