조건문

지금까지의 코드는 위에서부터 아래로 순차적으로 명령을 수행하는 프로그램을 작성하였다.

제어문(Control of Flow)은 크게 반복문과 조건문으로 나눌 수 있고, 이는 순서도(Flow chart로 표현이 가능하다.)

In [2]:

```
# 위의 flow chart로 표현하면 아래와 같다.
a = 5
if a > 5:
    print("5 초과")
else:
    print("5 이하")
print(a)
```

```
5 이하
5
```

조건문 문법

1. `if` 문은 반드시 일정한 참/거짓을 판단할 수 있는 `조건식`과 함께 사용이 되어야한다. `if <조건식>:`

2-1. `<조건식>`이 참인 경우 `:` 이후의 문장을 수행한다.

2-2. `<조건식>`이 거짓인 경우 `else:` 이후의 문장을 수행한다.

- 이때 반드시 **들여쓰기를** 유의해야한다. 파이썬에서는 코드 블록을 자바나 C언어의 `{}`와 달리 `들여쓰기`로 판단하기 때문이다.
- 앞으로 우리는 `PEP-8`에서 권장하는 `4spaces`를 사용할 것이다.

우리는 4spaces를 맞춰서 씁니다!

출처 : 400,000 GitHub repositories, 1 billion files, 14 terabytes of code: Spaces or Tabs?

실습문제1 - 조건문 기초 활용

> **조건문을 통해 변수 num의 값과 홀수/짝수 여부를 출력하세요.**

예시 출력)
3
홀수입니다.

In [4]:

```
# 실습!
num = int(input("점수를 입력하세요 : "))
# 아래에 코드를 작성하세요.
print(num)
if num % 2 != 0:
    print("홀수입니다.")
else :
    print("짝수입니다.")
```

```
점수를 입력하세요 : 32
32
짝수입니다.
```

복수 조건문

2개 이상의 조건문을 활용할 경우 `elif <조건식>:` 을 활용한다.

실습문제2 - 조건식 2개 이상 활용하기

> 조건문을 통해 변수 score에 따른 평점을 출력하세요.

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

예시 출력)

```
B
```

In [11]:

```
# 실습!
score = int(input("점수를 입력하세요 : "))
# 아래에 코드를 작성하세요.
if score >= 90:
    print("A")
elif score >= 80:
    print("B")
elif score >= 70:
    print("C")
elif score >= 60:
    print("D")
else:
    print("F")
```

```
점수를 입력하세요 : 97
A
```

실습문제3 - 중첩 조건문 활용

> **위의 실습문제 2코드를 활용하여 95점 이상이면, "참잘했어요"를 함께 출력해주세요**

예시 출력)

```
A
참잘했어요
```

In [12]:

```
# 실습!
score = 96
if score >= 90:
    print("A")
    if score >= 95:
        print("참잘했어요")
elif score >= 80:
```

```
    print("B")
elif score >= 70:
    print("C")
elif score >= 60:
    print("D")
else:
    print("F")
```

```
A
참잘했어요
```

조건 표현식(Conditional Expression)

활용법

```
true_value if <조건식> else false_value
```

와 같이 표현식을 작성할 수 있다. 이는 보통 다른 언어에서 활용되는 삼항연산자와 동일하다.

In [23]:

```
a = int(input("숫자를 입력하세요 : "))
print("3 맞아요") if a == 3 else print("3 아니에요")
```

```
숫자를 입력하세요 : 3
3 맞아요
```

- 표현식은 보통 조건에 따라 값을 정할 때 많이 활용된다.

In [31]:

```
num = int(input("숫자를 입력하세요 : "))
value = num if num >= 0 else 0
print(value)
```

```
숫자를 입력하세요 : 3
3
```

In [38]:

```
# 위의 코드와 동일한 코드입니다.
num = int(input("숫자를 입력하세요 : "))
if num >= 0:
    value = num
    print(value)
else:
    value = 0
    print(value)
```

```
숫자를 입력하세요 : -2
0
```

In [ ]:

```
# 다음의 코드와 동일한 조건 표현식을 작성해보세요.
num = 2
if num % 2:
    result = '홀수입니다.'
else:
    result = '짝수입니다.'
print(result)
```

In [43]:

```
# 여기에 코드를 작성하세요.
num = 2
```

```
result = '홀수입니다.' if num % 2 else '짝수입니다.'
print(result)
```

짝수입니다.

반복문

`while` 문

`while` 문은 조건식이 참(True)인 경우 반복적으로 코드를 실행합니다.

while 문은 종료조건을 반드시 설정해주어야 합니다.

In [45]:

```
# 위의 flow chart로 표현하면 아래와 같다.
a = 0
while a < 5:
    print(a)
    a += 1
print("끝")
```

```
0
1
2
3
4
끝
```

`while` 문 역시 `<조건식>` 이후에 `:` 이 반드시 필요하며,

이후 오는 코드 블록은 `4spaces` 로 **들여쓰기**를 해주셔야 합니다.

`for` 문

`for` 문은 정해진 범위 내(시퀀스)에서 순차적으로 코드를 실행합니다.

In [51]:

```
# 위의 flow chart로 표현하면 아래와 같다.
for i in range(5):
    print(i)
print("끝")
print(list(range(5)))
```

```
0
1
2
3
4
끝
[0, 1, 2, 3, 4]
```

```
for variable in sequence:
    code line1
    code line2
```

`for` 문은 `sequence` 를 순차적으로 **variable**에 값을 바인딩하며, 코드 블록을 시행합니다.

실습문제

> 반복문과 조건문만 활용하여 1~30까지 숫자 중에 홀수만 담긴 리스트를 만드세요

예시 출력)
```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29]
```

In [53]:

```
# 여기에 코드를 작성하세요.
list = []
for i in range(1, 31):
    if i % 2 == 1:
        list.append(i)
print(list)
```

```
[1, 3, 5, 7, 9, 11, 13, 15, 17, 19, 21, 23, 25, 27, 29]
```

index와 함께 `for` 문 활용하기

`enumerate()` 를 활용하면, 추가적인 변수를 활용할 수 있다.

In [55]:

```
lunch = ['짜장면', '초밥']
for idx, menu in enumerate(lunch, start=3):
    print(idx, menu)
```

```
3 짜장면
4 초밥
```

- `enumerate()` 는 파이썬 표준 라이브러리의 내장함수 중 하나이며, 다음과 같이 구성되어 있다.

In [2]:

```
classroom = ['Kim', 'Hong', 'Kang']
list(enumerate(classroom))
```

Out[2]:

```
[(0, 'Kim'), (1, 'Hong'), (2, 'Kang')]
```

In [ ]:

```

```

dictionary 반복문 활용하기

기본적으로 dictionary를 `for` 문을 시행시키면 다음과 같이 시행됩니다.

In [2]:

```
classroom = {"teacher": "Kim", "student1": "Hong", "student2": "Kang"}
for member in classroom:
    print(member)
```

```
teacher
student1
student2
```

dictionary의 `key`를 출력함으로써 `value`에도 접근할 수 있기 때문입니다.

따라서 dictionary의 value를 출력하기 위해서는 아래와 같이 작성합니다.

In [5]:

```
for member in classroom:
    print(classroom[member])
```

```
Kim
Hong
Kang
```

- dictionary에서 `for` 활용하는 4가지 방법

```
# 0. dictionary (key 반복)
for key in dict:
    print(key)

# 1. key 반복
for key in dict.keys():
    print(key)

# 2. value 반복
for val in dict.values():
    print(val)

# 3. key와 value 반복
for key, val in dict.items():
    print(key, val)
```

실습문제

> 한번 직접 4가지 반복문을 활용해보고 출력되는 결과를 확인해보세요.

```
classroom = {"teacher": "Kim", "student1": "Hong", "student2": "Kang"}
```

In [7]:

```
# 여기에 코드를 작성하세요.
classroom = {"teacher": "Kim", "student1": "Hong", "student2": "Kang"}
# 0. dictionary (key 반복)
for key in classroom:
    print(key)

# 1. key 반복
for key in classroom.keys():
    print(key)

# 2. value 반복
for val in classroom.values():
    print(val)

# 3. key와 value 반복
for key, val in classroom.items():
    print(key, val)
```

```
teacher
student1
student2
```

```
student2
teacher
student1
student2
Kim
Hong
Kang
teacher Kim
student1 Hong
student2 Kang
```

`break`, `continue`, `else` : 반복문 제어

1. `break` : (전체 루프문을 지나) 감싸고 있는 가장 근접한 루프의 바깥으로 이동한다.
2. `continue` : (루프의 헤더 라인으로) 감싸고 있는 가장 근접한 루프의 최상단으로 이동한다.
3. 루프 `else` 블록 : 루프가 정상적으로(즉, break 를 만나지 않고) 종료된 경우에만 실행한다.

In [ ]:

```
while test:
    statements
    if test:
        break           # 바로 루프를 빠져나감. else 문 생략.
        continue        # 바로 루프 상단에 있는 test 로 이동.
else:
    statements          # break 를 만나지 않고 루프를 종료할 경우 실행.
```

`break` : 반복문 끝내기

- `break` 문은 반복문을 종료하는 표현입니다.
- 반복문을 중단하고 빠져나옵니다.

In [8]:

```
# break 문을 활용해봅시다.
for i in range(10):
    if i != 0:
        break
    print(i)
```

```
0
```

실습문제

> 조건문과 반복문, break를 통해서 아래의 코드와 동일한 코드를 작성하세요.
>
> (3이 있을 경우 True를 print하고, 아닐 경우 False를 print 합니다.)

```
numbers = [1, 5, 10]
print(3 in numbers)
```

예시 출력)

```
False
```

In [3]:

```
numbers = [1, 5, 10]
# 여기에 코드를 작성하세요.
for idx, num in enumerate(numbers):
    if num == 3:
        print('True')
        break
    if idx+1 == len(numbers):
        print('False')
```

```
# check_num = False
# for num in numbers:
#     if num == 3:
#         check_num = True
#         break
# print(check_num)
```

```
False
```

`continue` : 코드실행 건너뛰기

- `continue` 문은 continue 이후의 코드를 수행하지 않고 다음 요소를 선택해 반복을 계속 수행합니다.
- break 와 약간 다른점은 continue 는 제어흐름(반복)을 유지한 상태에서 코드의 실행만 건너뛴다.

In [5]:

```
# continue 문을 활용해봅시다.
for i in range(6):
    if i % 2 == 0:
        continue
    print(f'{i}는 홀수다.')
```

```
1는 홀수다.
3는 홀수다.
5는 홀수다.
```

`else`

- `else` 문은 끝까지 반복문을 시행한 이후에 실행됩니다.
- `break` 를 통해 중간에 종료되지 않은 경우만 실행.

In [6]:

```
# break 시행 안됨 예시
for i in range(3):
    if i == 3:
        print(f'{i}에서 break 시행됨')
        break
else:
    print('break 시행 안됨')
```

```
break 시행 안됨
```

In [9]:

```
# break 시행 됨 예시
for i in range(3):
    if i == 0:
        print(f'{i}에서 break 시행됨')
        break
else:
    print('break 시행 안됨')
```

```
0에서 break 시행됨
```

In [18]:

```
# 앞선 실습문제를 else를 통해 개선해봅시다.
numbers = [1, 5, 10]
# 여기에 코드를 작성하세요.
for num in numbers:
    if num == 3:
        print('True')
        break
```

```
else:
    print('False')
# for idx, num in enumerate(numbers):
#     if num == 3:
#         print('True')
#         break
#     else:
#         print('False')
#         break
```

```
False
```