작성자 : 김기희 (cupid072@naver.com)

제어문(Statement)

if문 (조건문)

- if 문은 조건에 따라 수행할 문장이 선택되는 구조이다.

형식1)

```
if <조건>:
    문장
```

형식2)

```
if <조건>:
    문장1
else:
    문장2
```

형식3)

```
if <조건1>:
    문장1
elif <조건2>:
    문장2
elif <조건n>:
    문장n
```

형식4)

```
if <조건1>:
    문장1
elif <조건2>:
    문장2
elif <조건n>:
    문장n
else:
    문장m
```

예)

```
stock = 20
if stock > 0:
    pass                    # 블록 내에서 수행할 문장이 없는 경우
else:
    print("재고가 부족하여 판매가 불가능합니다.")
```

반복문(조건반복)

- while 문은 조건을 만족하는 동안 반복 수행한다.

- else 절은 while 루프를 종료한 후 수행할 블록이다. else 절도 while 구문에 포함되므로 break 문을 사용하여 while 루프를 탈출하는 경우 else 절의 문장은 수행되지 않는다.

- 형식)

```
while 조건:
    문장
[else:
    문장]
```

반복문(구간반복)

- for 문은 일정 범위를 반복적으로 수행한다.
- else 절은 for 루프를 종료한 후 수행할 블록이다. else 절도 for 구문에 포함되므로 break 문을 사용하여 for 루프를 탈출하는 경우 else 절의 문장은 수행되지 않는다.
- 형식1) 리스트, 튜플, 문자열에 대한 반복

```
for 변수 in 리스트|튜플|문자열:
        문장
[else:
        문장]
```

- 형식2) range() 함수에 의한 범위 지정 반복

```
for 변수 in range(start, end):
        문장
[else:
        문장]
```

range() 함수

- range() 함수는 연속된 숫자를 생성하는 제네레이터(Generator)를 생성하여 반환하는 함수이다.

- range() 함수의 사용법은 다음과 같다.

  ➢ range(stop) -> range(10) 0, 1, 2, 3, 4, 5, 6, 7, 8, 9
  0부터 stop-1 까지의 연속된 정수 값을 생성한다.

  ➢ range(start, stop) -> range(1, 11) 1, 2, 3, 4, 5, 6, 7, 8, 9, 10
  start부터 stop-1 까지의 연속된 정수 값을 생성한다.

  ➢ range(start, stop, step) -> range(1, 11, 2) 1 3 5 7 9
  start부터 stop-1 까지 step 간격의 숫자를 생성한다.

  ➢ start 값이 stop 값 보다 큰 경우 step 값은 음수이어야 한다.
  range(20, 0, -2) -> 20 18 16 14 12 10 8 6 4 2 (마지막 stop 값은 포함되지 않는다.)

분기문

- continue문은 반복문 내에서 사용되며 continue 문을 포함하는 가장 안쪽의 반복문의 조건식으로 제어가 이동한다.

- break문은 반복문 내에서 사용되며 break문을 포함하는 가장 안쪽의 반복문의 실행일 종료한다.

```
a = 0
while True:
    a = a + 1
    if (a % 2) == 0:
        continue                    # while 문의 조건식 True로 제어가 이동
    if (a >= 10):
        break                       # while 문의 루프를 종료한다.
    print("{} ".format(a))
```

for 문에서의 분기문

- for 문에서 continue 문의 반복 대상이 리스트나 튜플 인 경우 다음 원소를 추출한 후 평가한다.
- for 문에서 continue 문의 반복 대상이 문자열인 경우 다음 문자열을 추출한 후 평가한다.
- for 문에서 continue 문의 반복 대상이 값의 범위라면 다음 값을 추출한 후 평가한다.

- for 문에서 break 문을 만나면 for 루프를 종료한다.

```
numbers = [ 9, 2, 4, 1, 8, 6, 3, 7, 5, 0 ]
for num in numbers:
    if num != 0:
        print(num)
        continue                    # 반복 작업 대상의 다음 값을 추출로 제어가 이동한다.
    break                           # for 루프를 종료한다.
```

중첩된 제어문

- 중첩된 제어문이란? 제어문 블록 내에 또 다른 제어문이 기술되는 형식을 말한다.

- 중첩된 제어문을 사용하여 구구단 출력하기

```
for op1 in range(2, 10):
    print(f"----- {op1}단 -----")

    for op2 in range(2, 10):
        result = op1 * op2
        print(f"{op1} X {op2} = {result}")
```

실행결과

```
----- 2단 -----
2 X 2 = 4
2 X 3 = 6
2 X 4 = 8
2 X 5 = 10
2 X 6 = 12
2 X 7 = 14
2 X 8 = 16
2 X 9 = 18
----- 3단 -----
3 X 2 = 6
.....
```

pass 문

- pass 문은 공문(Empty Statement)를 기술하고자 할 때 사용한다.
- pass는 문법적으로 문장이 필요하지만 프로그램이 특별히 할 일이 없을 때 사용한다.

```
while True:                          # 무한 반복 실행
    pass

class MyEmptyClass:                  # 멤버를 가지지 않는 클래스 정의
    pass

def initlog(*args):                  # 아무런 동작을 하지 않는 함수의 정의
    pass
```

match 문

- match 문은 식의 결과와 일치하는 case로 분기하며 형식은 다음과 같다.

```
match 식:
        case 값1:                   # match 식의 값이 값1 이라면
            문장1                   # 문장1을 수행
        case 값2:                   # match 식의 값이 값2 라면
            문장2                   # 문장2를 수행
        case _:                     # 위의 모든 case 값에 일치하지 않으면
            문장n                   # 문장n을 수행
```

- case의 값은 | (Bitwise OR) 연산자를 사용하여 여러 개를 지정할 수 있다.

```
값1 | 값2 | 값n      : 값1, 값2, 값n 중에 하나와 일치하면
```

튜플(tuple)에 대한 match 문

- match 문의 식에는 튜플(tuple)이 기술될 수 있다.

```
point = (10, 20)

match point:
    case (10, 10):
        print('정사각형')
    case (20, 20):
        print('정사각형')
    case (10, 20) | (20, 10):
        print('직사각형')
    case _:
        print('오류')
```

클래스 인스턴스를 식으로 하는 match 문

```
class Point:
    def __init__(self, x, y):
        self.x = x
        self.y = y

point = Point(10, 20)

match point:
    case Point(x = 10, y = 10) | Point(x = 20, y = 20):
        print('정사각형')
    case Point(x = 10, y = 20) | Point(x = 20, y = 10):
        print('직사각형')
    case _:
        print('오류')
```

match문의 식과 비교할 인스턴스 생성을 위한 생성자 함수 호출 시 반드시 이름을 지정하여 인자 값을 전달해야 한다.