

단기 집중교육

2일차 실습



Computational Thinking



연세대학교
YONSEI MIRAE
CAMPUS



제어문, 반복문

YONSEI MIRAE CAMPUS

1. 점수가 90점 이상이면 첫번째 결과가, 90점 미만이면 두번째 결과가 나오도록 조건문을 작성하시오.

1) 축하합니다.
당신은 합격입니다.

2) 죄송합니다.
당신은 불합격입니다.

2. 점수가 90점 이상이면 A, 80점 이상이면 B, 70점 이상이면 C학점을 출력하는 조건문을 작성하시오.

예시) 학점 : C

3. 점수가 100점이면 만점으로 합격, 90점 이상이면 합격, 그 외의 경우는 불합격이 나오도록 조건문을 작성하시오.

◆ for문

- 구조: for < 반복 변수 > in < 반복 범위 >:
 < 코드 블록 >

- for index, value in enumerate(list_data)
 < 코드블록 >

-> 리스트 데이터의 요소와 인덱스 값을 출력함

예시)

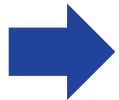
```
list_num = [10, 20, 30, 40]

for index, value in enumerate(list_num):
    print(index, value)
```

```
0 10
1 20
2 30
3 40
```

- ◆ 아래의 두 개의 리스트를 반복문을 통해 결과와 같이 출력하는 코드를 작성하시오.

```
names = ["동백", "용식", "자영", "규태", "종렬", "향미"]  
scores = [96, 85, 100, 70, 80, 75]
```



```
동백 96  
용식 85  
자영 100  
규태 70  
종렬 80  
향미 75
```

HINT!! **for var1, var2 in zip(list1, list2):**
 < 코드 블록 >
 -> list1, 2의 요소가 순서대로 동시에 적용됨

- ◆ 아래의 두 개의 리스트를 반복문을 통해 결과와 같이 출력하는 코드를 작성하시오.

```
names = ["동백", "용식", "자영", "규태", "종렬", "향미"]  
scores = [96, 85, 100, 70, 80, 75]
```

```
for k in range(len(names)):  
    print(names[k], scores[k])
```

동백 96
용식 85
자영 100
규태 70
종렬 80
향미 75

```
for name, score in zip(names, scores):  
    print(name, score)
```

동백 96
용식 85
자영 100
규태 70
종렬 80
향미 75

◆ while문

- 구조: while < 조건 >:
 < 코드 블록 >
- Q. 빈 리스트를 만들고, while문을 사용하여 0부터 50까지 2씩 증가하는 리스트를 만드시오.

```
list_num = []
```

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36,  
38, 40, 42, 44, 46, 48, 50]
```

◆ while문

- Q. 빈 리스트를 만들고, while문을 사용하여 0부터 50까지 2씩 증가하는 리스트를 만드시오.

```
list_num = []    # 빈 리스트 생성
count = 0        # count를 0으로 초기화

while (count <= 50):    # <조건> count가 조건과 일치하는지 검사
    list_num.append(count) # <코드 블록> list_num에 count 추가
    count = count + 2     # <코드 블록> count를 2씩 증가


print(list_num) # 리스트 list_num의 내용을 출력
```

```
[0, 2, 4, 6, 8, 10, 12, 14, 16, 18, 20, 22, 24, 26, 28, 30, 32, 34, 36,
38, 40, 42, 44, 46, 48, 50]
```


◆ while문

- Q. num리스트를 만들고 num리스트의 요소를 하나씩 더하여 출력하고, 합계가 10이상인 경우 "while문을 끝냅니다"라는 문구를 출력하고 종료하는 코드를 작성하시오.

num = [1, 2, 3, 4, 5, 6] ->리스트

 1
3
6
10
while 문을 끝냅니다.

◆ while문

```
num = [1, 2, 3, 4, 5, 6]
num_sum = 0 # 숫자의 합계를 0으로 초기화
count = 0   # count를 0으로 초기화

while True:
    num_sum = num_sum + num[count] # 리스트 num의 요소를 하나씩 더함
    print(num_sum)
    if (num_sum >= 10): # 합계(num_sum)가 10 이상인지 검사
        print("while 문을 끝냅니다.")
        break # while 문을 끝냄

    count = count + 1 # count를 1씩 증가
```

```
1
3
6
10
while 문을 끝냅니다.
```

◆ 한 줄 반복문

[<반복 실행문> for <반복 변수> in <반복 범위>]

```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] # 리스트 생성  
  
# 리스트의 각 요소에 2*x+1 연산을 수행해서 새로운 리스트 생성  
result = [2*x+1 for x in numbers]  
  
print(result) # 생성한 리스트 출력
```

[1, 3, 5, 7, 9, 11, 13, 15, 17, 19]

[<반복 실행문> for <반복 변수> in <반복 범위> if <조건>]

```
numbers = [0, 1, 2, 3, 4, 5, 6, 7, 8, 9] # 리스트 생성  
  
# x >= 3 조건을 만족할 때만 2*x+1 연산을 수행  
result = [2*x+1 for x in numbers if x>=3]  
  
print(result) # 생성한 리스트 출력
```

[7, 9, 11, 13, 15, 17, 19]



함수, 클래스, 모듈

YONSEI MIRAE CAMPUS

◆ 함수 정의

```
def 함수명([매개변수1, 매개변수2, ..., 매개변수n]):
```

```
    < 코드 블록 >
```

```
    [ return <반환값> ]
```

◆ 예제

```
=====  이모티콘을 출력하는 함수 my_emoticon()을 정의하시오.
(^o^)
=====
```

◆ 함수 정의

def 함수명([매개변수1, 매개변수2, ..., 매개변수n]):

< 코드 블록 >

[return <반환값>]

◆ 예제

=====
(^o^) 이모티콘을 출력하는 함수 my_emoticon()을 정의하시오.
=====

```
# 함수의 정의 (이모티콘 출력)
def my_emoticon():
    print("=====")
    print(" (^o^)")
    print("=====")

# 함수의 호출
my_emoticon()
```

◆ 매개변수에 기본값 할당

def 함수명 (매개변수1=기본값1, 매개변수2=기본값2, ...):

< 코드 블록 >

[return <반환 값>]

◆ 예시

```
def my_add(a=1, b=2, c=3):  
    y = a + b + c  
    print("{0} + {1} + {2} = {3}".format(a, b, c, y))
```

◆ 자료형 변환 함수

- int()
- float()
- str()
- list() : 튜플/세트 -> 리스트
- Tuple() : 리스트/세트 -> 튜플
- Set() : 리스트/튜플 -> 세트

```
print("튜플/세트 -> 리스트로 변환:", list((1,2,3)), list({1,2,3}))  
print("리스트/세트 -> 튜플로 변환:", tuple([1,2,3]), tuple({1,2,3}))  
print("리스트/튜플 -> 세트로 변환:", set([1,2,3]), set((1,2,3)))
```

튜플/세트 -> 리스트로 변환: [1, 2, 3] [1, 2, 3]
리스트/세트 -> 튜플로 변환: (1, 2, 3) (1, 2, 3)
리스트/튜플 -> 세트로 변환: {1, 2, 3} {1, 2, 3}

◆ 자료형 변환 함수

- `int()`
- `float()`
- `str()`
- `list()` : 튜플/세트 -> 리스트
- `Tuple()` : 리스트/세트 -> 튜플
- `Set()` : 리스트/튜플 -> 세트

◆ 최솟값, 최댓값, 합계 구하는 함수

- `min()`
- `max()`
- `sum()`

◆ 클래스: 객체를 만들기 위한 기본틀

◆ 객체: 클래스로 만들어진 결과물

◆ 클래스 선언

```
class Robot():  
    def __init__(self, name, position):    # 초기화 함수  
        self.name = name # 인스턴스 변수(로봇 객체의 이름) 초기화  
        self.position = position          # 인스턴스 변수(로봇 객체의 초기 위치) 초기화  
  
    def move(self):    # 앞으로 한 칸 이동을 위한 함수  
        self.position = self.position + 1 # 이전 위치에서 앞으로 한 칸 이동  
        print(f"{self.name}의 현재 위치: {self.position}")
```

◆ 클래스에서 객체 생성

```
robot1 = Robot('R1', 0) # 클래스에서 객체 생성
```

```
print(f"로봇의 이름: {robot1.name}, 초기 위치: {robot1.position}")
```

로봇의 이름: R1, 초기 위치: 0

◆ 메서드 호출

```
robot1.move() # 객체의 메서드 move 호출
```

R1의 현재 위치: 1

```
robot2 = Robot('R2', 10) # 클래스에서 객체 생성  
# 객체의 속성에 접근해 로봇의 이름과 초기 위치 출력  
print(f"로봇의 이름: {robot2.name}, 초기 위치: {robot2.position}")
```

```
robot2.move() # 객체의 메서드 move 호출(호출할 때마다 한 칸씩 이동)  
robot2.move() # 객체의 메서드 move 호출(호출할 때마다 한 칸씩 이동)
```

로봇의 이름: R2, 초기 위치: 10

R2의 현재 위치: 11

R2의 현재 위치: 12

◆ 모듈 만들기

```
%%writefile C:\myPyExcel\modules\calc_area.py
# File name: calc_area.py
PI = 3.14
def rectangle(l, w): # 직사각형(가로: l, 세로: w)의 넓이를 반환
    return l * w

def circle(r): # 원(반지름: r)의 넓이를 반환
    return PI * r ** 2
```

Writing C:\myPyExcel\modules\calc_area.py

```
%%writefile C:\myPyExcel\modules\car.py
# File name: car.py
class Car(): # 클래스 선언
    def __init__(self, size, color):
        self.size = size # 인스턴스 변수 생성 및 초기화
        self.color = color # 인스턴스 변수 생성 및 초기화

    def move(self):
        print("자동차({0} & {1})가 움직입니다.".format(self.size, self.color))
```

Writing C:\myPyExcel\modules\car.py

◆ 모듈 불러오기

```
cd C:\myPyExcel\modules
```

```
C:\myPyExcel\modules
```

```
import calc_area # 모듈 импорт

pi = calc_area.PI # 임포트한 모듈의 변수를 사용
rect = calc_area.rectangle(5, 2) # 임포트한 모듈의 함수를 호출
circ = calc_area.circle(3) # 임포트한 모듈의 함수를 호출

print(f"원주율:{pi}, 직사각형 넓이: {rect}, 원의 넓이: {circ}")
```

원주율:3.14, 직사각형 넓이: 10, 원의 넓이: 28.26

```
import car # 모듈 импорт

my_car = car.Car("중형", "검은색") # 임포트한 모듈의 클래스에서 객체를 생성
my_car.move() # 객체의 메서드를 호출
```

자동차(중형 & 검은색)가 움직입니다.

- ◆ import 모듈명
-> 사용법: 모듈명.함수
- ◆ from 모듈명 import 변수명/함수명/클래스명
-> 모듈명 없이 사용 가능
- ◆ from 모듈명 import *
-> 모듈의 모든 변수, 함수, 클래스 이용(모듈명 없이 바로 사용)
- ◆ import 모듈명 as 별명
-> 모듈명 대신 별명 이용



Question and Answer

YONSEI MIRAE CAMPUS