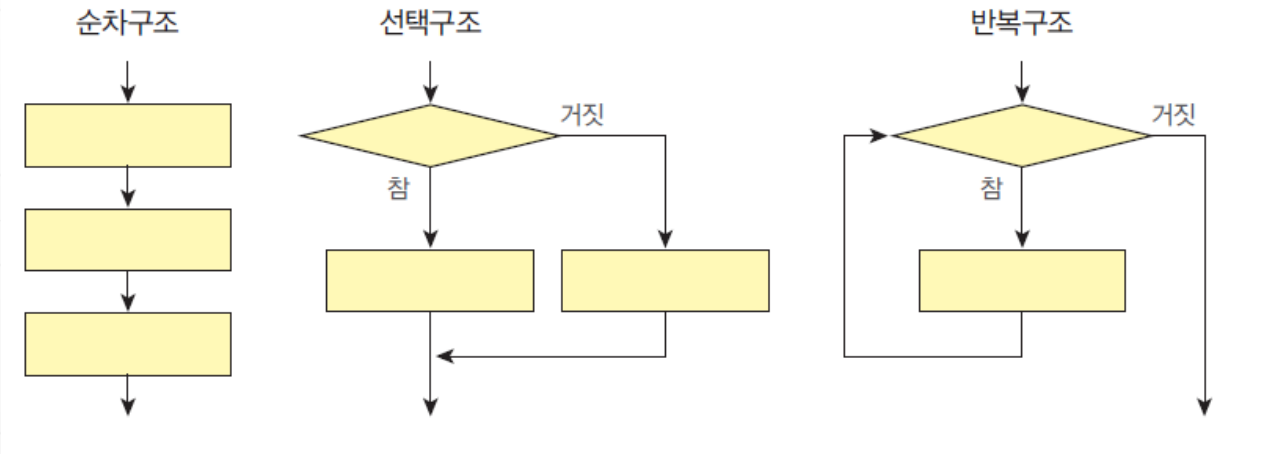
컴퓨터정보과

파이선 프로그래밍

05주차 – 제어문

제어문

- 프로그램 코드의 실행 방법
  - 순차구조 : 차례대로 실행
  - 선택(조건,분기) 구조 : **if**
    - 조건을 검사하여 여러 개의 실행 경로 중에서 하나를 선택
  - 반복 구조 : for / while
    - 조건이 만족될 때까지 반복
  - 기타 제어문 : continue / break / **pass**



if

- 조건문 if p.121
  - 실행 상황을 판단하고 그 판단 결과에 따라 상황을 수행하는 것을 돕는 문장
  - 기본구조

    조건식이 True로 판단되면 수행문장 블록(1~n)실행 False이면서 아래로 JUMP
    만약 else가 있는 경우 False는 else로 JUMP하여 수행문장 블록(m~z) 실행

```
if 조건식 :
    수행문장1
    ...
    수행문장n
```

```
if 조건식 :
    수행문장1
    ...
    수행문장n
else :
    수행문장m
    ...
    수행문장z
```

- 블록은 **들여쓰기(Indentation)**로 구분 (p.123)
  - 들여쓰기가 잘못되면 수행되지 않는다.
  - 추천 : 한 Step 당 공백문자(space) 4개

```
if 조건식 :
    수행문장1
    ...
    수행문장n
else :
    수행문장m
    ...
    수행문장z
```

if

- 조건식(문)이란 p.125
  - 참(True) 거짓(False)를 판단하는 식(문)
  - 조건식에 많이 사용하는 연산자 (결과 bool 타입)
    - 비교(관계) 연산자 p.126
      - <, >, ==, !=, >= , <=
    - 논리 연산자 p.127
      - **and, or, not**
    - 멤버십 연산자 p.128
      - 특정 값이 sequence나 collection에 포함되어 있는지 여부를 확인할 때 사용
      - **in, not in**
        - **‘a’ in ‘abc’**
        - **‘d’ not in ‘abc’**

elif

- 다양한 조건을 판단하는 elif p.130
  - C언어의 else if 구조
  - 위의 if의 조건식의 결과가 False인 경우 다른 조건식을 검사할 때 사용한다.
  - 연속으로 elif 사용 가능하며
  - 마지막에 else 또한 사용 가능하다.

```
if 조건식1 :
    수행문장1
    ...
if 조건식2 :
    수행문장2
    …
if 조건식3 :
    수행문장3
    …
else :
    수행문장4
```

```
if 조건식1 :
    수행문장1
    ...
elif 조건식2 :
    수행문장2
    …
elif 조건식3 :
    수행문장3
    …
else :
    수행문장4
```

조건부 표현식

- Conditioanl expression (일명: Ternary operator) p.132
  - 일종의 파이썬 삼항 연산자

```
if score >= 60:
    message = "succ"
else:
    message = "fail"
```

```
message = "succ"  if score >= 60 else "fail"
```

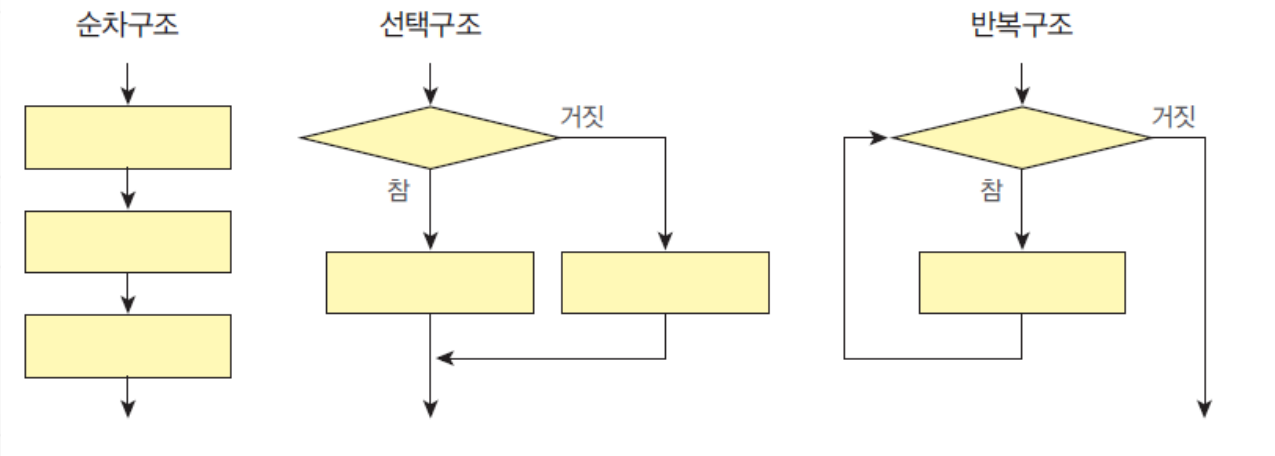
pass

- pass p.129
  - 빈 블록이 필요한 경우
  - 아무런 실행을 하지 않는다.
  - C언어에서는 {} 나 ;을 통해서 빈 문장, 빈 블록을 표시할 수 있지만 들여쓰기를 사용하는 파이썬에는 사용이 불가능하다.
    이를 대체해서 사용할 때 유용

```
if a < 10 :
    pass
else :
    print(a)
```

제어문

- 프로그램 코드의 실행 방법
  - 순차구조 : 차례대로 실행
  - 선택(조건,분기) 구조 : if
    - 조건을 검사하여 여러 개의 실행 경로 중에서 하나를 선택
  - 반복 구조 : **for / while**
    - 조건이 만족될 때까지 반복
  - 기타 제어 : **continue / break**/ pass



while

- 반복문 while p.133
  - 문장을 반복 수행할 경우 사용한다.
  - 조건식이 참(True)인 경우 코드 블록을 계속 수행한다.
- 기본구조

```
while 조건식 :
    수행문장1
    ...
    수행문장n
```

  - 조건식이 True이면 수행문장 1~n 실행하고, 다시 조건식 검사를 수행한다. 조건식이 False이면 while 루프 아래로 Jump
    - 블록은 들여쓰기로 구분

```
treeHit = 0
while treeHit < 10:
    treeHit = treeHit + 1
    print(f"나무를 {treeHit}번 찍었습니다.")
    if treeHit == 10:
        print("나무가 드디어 넘어갑니다.")
```

초기(시작)값
종료조건
증감식

```
나무를 1번 찍었습니다.
나무를 2번 찍었습니다.
나무를 3번 찍었습니다.
나무를 4번 찍었습니다.
나무를 5번 찍었습니다.
나무를 6번 찍었습니다.
나무를 7번 찍었습니다.
나무를 8번 찍었습니다.
나무를 9번 찍었습니다.
나무를 10번 찍었습니다.
나무가 드디어 넘어갑니다.
```

break

- break p.136 / p.143 + 무한 루프 p.139
  - 강제로 반복문(for 포함) 빠져나가기

```
print("이름 입력하세요.(단, quit 입력시 종료)")
while True:
    name = input("이름:").strip().lower()

    if name == 'quit':
        break
    else :
        print(name)
```

- 무한 루프 p.139

continue

- continue p.138
  - 강제로 반복문(for 포함)의 조건식 검사로 돌아가기

```
print("1~10사이의 숫자중 홀수의 합:", end="")
count = 0
sum = 0
while count < 10:
    count += 1

    if count % 2 == 0:
        continue

    sum += count

print(sum)
```

```
1~10사이의 숫자중 홀수의 합:25
>>>
```

for

- 반복문 for p.141
  - (순차형태인)군집형의 첫번째 요소부터 마지막 요소까지 차례로 변수에 대입해 반복적으로 문장을 수행할 경우 사용한다.
    (리스트와 같은 형태를 순차적으로 조회하여 작업할 필요가 있을 때 유용)
- 기본구조

(★)의 항목들을 하나씩 가리키는 임시 변수

(★)인덱스를 통해 다룰 수 있는 자료형
(리스트, 문자열, 튜플, ...)

```
1    numbers = ['one', 'two', 'three']
2    for number in numbers:
3        print(number.title())
```

들여쓰기 중요

반복시마다 수행할 문장 또는 루프 블록

```
One
Two
Three
```

- while과 for의 차이점?

```
1  numbers = ['one', 'two', 'three']
```

```
2  for number in numbers:
3      print(number.title())
```

```
One
Two
Three
```

for 활용

- 5명 학생 점수를 입력 받아, 평균을 구해보자 (1)

```
1   scores = [] #list()
2
3   data = input("2번:")
4   data = int(data)
5   scores.append(data)
6
7   scores.append(int(input("3번:")))
8   scores.append(int(input("4번:")))
9   scores.append(int(input("5번:")))
10  scores.insert(0, (int(input("1번:"))))
11
12  print("1번 학생 점수:", scores[0])
13  print("2번 학생 점수:", scores[1])
14  print("3번 학생 점수:", scores[2])
15  print("4번 학생 점수:", scores[3])
16  print("5번 학생 점수:", scores[4])
17
18  summary = scores[0] + scores[1] + scores[2] + scores[3] + scores[4]
19  avg = summary / len(scores)
20
21  print("총 평균:", avg)
```

```
2번:25
3번:67
4번:45
5번:80
1번:90
1번 학생 점수: 90
2번 학생 점수: 25
3번 학생 점수: 67
4번 학생 점수: 45
5번 학생 점수: 80
총 평균: 61.4
```

반복의 패턴이 보이는가!!!

for 활용

- 5명 학생 점수를 입력 받아, 평균을 구해보자 (2)

```
1  scores = [] #list()
2
3  data = input("2번:")
4  data = int(data)
5  scores.append(data)
6
7  scores.append(int(input("3번:")))
8  scores.append(int(input("4번:")))
9  scores.append(int(input("5번:")))
10 scores.insert(0, (int(input("1번:"))))
11
12 number = 0
13 summary = 0
14 for score in scores :
15     number += 1 # number = number + 1
16     summary += score # summary = summary + score
17     print(f"{number}번 학생 점수:", score)
18
19 avg = summary / len(scores)
20 print("총 평균:", avg)
```

```
2번:25
3번:67
4번:45
5번:80
1번:90
1번 학생 점수: 90
2번 학생 점수: 25
3번 학생 점수: 67
4번 학생 점수: 45
5번 학생 점수: 80
총 평균: 61.4
```

집계 함수 이용 법

```
avg = sum(scores)/len(scores)
max = max(scores)
min = min(scores)
```

range() 함수

- range() : range 타입의 연속된 숫자를 생성 (읽기전용) p.144

```
1   for value in range(10):
2       print(value, end="/")
3   print()
4
5   for value in range(1, 10):
6       print(value, end="/")
7   print()
8
9   for value in range(1, 10, 2):
10      print(value, end="/")
11  print()
```

```
0/1/2/3/4/5/6/7/8/9/
1/2/3/4/5/6/7/8/9/
1/3/5/7/9/
```

- range()로 리스트 만들기

```
1   a = range(1,10)
2   b = list(a)
3   print(a)
4   print(b)
5   b[0] = 11
6   a[0] = 11 #여기서 에러남
```

```
range(1, 10)
[1, 2, 3, 4, 5, 6, 7, 8, 9]
Traceback (most recent call last):
  File "E:/Python1/Week2/test8.py", line 6, in <module>
    a[0] = 11
TypeError: 'range' object does not support item assignment
```

숫자형 리스트 + for 루프 활용

- 5명 학생 점수를 입력 받아, 평균을 구해보자 (3)

```python
scores = [] #list()
for score in range(1,6) :
    scores.append(int(input(f"{score}번:")))

number = 0
for score in scores :
    number += 1 # number = number + 1
    print(f"{number}번 학생 점수:", score)

avg = sum(scores)/len(scores)
max = max(scores)
min = min(scores)
print("총 평균:", avg)
print("최고 점수:", max)
print("최저 점수:", min)
```

```python
scores = [] #list()
for score in range(1,6):
    scores.append(int(input(f"{score}번:")))

for idx in range(len(scores)):
    print(f"{idx+1}번 학생 점수:", scores[idx])

avg = sum(scores)/len(scores)
max = max(scores)
min = min(scores)
print("총 평균:", avg)
print("최고 점수:", max)
print("최저 점수:", min)
```

```
1번:10
2번:20
3번:30
4번:40
5번:50
1번 학생 점수: 10
2번 학생 점수: 20
3번 학생 점수: 30
4번 학생 점수: 40
5번 학생 점수: 50
총 평균: 30.0
최고 점수: 50
최저 점수: 10
```

리스트 내포

- **List Comprehension** p.147
  - for와 새 항목 생성을 한 행에 결합하여, 각 새 항목을 자동으로 리스트에 추가
    - 형식 : [표현식 for 항목 in 반복가능객체 if 조건문]

```
1   values = [1, 2, 3, 4, 5, 6, 7]
2
3   result1 = []
4   for value in values:
5       result1.append(value*3)
6
7   result2 = [value * 3 for value in values]
8
9   result3 = [value * 3 for value in values if value % 2 == 0]
10
11  print(result1)
12  print(result2)
13  print(result3)
```

```
[3, 6, 9, 12, 15, 18, 21]
[3, 6, 9, 12, 15, 18, 21]
[6, 12, 18]
```