3. 연산 및 흐름제어

연산

☐ 수치 연산

```
In [1]: 2+3 #더하기
Out[1]: 5

In [2]: 3-2 #빼기
Out[2]: 1

In [3]: 2*3 #곱하기
Out[3]: 6

In [4]: 3.14**2 #3.14의 2승
Out[4]: 9.8596

In [5]: 2**0.5 #2의 0.5승
Out[5]: 1.4142135623730951

In [6]: 5/2 #나누기
Out[6]: 2.5

In [7]: 5//2  #나눴을 때 몫
Out[7]: 2

In [8]: 5 % 2 #나눴을 때 나머지
Out[8]: 1
```

☐ 대입 연산

- 대입 연산자
  - x +=y 는 x = x + y와 같음

연산자	기능
+=	왼쪽의 변수에 오른쪽에 있는 값을 더한다.
-=	왼쪽의 변수에 오른쪽에 있는 값을 뺀다.
*=	왼쪽의 변수에 오른쪽에 있는 값을 곱한다.
/=	왼쪽의 변수를 오른쪽에 있는 값으로 나눈다.
%=	왼쪽의 변수에 오른쪽으로 나눈값으로 나눈 나머지를 대입한다.

```
In [9]: #대입연산

In [19]: a = 6

In [20]: b = 4

In [21]: a + b
Out[21]: 10

In [22]: a - b
Out[22]: 2

In [23]: a * b
Out[23]: 24

In [24]: a ** b #a의 b승
Out[24]: 1296

In [25]: a / b
Out[25]: 1.5

In [26]: a // b
Out[26]: 1

In [27]: a % b
Out[27]: 2
```

```
In [55]: a=20

In [56]: a += 5 # a=a+(5), a더하기5를 다시 a에 입력

In [57]: print(a)
25

In [58]: a -= 10 # a=a-(10), a빼기10을 다시 a에 입력

In [59]: print(a)
15

In [60]: a *= 2 #a = a*(2)

In [61]: print(a)
30

In [62]: a /= 3 #a=a/(3)

In [63]: print(a)
10.0

In [68]: a = 3

In [69]: b = 2

In [70]: a *= 2+b #a = a*(2+b)

In [71]: print(a)
12

In [72]: a *= b-5 #a = a*(b-5)

In [73]: print(a)
```

☐ 비교 연산

☐ 논리연산자

```
In [95]: a = True #참
In [96]: b = False #거짓
In [97]: #and 연산
In [104]: a and b #and 둘다 True일 때만 True
Out[104]: False
In [105]: a and a #and 둘다 True이므로 True
Out[105]: True
In [106]: b and b #and 둘다 False이므로 False
Out[106]: False
In [107]: # or연산
In [109]: a or b #or 둘중 하나만 True여도 True
Out[109]: True
In [110]: a or a #or 둘다 Ture여서 True
Out[110]: True
```

```
In [111]: b or b #or 둘다 False이므로 False
Out[111]: False
In [112]: # not 연산
In [113]: not a #반대되는 논리 연산
Out[113]: False
In [114]: not b
Out[114]: True
In [115]: not (a and b) #False의 반대이므로 True
Out[115]: True
```

☐ 구성원 연산자

```
In [116]: #구성원 연산자
In [117]: #in, not in을 사용해서 참,거짓 구분
In [119]: a = [1,2,'Hello Data']
In [120]: 1 in a #a에 1이 있으므로 참
Out[120]: True
In [121]: 1 not in a #a에 1이 있으므로 거짓
Out[121]: False
In [122]: 3 in a #a에 3이 없으므로 거짓
Out[122]: False
In [123]: 3 not in a #a에 3이 없으므로 참
Out[123]: True
In [124]: b = {'a':1,'b':2,'c':3}
In [125]: 'a' in b #b에 문자 a가 있으므로 참
Out[125]: True
In [126]: 'k' in b #b에 문자 k가 없으므로 거짓
Out[126]: False
In [127]: 1 in b #b에 key에 1이 없으므로 거짓(key값에만 있음)
Out[127]: False
```

☐ 연산자 활용

In [156]:
```
a = [
    {'num':'1','name':'김철수','kor':90,'eng':80,'math':85,'total':0,'avg':0.0},
    {'num':'2','name':'박민주','kor':90,'eng':85,'math':90,'total':0,'avg':0.0},
    {'num':'3','name':'박영호','kor':80,'eng':80,'math':80,'total':0,'avg':0.0},
]
```

In [157]:
```
a[0] #첫번째 딕셔너리 출력
```

Out[157]:
```
{'num': '1',
 'name': '김철수',
 'kor': 90,
 'eng': 80,
 'math': 85,
 'total': 0,
 'avg': 0.0}
```

In [158]:
```
a[0]['total'] #첫번째 딕셔너리에 total의 key값 출력
```

Out[158]:
```
0
```

In [159]:
```
a[0]['total'] = 1 #첫번째 딕셔너리에 total의 key값에 1을 입력
```

In [160]:
```
a[0]
```

Out[160]:
```
{'num': '1',
 'name': '김철수',
 'kor': 90,
 'eng': 80,
 'math': 85,
 'total': 1,
 'avg': 0.0}
```

In [161]:
```
#학생별 합계 구하기
a[0]['total'] = a[0]['kor']+a[0]['eng']+a[0]['math'] #total값 계산
a[1]['total'] = a[1]['kor']+a[1]['eng']+a[1]['math']
a[2]['total'] = a[2]['kor']+a[2]['eng']+a[2]['math']
```

In [162]:
```
a[0]['total']
```

Out[162]:
```
255
```

In [163]:
```
a[1]['total']
```

Out[163]:
```
265
```

In [164]:
```
a[2]['total']
```

Out[164]:
```
240
```

In [165]:
```
#학생별 평균 구하기
a[0]['avg']=a[0]['total']/3
a[1]['avg']=a[1]['total']/3
a[2]['avg']=a[2]['total']/3
```

In [166]:
```
print(a)
```

```
[{'num': '1', 'name': '김철수', 'kor': 90, 'eng': 80, 'math': 85, 'total': 255, 'avg': 85.0}, {'num': '2', 'name': '박민주', 'kor': 90, 'eng': 85, 'math': 90, 'total': 265, 'avg': 88.33333333333333}, {'num': '3', 'name': '박영호', 'kor': 80, 'eng': 80, 'math': 80, 'total': 240, 'avg': 80.0}]
```

In [167]:
```
#학급평균 구하기
b = (a[0]['total']+a[1]['total']+a[2]['total'])/3
print(b)
```

```
253.33333333333334
```

조건문

☐ if문

- if문은 "if조건식:..." 의 형태

- **만약 조건이 성립한다(TRUE)면...을 후반부를 실행**한다
  - 아래 식, x 가 10보다 작다면, a는 1을 입력
  - a를 확인해보니, 1이 입력되어 있음

```
In [8]: x = 5

In [9]: if x <10:
            a = 1

In [10]: a
Out[10]: 1
```

☐ **if-else문**

- **조건이 하나**일때 : **if-else문**의 구조
  - 파이썬에서 코드가 간결한 이유는 **:(콜론)을 사용**하여 자동으로 들여쓰기 하도록 만들었기 때문
  - R에서는 if문을 중괄호{ }로 감싸지만 파이썬은 들여쓰기로 해결

Python	R
- if-else · if (조건): (조건이 TURE일 때 실행될) 문장 또는 명령어 else: (조건이 FALSE일 때 실행될) 문장 또는 명령어	- if-else · if (조건) { (조건이 TURE일 때 실행될) 문장 또는 명령어 } else { (조건이 FALSE일 때 실행될) 문장 또는 명령어 }

In [1]:
```
#if문의 구조
```

In [5]:
```
if True:
    print('참입니다')
else:
    print('거짓입니다')
```
참입니다

In [6]:
```
if False:
    print('참입니다')
else:
    print('거짓입니다')
```
거짓입니다

In [7]:
```
a = 1000
```

In [9]:
```
if a > 5000:
    print('택시를 탑니다')
else:
    print('버스를 타야합니다')
```
버스를 타야합니다

In [10]:
```
if a <= 5000:
    print('택시를 탑니다')
else:
    print('버스를 타야합니다')
```
택시를 탑니다

In [11]:
```
if a > 5000:
print('택시를 탑니다') #들여쓰기 필요
else:
    print('버스를 타야합니다')
```
```
  File "<ipython-input-11-74aff16d5367>", line 2
    print('택시를 탑니다')
    ^
IndentationError: expected an indented block
```

In [12]:
```
if a > 5000:
    print('택시를 탑니다')
    else: #else 위치 앞으로
    print('버스를 타야합니다')
```
```
  File "<ipython-input-12-c39f89e01d63>", line 3
    else: #else 위치 앞으로
    ^
SyntaxError: invalid syntax
```

- Python과 R비교

Python	R

```
In [13]: x = 10

In [14]: if x>0:
             a = 1 #x>o면 a=1,b=1의값 할당
             b = 1
         else:
             a = 0 #x<=o면 a=0,b=0의값 할당
             b = 0

In [15]: a
Out[15]: 1

In [16]: b
Out[16]: 1
```

```
> #if-else 구분
> x <- 10
> if(x>0) {
+   a <- 1 #x>0 이면 a에 1을 할당
+   b <- 1
+ } else {
+   a <- 0 #x>0 이면 a에 1을 할당
+   b <- 0
+ }
> a
[1] 1
> b
[1] 1
```

```
In [29]: #input함수: 사용자가 어떤 값을 입력하게 하고, 그 값을 변수에 저장할 수 있음
         #float함수: 실수형으로 바꿔줌(소수점), int함수: 정수로 바꿔줌, str함수: 텍스트형식

In [30]: grade = float(input('총 평점을 입력해 주세요:'))
총 평점을 입력해 주세요:4.3

In [31]: if grade >= 4.3:
             print('당신은 장학금 수여 대상자입니다')
             print('축하합니다')
당신은 장학금 수여 대상자입니다
축하합니다

In [32]: data = int(input('숫자를 입력하시오:'))
숫자를 입력하시오:5

In [33]: if data % 2 == 0: #숫자를 2로 나눠었을때, 나머지가 없으면 짝수 그렇지 않으면 홀수
             print('입력된 값은 짝수입니다.')
         else:
             print('입력된 값은 홀수 입니다.')
입력된 값은 홀수 입니다.
```

☐ if-elif-else문

- 조건이 여러개 일때 : **if-elif-else**

Python	R
- if-else · if (조건1): (조건1 일 때 실행될) 문장 또는 명령어 elif (조건2): (조건1이 아니고 조건2일 때 실행될) 문장 또는 명령어 else: (조건1이 아니고 조건2도 아닐 때) 문장 또는 명령어	- if-else · if (조건1) { (조건1 일 때 실행될) 문장 또는 명령어 } else if { (조건1이 아니고 조건2일 때 실행될) 문장 또는 명령어 } else { (조건1이 아니고 조건2도 아닐 때) 문장 또는 명령어 }

Python	R
In [5]: ``` x=5 ``` In [6]: ``` if x < 10: a = 1 elif 10 <= x and x < 20: a =2 else: a =3 ``` In [7]: ``` a ``` Out [7]: 1	``` > #if-else if > x <- 5 > if (x < 10) { + a <- 1 + } else if(x >= 10 & x <20) { + a <- 2 + } else { + a <- 3 + } > a [1] 1 ```

☐ 중첩된 if 문

- if문 안에 또 if 문이 들어갈 수 있음

```
In [46]: #중첩된 if문

In [47]: age = int(input('나이를 입력하세요:'))
나이를 입력하세요:30

In [48]: height = int(input('키를 입력하세요:'))
키를 입력하세요:170

In [50]: if age >= 40: #나이가 40이상
             if height >= 170:
                 print('키가 보통 이상입니다')
             else:
                 print('키가 보통입니다')
         else: #나이가 40미만
             if height >= 175:
                 print('키가 보통 이상입니다')
             else:
                 print('키가 보통입니다')
키가 보통입니다
```

☐ 간략한 if-else문

- if문 뒤에 수행할 문장이 한 줄이고, else문 뒤에 수행할 문장이 한 줄일 때 간략하게 코드 작성 가능
- **(참일 때 실행문) if (조건문) else (거짓일 때 실행문)**

```
In [58]: score = 80

In [60]: if score >= 70:
             a = "good"
         else:
             a = "bad"

In [61]: a
Out[61]: 'good'

In [63]: a = "good" if score >= 70 else "bad" #위의 문장아 이와 같이 간단히 가능
```

반복문(for, while, break)

☐ for문

Python	R
· for 반복변수 in 리스트(튜플,문자열) 실행 문장	· for (반복변수 in data) { 반복변수가 들어간 문장 }

In [64]:
```
#for 문
```

In [65]:
```
for a in [1,2,3]: #list의 값이 차례로 입력
    print(a)
```

```
1
2
3
```

In [68]:
```
for a in [(1,3),{5,7},9]: #list 안에 튜플(1,3), 두번째 값 집합{5,7}, 숫자 9 가 차례로 출력
    print(a)
```

```
(1, 3)
{5, 7}
9
```

In [69]:
```
(a,b) = (1,2)
```

In [70]:
```
print(a)
```

```
1
```

In [71]:
```
print(b)
```

```
2
```

In [72]:
```
print(a,b)
```

```
1 2
```

In [73]:
```
for (a,b) in [(1,2),(3,4),(5,6)]: #리스트의 튜플 (1,2), (3,4), (
    print(a+b)
```

```
3
7
11
```

In [74]:
```
a = 'Hello!'
```

In [75]:
```
b = ['Hello!', 'Data'] #리스트
```

In [77]:
```
c=(1,2,3) #튜플
```

In [81]:
```
for x in a: #첫번째 문자인 H부터 !까지 순차적으로 출력
    print(x)
```

```
H
e
l
l
o
!
```

In [82]:
```
for x in a: #첫번째 문자인 H부터 !까지 순차적으로 출력
    print(x, end = "")
```

```
Hello!
```

In [87]:
```
for x in b:
    print(x)
```

```
Hello!
Data
```

In [88]:
```
for x in c: #튜플
    print(x)
```

```
1
2
3
```

In [89]:
```
d = {'circle':2,'rectangle':3,'line':1} #딕셔너리
```

In [90]:
```
e = {'red','green','blue','red'} #집합
```

In [92]:
```
for x in d: #딕셔너리는 key만 반환
    print(x)
```

```
circle
rectangle
line
```

In [94]:
```
for x in e: #집합자료형은 순서가 없으며 중복을 허용하지 않음. 순차적으로 입력되지 않음.
    print(x)
```

```
blue
red
green
```

- Python과 R비교
  - **range함수**
    - range는 range(시작숫자(이상), 종료숫자(미만), 간격)의 형태
    - range의 결과는 시작숫자부터 **종료숫자 바로 앞 숫자**까지 컬렉션을 만듦
    - 시작숫자와 step은 생략가능

Python	R
In [98]: `list(range(3,31,3)) #range(시작하는수, 끝나는수+1,간격)` Out[98]: `[3, 6, 9, 12, 15, 18, 21, 24, 27, 30]` In [104]: `result = 0` In [105]: `for i in list(range(1,11,2)):` `  result = result + 3` `  print(result)` 3 6 9 12 15	`> result <- 0` `> for (i in seq(1,10,by=2)) {` `+  result = result + 3` `+ }` `> result` `[1] 15`

- 리스트 내포 사용하기
  - **[표현식 for 항목 in 반복가능객체 if 조건문]**
  - 리스트 안에 for문을 포함하는 리스트 내포(List comprehension)를 사용하면 좀 더 편리하고 직관적인 프로그램을 만들 수 있음

```
In [167]: lst = [1,2,3,4,5]

In [168]: result = [ ]

In [169]: for num in lst:
              result.append(num*3) #append함수써서 하나씩 추가
          print(result)
          [3, 6, 9, 12, 15]

In [170]: lst = [1,2,3,4,5]

In [ ]:   #한줄로 간편하게 입력 방법

In [171]: result = [num*3 for num in lst]

In [172]: print(result)
          [3, 6, 9, 12, 15]

In [174]: result = [num*3 for num in lst if num%2 == 0] #조건문 추가 짝수인것 찾기

In [175]: print(result)
          [6, 12]
```

☐ 이중 for 문

- 이중 if 문과 마찬가지로, for문 안에 for문에 들어갈 수 있음

Python

```
In [129]: result = [ ] #괄호안에 띄워야함

In [130]: for i in list(range(1,6)):
              for j in list(range(1,11)):
                  k = i*j
                  result_temp = [i,j,k]
                  result.append(result_temp)

In [131]: print(result)

[[1, 1, 1], [1, 2, 2], [1, 3, 3], [1, 4, 4], [1, 5, 5],
3, 6], [2, 4, 8], [2, 5, 10], [2, 6, 12], [2, 7, 14],
5, 15], [3, 6, 18], [3, 7, 21], [3, 8, 24], [3, 9, 27],
[4, 7, 28], [4, 8, 32], [4, 9, 36], [4, 10, 40], [5, 1
40], [5, 9, 45], [5, 10, 50]]
```

R

```
> #중첩된 for문, 단계별 결과 누적 저장
> result <- data.frame()
> for (i in 1:5) {
+   for (j in 1:10) {
+     k <- i*j
+     result_temp <- c(i,j,k)
+     result <- rbind(result, result_temp)
+   }
+ }
> head(result)
  X1L X1L.1 X1L.2
1   1     1     1
2   1     2     2
3   1     3     3
4   1     4     4
5   1     5     5
6   1     6     6
```

- 이중 for문을 활용한 구구단
  - 매개변수 end를 넣어 준 이유는 해당 결과값을 출력할 때 다음 줄로 넘기지 않고 그 줄에 계속해서 출력하기 위해서임
  - 주의) print문의 위치에 따라서 다른 결과값 출력

In [164]:
```
for a in range(2,10):
    for b in range(1,10):
        print(a*b, end=" ") #아래 for문이 끝날 때 까지 이어서 결과 출력
    print('')
```

```
2 4 6 8 10 12 14 16 18
3 6 9 12 15 18 21 24 27
4 8 12 16 20 24 28 32 36
5 10 15 20 25 30 35 40 45
6 12 18 24 30 36 42 48 54
7 14 21 28 35 42 49 56 63
8 16 24 32 40 48 56 64 72
9 18 27 36 45 54 63 72 81
```

In [165]:
```
for a in range(2,10):
    for b in range(1,10):
        print(a*b)
    print('')
```

```
2
4
6
8
10
12
14
16
18

3
6
9
```

In [166]:
```
for a in range(2,10):
    for b in range(1,10):
        print(a*b,end=" ")
```

```
2 4 6 8 10 12 14 16 18 3 6 9 12 15 18 21 24 27 4 8 12 16 20 24 2
2 49 56 63 8 16 24 32 40 48 56 64 72 9 18 27 36 45 54 63 72 81
```

☐ for문 안에 조건문(if-else)

- 데이터 분석시에 반복문(for)과 조건문(if)이 함께 활용되는 경우가 많음

```
In [135]: #for-if-else

In [141]: Score=[100,7,75,32]

In [142]: num = 0

In [143]: for a in Score:
              num = num + 1
              if a >= 70:
                  print('%d번 합격'%num)
              else:
                  print('%d번 불합격'%num)

1번 합격
2번 불합격
3번 합격
4번 불합격
```

☐ While문

- **조건문이 참(TRUE)인 동안 while의 실행문이 실행**됨
  - 주의) **조건문이 계속 참이면 무한 반복**됨

Python	R
· while 조건문: 조건이 참일 때 수행할 문장 증가구문	· while (조건) { 조건이 참일 때 수행할 문장 증가구문 }

Python	R
In [182]: i = 1 In [183]: while i < 10: result = 2*i print(result) i = i + 1 2 4 6 8 10 12 14 16 18	> while (i < 10) { + result = 2*i + print(result) + i = i +1 + } [1] 2 [1] 4 [1] 6 [1] 8 [1] 10 [1] 12 [1] 14 [1] 16 [1] 18

```
In [184]: a = 1

In [185]: b = 0

In [187]: while a <= 100:
              b = b + a
              a = a + 1
          print('1부터100까지의 합은:',b)

          1부터100까지의 합은: 5050
```

☐ break 문

- 반복문(while, for)을 끝내기 위한 목적으로 사용
- **특정 조건을 만족하면 반복문 강제 종료**

```
In [191]: a = 1

In [192]: b = 0

In [193]: while a <= 100:
              b = b + a
              a = a + 1
              if b > 3000:
                  break

          print('1부터100까지의 합은:',b)

          1부터100까지의 합은: 3003
```

Python	R
In [194]: result = [ ] In [195]: for i in range(1,6): for j in range(1,11): k = i+j result_temp = [i,j,k] if k>10: break result.append(result_temp) In [196]: print(result) [[1, 1, 1], [1, 2, 2], [1, 3, 3], [1, 4, 4], [1, 5, 5], [1, 6, 3, 6], [2, 4, 8], [2, 5, 10], [3, 1, 3], [3, 2, 6], [3, 3, 9], 6], [1, 7, 7], [1, 8, 8], [1, 9, 9], [1, 10, 10], [2, 1, 2], [2, 2, 4], [2, [4, 1, 4], [4, 2, 8], [5, 1, 5], [5, 2, 10]]	> #중첩된 for문, 단계별 결과 누적 저장 > result <- data.frame() > for (i in 1:5) { + for (j in 1:10) { + k <- i*j + result_temp <- c(i,j,k) + if (k>10) { #k가 10보다 크면 변수 j에 대한 f + break + } + result <- rbind(result, result_temp) + } + } > result X1L X1L.1 X1L.2 1 1 1 1 2 1 2 2 3 1 3 3 4 1 4 4 5 1 5 5

- continue()함수
  - 반복문 실행시 continue()함수 아래에 있는 해당 step을 실행하지 않고 건너뜀

In [144]:
```
#Continue
```

In [145]:
```
Score=[100,7,75,32]
```

In [146]:
```
num = 0
```

In [147]:
```
for a in Score:
    num = num + 1
    if a < 70:
        continue
    else:
        print('%d번 합격'%num)
```

```
1번 합격
3번 합격
```