데이터기반 프로그래밍(2)

신봉균 20191624

2023-03-22

---

01 조건문

코드 4-3

```
score = 85
if(score>90){
  grade = 'A'
} else if (score >80) {
  grade = 'B'
} else if (score >70) {
  grade = 'C'
} else if (score >60) {
  grade = 'D'
} else if (score >50) {
  grade = 'F'
}
print(grade)
```

```
## [1] "B"
```

---

02 반복문

코드 4-8

```
for (i in 1:9){
  cat('2 *', i,'=',2*i,'\n')
}
```

```
## 2 * 1 = 2
## 2 * 2 = 4
## 2 * 3 = 6
## 2 * 4 = 8
## 2 * 5 = 10
## 2 * 6 = 12
## 2 * 7 = 14
```

```
## 2 * 8 = 16
## 2 * 9 = 18
```

코드 4-8 에서는 구구단을 출력하기 위해 **print()** 함수 대신에 **cat()** 함수를 사용하였다. **print()** 함수는 하나의 값을 출력할 때 사용하고,**cat()** 함수는 한 줄에 여러 개의 값을 결합하여 출력할 때 사용한다.

코드 4-11

```
norow = nrow(iris)              #iris 의 행의 수수
mylabel = c()                   #비어있는 벡터 선언
for(i in 1:norow){
  if (iris$Petal.Length[i] <= 1.6){   #꽃잎의 길이에 따라 레이블 결정정
    mylabel[i] = 'L'
  } else if(iris$Petal.Length[i] >= 5.1){
    mylabel[i]= 'H'
  } else {
    mylabel[i] = 'M'
  }
}
print(mylabel)                                  #레이플 출력
```

```
##   [1] "L" "L" "L" "L" "L" "M" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L"
"L"
##  [19] "M" "L" "M" "L" "L" "M" "M" "L" "L" "L" "L" "L" "L" "L" "L" "L" "L"
"L"
##  [37] "L" "L" "L" "L" "L" "L" "L" "L" "M" "L" "L" "L" "L" "L" "M" "M" "M"
"M"
##  [55] "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M"
"M"
##  [73] "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "H" "M" "M" "M" "M" "M"
"M"
##  [91] "M" "M" "M" "M" "M" "M" "M" "M" "M" "M" "H" "H" "H" "H" "H" "H" "M"
"H"
## [109] "H" "H" "H" "H" "H" "M" "H" "H" "H" "H" "H" "M" "H" "M" "H" "M" "H"
"H"
## [127] "M" "M" "H" "H" "H" "H" "H" "H" "H" "H" "H" "H" "M" "H" "H" "H" "H"
"H"
## [145] "H" "H" "M" "H" "H" "H"
```

```
newds = data.frame(iris$Petal.Length, mylabel)  #꽃잎의 길이와 레이블 결합
head(newds)                                     #새로운 데이터셋 내용 출력
```

```
##   iris.Petal.Length mylabel
## 1               1.4       L
## 2               1.4       L
```

```
## 3                1.3       L
## 4                1.5       L
## 5                1.4       L
## 6                1.7       M
```

코드 4-11 의 for 문을 보면 **iris$Petal.length** 의 **i** 번째 값에 따라 **mylabel** 의 **i** 번째 값 이 **'L','H','M'** 중의 하나로 결정되는 것을 알 수 있다. **mylabel** 은 처음에는 비어있는 벡터 였는데 for 문의 반복이 한 번 실행될 때마다 값들이 하나씩 추가되어 for 문이 종료되면 150 개의 레이블 값을 가지게 된다.

코드 4-14

```
sum = 0
for(i in 1:10){
  if(i%%2==0)next
  sum = sum + i
}
sum
```

```
## [1] 25
```

[코드 4-14] 역시 1~10 까지의 합계를 구하는데 **i** 가 짝수이면 **next** 가 실행되어 **sum <- sum+i** 를 실행하지 않고 다음 반복으로 넘어간다. 따라서 최종 결과에는 홀수들의 합계가 저장이된다. ***

03 apply() 함수

코드 4-15

```
apply(iris[,1:4], 1, FUN = mean) #row 방향으로 함수 작용
```

```
##   [1] 2.550 2.375 2.350 2.350 2.550 2.850 2.425 2.525 2.225 2.400 2.700
2.500
##  [13] 2.325 2.125 2.800 3.000 2.750 2.575 2.875 2.675 2.675 2.675 2.350
2.650
##  [25] 2.575 2.450 2.600 2.600 2.550 2.425 2.425 2.675 2.725 2.825 2.425
2.400
##  [37] 2.625 2.500 2.225 2.550 2.525 2.100 2.275 2.675 2.800 2.375 2.675
2.350
##  [49] 2.675 2.475 4.075 3.900 4.100 3.275 3.850 3.575 3.975 2.900 3.850
3.300
##  [61] 2.875 3.650 3.300 3.775 3.350 3.900 3.650 3.400 3.600 3.275 3.925
3.550
##  [73] 3.800 3.700 3.725 3.850 3.950 4.100 3.725 3.200 3.200 3.150 3.400
3.850
##  [85] 3.600 3.875 4.000 3.575 3.500 3.325 3.425 3.775 3.400 2.900 3.450
```

```
3.525
##  [97] 3.525 3.675 2.925 3.475 4.525 3.875 4.525 4.150 4.375 4.825 3.400
4.575
## [109] 4.200 4.850 4.200 4.075 4.350 3.800 4.025 4.300 4.200 5.100 4.875
3.675
## [121] 4.525 3.825 4.800 3.925 4.450 4.550 3.900 3.950 4.225 4.400 4.550
5.025
## [133] 4.250 3.925 3.925 4.775 4.425 4.200 3.900 4.375 4.450 4.350 3.875
4.550
## [145] 4.550 4.300 3.925 4.175 4.325 3.950
```

apply(iris[,1:4], 1, FUN = mean) 명령문은 iris 데이터셋에서 행 방향으로 진행을 하면서 각 행의 평균(mean)을 계산하여 출력한다. **iris** 데이터셋에서 150 개의 행이 있기 때문에 이 명령 문의 실행 결과는 150 개 행에 대한 행별 평균값이다.

```
apply(iris[,1:4], 2, FUN = mean) #col 방향으로 함수 작용

## Sepal.Length  Sepal.Width Petal.Length  Petal.Width
##     5.843333     3.057333     3.758000     1.199333
```

apply(iris[,1:4], 2, FUN = mean) 명령문은 **iris** 데이터셋에서 열 방향으로 진행을 하면서 각 열의 평균(mean)을 계산하여 출력한다. 그 결과 4 개의 열에 대한 평균이 출력된 것을 확일할 수 있다. **apply** 함수와 유사한 함수로 **lapply(), sapply(), tapply(), mapply()** 함수 등이 있 는데 **apply()** 함수를 이해하면 나머지 함수들도 쉽게 사용할 수 있다.

---

04 사용자 정의 함수

코드 4-19

```
myfunc = function(x,y){
  val.sum = x+y
  val.mul = x*y
  return(list(sum=val.sum, mul=val.mul))
}
result= myfunc(5,8)
s= result$sum
m= result$mul
cat('5+8=',s, '\n')   #5,8 의 합

## 5+8= 13

cat('5*8=',m,'\n')    #5,8 의 곱
```

```
## 5*8= 40
```

코드 4-20

```
setwd("C:/Users/Sin/Desktop/coding_study/R/R 학교 수업/") # myfunc.R 이 저장된 폴더
source('myfunc.R')                                        # myfunc.R 안에 있는 함수 실행

## 5+8= 13
## 5*8= 40

#함수 사용
a= mydiv(20,4)                                            #함수 호출
b = mydiv(30,4)                                           #함수 호출
a+b

## [1] 12.5

mydiv(mydiv(20,2),5)                                      #함수 호출

## [1] 2
```

mydiv() 함수를 호출하기 위해서는 먼저 파일에 있는 **mydiv()** 함수를 실행해야 하는데 그 명령어가 **source("myfunc.R")**이다. 이 명령문의 의미는 **myfunc.R** 파일에 저장되어 있는 함수나 명령문들을 실행하라는 것이다.

setwd("C:/Users/Sin/Desktop/coding_study/R/R 학교 수업/")는 **myfunc.R** 파일이 위치하는 폴더를 작업 폴더로 지정한다. **source("myfunc.R")**를 실행하면 **mydiv()** 함수를 사용할 준 비가 되는 것이므로 이후에는 필요하 곳에서 호출하여 사용하면 된다.

---

05 조건에 맞는 데이터의 위치 찾기

코드 4-23

```
idx = which(iris$Petal.Length>5.0)                        #꽃잎의 길이가 5.0 이상인 값들의 인덱스
idx

##  [1]  84 101 102 103 104 105 106 108 109 110 111 112 113 115 116 117 118
119 121
## [20] 123 125 126 129 130 131 132 133 134 135 136 137 138 140 141 142 143
```

```
144 145
## [39] 146 148 149 150

iris.big = iris[idx,]                                    #인덱스에 해당하는 값만 추출하
여 저장
iris.big
```

```
##     Sepal.Length Sepal.Width Petal.Length Petal.Width    Species
## 84           6.0         2.7          5.1         1.6 versicolor
## 101          6.3         3.3          6.0         2.5  virginica
## 102          5.8         2.7          5.1         1.9  virginica
## 103          7.1         3.0          5.9         2.1  virginica
## 104          6.3         2.9          5.6         1.8  virginica
## 105          6.5         3.0          5.8         2.2  virginica
## 106          7.6         3.0          6.6         2.1  virginica
## 108          7.3         2.9          6.3         1.8  virginica
## 109          6.7         2.5          5.8         1.8  virginica
## 110          7.2         3.6          6.1         2.5  virginica
## 111          6.5         3.2          5.1         2.0  virginica
## 112          6.4         2.7          5.3         1.9  virginica
## 113          6.8         3.0          5.5         2.1  virginica
## 115          5.8         2.8          5.1         2.4  virginica
## 116          6.4         3.2          5.3         2.3  virginica
## 117          6.5         3.0          5.5         1.8  virginica
## 118          7.7         3.8          6.7         2.2  virginica
## 119          7.7         2.6          6.9         2.3  virginica
## 121          6.9         3.2          5.7         2.3  virginica
## 123          7.7         2.8          6.7         2.0  virginica
## 125          6.7         3.3          5.7         2.1  virginica
## 126          7.2         3.2          6.0         1.8  virginica
## 129          6.4         2.8          5.6         2.1  virginica
## 130          7.2         3.0          5.8         1.6  virginica
## 131          7.4         2.8          6.1         1.9  virginica
## 132          7.9         3.8          6.4         2.0  virginica
## 133          6.4         2.8          5.6         2.2  virginica
## 134          6.3         2.8          5.1         1.5  virginica
## 135          6.1         2.6          5.6         1.4  virginica
## 136          7.7         3.0          6.1         2.3  virginica
## 137          6.3         3.4          5.6         2.4  virginica
## 138          6.4         3.1          5.5         1.8  virginica
## 140          6.9         3.1          5.4         2.1  virginica
## 141          6.7         3.1          5.6         2.4  virginica
## 142          6.9         3.1          5.1         2.3  virginica
## 143          5.8         2.7          5.1         1.9  virginica
## 144          6.8         3.2          5.9         2.3  virginica
## 145          6.7         3.3          5.7         2.5  virginica
## 146          6.7         3.0          5.2         2.3  virginica
## 148          6.5         3.0          5.2         2.0  virginica
```

```
## 149          6.2         3.4          5.4         2.3  virginica
## 150          5.9         3.0          5.1         1.8  virginica
```

which()함수를 이용하여 매트릭스, 데이터프레임 안에 있는 특정 값의 행과 열의 위치를 알 고 싶으면 코드 4-24 와 같이 **arr.ind = TRUE** 매개변수를 추가한다.

코드 4-24

```
#1~4 열의 값 중 5 보다 큰 값의 행과 열의 위치
idx = which(iris[,1:4]>5.0, arr.ind = TRUE)
head(idx, 20)                                #데이터가 너무 많아 20 개만 출력시킨다.

##       row col
##  [1,]   1   1
##  [2,]   6   1
##  [3,]  11   1
##  [4,]  15   1
##  [5,]  16   1
##  [6,]  17   1
##  [7,]  18   1
##  [8,]  19   1
##  [9,]  20   1
## [10,]  21   1
## [11,]  22   1
## [12,]  24   1
## [13,]  28   1
## [14,]  29   1
## [15,]  32   1
## [16,]  33   1
## [17,]  34   1
## [18,]  37   1
## [19,]  40   1
## [20,]  45   1
```