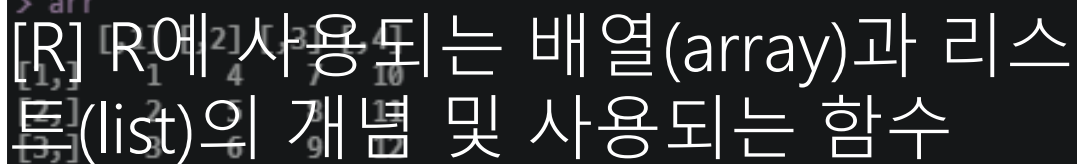


노트북: [TIL-MY]
만든 날짜: 2020-07-23 오전 8:57
URL: <https://continuous-development.tistory.com/35>

홈 태그



동일 타입의 데이터만 저장 가능

Deep learning

인덱싱 : [행의 인덱싱, 열의 인덱싱, 층(면)의 인덱스]

#배열 생성 array(value , dim=c(value))

```
190 ##배열(Array) -
191 #array(), dim()
192
193 m <- matrix(1:12, ncol=4)
194 m
195 class(m)
196
197 arr <- array(1:12, dim=c(3,4,3))
198 arr
199 class(arr)
200 |
201
```

200:1 (Top Level) ▾

Console Terminal × Jobs ×

~/ ↗

```
> arr
      [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
> arr <- array(1:12, dim=c(3,4,3))
> arr
, , 1
      [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
, , 2
      [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
, , 3
      [,1] [,2] [,3] [,4]
[1,]    1    4    7   10
[2,]    2    5    8   11
[3,]    3    6    9   12
```

3행 / 4열 / 3층 을 가진 배열

#배열에 대한 접근

AWS

ETC..

R 

공지사항

글 보실 때 주의사항

: 최근글 : 인기글

[R] R
에 ...



2020.07.23

[R] R
에 ...



2020.07.23

[R] R
에...



2020.07.22

[R] R
에 ...



2020.07.22

[Data
트...



2020.07.21

최근댓글

태그

rep함수, AWS,

Oracle SQL,

length함수,

```
201
202 # 배열에 대한 접근
203 #행렬과 유사한 방식으로 각 요소에 접근할 수 있다.
204
205 arr[1,2,3]
206 arr[c(1,3),c(1,2),3]
207 arr[1:3,1:2,3]
208 |
209
208:1 (Top Level)
Console Terminal Jobs
~/
> arr[1,2,3]
[1] 4
> arr[c(1,3),c(1,2),3]
      [,1] [,2]
[1,]    1    4
[2,]    3    6
> arr[1:3,1:2,3]
      [,1] [,2]
[1,]    1    4
[2,]    2    5
[3,]    3    6
> |
>
```

#리스트(list)

저장 가능한 데이터의 타입, 데이터 셋의 종류에 제한이 없다. 벡터, 행렬, 배열, 데이터 프레임 등의 서로 다른 구조의 데이터를 하나로 묶을 수 있는 자료구조이다.

list() 함수로 리스트를 생성하고, [, [[, \$ 을 통해 부분집합을 뽑아낸다.

R 정규표현식,
R 기본함수,
paste함수, DDL,
행렬,
R 정규표현식 사용법,
SQL,
str_extract_all,
setequal함수,
unlist함수,
array 함수,
matrix함수,
사용법, 설정,
str_extract,
날짜함수,
rbind함수,
인스턴스, R 배열,
strsplit, Oracle,
substr,
cbind함수,
테이블 생성,
names함수,
do.call

전체 방문자

102

Today : 0

Yesterday : 3

#리스트 - list(키, 값) 형태의 데이터를 담은 연관 배열이다.

```
229 #list(키, 값) 형태의 데이터를 담은 연관배열이다. ~ 데이터 셋의 종류에 제한은 없다 $키의 리스트가 포함된 원소 수를 / 개수만큼 간단히 처리
230 #key와 values를 담은 형태 (파이썬에서는 딕셔너리)
231
232 #list()
233 #apply -> lapply()-키 불륨을 담아서 보낸다 ,sapply() value만 보낸다.,
234
235
236 list <- list()
237 list
238
239 exList <- list(name="jslim", height = 70)
240 exList$name
241 exList$height
242
243 #name = 키 values
244
245 |
246 simpleList <- list(1:4, rep(3,5) , "cat")
247 simpleList
248
249
250 newList <- c(list(1, 2, simpleList) , c(3,4))
251 newList
```

```
> list <- list()
> list
list()
> exList <- list(name="jslim", height = 70)
> exList$name
[1] "jslim"
> exList$height
[1] 70
> simpleList <- list(1:4, rep(3,5) , "cat")
> simpleList
[[1]]
[1] 1 2 3 4

[[2]]
[1] 3 3 3 3 3

[[3]]
[1] "cat"
```

```
256
257 #리스트안에 리스트 중첩
258 list(a = list(c(1,2,3) ),
259       b = list(c(1,2,3,4)))
260
261 overList<-list(a = list(c(1,2,3) ),
262               b = list(c(1,2,3,4)))
263
264 overList
265
266 #어느 위치로 갈지 정하고 거기서 몇번째 리스트를 가져올지 하고 그다음 몇번째 인덱스를 할지를 본다
267 overList$a[[1]][2]
268 overList$b[[1]][3]
269
270
271
```

261:36 (Top Level) :

```
~/
> #리스트안에 리스트 중첩
> list(a = list(c(1,2,3) ),
+       b = list(c(1,2,3,4)))
$a
[[1]]
[1] 1 2 3

$b
[[1]]
[1] 1 2 3 4

>
> overList<-list(a = list(c(1,2,3) ),
+               b = list(c(1,2,3,4)))
>
> overList
$a
[[1]]
[1] 1 2 3

$b
[[1]]
[1] 1 2 3 4
```

```
>
> #어느 위치로 갈지 정하고 거기서 몇번째 리스트를 가져올지 하고 그다음 몇번째 인덱스를 할지를 본다
> overList$a[[1]][2]
[1] 2
> overList$b[[1]][3]
[1] 3
> |
```

#키에 대해 단일로 값을 가지는 요소들

```
270
271 #키를 가지는 단일 요소들
272 member <- list(
273   name  = "jslim",
274   address = "seoul",
275   tel    = "010-4603-2283",
276   age    = 48,
277   married = T
278 )
279 member
280
281 member$tel
282 member$name
283 |
283:1 (Top Level) ↕
```

Console Terminal x Jobs x

~/ ↗

```
+ name  = "jslim",
+ address = "seoul",
+ tel    = "010-4603-2283",
+ age    = 48,
+ married = T
+ )
> member
$name
[1] "jslim"

$address
[1] "seoul"

$tel
[1] "010-4603-2283"

$age
[1] 48

$married
[1] TRUE

> member$tel
[1] "010-4603-2283"
> member$name
[1] "jslim"
> |
```

#키에 대해 다중으로 값을 가지는 요소들

```

282
283 #키에대해 다중으로 값을 가지는 요소들
284 member <- list(
285   name = c("섭섭해", "임섭순"),
286   address = c("seoul", "gwangu"),
287   age = c("48", "29"),
288   gender = c("남자", "여자")
289 )
290
291 member$name
292 member$name[2]
293 member$age[1] <- 38
294 member$id <- c("jslim", "admin")
295 member
296 member$id <- NULL
297 member$address[2] <- "seoul"
298 member
299 |

```

```

> member$id <- c("jslim", "admin")
> member
$name
[1] "섭섭해" "임섭순"

$address
[1] "seoul" "seoul"

$age
[1] "38" "29"

$gender
[1] "남자" "여자"

$id
[1] "jslim" "admin"

> member$id <- NULL
> member$address[2] <- "seoul"
> member
$name
[1] "섭섭해" "임섭순"

$address
[1] "seoul" "seoul"

$age
[1] "38" "29"

$gender
[1] "남자" "여자"

```

#서로 다른 자료구조를 가지는 리스트

```

301 #서로 다른 자료구조(vector,matrix,array)를 가지는 리스트
302
303 multiList <- list(
304   one = c("one","two","three"),
305   second = matrix(1:9, nrow = 3),
306   third = array(1:12, dim = c(2,3,2)) # 2행 3열에 계층은 2개
307 )
308 multiList
309
310 multiList$one[1]
311 multiList$one[2]
312 multiList$second[2,2]
313 multiList$second[3,3]
314 multiList$third[1,3,2]
315

```

```

[3,] 3 6 9

$third
, , 1
     [,1] [,2] [,3]
[1,] 1 3 5
[2,] 2 4 6
, , 2
     [,1] [,2] [,3]
[1,] 7 9 11
[2,] 8 10 12

>
> multiList$one[1]
[1] "one"
> multiList$one[2]
[1] "two"
> multiList$second[2,2]
[1] 5
> multiList$second[3,3]
[1] 9
> multiList$third[1,3,2]
[1] 11
>

```

#unlist() - 리스트를 벡터로 형 변환

```

317 # unlist() list → 리스트를 벡터로 형변환
318 x <- list(1:5)
319 x
320 class(x)
321
322 vec <- unlist(x)
323 vec
324 class(vec)
325
326 matrixList <- list(
327   row1 = list(1,2,3),
328   row2 = list(10,20,30),
329   row3 = list(100,200,300)
330 )
331
332 matrixList
333 class(matrixList)
334

```

```

> x <- list(1:5)
> x
[[1]]
[1] 1 2 3 4 5

> class(x)
[1] "list"
>
> vec <- unlist(x)
> class(vec)
[1] "integer"
>

```

#do.call -리스트를 행렬로 바꾸는 함수

```

327 # do.call(func, data) - 리스트를 행렬로 바꾸는 명령 (rbind,cbind 만 가능)
328 matrixList <- list(
329   row1 = list(1,2,3),
330   row2 = list(10,20,30),
331   row3 = list(100,200,300)
332 )
333
334 matrixList
335 class(matrixList)
336
337 row_mat <- do.call(rbind,matrixList)
338 row_mat
339
340 row_mat <- do.call(cbind,matrixList)
341 row_mat
342

```

```

> # do.call(func, data) - 리스트를 행렬로 바꾸는 명령 (rbind,cbind 만 가능)
> matrixList <- list(
+   row1 = list(1,2,3),
+   row2 = list(10,20,30),
+   row3 = list(100,200,300)
+ )
> matrixList
$row1
$row1[[1]]
[1] 1

$row1[[2]]
[1] 2

$row1[[3]]
[1] 3

$row2
$row2[[1]]
[1] 10

$row2[[2]]

```



```

> class(matrixList)
[1] "list"
> row_mat <- do.call(rbind,matrixList)
> row_mat
      [,1] [,2] [,3]
row1  1    2    3
row2 10   20   30
row3 100  200  300
> row_mat <- do.call(cbind,matrixList)
> row_mat
      row1 row2 row3
[1,]  1   10  100
[2,]  2   20  200
[3,]  3   30  300

```

#list의 값의 길이 확인

```

344
345 listLength <- list(1:5, list("This is my First time R",c(T,F,T)))
346 listLength
347 length(listLength)
348 length(listLength[[1]])
349 length(listLength[[2]])
350 str_length(listLength[[2]][1])
351 str_length(listLength[[2]][2])
352
353

```

```

> length(listLength)
[1] 2
> length(listLength[[1]])
[1] 5
> length(listLength[[2]])
[1] 2
> str_length(listLength[[2]][1])
[1] 23
> str_length(listLength[[2]][2])
[1] 20
경고메시지(들):
In str_length(string) : argument is not an atomic vector; coercing
>

```

#list 처리 함수

```

354 |
355 # list 처리함수
356 #lapply() : list , key = value - list 형식
357 #sapply() : list , value      - vector 형식
358
359
360 x <- list(1:5)
361 y <- list(6:10)
362
363 lapply(c(x,y),FUN = sum)
364 class(lapply(c(x,y),FUN = sum))
365
366 sapply(c(x,y),FUN = sum)
367 class(sapply(c(x,y),FUN = sum))
368

```

```

> x <- list(1:5)
> y <- list(6:10)
> lapply(c(x,y),FUN = sum)
[[1]]
[1] 15

[[2]]
[1] 40

> class(lapply(c(x,y),FUN = sum))
[1] "list"
> sapply(c(x,y),FUN = sum)
[1] 15 40
> class(sapply(c(x,y),FUN = sum))
[1] "integer"
>

```

#리스트 -> 벡터 -> 행렬 -> 데이터 프레임 변환 예제

```

373
374 l_value<- lapply(iris[,1:4], mean)
375 s_value<-sapply(iris[,1:4], mean)
376
377 # unlist() 리스트를 벡터로 변환
378 vec <- unlist(l_value)
379
380 vec
381 class(vec)
382 # matrix() 벡터를 행렬로 변환
383 mat <- matrix(vec,ncol=4)
384 mat
385 class(mat)
386 #as.data.frame() 행렬을 데이터 프레임으로 변환
387 dataFrame <- as.data.frame(mat)
388 dataFrame
389 class(dataFrame)
390 #names() 사용해서 리스트로부터 변수명을 얻어와 데이터 프레임의 각 열에 이름 부여
391 rename<-names(s_value)
392 re_dataFrame<-rename(dataFrame,c("V1" = rename[1],"V2" = rename[2],"V3"=rename[3] , "V4" = rename[4]))
393 re_dataFrame
394

```

```

377
378 # unlist() 리스트를 벡터로 변환
379 vec <- unlist(l_value)
380 vec
381 class(vec)
382
383 # matrix() 벡터를 행렬로 변환
384 mat <- matrix(vec,ncol=4, byrow =T)
385 mat
386 class(mat)
387
388 #as.data.frame() 행렬을 데이터 프레임으로 변환
389 dataFrame <- as.data.frame(mat)
390 dataFrame
391 class(dataFrame)
392
393 #names() 사용해서 리스트로부터 변수명을 얻어와 데이터 프레임의 각 열에 이름 부여 /rename도 가능
394 names(dataFrame) <-names(iris[, 1:4])
395 #names(dataFrame) <-names(s_value)
396 dataFrame
397
398 rename<-names(s_value)
399 re_dataFrame<-rename(dataFrame,c("V1" = rename[1],"V2" = rename[2],"V3"=rename[3] , "V4" = rename[4]))
400 re_dataFrame

```

```
373 l_value<- lapply(iris[,1:4], mean)
374 s_value<-sapply(iris[,1:4], mean)
375
376
377 # unlist() 리스트를 벡터로 변환
378 vec <- unlist(l_value)
379
380 class(vec)
381 # matrix() 벡터를 행렬로 변환
382 mat <- matrix(vec,ncol=4)
383
384 class(mat)
385 #as.data.frame() 행렬을 데이터 프레임으로 변환
386 dataFrame <- as.data.frame(mat)
387
388 class(dataFrame)
389 #names() 사용해서 리스트로부터 변수명을 얻어와 데이터 프레임의 각 열에 이름 부여
390 rename<-names(s_value)
391 re_dataFrame<-rename(dataFrame,c("V1" = rename[1],"V2" = rename[2],"V3"=rename[3] , "V4" = rename[4]))
392 re_dataFrame
393
394 |
394:1 (Top Level) :-
Console Terminal Jobs
~/
> vec
Sepal.Length Sepal.Width Petal.Length Petal.Width
5.843333 3.057333 3.758000 1.199333
> class(vec)
[1] "numeric"
> # matrix() 벡터를 행렬로 변환
> mat <- matrix(vec,ncol=4)
> mat
      [,1] [,2] [,3] [,4]
[1,] 5.843333 3.057333 3.758 1.199333
> class(mat)
[1] "matrix" "array"
> #as.data.frame() 행렬을 데이터 프레임으로 변환
> dataFrame <- as.data.frame(mat)
> dataFrame
      V1      V2      V3      V4
1 5.843333 3.057333 3.758 1.199333
> class(dataFrame)
[1] "data.frame"
> #names() 사용해서 리스트로부터 변수명을 얻어와 데이터 프레임의 각 열에 이름 부여
> rename<-names(s_value)
> re_dataFrame<-rename(dataFrame,c("V1" = rename[1],"V2" = rename[2],"V3"=rename[3] , "V4" = rename[4]))
> re_dataFrame
Sepal.Length Sepal.Width Petal.Length Petal.Width
5.843333 3.057333 3.758 1.199333
```

이 과정을 짧게 나타내면

```
402
403 data.frame(do.call(cbind,lapply(iris[,1:4], mean)))
404
405 class(data.frame(do.call(cbind,lapply(iris[,1:4], mean))))
406 |
```

이렇게 된다.

'R' 카테고리의 다른 글

[R] R에 사용되는 배열(array)과 리스트(list)의 개념 및 사용되는 함수 (0)

03:32:42

[R] R에 사용되는 행렬(matrix)의 개념 및 사용되는 함수 (0)

02:26:20

[R] R에서 사용되는 정규표현식(Regex) 표현 방법과 함수를 통한 사용 예제 (0)

2020.07.22

[R] R에 사용되는 벡터(matrix)의 개념 및 사용되는 함수(출력,인덱싱,길이반환,문자열비교 등등) (0)

2020.07.22

태그

array

array 함수

do.call

list

list 함수

R list

R 배열

unlist함수

리스트

행렬

관련글



[R] R에 사... [R] R에서 ... [R] R에 사...

댓글 0

TEL. 02.1234.5678 / 경기 성남시 분당구 판교역로

© Kakao Corp.

