R & PYTHON

VICTOR TERRON

19/12/2020

ESTE DOCUMENTO ES UN EJEMPLO DE R & PHYTHON CON LIBRERÍA RETICULATE ## LIBRERÍA RETICULATE RETICULATE ES UNA INTERFACE DE PYTHON

```
library(reticulate)
use_python("C:/ProgramData/Anaconda3/python.exe")
py_install("numpy")
py_install("pandas")
os  <- import ("os")
os$listdir(".")
```

```
##  [1] ".RData"                     ".Rhistory"
##  [3] ".Rproj.user"                "DOCUMENTAION DE TEXTO.Rmd"
##  [5] "DOCUMENTAION-DE-TEXTO.html" "DOCUMENTAION-DE-TEXTO.pdf"
##  [7] "EJERCICIO 1.R"              "proyalgebralineal.Rproj"
##  [9] "PRUEA.pdf"                  "PRUEA.Rmd"
## [11] "prueba1.pdf"                "prueba1.rmd"
## [13] "PRUEBA3.pdf"                "PRUEBA3.Rmd"
## [15] "R&PYTHON.Rmd"               "R-PYTHON.html"
## [17] "R-PYTHON.pdf"               "R-PYTHON.Rmd"
## [19] "REFERENTE.html"             "REFERENTE.pdf"
## [21] "REFERENTE.RMD"              "RPYTHON.pdf"
## [23] "RPYTHON.Rmd"                "SA.R"
## [25] "SCRIPT INICIAL.R"
```

##OPERACIONES

```
np <- import("numpy",convert = FALSE)
x<-np$array(c(1:4))
sum<-x$cumsum()
print(sum)
```

```
## [ 1  3  6 10]
```

```
py_to_r(sum)
```

```
## [1]  1  3  6 10
```

##AYUDA

```
help(py_to_r)
```

```
## starting httpd help server ... done
```

```
py_help(os$chdir)
```

##ARRAYS

```
a <- np_array(c(1:10), order="C")
a
```

```
## [ 1  2  3  4  5  6  7  8  9 10]
```

##IRIS

```
datos <- iris
head(datos)
```

```
##   Sepal.Length Sepal.Width Petal.Length Petal.Width Species
## 1          5.1         3.5          1.4         0.2  setosa
## 2          4.9         3.0          1.4         0.2  setosa
## 3          4.7         3.2          1.3         0.2  setosa
## 4          4.6         3.1          1.5         0.2  setosa
## 5          5.0         3.6          1.4         0.2  setosa
## 6          5.4         3.9          1.7         0.4  setosa
```

```
datos_py <- r_to_py(datos)
```

```
import numpy as np
import pandas as pd
r.datos_py.head()
```

```
##    Sepal.Length  Sepal.Width  Petal.Length  Petal.Width Species
## 0           5.1          3.5           1.4          0.2  setosa
## 1           4.9          3.0           1.4          0.2  setosa
## 2           4.7          3.2           1.3          0.2  setosa
## 3           4.6          3.1           1.5          0.2  setosa
## 4           5.0          3.6           1.4          0.2  setosa
```

##SPARCE MATRIX

```
library(Matrix)
N <- 6
sparse_mat <- sparseMatrix(
  i = sample(N, N, replace=F),
  j = sample(N, N, replace=F),
  x = runif(N),
  dims=c(N,N)
)
sparse_mat
```

```
## 6 x 6 sparse Matrix of class "dgCMatrix"
##
## [1,] .         .         .         0.2673099 .         .
## [2,] .         .         .         .         0.2109255 .
## [3,] .         .         .         .         .         0.7919447
## [4,] .         0.3503466 .         .         .         .
## [5,] 0.3110567 .         .         .         .         .
## [6,] .         .         0.9553526 .         .         .
```

```
sparse_mat_py <- r_to_py(sparse_mat)
```

##SPARCE MATRIX EN PYTHON

```
r.sparse_mat_py
```

```
## <6x6 sparse matrix of type '<class 'numpy.float64'>'
##  with 6 stored elements in Compressed Sparse Column format>
```

Si alguna modificacion ha sido realizada en PYTHON se puede recuperar transformandola a R de nuevo:

```
py_to_r(sparse_mat_py)
```

haciendo el experimento de conversion de objeto de R a PYTHON:

```
np <- import("numpy",convert=FALSE)
x <- np$array(c(1:4))
sum <- x$cumsum()
print(sum)
```

```
## [ 1  3  6 10]
```

```
py_to_r(sum)
```

```
## [1]  1  3  6 10
```

AYUDA

{r} #help(reticulate:py_to_r) #py_help(os$chdir) <!-- –>

##ARRAYS

```
a <- np_array(c(1:10),dtype="float16")
```

Para definir un vector en R utilizamos c(1,2,3), aunque tambien se puede utilizar scan() para escanear datos que se escriben desde teclado, suponiendo que tengamos ya un vector en la variable x y quieramos modificar visualmente el vector x rep(a,n) donde a es el elemento que queremos repetir y el numero de veces en n que queremos repetirlo