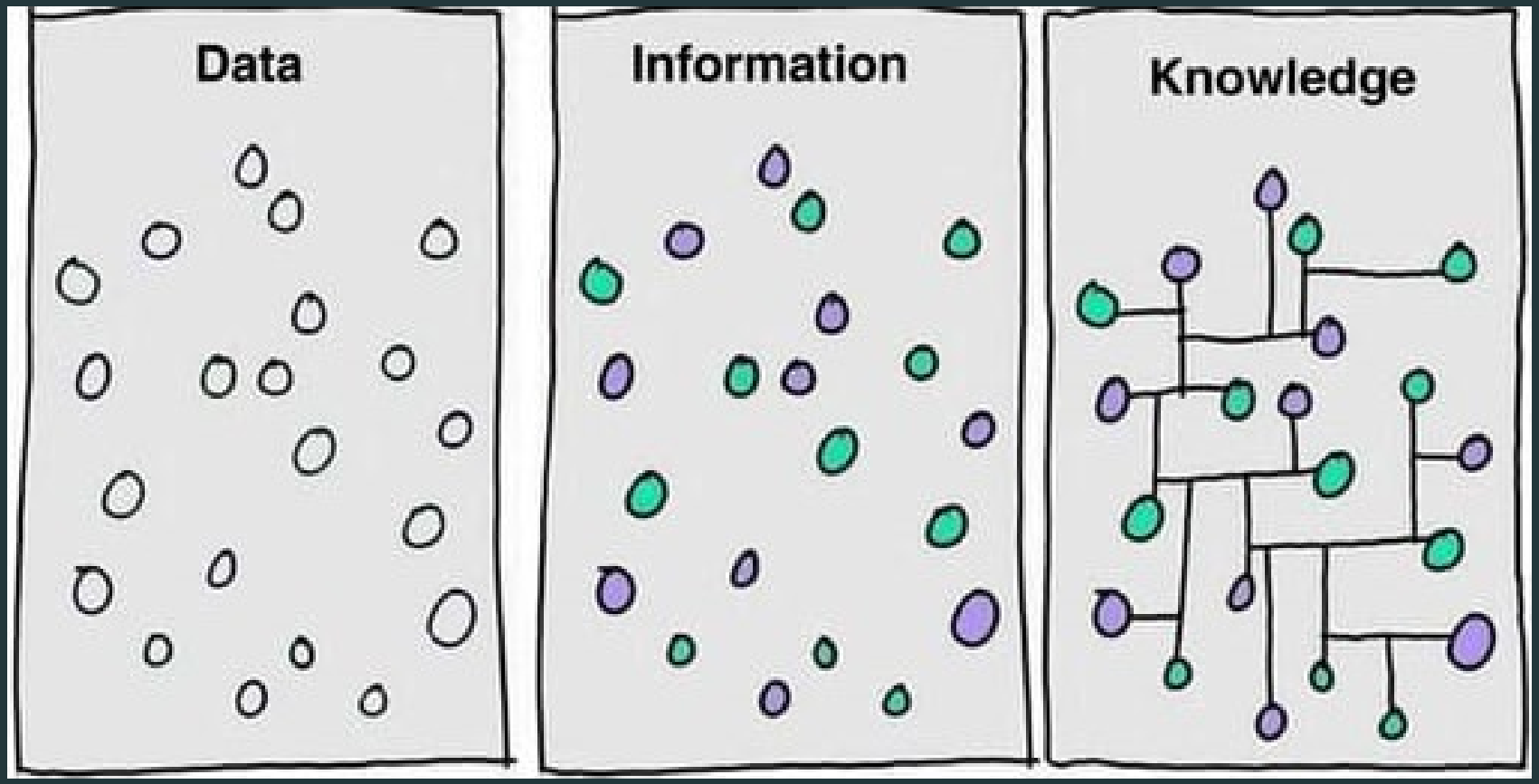
Importar achivos del computador

Programación Unidad 3: Python y algoritmia

Ph.D. Santiago Echeverri-Arteaga


Data
Information
Knowledge

Índice

Índice

Importancia de la importación de datos

¿Para qué necesita un físico importar datos en un programa?

Los datos pueden importarse desde el teclado o la CPU. Cada una de ellas tiene sus ventajas, pero ¿Para qué es importante **para un físico** aprender a importar y a trabajar con datos?

¿Para qué necesita un físico importar datos en un programa?

Los datos pueden importarse desde el teclado o la CPU. Cada una de ellas tiene sus ventajas, pero ¿Para qué es importante **para un físico** aprender a importar y a trabajar con datos?

1. Datos experimentales (Equipos con software comercial o propio como LabView)

¿Para qué necesita un físico importar datos en un programa?

Los datos pueden importarse desde el teclado o la CPU. Cada una de ellas tiene sus ventajas, pero ¿Para qué es importante **para un físico** aprender a importar y a trabajar con datos?

1. Datos experimentales (Equipos con software comercial o propio como LabView)
2. Física de altas energías (Pandas)

¿Para qué necesita un físico importar datos en un programa?

Los datos pueden importarse desde el teclado o la CPU. Cada una de ellas tiene sus ventajas, pero ¿Para qué es importante **para un físico** aprender a importar y a trabajar con datos?

1. Datos experimentales (Equipos con software comercial o propio como LabView)
2. Física de altas energías (Pandas)
3. Física médica

¿Para qué necesita un físico importar datos en un programa?

Los datos pueden importarse desde el teclado o la CPU. Cada una de ellas tiene sus ventajas, pero ¿Para qué es importante **para un físico** aprender a importar y a trabajar con datos?

1. Datos experimentales (Equipos con software comercial o propio como LabView)
2. Física de altas energías (Pandas)
3. Física médica
4. Simulaciones computacionales

¿Para qué necesita un físico importar datos en un programa?

Los datos pueden importarse desde el teclado o la CPU. Cada una de ellas tiene sus ventajas, pero ¿Para qué es importante **para un físico** aprender a importar y a trabajar con datos?

1. Datos experimentales (Equipos con software comercial o propio como LabView)
2. Física de altas energías (Pandas)
3. Física médica
4. Simulaciones computacionales
5. Sistemas complejos

¿Para qué necesita un físico importar datos en un programa?

Los datos pueden importarse desde el teclado o la CPU. Cada una de ellas tiene sus ventajas, pero ¿Para qué es importante **para un físico** aprender a importar y a trabajar con datos?

1. Datos experimentales (Equipos con software comercial o propio como LabView)
2. Física de altas energías (Pandas)
3. Física médica
4. Simulaciones computacionales
5. Sistemas complejos
6. Ciencia de datos

¿Para qué necesita un físico importar datos en un programa?

Los datos pueden importarse desde el teclado o la CPU. Cada una de ellas tiene sus ventajas, pero ¿Para qué es importante **para un físico** aprender a importar y a trabajar con datos?

1. Datos experimentales (Equipos con software comercial o propio como LabView)
2. Física de altas energías (Pandas)
3. Física médica
4. Simulaciones computacionales
5. Sistemas complejos
6. Ciencia de datos

¿Cómo son esos datos?

¿Para qué necesita un físico importar datos en un programa?

Los datos pueden importarse desde el teclado o la CPU. Cada una de ellas tiene sus ventajas, pero ¿Para qué es importante **para un físico** aprender a importar y a trabajar con datos?

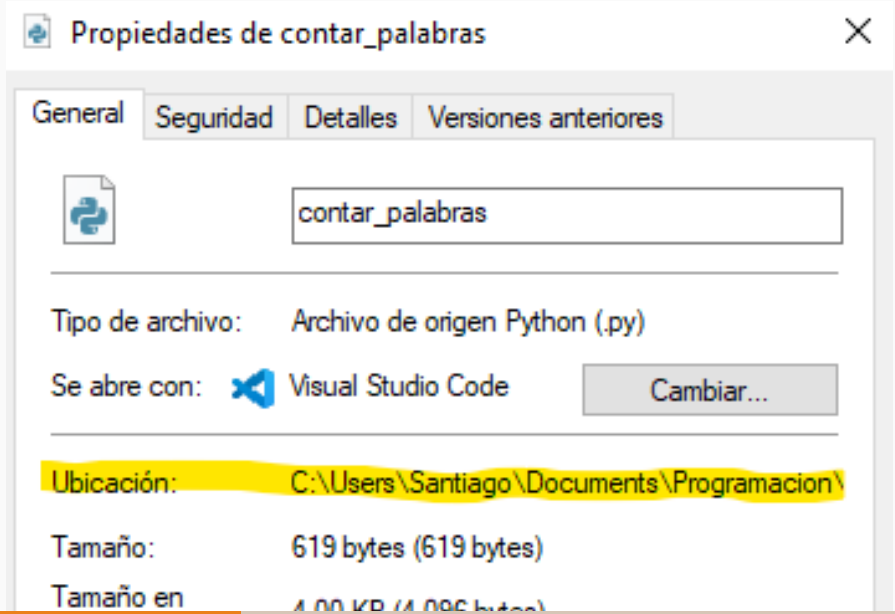
1. Datos experimentales (Equipos con software comercial o propio como LabView)
2. Física de altas energías (Pandas)
3. Física médica
4. Simulaciones computacionales
5. Sistemas complejos
6. Ciencia de datos

¿Cómo son esos datos?
Tablas grandes (o muy grandes)

Importar/Leer desde la CPU sin librerías adicionales

open

- Para leer un archivo de texto como una lista de cadenas, siendo cada cadena un renglón del archivo:
  lines = [line.strip() for line in open('example.txt')]
- Para leer un archivo de texto tal cual está
  s = open('example.txt').read()
- Si el archivo está en el mismo directorio que el .py solo se necesita indicar el nombre. De lo contrario es necesario indicar el directorio completo
  'C:/Users/Santiago/Documents/archivo.txt'



write

Para escribir en un archivo primero se debe crear el archivo, luego guardar la información en él y finalmente guardarlo. Esto se hace así:

1. f = open('writefile.txt', 'w')
2. print('Esta es la línea 1.', file=f)
   print('Esta es la línea 2.', file=f)
3. f.close()

Numpy

loadtxt

En numpy existen dos opciones para importar datos desde la CPU, la primero de ellos es loadtxt. **Éste tiene multiples parametros OPCIONALES que nos hacen la vida más fácil**

- **usecols=[0, 2]** Selecciona las columnas a leer
- **delimiter=',** Caracter que separa las columnas.
- **skiprows=1** Se salta 1 columna por ser el encabezado
- **comments='#'** Cuál es el caracter que indica los comentarios
- **dtype=str** Es NECESARIO ponerlo si los datos a leer son cadenas

para usarlo se pone
Variable = np.loadtxt('C:/Users/Santiago/ejemplo.txt', OPCIONES)

genfromtxt

Es otra forma de leer archivos de datos en numpy, al igual que loadtxt tiene múltiples opciones que nos facilitan la vida. **Es preferible cuando se desea importar una lista de múltiples tipos de datos**

- **usecols=[0, 2]** Selecciona las columnas a leer
- **delimiter=','** Caracter que separa las columnas.
- **skip_header=1** y **skip_footer=3** Se salta 1 columna por ser el encabezado y 3 por ser el pie de página
- **comments='#'** Cuál es el caracter que indica los comentarios
- **dtype=str** Es NECESARIO ponerlo si los datos a leer son cadenas
- **dtype=None, encoding=None** Es NECESARIO ponerlo si los datos a leer son **de múltiples tipos**

para usarlo se pone
Variable = np.genfromtxt('C:/Users/Santiago/ejemplo.txt', OPCIONES)

savetxt

Para **exportar variables** numpy tiene una función llamada **savetxt**, la cual tiene múltiples opciones para personalizar el archivo guardado

1. **fmt** Indica el formato que tiene cada columna **delimiter=’,’** Caracter que separa las columnas.
2. **header** Comentario al inicio
3. **comments** Caracter a insertar al inicio de los comentarios
4. **footer** Comentario al final

savetxt guarda el contenido de la lista, **write** guarda lo que arrojaría el Shell.
Ejemplo: Si se declara A = [[1, 2],[3, 4]], **write** imprime [[1, 2],[3, 4]], pero savetxt imprime
1 2
3 4

savetxt en tablas de cadenas

Se puede usar con una tabla de cadenas y números, pero **con ésta forma de usar savetxt no se les puede dar formato a los números.** Para usarlo se pone

np.savetxt('ARCHIVO', Variable, fmt=[' %s' , ' %s'],OPCIONES) *#Se debe poner un % por cada columna*

savetxt en tablas de números

Para usarlo se pone
np.savetxt('ARCHIVO', Variable, OPCIONES)

savetxt en tablas de números y cadenas

Se debe creear un array estructurado, es decir, un array de Numpy donde cada columna es de un tipo de datos diferente.
Primero se debe definir el tipo de dato de cada columna poniendo una lista de tuplas (una por columna). El primer elemento de la tupla es el nombre de la columna y el segundo el tipo de datos que va a tener. Si son cadenas se pone **np.str_10**, si son flotantes **np.float64** y si son enteros **np.int32**
dtype = [('A', (np.str_, 10)), ('B', np.float64), ('C', np.int32)]
N = np.array([('Ana', 33.3, 1), ('Bob', 44.4, 5), ('Cairne', 66.6, 2), ('Dana', 88.8, 20)], dtype=dtype)
Si se desea acceder a los nombres se pone N['A']
Para guardarlo con savetxt:
np.savetxt('ejemplo.dat', N, fmt=['%s', '%.2f', '%d'], OPCIONES)

¿Y si es un diccionario?

D = np.array(list(diccionario.items()), dtype=dtype)
Con el dtype como en el caso anterior:
dtype = [('A', (np.str_, 10)), ('B', np.float64)]
y finalmente se guarda como en el caso anterior
np.savetxt('ejemplo.dat', D, fmt=['%s' , '%d'], OPCIONES)

Ejercicios

Ejercicios

Realice los siguientes ejercicios para entregar

1. Construya un programa que pida el nombre de los productos y sus respectivos valores. Posteriormente pida el nombre de los productos que se desean comprar y la cantidad de cada uno. Debe imprimir el valor total de la compra.
2. Con el mismo diccionario del ejercicio anterior, realice un programa en que el usuario ingrese un valor en efectivo y el programa imprima el nombre y precio de los items que puede comprar