

Práctica de laboratorio 3: Laboratorio:

2.5.1.4 Análisis de datos del medidor de Internet

Nevarez García Litzy Yulissa

Objetivos

Parte 1: Recopile y guarde los datos Parte 2: Manipule los datos

Aspectos básicos/situación

En esta práctica de laboratorio, usted adquirirá estadísticas de la velocidad de Internet y almacenará los datos en vivo en un archivo de valores separados por comas (csv). También cargará los datos almacenados del archivo csv a una estructura de datos de Python, los pandas marco de datos, y utilizará sus funcionalidades para explorar los datos y manipularlos de manera que sean fácilmente legibles.

Recursos necesarios

- 1 computadora con acceso a Internet
- Raspberry Pi versión 2 o superior
- Bibliotecas de Python: datetime, csv, subprocess, pandas, numpy
- Archivos de datos: rpi_data_long.csv

Parte 1: Recopile y guarde los datos

El objetivo de esta primera parte de la práctica de laboratorio es reunir mediciones de la velocidad de Internet a través de Raspberry Pi. Se recogerán tres tipos de mediciones:

1. Velocidad de ping
2. Velocidad de descarga
3. Velocidad de carga

Paso 1: Instale Speedtest e importe las bibliotecas de Python.

En este paso, instalará Speedtest e importará las bibliotecas de Python.

- a) Instale speedtest-cli.

Este cli permite que la notebook de Jupyter se conecte a la página web y guarde los datos.

In [12]: `!pip install speedtest-cli`

Requirement already satisfied: speedtest-cli in c:\programdata\anaconda3\lib\site-packages (2.1.3)

b) Importe las bibliotecas de Python necesarias.

```
In [1]: # Code cell 2
# Python Library to manage date and time data
import datetime
# Python Library to read and write csv files
import csv
# Python Library to execute bash commands from the notebook.
# If you want to know more about this, check this resource:
# http://www.pythonforbeginners.com/os/subprocess-for-system-administrators
import subprocess
```

Paso 2: Genere las marcas de hora mediante el paquete datetime.

En esta práctica de laboratorio, se generarán mediciones de las estadísticas de la velocidad de Internet. Un paso crucial en la adquisición de datos para la mayoría de las aplicaciones de análisis de datos es asociar una marca de hora a las mediciones.

a) Para generar una marca de hora, utilice la función datetime.now del paquete datetime:

```
In [2]: date_time = datetime.datetime.now()
print(date_time, type(date_time))

2022-09-28 09:10:20.546223 <class 'datetime.datetime'>
```

b) Una instancia de la clase datetime no se puede escribir directamente en forma de texto. La función strftime analiza la información de fecha en una cadena. Los argumentos de esta función determinan el formato del sting de salida. Una descripción de estos parámetros se encuentra en la documentación de la función strftime en <https://docs.python.org/2/library/time.html>.

```
In [3]: date_time.strftime('%a, %d %b %Y %H:%M:%S')

Out[3]: 'Tue, 27 Sep 2022 10:34:38'
```

Después de leer la documentación de la función strftime, genere una marca de hora y analícela en una cadena con el siguiente formato: AAAA-MM-DD HH:MM:SS.

```
In [4]: # Code cell 5
# Enter your code
date_time.strftime('%Y-%d-%d,%H:%M:%S')

Out[4]: '2022-27-27,08:51:33'
```

Paso 3: Ejecute el proceso y recopile la salida con Python.

El comando speedtest-cli, si se ejecuta desde un terminal, devuelve una cadena con las velocidades de carga y descarga. Para ejecutar el comando de la computadora portátil, es necesario utilizar el subprocesso del módulo de Python, que permite la ejecución de un proceso directamente de las celdas de código de la computadora portátil.

a) Ejecute una prueba de velocidad mediante el comando speedtest-cli de Python. La salida se almacenará en la variable process_output.

```
In [3]: # Code cell 6
# This string contains the command Line to interface with speedtest.net

speedtest_cmd = "speedtest-cli --version"
# Execute the process
process = subprocess.Popen(speedtest_cmd.split(), stdout=subprocess.PIPE)
# Collect the command output
process_output = process.communicate()[0]
```

b) Imprima la salida del proceso. Observe el tipo para la variable process_output.

```
In [4]: # Code cell 7
print(process_output, type(process_output))

b' ' <class 'bytes'>
```

c) El resultado de la prueba de velocidad se divide, y una marca de hora se adjunta a los resultados.

```
In [5]: # Code cell 8
# Store the time at which the speedtest was executed
date_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
process_output = process_output.split()
process_output.append(date_time)

print(process_output, type(process_output))

['2022-09-28 09:10:32'] <class 'list'>
```

d) La función speedtest() se crea para devolver los resultados del comando speedtest-cli.

```
In [6]: # Code cell 9
# function to execute the speed test
def speedtest():
    # We need to store the time at which the speedtest was executed
    date_time = datetime.datetime.now().strftime("%Y-%m-%d %H:%M:%S")
    # This is a string that contains what we would write on the command Line
    #to interface with speedtest.net
    speedtest_cmd = "speedtest-cli --simple"
    # We now execute the process:
    process = subprocess.Popen(speedtest_cmd.split(), stdout=subprocess.PIPE)
    process_output = process.communicate()[0]
    process_output = process_output.split()
    # and we add the date and time
    process_output.append(date_time)
    return process_output
```

¿Qué devuelve la función speedtest()? ¿Cuál es el código para ver resultados de la función speedtest()?

Escriba la respuesta aquí

```
In [7]: print(speedtest())

['2022-09-28 09:10:37']
```

Paso 4: Guarde la salida de la función speedtest().

Los valores separados por comas (csv) son el formato de importación y exportación más común para las hojas de cálculo y las bases de datos. Para obtener más información sobre el

trabajo con los csv en Python, navegue a <https://docs.python.org/2/library/csv.html>.

a) Cree un archivo denominado test.txt en el directorio /tmp y escriba "test_msg" en el archivo.

```
In [8]: # Code cell 11
with open('test.txt', 'w') as f:
    f.write('test_msg')

#file_name = "C:\\\\Usuarios\\\\yulis\\\\AppData\\\\Local\\\\Temp"
#with open(file_name, "w") as f:
#    f.write('test_msg')
```

b) Use el comando de Linux cat para verificar la creación y el contenido del archivo.

```
In [49]: !cat /tmp/test.txt
```

"cat" no se reconoce como un comando interno o externo,
programa o archivo por lotes ejecutable.

c) Para verificar que el archivo se haya abierto correctamente:

```
In [9]: # Code cell 13
with open("test.txt", 'r') as f:
    str = f.read()
print(str)
```

test_msg

d) La comprensión del significado de la declaración with, especialmente en combinación con try y except no es necesaria para el resto de esta práctica de laboratorio, pero un recurso útil sobre esto es <http://effbot.org/zone/python-with-statement.htm>.

Para escribir en el archivo CSV, es necesario crear un objeto csv.writer. Marque <https://docs.python.org/2/library/csv.html> y descubra qué función del objeto "csv.writer" se puede utilizar para agregar una fila al archivo csv.

```
In [40]: # Code cell 14
# Function to save data to csv
def save_to_csv(data, filename):
    try:
        # If the file exists, append a new Line to it, with the
        # results of the current experiment
        with open(filename + '.csv', 'a') as f:
            wr = csv.writer(f)
            wr.writerow(data)
    except:
        # If it does not exist, create the file first
        with open(filename + '.csv', 'w') as f:
            # Hint: This is similar to appending new Lines to a file.
            # Create a csv writer object
            wr = csv.writer(f)
            # Save (write) to file
            wr.writerow(data)
```

Paso 5: Verifique los datos recolectados.

Escriba una función para abrir un archivo csv e imprima su contenido en la pantalla. Puede encontrar un ejemplo en la sección 13.1.5 de <https://docs.python.org/2/library/csv.html>

In [41]:

```
# Code cell 15
def print_from_csv(filename):
    with open(filename + '.csv', 'r') as f:
        re = csv.read(f)
        # 1. Loop over the rows
        for row in re:
            # 2. print
            print(' '.join(row))
```

En este momento, todas las funciones necesarias para recopilar y almacenar los datos de la velocidad de Internet han finalizado.

Paso 6: Ejecute varias veces Speedtest y guarde los datos.

a) Escriba un bucle for que ejecute speedtest 5 veces, imprima la salida de las pruebas y guarde los datos en un archivo csv.

In [45]:

```
# Code cell 16
for i in range(5):
    # Measure internet speed (speedtest_output) using the speedtest() created earlier
    speedtest_output = speedtest()
    # Print the test number
    print('Test number {}'.format(i))
    # Print the result (The needed variable is speedtest_output)
    print(speedtest_output)
    save_to_csv(speedtest_output, 'rpi_data_test')
```

```
Test number 0
['2022-09-28 10:17:47']
Test number 1
['2022-09-28 10:17:47']
Test number 2
['2022-09-28 10:17:47']
Test number 3
['2022-09-28 10:17:47']
Test number 4
['2022-09-28 10:17:48']
```

b) Muestre el archivo para verificar que los datos se hayan guardado correctamente.

In [47]:

```
# Code cell 17
print_from_csv('rpi_data_test')
```

```
-----
AttributeError                                     Traceback (most recent call last)
Input In [47], in <cell line: 2>()
      1 # Code cell 17
----> 2 print_from_csv('rpi_data_test')

Input In [41], in print_from_csv(filename)
      2 def print_from_csv(filename):
      3     with open(filename + '.csv', 'r') as f:
----> 4         re = csv.read(f)
      5         # 1. Loop over the rows
      6         for row in re:
      7             # 2. print

AttributeError: module 'csv' has no attribute 'read'
```

Si un conjunto de datos más grande es necesario, speedtest puede ejecutarse en segundo plano para más muestras.

¿Cómo cambiaría el código si quisiera ejecutar el código 100 veces?

Escriba la respuesta aquí

```
In [ ]: # Code cell 18
# Code to run 100 times
for i in range(100):
    speedtest_output = speedtest()
    print('Test number: {}'.format(i))
    print(speedtest_output)
    save_to_csv(speedtest_output, 'rpi_data_test')
```

Parte 2: Manipule los datos

Los pandas de la biblioteca de Python son muy útiles para trabajar con datos estructurados. La documentación completa se encuentra aquí: <http://pandas.pydata.org/pandas-docs/version/0.14.1/>

Se utilizará un conjunto de datos más grande recopilado de antemano para esta parte de la práctica de laboratorio. El nombre de archivo es rpi_data_long.csv.

Paso 1: Importe las bibliotecas de Python.

Importe los pandas y otras bibliotecas utilizadas para las siguientes tareas.

```
In [11]: # Code cell 19
import datetime
import csv
import pandas as pd
# NumPy is a library that adds support for large, multi-dimensional arrays and matrices
# along with high-level mathematical functions to operate on these arrays
import numpy as np
```

Paso 2: Cargue el archivo csv en un objeto del marco de datos mediante pandas.

Un marco de datos de pandas es una estructura de datos etiquetada de 2 dimensiones con columnas potencialmente de diferentes tipos. Puede verlo como una hoja de cálculo o una tabla de SQL. La función de la biblioteca de pandas read_csv convierte automáticamente un archivo csv en un objeto del marco de datos.

Lea la documentación de read_csv en http://pandas.pydata.org/pandas-docs/version/0.14.1/generated/pandas.read_csv.html. Esta función contiene muchos parámetros. El único no opcional es filepath, es decir, la ubicación del archivo csv. Todos los demás parámetros son opcionales.

En este paso, importará y verá el contenido del archivo csv, rpi_data_long.csv. Esto archivo csv se encuentra en el mismo directorio que esta notebook de Jupyter.

a) Asigne el archivo rpi_data_long.csv a la variable data_file.

```
In [12]: # Code cell 20
data_file = './Data/rpi_data_long.csv'
#data_file = pd.read_csv("./Data/rpi_data_Long.csv")
data_file

Out[12]: './Data/rpi_data_long.csv'
```

b) Utilice el comando head de Linux para ver las primeras 10 líneas del archivo csv.

```
In [14]: !head -n 5 rpi_data_long.csv
```

c) Utilice el parámetro names de la función read_csv para especificar el nombre de las columnas del marco de datos.

Code cell 22 column_names = ['Type A', 'Measure A', 'Units A', 'Type B', 'Measure B', 'Units B', 'Type C', 'Measure C', 'Units C', 'Datetime']

```
In [13]: column_names = [ 'Type A', 'Measure A', 'Units A',
                     'Type B', 'Measure B', 'Units B',
                     'Type C', 'Measure C', 'Units C',
                     'Datetime']
```

d) Utilice la función read_csv para leer de data_file y asigne column_names como los nombres de columna en el marco de datos.

```
In [14]: # Code cell 23
with open(data_file, 'r') as f:
    df_redundant = pd.read_csv(f, names = column_names)

#df_redundant = pd.read_csv("./Data/rpi_data_Long.csv", names = column_names)
#df_redundant
```

e) El comando head() muestra las primeras filas del marco de datos.

```
In [15]: df_redundant.head()
```

Out[15]:	Type A	Measure A	Units A	Type B	Measure B	Units B	Type C	Measure C	Units C	Datetime
0	Ping:	26.992	ms	Download:	91.80	Mbit/s	Upload:	14.31	Mbit/s	2016-11-24 13:36:25
1	Ping:	24.532	ms	Download:	88.19	Mbit/s	Upload:	14.12	Mbit/s	2016-11-24 13:36:55
2	Ping:	20.225	ms	Download:	59.86	Mbit/s	Upload:	14.11	Mbit/s	2016-11-24 13:37:25
3	Ping:	19.332	ms	Download:	91.81	Mbit/s	Upload:	14.22	Mbit/s	2016-11-24 13:37:57
4	Ping:	22.494	ms	Download:	92.05	Mbit/s	Upload:	14.08	Mbit/s	2016-11-24 13:38:27

¿Cuál es el código para leer las primeras 20 líneas del archivo csv?

Escriba la respuesta aquí

In [16]: `df_redundant.head(20)`

Out[16]:	Type A	Measure A	Units A	Type B	Measure B	Units B	Type C	Measure C	Units C	Datetime
0	Ping:	26.992	ms	Download:	91.80	Mbit/s	Upload:	14.31	Mbit/s	2016-11-24 13:36:25
1	Ping:	24.532	ms	Download:	88.19	Mbit/s	Upload:	14.12	Mbit/s	2016-11-24 13:36:55
2	Ping:	20.225	ms	Download:	59.86	Mbit/s	Upload:	14.11	Mbit/s	2016-11-24 13:37:25
3	Ping:	19.332	ms	Download:	91.81	Mbit/s	Upload:	14.22	Mbit/s	2016-11-24 13:37:57
4	Ping:	22.494	ms	Download:	92.05	Mbit/s	Upload:	14.08	Mbit/s	2016-11-24 13:38:27
5	Ping:	17.586	ms	Download:	91.88	Mbit/s	Upload:	14.18	Mbit/s	2016-11-24 13:39:01
6	Ping:	21.835	ms	Download:	92.18	Mbit/s	Upload:	14.07	Mbit/s	2016-11-24 13:39:30
7	Ping:	20.464	ms	Download:	92.05	Mbit/s	Upload:	14.13	Mbit/s	2016-11-24 13:40:00
8	Ping:	19.293	ms	Download:	90.79	Mbit/s	Upload:	14.14	Mbit/s	2016-11-24 13:40:30
9	Ping:	20.354	ms	Download:	92.64	Mbit/s	Upload:	14.07	Mbit/s	2016-11-24 13:41:00
10	Ping:	19.01	ms	Download:	91.86	Mbit/s	Upload:	14.38	Mbit/s	2016-11-24 13:41:30
11	Ping:	18.092	ms	Download:	92.24	Mbit/s	Upload:	14.23	Mbit/s	2016-11-24 13:42:00
12	Ping:	18.137	ms	Download:	91.64	Mbit/s	Upload:	14.32	Mbit/s	2016-11-24 13:42:30
13	Ping:	18.344	ms	Download:	92.85	Mbit/s	Upload:	14.20	Mbit/s	2016-11-24 13:43:00
14	Ping:	23.328	ms	Download:	92.21	Mbit/s	Upload:	14.20	Mbit/s	2016-11-24 13:43:29
15	Ping:	17.908	ms	Download:	92.24	Mbit/s	Upload:	11.36	Mbit/s	2016-11-24 13:43:59

	Type A	Measure A	Units A	Type B	Measure B	Units B	Type C	Measure C	Units C	Datetime
16	Ping:	19.291	ms	Download:	92.41	Mbit/s	Upload:	14.03	Mbit/s	2016-11-24 13:44:31
17	Ping:	20.117	ms	Download:	90.39	Mbit/s	Upload:	14.17	Mbit/s	2016-11-24 13:45:01
18	Ping:	19.579	ms	Download:	92.55	Mbit/s	Upload:	14.06	Mbit/s	2016-11-24 13:45:32
19	Ping:	16.364	ms	Download:	78.51	Mbit/s	Upload:	13.71	Mbit/s	2016-11-24 13:46:02

Paso 3: Cree una representación concisa.

En este paso, creará una representación más compacta mediante una copia del marco de datos df冗余.

a) Copie df冗余 en otro marco de datos llamado df_compact mediante copy().

```
In [17]: df_compact = df冗余.copy()
```

b) Cambie el nombre de las columnas en relación con las medidas como se muestra:

Measure A -> Ping (ms)

Measure B -> Download (Mbit/s)

Measure C -> Upload (Mbit/s)

```
In [18]: df_compact.rename(columns={'Measure A': 'Ping (ms)',  
                                'Measure B': 'Download (Mbit/s)',  
                                'Measure C': 'Upload (Mbit/s)'}, inplace=True)  
df_compact.head(3)
```

	Type A	Ping (ms)	Units A	Type B	Download (Mbit/s)	Units B	Type C	Upload (Mbit/s)	Units C	Datetime
0	Ping:	26.992	ms	Download:	91.80	Mbit/s	Upload:	14.31	Mbit/s	2016-11-24 13:36:25
1	Ping:	24.532	ms	Download:	88.19	Mbit/s	Upload:	14.12	Mbit/s	2016-11-24 13:36:55
2	Ping:	20.225	ms	Download:	59.86	Mbit/s	Upload:	14.11	Mbit/s	2016-11-24 13:37:25

c) Como las columnas de los tipos y las unidades ya no son necesarias, estas columnas pueden descartarse.

```
In [19]: df_compact.drop(['Type A', 'Type B', 'Type C',
```

```
'Units A', 'Units B', 'Units C'], axis=1, inplace=True)
df_compact.head()
```

Out[19]:

	Ping (ms)	Download (Mbit/s)	Upload (Mbit/s)	Datetime
0	26.992	91.80	14.31	2016-11-24 13:36:25
1	24.532	88.19	14.12	2016-11-24 13:36:55
2	20.225	59.86	14.11	2016-11-24 13:37:25
3	19.332	91.81	14.22	2016-11-24 13:37:57
4	22.494	92.05	14.08	2016-11-24 13:38:27

En la tabla anterior, el campo Datetime es una cadena. Los pandas y Python ofrecen varias operaciones para trabajar con la fecha y la hora que pueden ser muy útiles.

En el siguiente paso, la cadena de la columna Datetime se separará en dos columnas nuevas.

Paso 4: Separe los datos en dos columnas.

En este paso, utilizará los pandas para generar las columnas Date y Time a partir de la columna Datetime y luego descartará la columna Datetime.

Se utiliza la función lambda para crear dos funciones anónimas que recuperen sólo la fecha y la hora de un objeto datetime, respectivamente. Luego utilice la función de los pandas apply para aplicar esta función a una columna completa (en la práctica, apply implícitamente define un bucle for y pasa las filas una por una en nuestra función lambda). Guarde el resultado de las funciones apply en dos nuevas columnas del marco de datos.

a) Aplique la función lambda para iterar a través del marco de datos y dividir la fecha de la columna Datetime.

In [21]:

```
# Code cell 28
df_compact['Date'] = df_compact['Datetime'].apply(lambda dt_str: pd.to_datetime(dt_
```

b) Repita el paso a para dividir la hora de la columna Datetime.

In [23]:

```
temp = df_compact['Datetime'].apply(lambda dt_str: pd.to_datetime(dt_str))
df_compact['Time'] = temp.dt.time
```

c) Toda la información de la columna Datetime se ha copiado ahora a las columnas Date y Time. La columna Datetime carece de propósito. La columna Datetime se puede descartar del marco de datos.

Ingrese el código para descartar la columna Datetime en la celda a continuación.

In [24]:

```
df_compact.drop(['Datetime'], axis=1, inplace=True)
df_compact.head()
```

	Ping (ms)	Download (Mbit/s)	Upload (Mbit/s)	Date	Time
0	26.992	91.80	14.31	2016-11-24	13:36:25
1	24.532	88.19	14.12	2016-11-24	13:36:55
2	20.225	59.86	14.11	2016-11-24	13:37:25
3	19.332	91.81	14.22	2016-11-24	13:37:57
4	22.494	92.05	14.08	2016-11-24	13:38:27

Ingrese el código para imprimir las primeras 3 filas del marco de datos para verificar los cambios.

In [25]: `df_compact.head(3)`

	Ping (ms)	Download (Mbit/s)	Upload (Mbit/s)	Date	Time
0	26.992	91.80	14.31	2016-11-24	13:36:25
1	24.532	88.19	14.12	2016-11-24	13:36:55
2	20.225	59.86	14.11	2016-11-24	13:37:25

d) Utilice la función type para imprimir el tipo de variable de los valores en las columnas Date y Time.

In [27]: `print(df_compact['Date'][0], type(df_compact['Date'][0]))`

2016-11-24 <class 'datetime.date'>

In [28]: `print(df_compact['Time'][0], type(df_compact['Time'][0]))`

13:36:25 <class 'datetime.time'>

Paso 5: Guarde el nuevo marco de datos.

Guarde el marco de datos de pandas df_compact como un archivo csv en rpi_data_compact.csv.

In [26]: `# Code cell 33
df_compact.to_csv('./Data/rpi_data_compact.csv')`