MÁSTER DE DATA SCIENCE

Práctica 1

Estudiante Diego Armando Cale Pillco **Fecha** 11/11/2019

En esta práctica se elabora un caso práctico orientado a aprender a identificar los datos relevante herramientas de extracción de datos.

1. Contexto. Explicar en qué contexto se ha recolectado la información. Explique por qué el sitio

Para el desarrollo de la presente página se ha realizado de la página https://www.hubertiming.con cortometraje de las carreras desde 2009 y actualmente cronometra más de 60 carreras al año. Pr comprensión profunda de la gestión de eventos y las operaciones de carrera, y un alto nivel de ate

El conjunto de datos utilizado en esta práctica se tomó de una carrera de 5 km que tuvo lugar en M febrero de 2019. Del siguiente sitio web: https://www.hubertiming.com/results/2019WorstDay

Específicamente, analizará el rendimiento de los corredores de 5 km y responderá preguntas com

- ¿Cuál fue el tiempo promedio de finalización para los corredores?
- ¿Los tiempos de finalización de los corredores siguieron una distribución normal?
- ¿Hubo alguna diferencia de rendimiento entre hombres y mujeres de varios grupos de edad?

Web Scraping usando Python

Web Scraping usando Beautiful Soup

Comienzo a importando los módulos necesarios (pandas, numpy, matplotlib.pyplot, seaborn). Par de incluir la línea% matplotlib en línea como se muestra a continuación.

```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
%matplotlib inline
```

Para realizar el raspado web, también debe importar las bibliotecas que se muestran a continuaci El paquete Beautiful Soup se usa para extraer datos de archivos html. El nombre de la biblioteca d Beautiful Soup, versión 4.

```
import requests
from bs4 import BeautifulSoup
```

Después de importar los módulos necesarios, debe especificar la URL que contiene el conjunto de

```
url='https://www.hubertiming.com/results/2019WorstDay'
pagina_html = requests.get(url)
pagina_html=pagina_html.text
```

Obtener el html de la página es solo el primer paso. El siguiente paso es crear un objeto Beautiful el html a la función BeautifulSoup (). El paquete Beautiful Soup se usa para analizar el html, es dec en objetos Python. El segundo argumento 'lxml' es el analizador html cuyos detalles no necesita p

```
from bs4 import BeautifulSoup
soup = BeautifulSoup(pagina_html, 'lxml')
type(soup)
```

```
bs4.BeautifulSoup
```

El objeto soup le permite extraer información interesante sobre el sitio web que está raspando, co muestra a continuación.

```
# Obtener título de la página
title = soup.title
print(title)
```

```
<title>2019 Worst Day of the Year 5K Race Results</title>
```

También puede obtener el texto de la página web e imprimirlo rápidamente para verificar si es lo q

```
# Imprimir los elementos de la págna Html
text = soup.get_text()
#print(soup.text)
```

Para imprimir solo las filas de la tabla, pase el argumento 'tr' en soup.find_all ().

```
# Print the first 100 rows for sanity check
rows = soup.find_all('tr')
print(rows[:100])
```

```
[<tr colspan="2"><b>5K:</b></tr>, <tr><td>Finishers:</td><td>123</td></tr>, <tr><td>M
```

El objetivo de este tutorial es tomar una tabla de una página web y convertirla en un marco de dat Python. Para llegar allí, primero debe obtener todas las filas de la tabla en forma de lista y luego c continuación se muestra un bucle for que recorre las filas de la tabla e imprime las celdas de las fi

```
for row in rows:
    row_td = row.find_all('td')
print(row_td)
type(row_td)
```

```
[<td>123</td>, <td>1444</td>, <td>DAWN GEOPPINGER</td>, <td>F</td>, <td>40</td>, <td>
bs4.element.ResultSet
```

El resultado anterior muestra que cada fila se imprime con etiquetas html incrustadas en cada fila eliminar las etiquetas html usando Beautiful Soup o expresiones regulares.

La forma más fácil de eliminar etiquetas html es usar Beautiful Soup, y solo se necesita una línea interés a BeautifulSoup () y use el método get_text () para extraer el texto sin etiquetas html.

```
str_cells = str(row_td)
cleantext = BeautifulSoup(str_cells, "lxml").get_text()
print(cleantext)
```

```
[123, 1444, DAWN GEOPPINGER, F, 40, PORTLAND, OR, 2:09:59, 41:56, 79 of 79, F 40-44,
```

Se desaconseja el uso de expresiones regulares, ya que requiere varias líneas de código y uno pue importar el módulo re (para expresiones regulares). El siguiente código muestra cómo crear una e caracteres dentro de las etiquetas html y reemplazarlos con una cadena vacía para cada fila de la pasando una cadena para que coincida con re.compile (). El punto, la estrella y el signo de interrog angular de apertura seguido de cualquier cosa y seguido de un paréntesis angular de cierre. Coinc es decir, coincide con la cadena más corta posible. Si omite el signo de interrogación, coincidirá c angular de apertura y el último paréntesis angular de cierre. Después de compilar una expresión re buscar todas las subcadenas donde coincida la expresión regular y reemplazarlas con una cadena genera una lista vacía, extrae el texto entre las etiquetas html para cada fila y lo agrega a la lista as

```
import re

list_rows = []
for row in rows:
    cells = row.find_all('td')
    str_cells = str(cells)
    clean = re.compile('<.*?>')
    clean2 = (re.sub(clean, '',str_cells))
    list_rows.append(clean2)
print(clean2)
type(clean2)
```

```
[123, 1444, DAWN GEOPPINGER, F, 40, PORTLAND, OR, 2:09:59, 41:56, 79 of 79, F 40-44,
str
```

El siguiente paso es convertir la lista en un marco de datos y obtener una vista rápida de las prime

```
import pandas as pd
df = pd.DataFrame(list_rows)
df.head(10)
```

	0
0	[]
1	[Finishers:, 123]
2	[Male:, 44]
3	[Female:, 79]
4	[]
5	[1, 1378, BRADY BEAGLEY, M, 27, PORTLAND, OR, ...
6	[2, 1302, SCOTT GULLICKSON, M, 55, RIDGEFIELD,...
7	[3, 1364, JAMES DAVIS, M, 15, SHERWOOD, OR, 22...
8	[4, 1330, KEVIN ERICKSON, M, 34, VANCOUVER, WA...
9	[5, 1372, STEPHEN WILLIAMS, M, 24, PORTLAND, O...

Manipulación y limpieza de datos

El marco de datos no está en el formato que queremos. Para limpiarlo, debe dividir la columna "0" Esto se logra utilizando el método str.split ().

```
df1 = df[0].str.split(',', expand=True)
df1.head(10)
```

	0	1	2	3	4	5	6	7	8	9
0	[]	None	None	None	None	None	None	None	None	None
1	[Finishers:	123]	None	None	None	None	None	None	None	None
2	[Male:	44]	None	None	None	None	None	None	None	None
3	[Female:	79]	None	None	None	None	None	None	None	None
4	[]	None	None	None	None	None	None	None	None	None
5	[1	1378	BRADY BEAGLEY	M	27	PORTLAND	OR	19:17	6:13	1 of 44
6	[2	1302	SCOTT GULLICKSON	M	55	RIDGEFIELD	WA	20:55	6:45	2 of 44
7	[3	1364	JAMES	M	15	SHERWOOD	OR	22:47	7:21	3 of

Esto se ve mucho mejor, pero aún queda trabajo por hacer. El marco de datos tiene corchetes no c método strip () para quitar el corchete de apertura en la columna "0".

```
df1[0] = df1[0].str.strip('[')
```

```
df1[0] = df1[0].str.strip(']')
df1.head(10)
```

	0	1	2	3	4	5	6	7	8	9	
0		None	None	None	None	None	None	None	None	None	N
1	Finishers:	123]	None	None	None	None	None	None	None	None	N
2	Male:	44]	None	None	None	None	None	None	None	None	N
3	Female:	79]	None	None	None	None	None	None	None	None	N
4		None	None	None	None	None	None	None	None	None	N
5	1	1378	BRADY BEAGLEY	M	27	PORTLAND	OR	19:17	6:13	1 of 44	
6	2	1302	SCOTT GULLICKSON	M	55	RIDGEFIELD	WA	20:55	6:45	2 of 44	
7	3	1364	JAMES	M	15	SHERWOOD	OR	22:47	7:21	3 of	

A la tabla le faltan encabezados de tabla. Puede usar el método find_all () para obtener los encabe

```
col_labels = soup.find_all('th')
```

Similar a las filas de la tabla, puede usar Beautiful Soup para extraer texto entre etiquetas html par

```
all_header = []
col_str = str(col_labels)
cleantext2 = BeautifulSoup(col_str, "lxml").get_text()
all_header.append(cleantext2)
print(all_header)
```

```
[u'[Place, Bib, Name, Gender, Age, City, State, Chip Time, Chip Pace, Gender Place, A
```

Luego puede convertir la lista de encabezados en un marco de datos de pandas.

```
df2 = pd.DataFrame(all_header)
all_header
```

```
[u'[Place, Bib, Name, Gender, Age, City, State, Chip Time, Chip Pace, Gender Place, A
```

Del mismo modo, puede dividir la columna "0" en varias columnas en la posición de coma para toc

```
df3 = df2[0].str.split(',', expand=True)
df3.head()
```

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	[Place	Bib	Name	Gender	Age	City	State	Chip Time	Chip Pace	Gender Place	Age Group	Age Group	Time to

Los dos marcos de datos se pueden concatenar en uno usando el método concat () como se ilust

```
frames = [df3, df1]

df4 = pd.concat(frames)
df4.head(10)
```

	0	1	2	3	4	5	6	7	8	
0	[Place	Bib	Name	Gender	Age	City	State	Chip Time	Chip Pace	Gende Plac
0		None	None	None	None	None	None	None	None	Non
1	Finishers:	123]	None	None	None	None	None	None	None	Non
2	Male:	44]	None	None	None	None	None	None	None	Non
3	Female:	79]	None	None	None	None	None	None	None	Non
4		None	None	None	None	None	None	None	None	Non
5	1	1378	BRADY BEAGLEY	M	27	PORTLAND	OR	19:17	6:13	1 of 4
6	2	1302	SCOTT GULLICKSON	M	55	RIDGEFIELD	WA	20:55	6:45	2 of 4

A continuación se muestra cómo asignar la primera fila para que sea el encabezado de la tabla.

```
df5 = df4.rename(columns=df4.iloc[0])
df5.head()
```

	[Place	Bib	Name	Gender	Age	City	State	Chip Time	Chip Pace	Gender Place	Age Group	Ag Grou Plac
0	[Place	Bib	Name	Gender	Age	City	State	Chip Time	Chip Pace	Gender Place	Age Group	Ag Grou Plac
0		None	None	None	None	None	None	None	None	None	None	Non
1	Finishers:	123]	None	None	None	None	None	None	None	None	None	Non

En este punto, la tabla está formateada casi correctamente. Para el análisis, puede comenzar obte como se muestra a continuación.

```
df5.info()
```

```
df5.shape
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 129 entries, 0 to 127
Data columns (total 14 columns):
[Place              129 non-null object
 Bib                127 non-null object
 Name               124 non-null object
 Gender             124 non-null object
 Age                124 non-null object
 City               124 non-null object
 State              124 non-null object
 Chip Time          124 non-null object
 Chip Pace          124 non-null object
 Gender Place       124 non-null object
 Age Group          124 non-null object
 Age Group Place    124 non-null object
 Time to Start      124 non-null object
 Gun Time]          124 non-null object
dtypes: object(14)
memory usage: 15.1+ KB
(129, 14)
```

La tabla tiene 129 filas y 14 columnas. Puede soltar todas las filas con cualquier valor faltante.

```
df6 = df5.dropna(axis=0, how='any')
```

Además, observe cómo el encabezado de la tabla se replica como la primera fila en df5. Se puede

```
df7 = df6.drop(df6.index[0])
df7.head()
```

	[Place	Bib	Name	Gender	Age	City	State	Chip Time	Chip Pace	Gender Place	(
5	1	1378	BRADY BEAGLEY	M	27	PORTLAND	OR	19:17	6:13	1 of 44	
6	2	1302	SCOTT GULLICKSON	M	55	RIDGEFIELD	WA	20:55	6:45	2 of 44	
7	3	1364	JAMES DAVIS	M	15	SHERWOOD	OR	22:47	7:21	3 of 44	

Puede realizar más limpieza de datos cambiando el nombre de las columnas [Lugar y Equipo . Pyt
Asegúrese de incluir espacio después de las comillas en Equipo].

```
df7.rename(columns={'[Place': 'Place'},inplace=True)
df7.rename(columns={' Gun Time]': 'Team'},inplace=True)
df7.head()
```

	Place	Bib	Name	Gender	Age	City	State	Chip Time	Chip Pace	Gender Place	Gr
5	1	1378	BRADY BEAGLEY	M	27	PORTLAND	OR	19:17	6:13	1 of 44	M
6	2	1302	SCOTT GULLICKSON	M	55	RIDGEFIELD	WA	20:55	6:45	2 of 44	M
7	3	1364	JAMES DAVIS	M	15	SHERWOOD	OR	22:47	7:21	3 of 44	M

El paso final de limpieza de datos consiste en eliminar el corchete de cierre para las celdas en la c

```
df7['Team'] = df7['Team'].str.strip(']')
df7.head()
```

	Place	Bib	Name	Gender	Age	City	State	Chip Time
5	1	1378	BRADY BEAGLEY	M	27	PORTLAND	OR	19:17
6	2	1302	SCOTT GULLICKSON	M	55	RIDGEFIELD	WA	20:55
7	3	1364	JAMES DAVIS	M	15	SHERWOOD	OR	22:47
8	4	1330	KEVIN ERICKSON	M	34	VANCOUVER	WA	22:52

Guardo los datos en un archivo csv para luego comenzar a trazar los datos y calcular estadísticas

```
df7.to_csv("datosCarrera5k.csv")
```

Análisis de datos y visualización

La primera pregunta a responder es, ¿cuál fue el tiempo promedio de finalización (en minutos) par columna "Tiempo de chip" en solo minutos. Una forma de hacerlo es convertir la columna a una lis

```
time_list = df7[' Chip Time'].tolist()

# You can use a for loop to convert 'Chip Time' to minutes
time_mins = []
for i in time_list:
  tam=len(i.split(':'))
  if tam==2:
    m, s = i.split(':')
    math = (int(m) * 60 + int(s))/60
    time_mins.append(math)
  else:
    h, m, s = i.split(':')
    math = (int(h) * 3600 + int(m) * 60 + int(s))/60
```

```
    time_mins.append(math)
print(time_mins)
```

```
[19, 20, 22, 22, 24, 25, 25, 25, 26, 26, 26, 26, 26, 27, 27, 27, 28, 28, 28, 28, 28,
```

El siguiente paso es convertir la lista nuevamente en un marco de datos y crear una nueva column
del corredor expresados en solo minutos.

```
df7['Runner_mins'] = time_mins
df7.head()
```

	Place	Bib	Name	Gender	Age	City	State	Chip Time	Chip Pace	Gender Place	Gı
5	1	1378	BRADY BEAGLEY	M	27	PORTLAND	OR	19:17	6:13	1 of 44	M
6	2	1302	SCOTT GULLICKSON	M	55	RIDGEFIELD	WA	20:55	6:45	2 of 44	M
7	3	1364	JAMES DAVIS	M	15	SHERWOOD	OR	22:47	7:21	3 of 44	M
8	4	1330	KEVIN ERICKSON	M	34	VANCOUVER	WA	22:52	7:22	4 of 44	M

El siguiente código muestra cómo calcular estadísticas para columnas numéricas solo en el marc

```
df7.describe(include=[np.number])
```

	Runner_mins
count	123.000000
mean	38.390244
std	13.063925
min	19.000000
25%	29.000000
50%	36.000000
75%	44.500000
max	129.000000

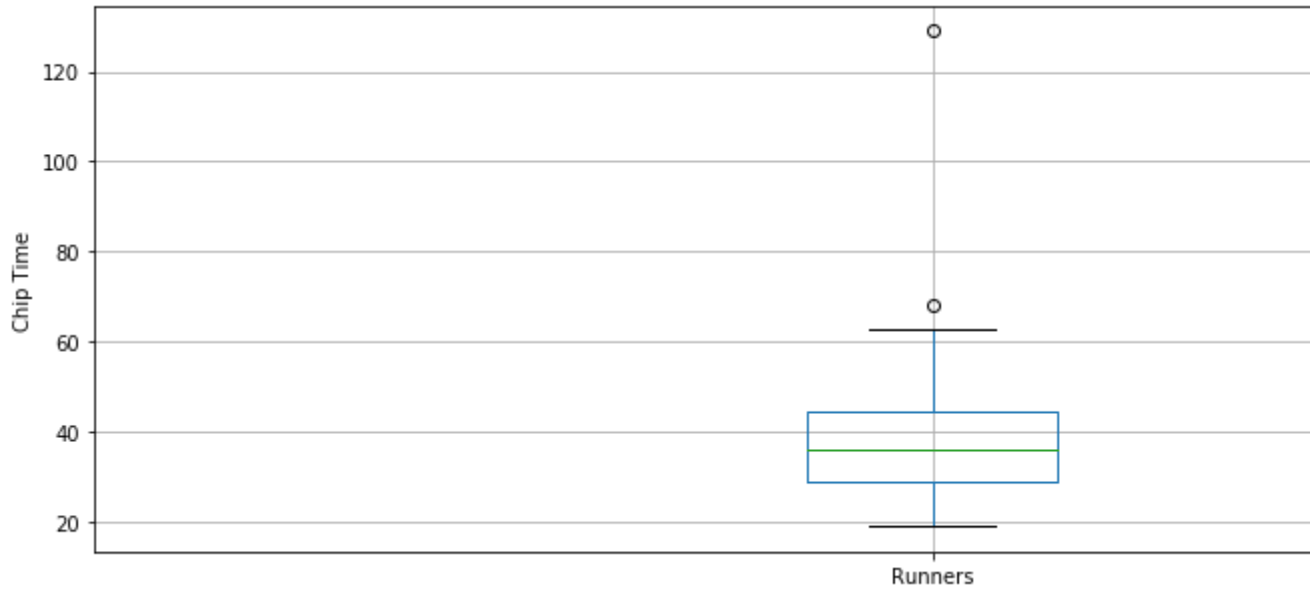
Curiosamente, el tiempo promedio de chip para todos los corredores fue de ~ 38.39 minutos. El c
minutos, y el corredor más lento terminó en 129 minutos.

Un diagrama de caja es otra herramienta útil para visualizar estadísticas resumidas (máximo, mín
incluidos los valores atípicos). A continuación se presentan estadísticas de resumen de datos par

diagrama de caja. Para la visualización de datos, es conveniente importar primero los parámetros establecer el mismo tamaño para todas las figuras para evitar hacerlo para cada figura.

```
from pylab import rcParams
rcParams['figure.figsize'] = 15, 5
df7.boxplot(column='Runner_mins')
plt.grid(True, axis='y')
plt.ylabel('Chip Time')
plt.xticks([1], ['Runners'])
```

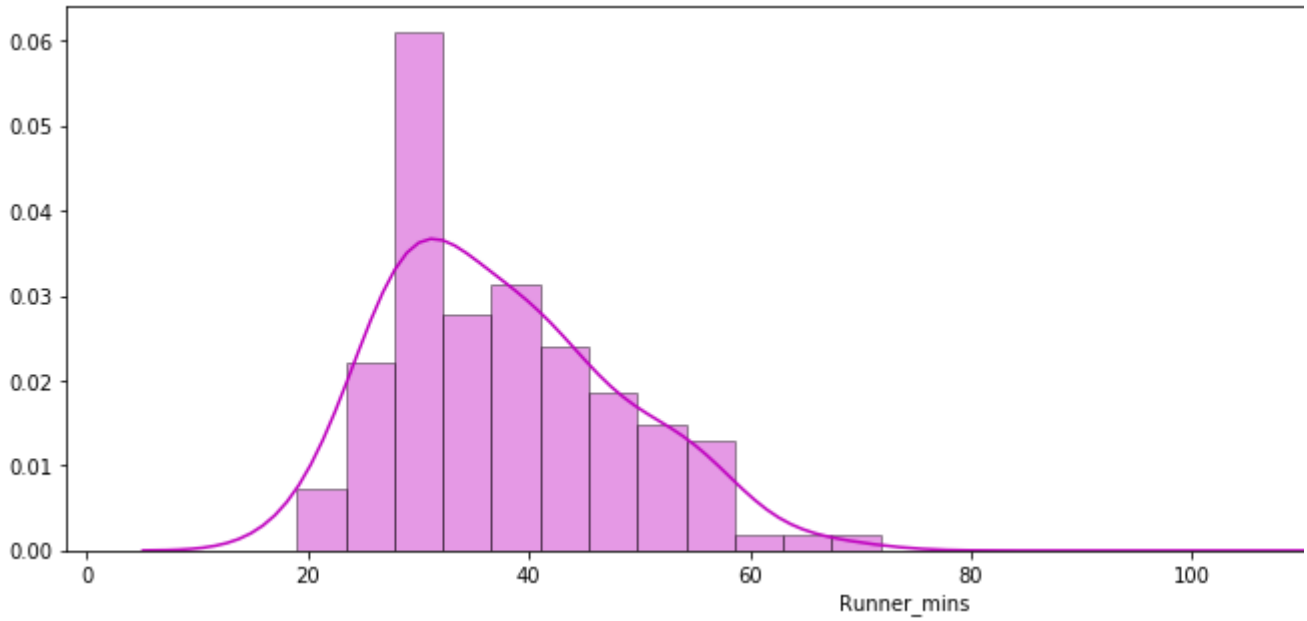
```
([<matplotlib.axis.XTick at 0x7f417c0f8c10>],
 <a list of 1 Text xticklabel objects>)
```



La segunda pregunta a responder es: ¿Los tiempos de finalización de los corredores siguieron una

A continuación se muestra un diagrama de distribución de los tiempos de chip de los corredores t distribución se ve casi normal.

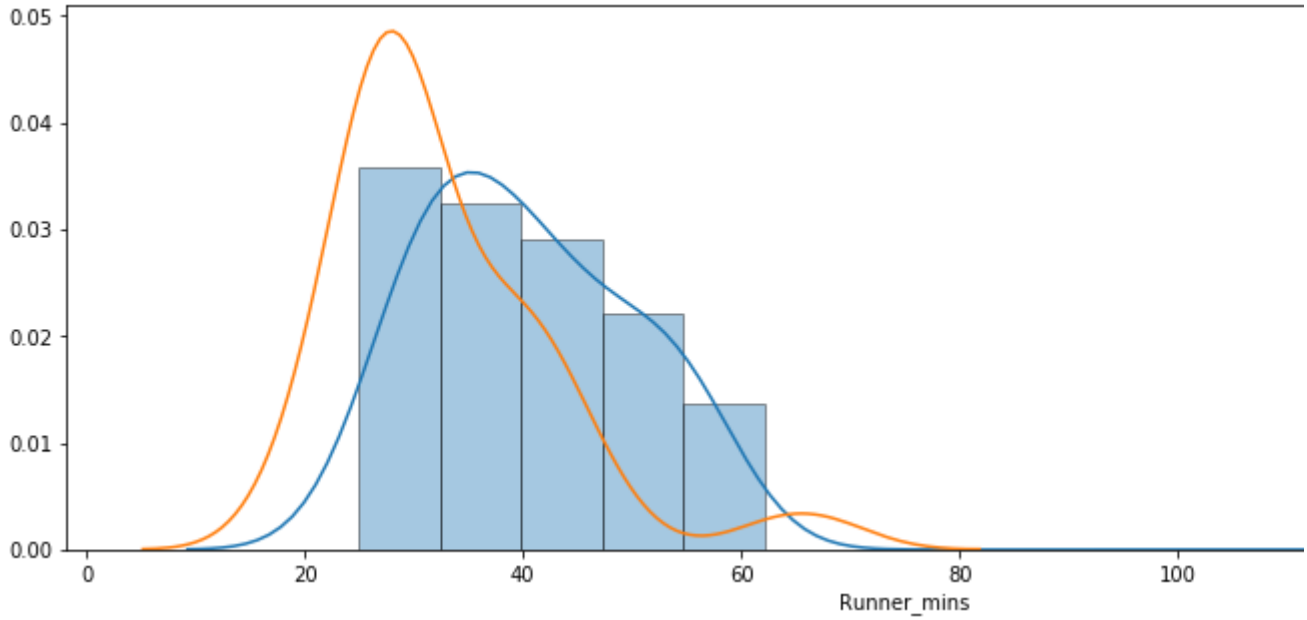
```
x = df7['Runner_mins']
ax = sns.distplot(x, hist=True, kde=True, rug=False, color='m', bins=25, hist_kws={'edgeco
plt.show()
```



La tercera pregunta se refiere a si hubo diferencias de rendimiento entre hombres y mujeres de va
muestra un diagrama de distribución de tiempos de chip para hombres y mujeres.

```
f_fuko = df7.loc[df7[' Gender']==' F'][' Runner_mins']
m_fuko = df7.loc[df7[' Gender']==' M'][' Runner_mins']
sns.distplot(f_fuko, hist=True, kde=True, rug=False, hist_kws={'edgecolor':'black'}, label
sns.distplot(m_fuko, hist=False, kde=True, rug=False, hist_kws={'edgecolor':'black'}, labe
plt.legend()
```

```
<matplotlib.legend.Legend at 0x7f417c141e10>
```



La distribución indica que las mujeres fueron más lentas que los hombres en promedio. Puede us
estadísticas de resumen para hombres y mujeres por separado, como se muestra a continuación

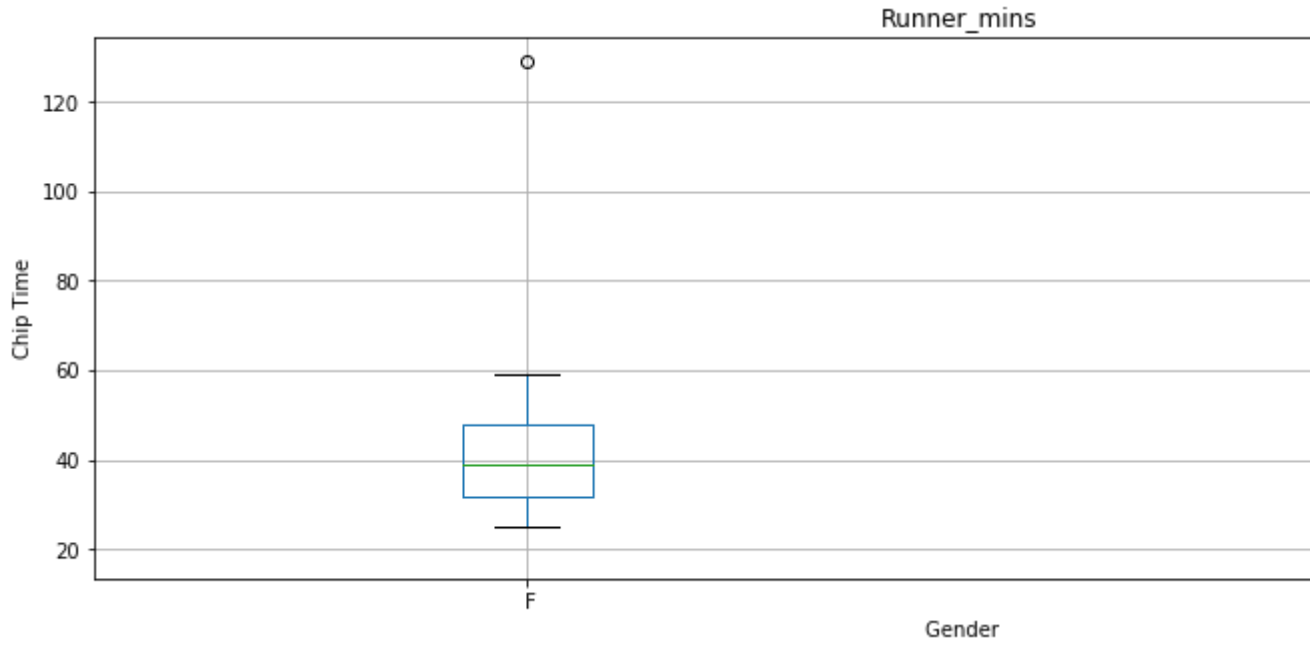
```
g_stats = df7.groupby(" Gender", as_index=True).describe()
print(g_stats)
```

```
        Runner_mins
              count       mean        std   min    25%   50%    75%    max
Gender
F              79.0  41.506329  13.544678  25.0  32.00  39.0  48.00  129.0
M              44.0  32.795455  10.077789  19.0  26.75  29.0  39.25   68.0
```

El tiempo promedio de astillas para todas las mujeres y los hombres fue de ~ 41 minutos y ~ 44 m
muestra una comparación de diagrama de caja de lado a lado de los tiempos de acabado masculi

```
df7.boxplot(column='Runner_mins', by=' Gender')
plt.ylabel('Chip Time')
plt.suptitle("")
```

```
Text(0.5,0.98,'')
```



Conclusión

En esta práctica, realizó el raspado web con Python. Usó la biblioteca Beautiful Soup para analizar
que pueda usarse para análisis. Realizó la limpieza de los datos en Python y creó gráficos útiles (g
de distribución) para revelar tendencias interesantes utilizando matplotlib y las bibliotecas navales
poder usar Python para extraer fácilmente datos de la web, aplicar técnicas de limpieza y extraer i