Python Formulario

Pandas | Numpy | Sklearn
Matplotlib | Seaborn
BS4 | Selenium | Scrapy

Escrito por Frank Andrade

Pandas Formulario

Pandas proporciona herramientas para análisis de datos en Python. Los siguientes ejemplos con código están relacionados con el dataframe debajo.

df =

	col1	col2
A	1	4
B	2	5
C	3	6

axis 1 (eje 1)

axis 0 (eje 0)

Introducción

Importando pandas:
```
import pandas as pd
```

Creando una serie:
```
s = pd.Series([1, 2, 3],
              index=['A', 'B', 'C'],
              name='col1')
```

Creando un dataframe:
```
data = [[1, 4], [2, 5], [3, 6]]
index = ['A', 'B', 'C']
df = pd.DataFrame(data, index=index,
                  columns=['col1', 'col2'])
```

Cargar dataframe:
```
df = pd.read_csv('nombre_archivo.csv',sep=',',
                 names=['col1', 'col2'],
                 index_col=0,
                 encoding='utf-8',
                 nrows=3)
```

Seleccionar filas y columnas

Seleccionar una columna:
```
df['col1']
```

Seleccionar más de una columna:
```
df[['col1', 'col2']]
```

Mostrar primeras n filas:
```
df.head(2)
```

Mostrar últimas n filas:
```
df.tail(2)
```

Seleccionar filas por valores del índice (index):
```
df.loc['A'] df.loc[['A', 'B']]
```

Seleccionar filas por posición:
```
df.loc[1] df.loc[1:]
```

Manejo de Datos

Filtro por valor:
```
df[df['col1'] > 1]
```

Ordenar por columnas:
```
df.sort_values(['col2', 'col2'],
               ascending=[False, True])
```

Identificar filas duplicadas:
```
df.duplicated()
```

Identificar filas únicas:
```
df['col1'].unique()
```

Intercambiar filas y columnas:
```
df = df.transpose()
df = df.T
```

Eliminar una columna:
```
df = df.drop('col1', axis=1)
```

Clonar un dataframe:
```
clon = df.copy()
```

Conectar múltiples dataframes verticalmente:
```
df2 = df + 5 #nuevo dataframe
pd.concat([df,df2])
```

Unir múltiples dataframes horizontalmente:
```
df3 = pd.DataFrame([[1, 7],[8,9]],
                   index=['B', 'D'],
             columns=['col1', 'col3'])
#df3: nuevo dataframe
```

Unir solo filas completas (INNER JOIN):
```
df.merge(df3)
```

Columna de la izquierda completa (LEFT OUTER JOIN):
```
df.merge(df3, how='left')
```

Columna de la derecha completa (RIGHT OUTER JOIN):
```
df.merge(df3, how='right')
```

Preservar todos los valores (OUTER JOIN):
```
df.merge(df3, how='outer')
```

Unir filas por indice (index):
```
df.merge(df3,left_index=True,
         right_index=True)
```

Completar valor NaN:
```
df.fillna(0)
```

Aplicar propia función:
```
def func(x):
    return 2**x
df.apply(func)
```

Aritmética y estadística

Agregar valor a todos los valores:
```
df + 10
```

Suma en columnas:
```
df.sum()
```

Suma acumulada sobre columnas:
```
df.cumsum()
```

Promedio en columnas:
```
df.mean()
```

Desviación estándar sobre columnas:
```
df.std()
```

Contar valores únicos:
```
df['col1'].value_counts()
```

Resumir valores estadísticos:
```
df.describe()
```

Indexación jerárquica

Crear índice jerárquico:
```
df.stack()
```

Disolver índice jerárquico:
```
df.unstack()
```

Agrupación

Crear grupo (objeto):
```
g = df.groupby('col1')
```

Iterar sobre grupos:
```
for i, grupo in g:
      print(i, grupo)
```

Operaciones en grupos:
```
g.sum()
g.prod()
g.mean()
g.std()
g.describe()
```

Seleccionar columnas de grupos:
```
g['col2'].sum()
g[['col2', 'col3']].sum()
```

Transformar valores:
```
  import math
  g.transform(math.log)
```

Aplicar una lista función en cada grupo:
```
def strsum(grupo):
 return ''.join([str(x) for x in grupo.value])

 g['col2'].apply(strsum)
```

Exportar Data

Data como NumPy array:
```
df.values
```

Guardar data como archivo CSV:
```
df.to_csv('output.csv', sep=",")
```

Dar formato a dataframe como cadena tabular:
```
df.to_string()
```

Convertir un dataframe a diccionario:
```
df.to_dict()
```

Guardar un dataframe como una tabla Excel :
```
df.to_excel('output.xlsx')
```

Visualización

Importar matplotlib:
```
import matplotlib.pyplot as plt
```

Empezar un nuevo diagrama:
```
plt.figure()
```

Gráfico de dispersión (Scatter plot):
```
df.plot.scatter('col1', 'col2',
                style='ro')
```

Gráfico de barras (Bar plot):
```
df.plot.bar(x='col1', y='col2',
            width=0.7)
```

Gráfico de áreas (Area plot):
```
df.plot.area(stacked=True,
             alpha=1.0)
```

Gráfoco de cajas (Box-and-whisker plot):
```
df.plot.box()
```

Histograma sobre una columna:
```
df['col1'].plot.hist(bins=3)
```

Histograma sobre todas las columnas:
```
df.plot.hist(bins=3, alpha=0.5)
```

Establecer tick marks:
```
labels = ['A', 'B', 'C', 'D']
posicion = [1, 2, 3, 4]
plt.xticks(posicion, labels)
plt.yticks(posicion, labels)
```

Seleccionar área para trazar:
```
plt.axis([0, 2.5, 0, 10]) #[desde x,
hasta x, desde y, hasta y]
```

Etiquetar gráfico y ejes:
```
plt.title('Correlation')
plt.xlabel('Nunstück')
plt.ylabel('Slotermeyer')
```

Guardar gráfico reciente:
```
plt.savefig('plot.png')
plt.savefig('plot.png',dpi=300)
plt.savefig('plot.svg')
```

Encuentra ejemplos prácticos en estos videos/guías que hice:
- Pandas para usuarios Excel(link)
- Limpieza de Datos con Pandas(link)
- Expresiones Regulares (link)

Frank Andrade www.youtube.com/andradefrank

NumPy Formulario

NumPy proporciona herramientas para trabajar con arrays. Los ejemplos con código están relacionados al array debajo.

NumPy Arrays

Introducción

Importar numpy:

```
import numpy as np
```

Crear arrays:

```
a = np.array([1,2,3])
b = np.array([(1.5,2,3), (4,5,6)], dtype=float)
c = np.array([[(1.5,2,3), (4,5,6)],
              [(3,2,1), (4,5,6)]],
              dtype = float)
```

Marcadores de posición iniciales:

```
np.zeros((3,4)) #Crear un array de ceros
np.ones((2,3,4),dtype=np.int16)
d = np.arange(10,25,5)
np.linspace( 0,2, 9)
e = np.full((2,2), 7)
f = np.eye(2)
np.random.random((2,2))
np.empty((3,2))
```

Guardar & Cargar En Disco:

```
np.save('mi_array', a)
np.savez('array.npz', a, b)
np.load('mi_array.npy')
```

Guardar & Cargar Archivos de Texto

```
np.loadtxt('mi_archivo.txt')
np.genfromtxt('mi_archivo.csv',
              delimiter=',')
np.savetxt('mi_array.txt', a,
           delimiter= ' ')
```

Inspeccionar Array

```
a.shape #forma array
len(a) #tamaño
b.ndim #dimensión
e.size #tamaño
b.dtype #tipo de data
b.dtype.name
b.astype(int) #cambiar tipo de data
```

Tipos de Data

```
np.int64
np.float32
np.complex
np.bool
np.object
np.string_
np.unicode_
```

Matemáticas con Arrays

Operaciones

```
>>> g = a-b
 array([[-0.5, 0. , 0. ],
        [-3. , 3. , 3. ]])
>>> np.subtract(a,b)

>>> b+a
 array([[2.5, 4. , 6. ],
        [ 5. , 7. , 9. ]])
>>> np.add(b,a)

>>> a/b
 array([[ 0.66666667, 1. , 1. ],
        [ 0.2 5 , 0.4 , 0 . 5 ]])
>>> np.divide(a,b)

>>> a*b
 array([[ 1 . 5, 4. , 9. ],
        [ 4. , 10. , 18. ]])
>>> np.multiply(a,b)

>>> np.exp(b) #exponencial
>>> np.sqrt(b) #raiz
>>> np.sin(a) #seno
>>> np.log(a) #logaritmo
>>> e.dot(f) #producto escalar
```

Operaciones:

```
a.sum()
a.min()
b.max(axis= 0)
b.cumsum(axis= 1) #Suma acumulada
a.mean() #Promedio
b.median() #Mediana
a.corrcoef() #Coeficiente Correlacion
np.std(b) #Deviacion estandar
```

Copiar arrays:

```
h = a.view() #Crear vista
np.copy(a)
h = a.copy() #Crear copia total
```

Ordenar arrays:

```
a.sort() #Ordenar array
c.sort(axis=0)
```

Manipulación de Array

Transposición de Array:

```
i = np.transpose(b)
i.T
```

Cambiar Forma de Array:

```
b.ravel()
g.reshape(3,-2)
```

Agregar/eliminar elementos:

```
h.resize((2,6))
np.append(h,g)
np.insert(a, 1, 5)
np.delete(a,[1])
```

Combinar arrays:

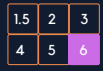
```
np.concatenate((a,d),axis=0)
np.vstack((a,b))#apilar verticalmente
np.hstack((e,f))#apilar horizontalmente
```

Dividir arrays:

```
np.hsplit(a,3) #Dividir horizontalmente
np.vsplit(c,2) #Dividir verticalmente
```

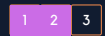
Subconjunto

```
b[1,2]
```



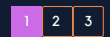
Slicing:

```
a[0:2]
```



Indexación Booleana:

```
a[a<2]
```



Scikit-Learn Formulario

Sklearn es un software gratuito y libreria de machine learning en Python. Cuenta con varios algoritmos de clasificación, regresión y agrupamiento.

Introducción

El código abajo demuestra los pasos básicos al usar sklearn para crear y correr un modelo en un set de data.
Los pasos en el código incluyen cargar (load) la data, dividir (split) la data en train y test sets, scalar los sets, crear el modelo, ajustar (fit) el modelo en la data usando el modelo entrenado para hacer predicciones en el test set, y finalmente evaluar el rendimiento del modelo.

```
from sklearn import neighbors,datasets,preprocessing
from sklearn.model_selection import train_test_split
from sklearn.metrics import accuracy_score
iris = datasets.load_iris()
X,y = iris.data[:,:2], iris.target
X_train, X_test, y_train, y_test=train_test_split(X,y)
scaler = preprocessing_StandardScaler().fit(X_train)
X_train = scaler.transform(X_train)
X_test = scaler.transform(X_test)
knn = neighbors.KNeighborsClassifier(n_neighbors = 5)
knn.fit(X_train, y_train)
y_pred = knn.predict(X_test)
accuracy_score(y_test, y_pred)
```

Cargar la Data

La data necesita ser numérica y ser manipulada como NumPy array o SciPy spare matrix (arrays numericos como Pandas DataFrame's también pueden ser usados)

```
>>> import numpy as np
>>> X = np.random.random((10,5))
array([[0.21,0.33],
       [0.23, 0.60],
       [0.48, 0.62]])
>>> y = np.array(['A','B','A'])
array(['A', 'B', 'A'])
```

Entrenando y Testeando la Data

```
from sklearn.model_selection import train_test_split
X_train,X_test,y_train,y_test = train_test_split(X,y,
random_state = 0)#divide data en train y test set
```

Pre-procesando la Data

Estandarización

Estandariza "features" eliminando media y escalando a la varianza de la unidad.

```
from sklearn.preprocessing import StandardScaler
scaler = StandardScaler().fit(X_train)
standarized_X = scaler.transform(X_train)
standarized_X_test = scaler.transform(X_test)
```

Normalización

Cada muestra con al menos un componente distinto de cero se reescala independientemente de otras muestras para que su norma sea igual a uno.

```
from sklearn.preprocessing import Normalizer
scaler = Normalizer().fit(X_train)
normalized_X = scaler.transform(X_train)
normalized_X_test = scaler.transform(X_test)
```

Binarización

Binarizar datos (valores de "feature" en 0 o 1) de acuerdo con un umbral.

```
from sklearn.preprocessing import Binarizer
binarizer = Binarizer(threshold = 0.0).fit(X)
binary_X = binarizer.transform(X_test)
```

Codificación de características (feature) categóricas

Transformador de imputación para completar valores perdidos.

```
from sklearn import preprocessing
le = preprocessing.LabelEncoder()
le.fit_transform(X_train)
```

Imputación de valores perdidos

```
from sklearn.impute import SimpleImputer
imp = SimpleImputer(missing_values=0, strategy ='mean')
imp.fit_transform(X_train)
```

Generación de características (feature) polinomiales

```
from sklearn.preprocessing import PolynomialFeatures
poly = PolynomialFeatures(5)
poly.fit_transform(X)
```

Encuentra ejemplos prácticos en estos videos/guías que hice:
- Guía de Scikit-Learn (link)
- Tokenización de textos (link)
- Predecir partidos de futbol (link)

Frank Andrade www.youtube.com/andradefrank

Crear Modelo

Modelos Supervisados

Regresión Lineal

```
from sklearn.linear_model import LinearRegression
lr  = LinearRegression(normalize = True)
```

Support Vector Machines (SVM)

```
from sklearn.svm import SVC
svc = SVC(kernel = 'linear')
```

Naive Bayes

```
from sklearn.naive_bayes import GaussianNB
gnb = GaussianNB()
```

KNN

```
from sklearn import neighbors
knn = neighbors.KNeighborsClassifier(n_neighbors = 5)
```

Modelos No Supervisados

Principal Component Analysis (PCA)

```
from sklearn.decomposition import PCA
pca = PCA(n_components = 0.95)
```

K means

```
from sklearn.cluster import KMeans
k_means = KMeans(n_clusters = 3, random_state = 0)
```

Ajuste del modelo (Fitting)

Ajustar modelos de aprendizaje supervisado y no supervisado a los datos.

Aprendizaje Supervisado

```
lr.fit(X, y) #ajustar el modelo a la data
knn.fit(X_train,y_train)
svc.fit(X_train,y_train)
```

Aprendizaje No Supervisado

```
k_means.fit(X_train) #ajustar el modelo a la data
pca_model = pca.fit_transform(X_train)#ajustar,luego transformar
```

Predecir

Predecir Labels

```
y_pred = lr.predict(X_test) #Estimadores Supervisados
y_pred = k_means.predict(X_test) #Estimadores No Supervisados
```

Estimador de probabilidad de un label

```
y_pred = knn.predict_proba(X_test)
```

Evaluar Rendimiento del Modelo

Métricas de Clasificación

Accuracy Score

```
knn.score(X_test,y_test)
from sklearn.metrics import accuracy_score
accuracy_score(y_test,y_pred)
```

Reporte de Clasificación

```
from sklearn.metrics import classification_report
print(classification_report(y_test,y_pred))
```

Confusion Matrix

```
from sklearn .metrics import confusion_matrix
print(confusion_matrix(y_test,y_pred))
```

Métricas de Regresión

Error Absoluto Medio

```
from sklearn.metrics import mean_absolute_error
mean_absolute_error(y_test,y_pred)
```

Error Medio Cuadrado

```
from sklearn.metrics import mean_squared_error
mean_squared_error(y_test,y_pred)
```

R^2 Score

```
from sklearn.metrics import r2_score
r2_score(y_test, y_pred)
```

Métricas Clustering

Indice Rand Adjustado

```
from sklearn.metrics import adjusted_rand_score
adjusted_rand_score(y_test,y_pred)
```

Homogeneidad

```
from sklearn.metrics import homogeneity_score
homogeneity_score(y_test,y_pred)
```

V-measure

```
from sklearn.metrics import v_measure_score
v_measure_score(y_test,y_pred)
```

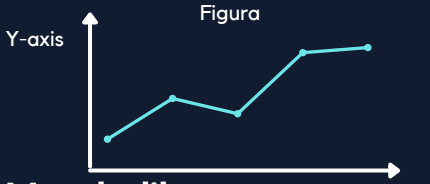
Optimizar el Modelo

Grid Search

```
from sklearn.model_selection import GridSearchCV
params = {'n_neighbors':np.arange(1,3),
          'metric':['euclidean','cityblock']}
grid = GridSearchCV(estimator = knn, param_grid = params)
grid.fit(X_train, y_train)
print(grid.best_score_)
print(grid.best_estimator_.n_neighbors)
```

Visualización Formulario

Matplotlib es una libreria de gráficos 2D de Python que produce figuras en una variedad de formatos.



Matplotlib

Flujo de Trabajo

Los pasos básicos para crear una gráfica con matplotlib son Preparar Data, Graficar, Personalizar Gráfico, Guardar Gráfico y Mostrar Gráfico.

```
import matplotlib.pyplot as plt
```

Ejemplo con Gráfico de linea

Preparar data

```
x = [2017, 2018, 2019, 2020, 2021]
y = [43, 45, 47, 48, 50]
```

Graficar & Personalizar

```
plt.plot(x,y,marker='o',linestyle='--',
color='g', label='Colombia')
plt.xlabel('Años')
plt.ylabel('Poblacion (M)')
plt.title('Años vs Poblacion')
plt.legend(loc='lower right')
plt.yticks([41, 45, 48, 51])
```

Guardar Gráfico

```
plt.savefig('ejemplo.png')
```

Mostrar Gráfico

```
plt.show()
```

Marcadores: '.', 'o', 'v', '<', '>'

Estilos de linea: '-', '--', '-.', ':'

Colores: 'b', 'g', 'r', 'y' #azul, verde, rojo, amarillo

Gráfico de Barras (Barplot)

```
x = ['Argentina', 'Colombia', 'Peru']
y = [40, 50, 33]
plt.bar(x, y)
plt.show()
```

Piechart

```
plt.pie(y, labels=x, autopct='%.0f %%')
plt.show()
```

Histograma

```
ages = [15, 16, 17, 30, 31, 32, 35]
bins = [15, 20, 25, 30, 35]
plt.hist(ages, bins, edgecolor='black')
plt.show()
```

Gráfico de Caja (Boxplots)

```
ages = [15, 16, 17, 30, 31, 32, 35]
plt.boxplot(ages)
plt.show()
```

Gráfico de Dispersión (Scatterplot)

```
a = [1, 2, 3, 4, 5, 4, 3 ,2, 5, 6, 7]
b = [7, 2, 3, 5, 5, 7, 3, 2, 6, 3, 2]
plt.scatter(a, b)
plt.show()
```

Subplots

Agrega el código debajo para hacer multples gráficos con 'n' números de filas y columnas.

```
fig, ax = plt.subplots(nrows=1,
                       ncols=2,
                       sharey=True,
                       figsize=(12, 4))
```

Graficar & Personalizar Cada Gráfico

```
ax[0].plot(x, y, color='g')
ax[0].legend()
ax[1].plot(a, b, color='r')
ax[1].legend()
plt.show()
```

Encuentra ejemplos prácticos en estos videos/guías que hice:
- Guía de Matplotlib (link)
- Guía de Nube de Palabras (link)
- Ejemplo Visualización Datos(link)

Frank Andrade www.youtube.com/andradefrank

Seaborn

Flujo de Trabajo

```
import seaborn as sns
import matplotlib.pyplot as plt
import pandas as pd
```

Gráfico de Lineas

```
plt.figure(figsize=(10, 5))
flights = sns.load_dataset("flights")
may_flights=flights.query("month=='May'")
ax = sns.lineplot(data=may_flights,
                  x="year",
                  y="passengers")
ax.set(xlabel='x', ylabel='y',
       title='my_title, xticks=[1,2,3])
ax.legend(title='my_legend,
          title_fontsize=13)
plt.show()
```

Gráfico de Barras (Barplot)

```
tips = sns.load_dataset("tips")
ax = sns.barplot(x="day",
                 y="total_bill,
                 data=tips)
```

Histograma

```
penguins = sns.load_dataset("penguins")
sns.histplot(data=penguins,
             x="flipper_length_mm")
```

Gráfico de Cajas (Boxplot)

```
tips = sns.load_dataset("tips")
ax = sns.boxplot(x=tips["total_bill"])
```

Gráfico de Dispersión (Scatterplot)

```
tips = sns.load_dataset("tips")
sns.scatterplot(data=tips,
                x="total_bill",
                y="tip")
```

Estética de la figura

```
sns.set_style('darkgrid') #estilos
sns.set_palette('husl', 3) #paletas
sns.color_palette('husl') #colores
```

Tamaño de letra de titulo de ejes, x e y labels, tick labels y leyendas:

```
plt.rc('axes', titlesize=18)
plt.rc('axes', labelsize=14)
plt.rc('xtick', labelsize=13)
plt.rc('ytick', labelsize=13)
plt.rc('legend', fontsize=13)
plt.rc('font', size=13)
```

Web Scraping Formulario

El web scraping nos permite extraer data de la web. Antes de aprender Beautiful Soup, Selenium o Scrapy, vamos a revisar conceptos básicos de HTML.

HTML básico para Web Scraping

Analicemos el siguiente elemento HTML.

Este es solo un elemento HTML, pero el documento HTML detrás de una página web tiene varios elmentos como este.

Código HTML ejemplo

```
<article class="main-article">
 <h1> Titanic (1997) </h1>
 <p class="plot"> 84 years later ... </p>
 <div class="full-script"> 13 meters. You ... </div>
</article>
```

El documento HTML está estructurado con "nodos". Cada rectángulo debajo representa un nodo (elemento, atributo o texto)

- "Hermanos" son nodos con los mismos padres.
- El hijo de un nodo y los hijos de sus hijos son llamados sus "descendientes". Del mismo modo, el padre de un nodo y el padre de su padre son llamados "ancestros".
- Es recomendado buscar elementos en este orden
  a. ID
  b. Class name
  c. Tag name
  d. Xpath

Beautiful Soup

Flujo de Trabajo

```
Importar librerías
 from bs4 import BeautifulSoup
 import requests

Obtener páginas
 result=requests.get("www.google.com")
 result.status_code #obtener status
 result.headers #obtener encabezados

Contenido de la página
 contenido = result.text

Crear soup
 soup=BeautifulSoup(contenido,"lxml")

HTML en formato legible
 print(soup.prettify())

Encontrar un elemento
 soup.find(id="mi_id")

Encontrar elementos
 soup.find_all("a")
 soup.find_all("a","css_class")
 soup.find_all("a",class_="mi_class")
 soup.find_all("a",attrs={"class":
                          "mi_class"})

Obtener texto
 ejemplo=elemento.get_text()
 ejemplo=elemento.get_text(strip=True,
                           separator=' ')

Obtener atributos
 ejemplo = elemento.get('href')
```

XPath

Necesitamos aprender XPath para hacer web scraping con Selenium y Scrapy.

XPath Sintaxis

Un XPath usualmente contiene un tag, nombre de atributo y valor de atributo.

```
//tag[@Atributo="Valor"]
```

Veamos algunos ejemplos de como localizar el elemento article, el titulo de la película y transcript del código HTML que vimos antes.

```
//article[@class="main-article"]
//h1
//div[@class="full-script"]
```

XPath Funciones y Operadores

XPath funciones

```
//tag[contains(@Atributo, "Valor")]
```

XPath Operadores: and, or

```
//tag[(expresion 1) and (expresion 2)]
```

XPath Caracteres Especiales

/	Selecciona los hijos del nodo ubicado a la izquiera de este caracter
//	Especifica que el nodo a emparejar puede estar en cualquier nivel del documento
.	Especifica que el contexto actual debería ser usado (el nodo referencia)
..	Selecciona a un nodo padre
*	Caracter comodín que selecciona todos los elementos sin importar el nombre
@	Selecciona un atributo
()	Indica una agrupación dentro de un XPath
[n]	Indica que un nodo con index "n" debe ser seleccionado

Selenium

Flujo de Trabajo

```
from selenium import webdriver
web="www.google.com"
path='introduce ruta del chromedriver'
driver = webdriver.Chrome(path)
driver.get(web)
```

Encontrar un elemento
```
driver.find_element_by_id('nombre')
```

Encontrar elementos
```
driver.find_elements_by_class_name()
driver.find_elements_by_css_selector
driver.find_elements_by_xpath()
driver.find_elements_by_tag_name()
driver.find_elements_by_name()
```

Cerrar driver
```
driver.quit()
```

Obtener el texto
```
data = elemento.text
```

Espera Implícita
```
import time
time.sleep(2)
```

Espera Explícita
```
from selenium.webdriver.common.by import By
from selenium.webdriver.support.ui import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC

WebDriverWait(driver,5).until(EC.element_to_be_clickable((By.ID,
'id_name'))) #esperar 5 segundos hasta poder encontrar elemento
```

Opciones: Headless mode, cambiar tamaño de ventana
```
from selenium.webdriver.chrome.options import Options
opciones = Options()
opciones.headless = True
opciones.add_argument('window-size=1920x1080')
driver = webdriver.Chrome(path,options=opciones)
```

Encuentra ejemplos prácticos en estos videos/guías que hice:
- Web Scraping con Selenium (link)
- Web Scraping con BS4 (link)
- Web Scraping Guía Completa (link)

Frank Andrade www.youtube.com/andradefrank

Scrapy

Scrapy es el framework más complete de web scraping en Python. Para configurarlo revisa la documentación de Scrapy.

Crear un Proyecto y Spider

Para crear un nuevo proyecto, corre el siguiente comando en el terminal o cmd
```
scrapy startproject mi_primer_spider
```
Para crear un nuevo spider, primero cambia el directorio
```
cd mi_primer_spider
```
Crear un spider
```
scrapy genspider ejemplo ejemplo.com
```

La plantilla básica

Cuando creamos un spider, obtenemos una plantilla con el siguiente contenido.

```
import scrapy
class ExampleSpider(scrapy.Spider):
    name = 'ejemplo'
    allowed_domains = ['ejemplo.com']
    start_urls = ['http://ejemplo.com/']

    def parse(self, response):
        pass
```

Clase

Método Parse

La clase es contruida con la data que introducimos en el comando previo, pero el método parse tenemos que construirlo nosotros.

Buscando elementos

Para buscar elementos con Scrapy, usa el argumento "response" del método parse
```
response.xpath('//tag[@Atributo="Valor"]')
```

Obtener texto

Para obtener el elemento texto usamos text() y luego .get() o .getall(). Por ejemplo:
```
response.xpath('//h1/text()').get()
response.xpath('//tag[@Atributo="Valor"]/text()').getall()
```

Devolver la data extraída

Para ver la data extraída tenemos que usar la palabra clave yield

```
def parse(self, response):
    title = response.xpath('//h1/text()').get()

    # Devolver data extraída
    yield {'titles': title}
```

Correr el spider y exportar data a CSV o JSON
```
scrapy crawl ejemplo
scrapy crawl ejemplo -o nombre_archivo.csv
scrapy crawl ejemplo -o nombre_archivo.json
```