Explicació detallada del funcionament del codi de *Escales Vaixells.py*

```
#llibreries i funcions
from random import randint
from time import sleep
from selenium import webdriver
from selenium.webdriver.chrome.service import Service
from selenium.webdriver.common.by import By
from selenium.webdriver.support.wait import WebDriverWait
from selenium.webdriver.support import expected_conditions as EC
from selenium.webdriver.chrome.options import Options
import pandas
def s(t1,t2):
    sleep(randint(1000*t1,1000*t1+1000*t2)/1000)
```

Inicialment es carreguen les llibreries i funcions que s'utilitzaran.

Amb randint i sleep crearem la funció s(t1,t2) que ens permetrà fer descansar el programa un temps aleatori de t1 segons de mitjana, i t2 segons de variància, per donar-li l'aparença de comportament no robòtic.

Com que el nostre scraper el controlarem amb la llibreria selenium en necessitem les propietats de webdriver, Service i By.

També utilitzem WebDriverWait i expected_conditions per esperar a que es carregui la informació quan la necessitem.

Amb Options li donarem el nom que volem al user-agent.

```
#canviar l'user-agent
opts = Options()
opts.add_argument("user-agent=Real Human Beep Bop")

#iniciar chrome.driver
serv = Service('../chromedriver.exe')
driver = webdriver.Chrome(service=serv, options=opts)

link = 'https://tarragona.posidoniaport.com/'
data_inici="01/01/2019 00:00"
data_fi="31/12/2019 23:59"

#inicia navegador
driver.get(link)

#comprova que l'user-agent és el correcte
#driver.execute_script("return navigator.userAgent")
```

Li posem el nom humà de *Real Human Beep Bop* a l'user-agent, iniciem el chrome.driver i establim l'enllaç de la pàgina i les dates que ens interessen. Finalment hi ha la comprovació (que tenim comentada) del nom de l'user-agent.

```
#clica a històrics quan la pàgina s'hagi carregat
WebDriverWait(driver, 20).until(EC.element_to_be_clickable(
    (By.XPATH, "//span[@id='historicos']"))).click()

#esborra el text per defecte i introdueix la data d'inici
WebDriverWait(driver, 20).until(EC.presence_of_element_located(
    (By.CSS_SELECTOR, "[data-bind='datetimepicker: fecatr']"))).clear()
driver.find_elements(By.CSS_SELECTOR, "[data-bind='datetimepicker: fecatr']")[0].send_keys(data_inici)

#clica a fet
WebDriverWait(driver, 20).until(EC.presence_of_element_located(
    (By.CSS_SELECTOR, "[class='ui-datepicker-close ui-state-default ui-priority-primary ui-corner-all']"))).click()

#esborra el text per defecte i introdueix la data final
WebDriverWait(driver, 20).until(EC.presence_of_element_located(
    (By.CSS_SELECTOR, "[data-bind='datetimepicker: fecsal, mindate: fecatr']"))).clear()
driver.find_elements(By.CSS_SELECTOR, "[data-bind='datetimepicker: fecsal, mindate: fecatr']")[0].send_keys(data_fi)

#clica a fet
WebDriverWait(driver, 20).until(EC.presence_of_element_located(
    (By.CSS_SELECTOR, "[class='ui-datepicker-close ui-state-default ui-priority-primary ui-corner-all']"))).click()

s(5,1)
#clica a Veure
WebDriverWait(driver, 20).until(EC.element_to_be_clickable(
    (By.XPATH, "//button[text()='Veure']"))).click()

s(5,1)
#estableix 30 registres per pàgina si les dades s'han acabat de carregar
```

A partir d'ara ja és quan treballa el scraper i hem d'anar amb compte de que no vagi massa ràpid i no doni temps de carregar les pàgines. Per això utilitzem el WebDriverWait amb 20 segons d'espera fins a que els elements que busquem estiguin disponibles per a ésser clicats o detectats, en funció de cada acció que vulguem prendre.

Hem anat a històrics, li hem passat les dates d'interès i hem clicat a Veure. Li donem uns segons de marge per a que aparegui el text *Carregant* abans de continuar.

```
#estableix 30 registres per pàgina si les dades s'han acabat de carregar
WebDriverWait(driver, 30).until(EC.invisibility_of_element_located(
    (By.XPATH, "//div[text()='Carregant...']")));
WebDriverWait(driver, 20).until(EC.presence_of_element_located(
    (By.XPATH, "//select[@class='ui-pg-selbox']"))).send_keys('30')

#creo dataframe buit
df = pandas.DataFrame(columns = ['Escala', 'Vaixell', 'Moll', 'Entrada', 'Sortida', 'Consignatari',
                                 'Eslora', 'Tipus', 'Mercaderia', 'Tn', 'Estibador'])
#identifico número de pàgines
num_pagines=int(WebDriverWait(driver, 20).until(EC.presence_of_element_located(
    (By.XPATH, "//td[@dir='ltr']//span"))).text)

pagines=range(0,num_pagines)
k=0
print('Copiant dades... Pàgina:',end=' ')
```

Quan l'element *Carregant* desaparegui ja tindrem les dades carregades. Aquí és on creem la base de dades buida (però amb els noms de columnes) i identifiquem el número de pàgines que el scraper haurà de navegar. A partir d'aquí començarà el procés iteratiu.

```
77  #entro a cada pàgina
78  for pag in pagines:
79      k += 1
80      print(k, end=' ')
81
82      table_id = WebDriverWait(driver, 20).until(EC.presence_of_element_located(
83      (By.CSS_SELECTOR, "[class='grid_gisgrid ui-jqgrid-btable']")))
84      rows = table_id.find_elements(By.CSS_SELECTOR, "[tabindex='-1']")
85
86      #copio dades fila a fila
87      for row in rows:
```

A cada pàgina identificarà la taula i n'agafarà la llista de totes les files. Dins de cada pàgina hi haurà un altre procés iteratiu per a cada fila de la taula.

```
89          df_tmp = pandas.DataFrame({
90          'Escala':row.find_elements(By.CSS_SELECTOR, "[aria-describedby$='_esccod']")[0].text,
91          'Vaixell':row.find_elements(By.CSS_SELECTOR, "[aria-describedby$='_nombuq']")[0].text,
92          'Moll':row.find_elements(By.CSS_SELECTOR, "[aria-describedby$='_desmue']")[0].text,
93          'Entrada':row.find_elements(By.CSS_SELECTOR, "[aria-describedby$='_fecatr']")[0].text,
94          'Sortida':row.find_elements(By.CSS_SELECTOR, "[aria-describedby$='_fecsal']")[0].text,
95          'Consignatari':row.find_elements(By.CSS_SELECTOR, "[aria-describedby$='_nomcsg']")[0].text,
96          'Eslora':row.find_elements(By.CSS_SELECTOR, "[aria-describedby$='_eslora']")[0].text,
97          'Tipus':row.find_elements(By.CSS_SELECTOR, "[aria-describedby$='_optipo']")[0].text,
98          'Mercaderia':row.find_elements(By.CSS_SELECTOR, "[aria-describedby$='_opmercancia']")[0].text,
99          'Tn':row.find_elements(By.CSS_SELECTOR, "[aria-describedby$='_optoneladas']")[0].text,
100         'Estibador':row.find_elements(By.CSS_SELECTOR, "[aria-describedby$='_opestibador']")[0].text},
101         index=[0])
102
103         df_tmp['Escala']=df_tmp['Escala'].str.replace('.','', regex=True)
104         df_tmp['Eslora']=df_tmp['Eslora'].str.replace('.','', regex=True)
105         df_tmp['Tn']=df_tmp['Tn'].str.replace('.','', regex=True)
106         df=pandas.concat([df,df_tmp])
107
108     #canvi de pàgina
109     WebDriverWait(driver, 20).until(EC.presence_of_element_located(
110     (By.CSS_SELECTOR, "[id*='next_pager']"))).click()
```

Per a cada fila detectarà cada valor que ens interessa: escala, vaixell, moll, etc. Ho posa tot en un DataFrame temporal d'una fila, corregeix el format dels números (eliminar els punts de milers) i ho concatena a la base de dades.

En acabar la iteració de totes les files de la pàgina, clica al botó de pàgina següent i continua el procés a la pàgina següent. Quan arriba a l'última pàgina clica al botó però no té efecte.

```
111
112  #formatejo base de dades
113  df2 = df.astype({"Escala":"float",
114                   "Eslora":"float"})
115
116  df2.index = range(1,len(df2)+1)
117
118  #guardo arxiu
119  df2.to_excel('../dataset/Escales Vaixells_2019.xlsx')
```

Finalment es formata les columnes numèriques – a un nou DataFrame per no perdre les dades originals en cas que hi hagi algun error –, es dona a cada registre el número d'índex pertinent i es guarda l'arxiu en format .xlsx a la carpeta dataset.