

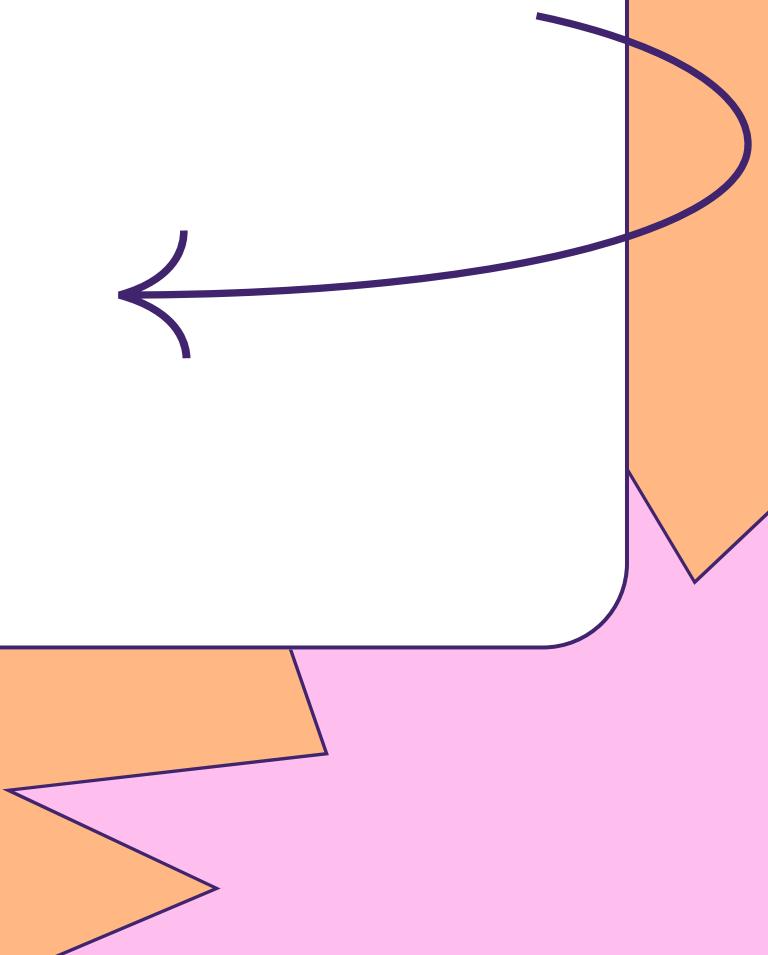
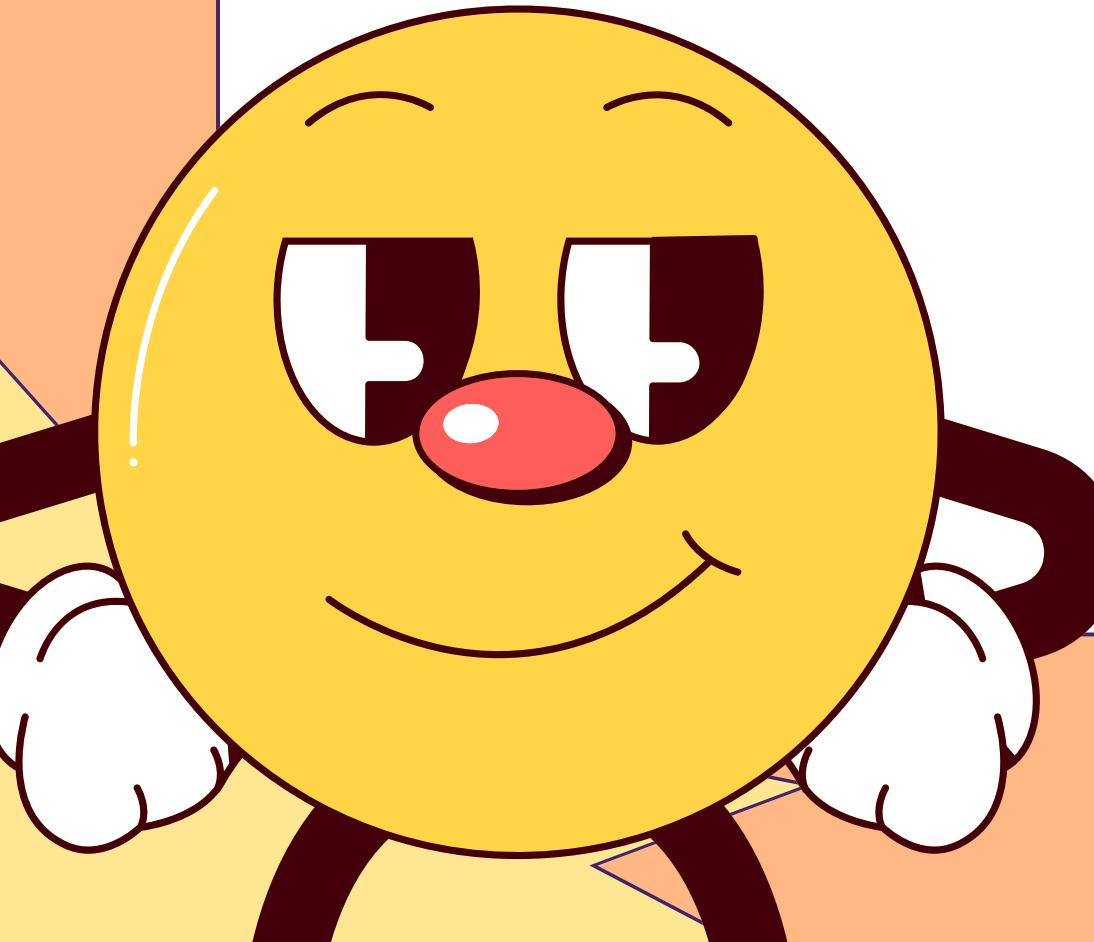
PRESENTADO POR:

Mateo Alís, Manuel Caballero, Vicente Llacer



# WEB SCRAPING

*Selección de viajes de trenes/aves y  
autobuses*



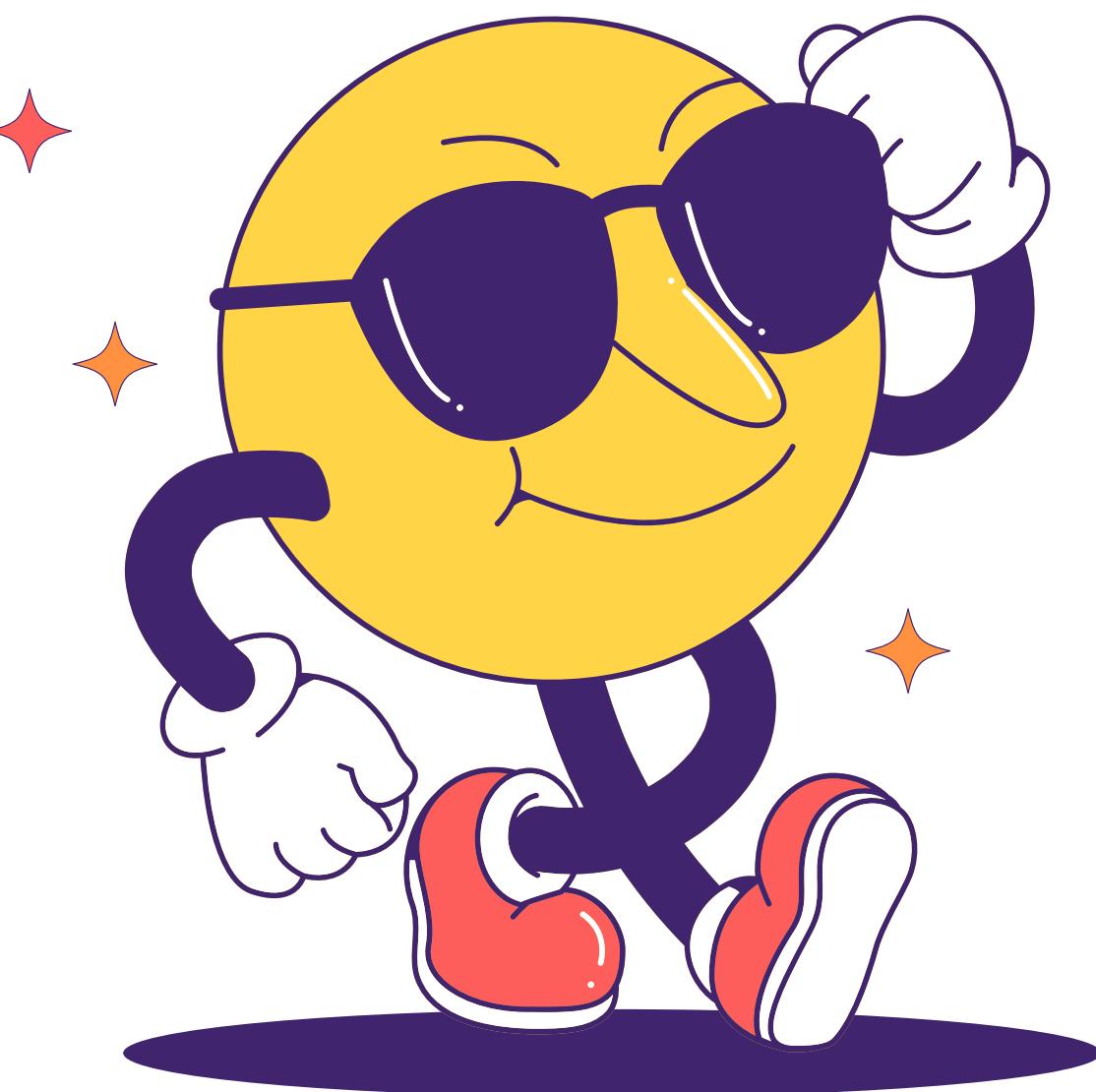
# Introducción

¿Como surgió la idea?

¿Con que objetivo?

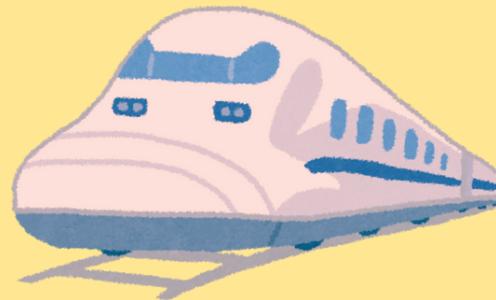
¿Que url's hemos utilizado?

¿Como es el procedimiento?



# Procedimiento

TRENES/  
AVES



AUTOBUSES



EL USUARIO DEBE  
INTRODUCIR:

FECHA DE IDA Y  
VUELTA

MEDIO DE  
TRANSPORTE A  
UTILIZAR

CIUDAD DE ORIGEN Y  
DESTINO

# Procedimiento

TRENES/  
AVES

AUTOBUSES

- Abre la página de RENFE y de OIUGO
- Cuenta el número de viajes que hay entre las 2 webs.
- Nos muestra el viaje de ida y vuelta más barato.
- Además muestra el precio, duración del trayecto y de que web son los billetes, etc...
- Nos da la opción de consultar los billetes y precios de autobuses también.

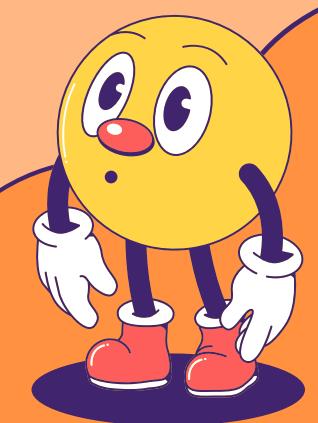
- Abre la página de ALSA
- Cuenta el número de viajes que hay en autobus.
- Nos muestra el viaje de ida y vuelta más barato.
- Además muestra el precio, duración del trayecto y de que web son los billetes, etc...
- Si miraramos primero autobuses nos daría opción a consultar billetes y precios de trenes/ave.

# ♦ Librerías y funciones auxiliares ♦

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

def mes_a_numero(nombre_mes):
    # Diccionario que mapea el nombre del mes al número correspondiente
    meses = {
        'enero': 1,
        'febrero': 2,
        'marzo': 3,
        'abril': 4,
        'mayo': 5,
        'junio': 6,
        'julio': 7,
        'agosto': 8,
        'septiembre': 9,
        'octubre': 10,
        'noviembre': 11,
        'diciembre': 12
    }

    # Convertir el nombre del mes a minúsculas y buscar el número correspondiente
    return meses.get(nombre_mes.lower(), 'Mes inválido')
```



# INICIO

```
texto = f" Hola, somos Manuel, Mateo y Vicente. Ante la reciente aparición del verano Joven, pretendemos facilitar su uso en la selección de viajes en tren y autobus. Por ello queremos mantener informados a nuestros usuarios sobre viajes de tren y autobus de toda España."  
print(texto)  
ok=True  
while ok:  
    modo=input("¿Cuál es tu medio de transporte? (tren/autobus)")  
    m=modo.lower()  
    fecha_ida_usuario = input('Introduce la fecha de vuelta')  
    fecha_vuelta_usuario = input('Introduce la fecha de ida')  
    origen_usuario = input('Introduce la ciudad de origen')  
    destino_usuario = input('Introduce la ciudad de destino')  
    if m=='tren' or m=='autobus':  
        ok=False  
    else:  
        print(f"No has introducido ninguna consulta que podamos resolver. Por favor introduce algunos de los valores especificados")  
  
if m=='tren':  
    tren()  
    r = input('¿Quieres consultar los precios de los autobuses? (si/no)')  
    if r.lower() == 'si':  
        autobus()  
  
elif m=='autobus':  
    autobus()  
    r = input('¿Quieres consultar los precios de los trenes? (si/no)')  
    if r.lower() == 'si':  
        tren()
```

Hola, somos Manuel, Mateo y Vicente. Ante la reciente aparición del verano Joven, pretendemos facilitar su uso en la selección de viajes en tren y autobus. Por ello queremos mantener informados a nuestros usuarios sobre viajes de tren y autobus de toda España.

¿Cuál es tu medio de transporte? (tren/autobus)



# TREN/AVE RENFE

```
def tren():
    driver = webdriver.Chrome()
    driver.get('https://www.renfe.com')
    driver.maximize_window()
    esperar = WebDriverWait(driver, 10)

    time.sleep(3)

    cookies = driver.find_element(By.CSS_SELECTOR, '#onetrust-accept-btn-handler')
    cookies.click()

    time.sleep(1)
    driver.refresh()

    time.sleep(5)

    origen = driver.find_element(By.CSS_SELECTOR, '#origin')
    origen.send_keys(origen_usuario)
    ul_element = esperar.until(EC.visibility_of_element_located((By.ID, "awesomplete_list_1")))
    primer_elemento = ul_element.find_element(By.CSS_SELECTOR, "#awesomplete_list_1_item_0")
    primer_elemento.click()

    time.sleep(1)

    destino = driver.find_element(By.CSS_SELECTOR, '#destination')
    destino.send_keys(destino_usuario)
    ul_element = esperar.until(EC.visibility_of_element_located((By.ID, "awesomplete_list_2")))
    primer_elemento = ul_element.find_element(By.CSS_SELECTOR, "#awesomplete_list_2_item_0")
    primer_elemento.click()

    time.sleep(1)

    #fecha de ida
    # Extraer el día, mes y año de la fecha ingresada
    dia_ida, mes_ida, año_ida = fecha_ida_usuario.split('/')
    dia_ida = int(dia_ida)
    mes_ida = int(mes_ida)
    año_ida = int(año_ida)

    # Abrir el selector de fecha
    fecha_de_ida_boton = driver.find_element(By.ID, 'first-input')
    fecha_de_ida_boton.click()

    time.sleep(1)

    # Esperar que se abra el calendario
    calendario_ida_abierto = esperar.until(
        EC.visibility_of_element_located((By.CSS_SELECTOR, '#daterangev2 > section > div.lightpick_inner > div.lightpick_months')))
```

```
# Ajustar el calendario al mes y año correctos
# Verificar si el mes del calendario actual es el que el usuario eligió
mes_calendario_ida = driver.find_element(By.CSS_SELECTOR, '#daterangev2 > section > div.lightpick_inner > div.lightpick_month')
año_calendario_ida = driver.find_element(By.CSS_SELECTOR, '#daterangev2 > section > div.lightpick_inner > div.lightpick_year')
mes_calendario_ida = mes_a_numero(mes_calendario_ida)
año_calendario_ida = int(año_calendario_ida)

time.sleep(1)

while mes_calendario_ida != mes_ida or año_calendario_ida != año_ida:
    # Hacer clic para avanzar al siguiente mes
    siguiente_mes_ida = driver.find_element(By.CSS_SELECTOR,
                                              '#daterangev2 > section > div.lightpick_inner > div.lightpick_toolbar > button.lightpick_next-action')
    siguiente_mes_ida.click()

    # Pausa breve para esperar que el calendario se actualice
    time.sleep(1)

    # Actualizar mes y año del calendario después de avanzar
    mes_calendario_ida = driver.find_element(By.CSS_SELECTOR, '#daterangev2 > section > div.lightpick_inner > div.lightpick_month')
    año_calendario_ida = driver.find_element(By.CSS_SELECTOR, '#daterangev2 > section > div.lightpick_inner > div.lightpick_year')
    mes_calendario_ida = mes_a_numero(mes_calendario_ida)
    año_calendario_ida = int(año_calendario_ida)

    fecha_ida_elemento = driver.find_element(By.XPATH, f"//div[contains(@class, 'lightpick_day') and text()='{dia_ida}']")
    fecha_ida_elemento.click()

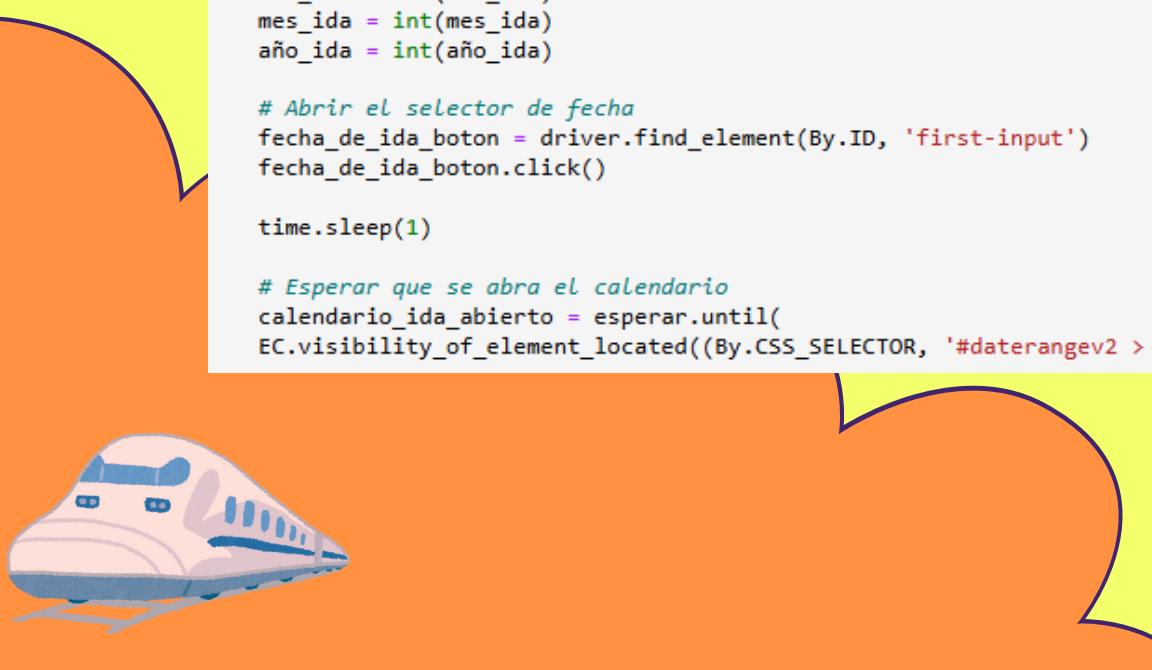
    time.sleep(1)

    #fecha de vuelta
    # Extraer el día, mes y año de la fecha ingresada
    dia_vuelta, mes_vuelta, año_vuelta = fecha_vuelta_usuario.split('/')
    dia_vuelta = int(dia_vuelta)
    mes_vuelta = int(mes_vuelta)
    año_vuelta = int(año_vuelta)

    # Ajustar el calendario al mes y año correctos
    # Verificar si el mes del calendario actual es el que el usuario eligió
    mes_calendario_vuelta = driver.find_element(By.CSS_SELECTOR, '#daterangev2 > section > div.lightpick_inner > div.lightpick_month')
    año_calendario_vuelta = driver.find_element(By.CSS_SELECTOR, '#daterangev2 > section > div.lightpick_inner > div.lightpick_year')
    mes_calendario_vuelta = mes_a_numero(mes_calendario_vuelta)
    año_calendario_vuelta = int(año_calendario_vuelta)

    time.sleep(1)

    while mes_calendario_vuelta != mes_vuelta or año_calendario_vuelta != año_vuelta:
        # Hacer clic para avanzar al siguiente mes
        siguiente_mes_vuelta = driver.find_element(By.CSS_SELECTOR,
                                                    '#daterangev2 > section > div.lightpick_inner > div.lightpick_toolbar > button.lightpick_next-action')
        siguiente_mes_vuelta.click()
```



# TREN/AVE RENFE

```

# Pausa breve para esperar que el calendario se actualice
time.sleep(1)

# Actualizar mes y año del calendario después de avanzar
mes_calendario_vuelta = driver.find_element(By.CSS_SELECTOR, '#daterangev2 > section > div.lightpick_inner > div.lightpick_header > div > span')
año_calendario_vuelta = driver.find_element(By.CSS_SELECTOR, '#daterangev2 > section > div.lightpick_inner > div.lightpick_header > div > span')
mes_calendario_vuelta = mes_a_numero(mes_calendario_vuelta)
año_calendario_vuelta = int(año_calendario_vuelta)

# Seleccionar el día del mes elegido
dia_vuelta_elemento = driver.find_element(By.XPATH, f"//div[contains(@class, 'lightpick_day') and text()='{dia_vuelta}']")
dia_vuelta_elemento.click()

time.sleep(1)

septar = driver.find_element(By.CSS_SELECTOR, '#daterangev2 > section > div.lightpick_footer-buttons > button.lightpick_apuntar')
septar.click()

time.sleep(5)

secar_billetes = driver.find_element(By.CSS_SELECTOR, '#ticketSearchBt > div > div > button')
secar_billetes.click()

time.sleep(5)

viajes = driver.find_elements(By.CSS_SELECTOR, 'div.row.selectedTren')
time.sleep(2)

viaje_data = []

for viaje in viajes:
    try:
        # Extraer los datos de cada viaje
        compania = 'RENFE'
        tipo_viaje = 'ida'
        origin = driver.find_element(By.CSS_SELECTOR, '#stv-ida > div.lugares > div:nth-child(1) > span:nth-child(3)').text
        destination = driver.find_element(By.CSS_SELECTOR, '#stv-ida > div.lugares > div:nth-child(2) > span:nth-child(3)').text
        departure_time = viaje.find_element(By.CSS_SELECTOR, 'div.col-md-8.trenes > h5').text
        arrival_time = viaje.find_element(By.CSS_SELECTOR, 'div.col-md-8.trenes > h5:nth-of-type(2)').text
        duration = viaje.find_element(By.CSS_SELECTOR, 'div.col-md-8.trenes > span.col.entre-horas > span.text-number').text
        price = viaje.find_element(By.CSS_SELECTOR, "div.col-md-4 > span.precio-final").text
        precio_final = price.splitlines()[-1]

        # Crear un diccionario para el viaje actual
        viaje_dict = {
            'compania': compania,
            'tipo_viaje': tipo_viaje,
            'origen': origin,
            'destino': destination,
            'hora_de_salida': departure_time,
            'hora_de_llegada': arrival_time,
            'duration': duration,
            'precio': precio_final
        }

        # Crear un diccionario para el viaje actual
        viaje_dict = {
            'compania': compania,
            'tipo_viaje': tipo_viaje,
            'origen': origin,
            'destino': destination,
            'hora_de_salida': departure_time,
            'hora_de_llegada': arrival_time,
            'duration': duration,
            'precio': precio_final
        }

        # Agregar el diccionario a la lista
        if all(viaje_dict.values()):
            viaje_data.append(viaje_dict)

    except:
        pass

    time.sleep(1)

    primer_billete = driver.find_element(By.CSS_SELECTOR, '#tren_i_1')
    primer_billete.click()
    time.sleep(1)

    primer_billete2 = driver.find_element(By.CSS_SELECTOR, '#planes-opciones_i_1')
    primer_billete2.click()
    time.sleep(1)

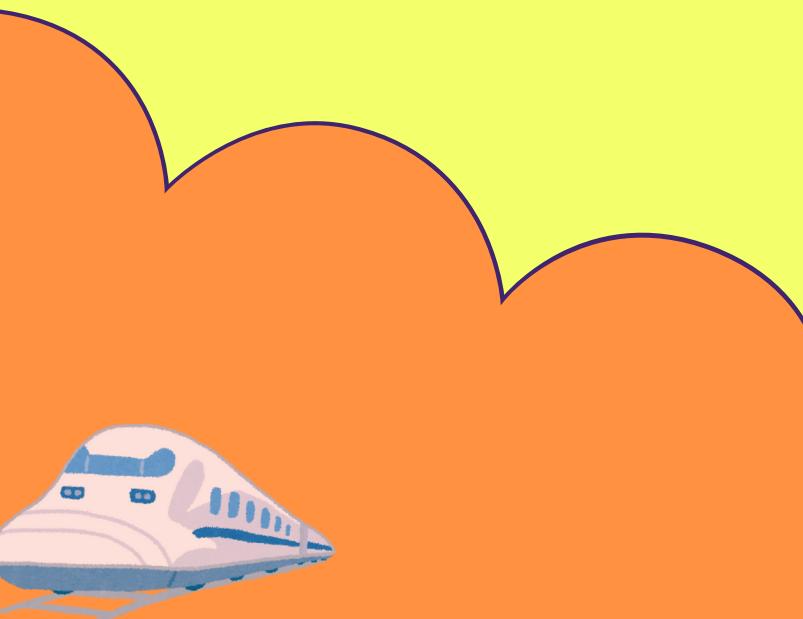
    primer_billete3 = driver.find_element(By.CSS_SELECTOR, '#btnSeleccionar')
    primer_billete3.click()
    time.sleep(1)

    primer_billete4 = driver.find_element(By.CSS_SELECTOR, '#cerrarConfirmacionMaletasAVLO')
    primer_billete4.click()
    time.sleep(5)

    viajes_vuelta = driver.find_elements(By.CSS_SELECTOR, 'div.row.selectedTren')
    viaje_vuelta_data = []

    for viaje in viajes_vuelta:
        try:
            # Extraer los datos de cada viaje
            compania = 'RENFE'
            tipo_viaje = 'vuelta'
            origin = driver.find_element(By.CSS_SELECTOR, '#stv-vuelta > div.lugares > div:nth-child(1) > span:nth-child(3)').text
            destination = driver.find_element(By.CSS_SELECTOR, '#stv-vuelta > div.lugares > div:nth-child(2) > span:nth-child(3)').text
            departure_time = viaje.find_element(By.CSS_SELECTOR, 'div.col-md-8.trenes > h5').text
            arrival_time = viaje.find_element(By.CSS_SELECTOR, 'div.col-md-8.trenes > h5:nth-of-type(2)').text
            duration = viaje.find_element(By.CSS_SELECTOR, 'div.col-md-8.trenes > span.col.entre-horas > span.text-number').text
            price = viaje.find_element(By.CSS_SELECTOR, "div.col-md-4 > span.precio-final").text
            precio_final = price.splitlines()[-1]
        
```

# TREN/AVE RENFE



```
primer_billete3 = driver.find_element(By.CSS_SELECTOR, '#btnSeleccionar')
primer_billete3.click()
time.sleep(1)

primer_billete4 = driver.find_element(By.CSS_SELECTOR, '#cerrarConfirmacionMaletasAVLO')
primer_billete4.click()
time.sleep(5)

viajes_vuelta = driver.find_elements(By.CSS_SELECTOR, 'div.row.selectedTren')

viaje_vuelta_data = []

for viaje in viajes_vuelta:
    try:
        # Extraer los datos de cada viaje
        compania = 'RENFE'
        tipo_viaje = 'vuelta'
        origin = driver.find_element(By.CSS_SELECTOR, '#stv-vuelta > div.lugares > div:nth-child(1) > span:nth-child(3)').text
        destination = driver.find_element(By.CSS_SELECTOR, '#stv-vuelta > div.lugares > div:nth-child(2) > span:nth-child(3)').text
        departure_time = viaje.find_element(By.CSS_SELECTOR, 'div.col-md-8.trenes > h5').text
        arrival_time = viaje.find_element(By.CSS_SELECTOR, 'div.col-md-8.trenes > h5:nth-of-type(2)').text
        duration = viaje.find_element(By.CSS_SELECTOR, 'div.col-md-8.trenes > span.col.entre-horas > span.text-number').text
        price = viaje.find_element(By.CSS_SELECTOR, "div.col-md-4 > span.precio-final").text
        precio_final = price.splitlines()[-1]

        # Crear un diccionario para el viaje actual
        viaje_vuelta_dict = {
            'compania' : compania,
            'tipo_viaje' : tipo_viaje,
            'origen': origin,
            'destino': destination,
            'hora_de_salida': departure_time,
            'hora_de_llegada': arrival_time,
            'duracion': duration,
            'precio': precio_final
        }

        # Agregar el diccionario a la lista
        if all(viaje_vuelta_dict.values()):
            viaje_vuelta_data.append(viaje_vuelta_dict)

    except:
        pass

driver.quit()
time.sleep(2)
```



# TREN/AVE OUIGO

```
driver = webdriver.Chrome()
driver.get('https://www.ouigo.com')
driver.maximize_window()
esperar = WebDriverWait(driver, 10)
time.sleep(5)
cookies = driver.find_element(By.CSS_SELECTOR, '#didomi-notice-agree-button')
cookies.click()

time.sleep(5)

iframe = driver.find_element(By.XPATH, "//iframe[@id='ouigo-sse-iframe-desktop']")
driver.switch_to.frame(iframe)

time.sleep(1)

origen = driver.find_element(By.XPATH, "//input[@id='origin-station-input-field_input']")
origen.click()
origen.send_keys(origen_usuario)
ul_element = esperar.until(EC.visibility_of_element_located((By.ID, "origin-station-input-listbox")))
primer_elemento = ul_element.find_element(By.XPATH, "//ul[@id='origin-station-input-listbox']/li[2]")
primer_elemento.click()

destino = driver.find_element(By.CSS_SELECTOR, '#destination-station-input-field_input')
destino.click()
destino.send_keys(destino_usuario)
ul_element = esperar.until(EC.visibility_of_element_located((By.ID, "destination-station-input-listbox")))
primer_elemento = ul_element.find_element(By.XPATH, "//ul[@id='destination-station-input-listbox']/li[1]")
primer_elemento.click()

#fecha de ida
# Extraer el día, mes y año de la fecha ingresada
dia_ida, mes_ida, año_ida = fecha_ida_usuario.split('/')
dia_ida = int(dia_ida)
mes_ida = int(mes_ida)
año_ida = int(año_ida)

# Abrir el selector de fecha
fecha_de_ida_boton = driver.find_element(By.ID, 'search-engine_inputfield_outbound-date_input')
fecha_de_ida_boton.click()

# Esperar que se abra el calendario
calendario_ida_abierto = esperar.until(
EC.visibility_of_element_located((By.CSS_SELECTOR, '#search-engine_inputfield_outbound-date_input-wrapper > div.sc-iGctRS.DHLjz')))

# Ajustar el calendario al mes y año correctos
# Verificar si el mes del calendario actual es el que el usuario eligió
fecha_calendario_ida = driver.find_element(By.CSS_SELECTOR, '#search-engine_inputfield_outbound-date_input-wrapper > div.sc-iGctRS.DHLjz')
mes_calendario_ida, año_calendario_ida = fecha_calendario_ida.split(' ')
mes_calendario_ida = mes_a_numero(mes_calendario_ida)
año_calendario_ida = int(año_calendario_ida)
```

```
while mes_calendario_ida != mes_ida or año_calendario_ida != año_ida:
    # Hacer clic para avanzar al siguiente mes
    siguiente_mes_ida = driver.find_element(By.CSS_SELECTOR,
        '#search-engine_inputfield_outbound-date_input-wrapper > div.sc-iGctRS.DHLjz > div > div.sc-kiYtDG.hHYCJM > div.sc-iGctRS.DHLjz')
    siguiente_mes_ida.click()

    # Pausa breve para esperar que el calendario se actualice
    time.sleep(1)

    # Actualizar mes y año del calendario después de avanzar
    fecha_calendario_ida = driver.find_element(By.CSS_SELECTOR,
        '#search-engine_inputfield_outbound-date_input-wrapper > div.sc-iGctRS.DHLjz > div > div.sc-kiYtDG.hHYCJM > div.sc-iGctRS.DHLjz')
    mes_calendario_ida, año_calendario_ida = fecha_calendario_ida.split(' ')
    mes_calendario_ida = mes_a_numero(mes_calendario_ida)
    año_calendario_ida = int(año_calendario_ida)

    # Ahora seleccionar el día del mes elegido
    fecha_ida_elemento = driver.find_element(By.XPATH, f"//li[@role='button' and @title='{fecha_ida_usuario}']//small[text()='{dia_ida}']")
    fecha_ida_elemento.click()

    #fecha de vuelta
    # Extraer el día, mes y año de la fecha ingresada
    dia_vuelta, mes_vuelta, año_vuelta = fecha_vuelta_usuario.split('/')
    dia_vuelta = int(dia_vuelta)
    mes_vuelta = int(mes_vuelta)
    año_vuelta = int(año_vuelta)

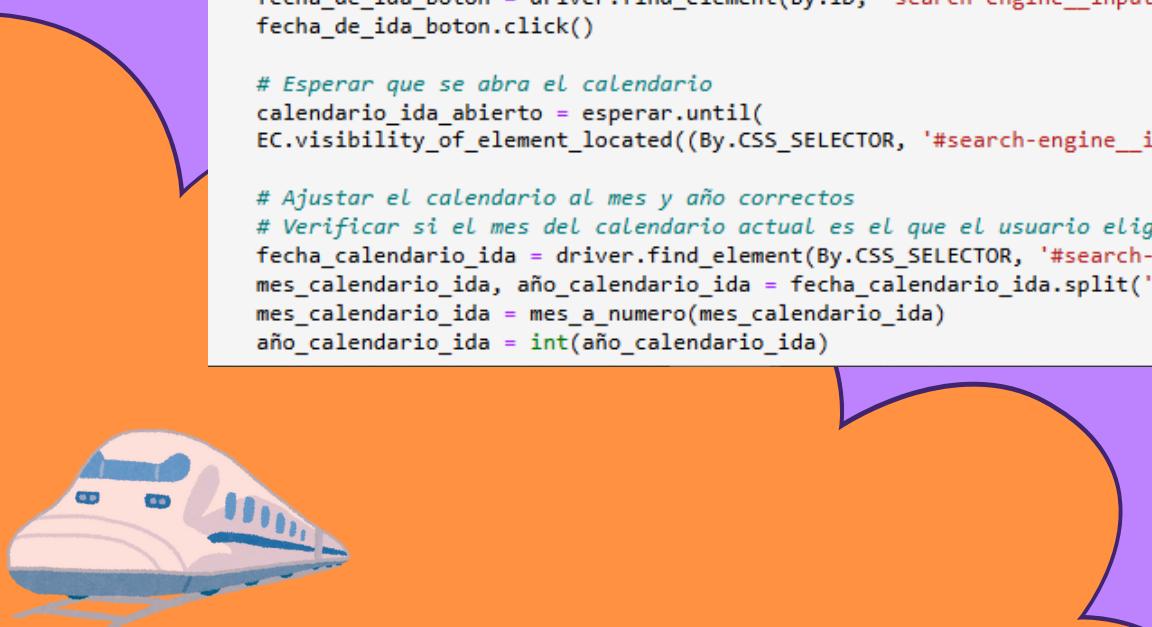
    # Abrir el selector de fecha
    fecha_de_vuelta_boton = driver.find_element(By.ID, 'search-engine_inputfield_inbound-date_input')
    fecha_de_vuelta_boton.click()

    # Esperar que se abra el calendario
    calendario_vuelta_abierto = esperar.until(
    EC.visibility_of_element_located((By.CSS_SELECTOR, '#search-engine_inputfield_inbound-date_input-wrapper > div.sc-iGctRS.DHLjz')))

    # Ajustar el calendario al mes y año correctos
    # Verificar si el mes del calendario actual es el que el usuario eligió
    fecha_calendario_vuelta = driver.find_element(By.CSS_SELECTOR, '#search-engine_inputfield_inbound-date_input-wrapper > div.sc-iGctRS.DHLjz')
    mes_calendario_vuelta, año_calendario_vuelta = fecha_calendario_vuelta.split(' ')
    mes_calendario_vuelta = mes_a_numero(mes_calendario_vuelta)
    año_calendario_vuelta = int(año_calendario_vuelta)

    while mes_calendario_vuelta != mes_vuelta or año_calendario_vuelta != año_vuelta:
        # Hacer clic para avanzar al siguiente mes
        siguiente_mes_vuelta = driver.find_element(By.CSS_SELECTOR,
            '#search-engine_inputfield_inbound-date_input-wrapper > div.sc-iGctRS.DHLjz > div > div.sc-kiYtDG.hHYCJM > div.sc-iGctRS.DHLjz')
        siguiente_mes_vuelta.click()

        # Pausa breve para esperar que el calendario se actualice
        time.sleep(1)
```



# TREN/AVE OUIGO

```
# Actualizar mes y año del calendario después de avanzar
fecha_calendario_vuelta = driver.find_element(By.CSS_SELECTOR,
    '#search-engine__inputfield__inbound-date__input-wrapper > div.sc-iGctRS.DHLjz > div > div.sc-kiYtDG.hHYCJM > div.sc-kiYtDG.hHYCJM > div.sc-kiYtDG.hHYCJM')
mes_calendario_vuelta, año_calendario_vuelta = fecha_calendario_vuelta.split(' ')
mes_calendario_vuelta = mes_a_numero(mes_calendario_vuelta)
año_calendario_vuelta = int(año_calendario_vuelta)

# Ahora seleccionar el día del mes elegido
fecha_vuelta_elemento = driver.find_element(By.XPATH, f"//li[@role='button' and @title='{fecha_vuelta_usuario}']//small[text()='{fecha_vuelta_usuario}']")
fecha_vuelta_elemento.click()

time.sleep(1)

mostrar_viajes = driver.find_element(By.CSS_SELECTOR, '#sse > form > div:nth-child(3) > div.styled__FormField-sc-1u9qghp-3.st')
mostrar_viajes.click()

time.sleep(5)

driver.switch_to.default_content()

time.sleep(1)

viajes = driver.find_elements(By.CSS_SELECTOR, 'div.sc-dBoTce.jxwSFw')

for viaje in viajes:
    try:
        # Extraer los datos de cada viaje
        compania = 'OUIGO'
        tipo_viaje = 'ida'
        origin = viaje.find_element(By.CSS_SELECTOR, 'div.sc-fBvYcn.hKdbKO > div.sc-lefGxV.kKDvDE').text
        destination = viaje.find_element(By.CSS_SELECTOR, 'div.sc-fBvYcn.hKdbKO > div.sc-kTLXwr.hwIEgc').text
        departure_time = viaje.find_element(By.CSS_SELECTOR, 'div.sc-fBvYcn.hKdbKO > div.sc-fMp0BE.jhCoH').text
        arrival_time = viaje.find_element(By.CSS_SELECTOR, 'div.sc-fBvYcn.hKdbKO > div.sc-fBVFAa.hcwhqo').text
        duration = viaje.find_element(By.CSS_SELECTOR, 'div.sc-dBoTce.jxwSFw > div.sc-tUeKj.fCtpvv').text
        price = viaje.find_element(By.CSS_SELECTOR, "div.sc-dBoTce.jxwSFw > span > div.sc-euehnN.dxXwnp").text

        # Crear un diccionario para el viaje actual
        viaje_dict = {
            'compania': compania,
            'tipo_viaje': tipo_viaje,
            'origen': origin,
            'destino': destination,
            'hora_de_salida': departure_time,
            'hora_de_llegada': arrival_time,
            'duracion': duration,
            'precio': price
        }

        # Agregar el diccionario a la lista
        if all(viaje_dict.values()):
            viaje_data.append(viaje_dict)

    except:
        pass

# Agregar el diccionario a la lista
if all(viaje_dict.values()):
    viaje_data.append(viaje_dict)

except:
    pass

primer_billete = driver.find_element(By.CSS_SELECTOR, 'div.sc-gqARDb.dqlszy > div > section > div > ul.sc-egAHoP.bzCyQy')
primer_billete.click()
time.sleep(5)
continuar = driver.find_element(By.CSS_SELECTOR, 'div.sc-kYnaAv.bfrGKw > button.sc-bdfBwQ.lkeddT.sc-kkjstb.cLiTKz')
continuar.click()
time.sleep(5)

reservar = driver.find_element(By.CSS_SELECTOR, 'div.sc-hKkhaR.iLjDAf > button.sc-bdfBwQ.lkeddT')
reservar.click()
time.sleep(5)

seguir = driver.find_element(By.CSS_SELECTOR, 'button.sc-bdfBwQ.eaWgkD')
seguir.click()
time.sleep(5)

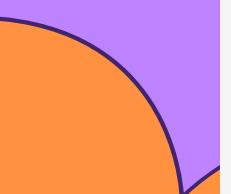
viajes_vuelta = driver.find_elements(By.CSS_SELECTOR, 'div.sc-dBoTce.jxwSFw')

for viaje in viajes_vuelta:
    try:
        # Extraer los datos de cada viaje
        compania = 'OUIGO'
        tipo_viaje = 'vuelta'
        origin_vuelta = viaje.find_element(By.CSS_SELECTOR, 'div.sc-fBvYcn.hKdbKO > div.sc-lefGxV.kKDvDE').text
        destination_vuelta = viaje.find_element(By.CSS_SELECTOR, 'div.sc-fBvYcn.hKdbKO > div.sc-kTLXwr.hwIEgc').text
        departure_time_vuelta = viaje.find_element(By.CSS_SELECTOR, 'div.sc-fBvYcn.hKdbKO > div.sc-fMp0BE.jhCoH').text
        arrival_time_vuelta = viaje.find_element(By.CSS_SELECTOR, 'div.sc-fBvYcn.hKdbKO > div.sc-fBVFAa.hcwhqo').text
        duration_vuelta = viaje.find_element(By.CSS_SELECTOR, 'div.sc-dBoTce.jxwSFw > div.sc-tUeKj.fCtpvv').text
        price_vuelta = viaje.find_element(By.CSS_SELECTOR, "div.sc-dBoTce.jxwSFw > span > div.sc-euehnN.dxXwnp").text

        # Crear un diccionario para el viaje actual
        viaje_vuelta_dict = {
            'compania': compania,
            'tipo_viaje': tipo_viaje,
            'origen': origin_vuelta,
            'destino': destination_vuelta,
            'hora_de_salida': departure_time_vuelta,
            'hora_de_llegada': arrival_time_vuelta,
            'duracion': duration_vuelta,
            'precio': price_vuelta
        }

        # Agregar el diccionario a la lista
        if all(viaje_vuelta_dict.values()):
            viaje_vuelta_data.append(viaje_vuelta_dict)

    except:
        pass
```



# TREN/AVE OUIGO

```
# Crear un diccionario para el viaje actual
viaje_vuelta_dict = {
    'compania': compania,
    'tipo_viaje': tipo_viaje,
    'origen': origin_vuelta,
    'destino': destination_vuelta,
    'hora_de_salida': departure_time_vuelta,
    'hora_de_llegada': arrival_time_vuelta,
    'duracion': duration_vuelta,
    'precio': price_vuelta
}

# Agregar el diccionario a la lista
if all(viaje_vuelta_dict.values()):
    viaje_vuelta_data.append(viaje_vuelta_dict)

except:
    pass

driver.quit()

numero_viajes_de_ida = len(viaje_data)
# Imprimir los datos recopilados
print(f'Número de viajes de ida en tren: {numero_viajes_de_ida}')

if viaje_data:
    viaje_mas_barato = min(viaje_data, key=lambda x: x['precio'])
    print("\nViaje con el precio más barato en tren:")
    print(viaje_mas_barato)
else:
    print("No se encontraron viajes.")

numero_viajes_de_vuelta = len(viaje_vuelta_data)
# Imprimir los datos recopilados
print(f'Número de viajes de vuelta en tren: {numero_viajes_de_vuelta}')

if viaje_vuelta_data:
    viaje_mas_barato = min(viaje_vuelta_data, key=lambda x: x['precio'])
    print("\nViaje con el precio más barato en tren:")
    print(viaje_mas_barato)
else:
    print("No se encontraron viajes.")
```



# AUTOBÚS

```
def autobus():
    driver = webdriver.Chrome()
    esperar = WebDriverWait(driver, 10)
    driver.get('https://www.alsa.es')
    driver.maximize_window()

    time.sleep(2)

    cookies = driver.find_element(By.CSS_SELECTOR, '#didomi-notice-agree-button')
    cookies.click()

    time.sleep(2)

    #origen
    origen = driver.find_element(By.CSS_SELECTOR, '#_originStationNameId_')
    origen.click()
    time.sleep(1)

    buscador = driver.find_element(By.CSS_SELECTOR, '#_JourneySearchPortlet_WAR_Alsaportlet_INSTANCE_JourneySearch_21651890_origen')
    buscador.click()
    buscador.send_keys(origen_usuario)

    time.sleep(10)

    ul_element = esperar.until(EC.visibility_of_element_located((By.CSS_SELECTOR, "#ui-id-1")))
    primer_elemento = ul_element.find_element(By.CSS_SELECTOR, "li:first-child")
    primer_elemento.click()

    #destino
    destino = driver.find_element(By.CSS_SELECTOR, '#_destinationStationNameId_')
    destino.click()
    time.sleep(1)

    buscador_destino = driver.find_element(By.CSS_SELECTOR, '#_JourneySearchPortlet_WAR_Alsaportlet_INSTANCE_JourneySearch_21651890_destino')
    buscador_destino.click()
    buscador_destino.send_keys(destino_usuario)

    time.sleep(1)

    destino_ul_element = esperar.until(EC.visibility_of_element_located((By.ID, "ui-id-2")))
    destino_primer_elemento = destino_ul_element.find_element(By.CSS_SELECTOR, "li:first-child")
    destino_primer_elemento.click()

    time.sleep(1)

#fecha de ida
# Extraer el día, mes y año de la fecha ingresada
dia_ida, mes_ida, año_ida = fecha_ida_usuario.split('/')
dia_ida = int(dia_ida)
mes_ida = int(mes_ida)

# Abrir el selector de fecha
fecha_de_ida_boton = driver.find_element(By.ID, '_JourneySearchPortlet_WAR_Alsaportlet_INSTANCE_JourneySearch_21651890_departureDatepicker')
fecha_de_ida_boton.click()

time.sleep(2)

# Esperar que se abra el calendario
calendario_ida_abierto = esperar.until(
    EC.visibility_of_element_located((By.ID, 'containerDatepickerOutward')))

# Ajustar el calendario al mes y año correctos
# Verificar si el mes del calendario actual es el que el usuario eligió
mes_calendario_ida = driver.find_element(By.CSS_SELECTOR, 'div.ui-datepicker-title > span.ui-datepicker-month').text
mes_calendario_ida = mes_a_numero(mes_calendario_ida)

# Si el mes y el año no coinciden con lo que el usuario ha elegido, navegar al mes y año correcto
while mes_calendario_ida != mes_ida:
    siguiente_mes_ida = driver.find_element(By.CSS_SELECTOR, 'a.ui-datepicker-next')
    siguiente_mes_ida.click()

    time.sleep(1)

# Actualizar el mes y año del calendario después de navegar
mes_calendario_ida = driver.find_element(By.CSS_SELECTOR, 'div.ui-datepicker-title > span.ui-datepicker-month').text
mes_calendario_ida = mes_a_numero(mes_calendario_ida)

# Ahora seleccionar el día del mes elegido
fecha_ida_elemento = driver.find_element(By.XPATH, f"//a[text()='{dia_ida}']")
fecha_ida_elemento.click()

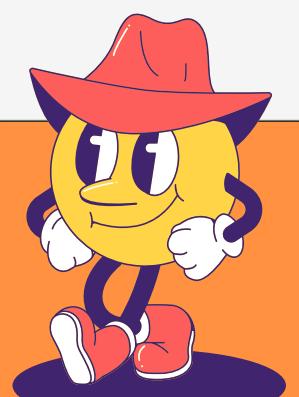
time.sleep(2)

#fecha de vuelta
# Extraer el día, mes y año de la fecha ingresada
dia_vuelta, mes_vuelta, año_vuelta = fecha_vuelta_usuario.split('/')
dia_vuelta = int(dia_vuelta)
mes_vuelta = int(mes_vuelta)

# Abrir el selector de fecha
fecha_de_vuelta_boton = driver.find_element(By.ID, '_JourneySearchPortlet_WAR_Alsaportlet_INSTANCE_JourneySearch_21651890_returnDatepicker')
fecha_de_vuelta_boton.click()

time.sleep(2)

# Esperar que se abra el calendario
```



# AUTOBÚS

```
# Esperar que se abra el calendario
calendario_vuelta_abierto = esperar.until(EC.visibility_of_element_located((By.ID, 'containerDatepickerReturn')))

# Ajustar el calendario al mes y año correctos
# Verificar si el mes del calendario actual es el que el usuario eligió
mes_calendario_vuelta = driver.find_element(By.CSS_SELECTOR, 'div.ui-datepicker-title > span.ui-datepicker-month').text
mes_calendario_vuelta = mes_a_numero(mes_calendario_vuelta)

# Si el mes y el año no coinciden con lo que el usuario ha elegido, navegar al mes y año correcto
while mes_calendario_vuelta != mes_vuelta:
    siguiente_mes_vuelta = driver.find_element(By.CSS_SELECTOR, 'a.ui-datepicker-next')
    siguiente_mes_vuelta.click()

    time.sleep(1)

# Actualizar el mes y año del calendario después de navegar
mes_calendario_vuelta = driver.find_element(By.CSS_SELECTOR, 'div.ui-datepicker-title > span.ui-datepicker-month').text
mes_calendario_vuelta = mes_a_numero(mes_calendario_vuelta)

# Ahora seleccionar el día del mes elegido
fecha_vuelta_elemento = driver.find_element(By.XPATH, f"//a[text()='{dia_vuelta}']")
fecha_vuelta_elemento.click()

time.sleep(1)

mostrar_viajes = driver.find_element(By.CSS_SELECTOR, '#journeySearchFormButtonjs')
mostrar_viajes.click()

time.sleep(10)

# Esperar hasta que se carguen los elementos de los viajes
viajes = esperar.until(EC.presence_of_all_elements_located((By.CSS_SELECTOR, 'div.content-journey.body-row.row.slide-in-resu

# Lista para almacenar los datos de los viajes
viaje_data = []

for viaje in viajes:
    try:
        # Extraer los datos de cada viaje
        tipo_viaje = 'ida'
        origin = viaje.find_element(By.CSS_SELECTOR, 'p.destinos > span.resaltado.left.icn-arrow-right.ng-binding').text
        destination = viaje.find_element(By.CSS_SELECTOR, 'p.destinos > span.right.ng-binding').text
        departure_time = viaje.find_element(By.CSS_SELECTOR, 'p.return.ng-scope > span.hora.hora-salida.left.ng-binding').text
        arrival_time = viaje.find_element(By.CSS_SELECTOR, 'p.return.ng-scope > span.hora.hora-llegada.right.ng-binding').text
        duration = viaje.find_element(By.CSS_SELECTOR, 'p.return.ng-scope > span.time-travel > span.time-travel.time.ng-bind
        price = viaje.find_element(By.CSS_SELECTOR, "span.bottom.btn.btn-primary.ng-binding.ng-scope").text

        # Crear un diccionario para el viaje actual
        viaje_dict = {
            'tipo_viaje': tipo_viaje,
```

```
# Crear un diccionario para el viaje actual
viaje_dict = {
    'tipo_viaje': tipo_viaje,
    'origen': origin,
    'destino': destination,
    'hora_de_salida': departure_time,
    'hora_de_llegada': arrival_time,
    'duracion': duration,
    'precio': price
}

# Agregar el diccionario a la lista
if all(viaje_dict.values()):
    viaje_data.append(viaje_dict)

except:
    pass

numero_viajes_de_ida = len(viaje_data)
# Imprimir los datos recopilados
print(f'Número de viajes de ida en bus: {numero_viajes_de_ida}')

if viaje_data:
    viaje_mas_barato = min(viaje_data, key=lambda x: x['precio'])
    print("\nViaje con el precio más barato de ida en bus:")
    print(viaje_mas_barato)
else:
    print("No se encontraron viajes en bus.")

time.sleep(10)

primer_billete = driver.find_element(By.CSS_SELECTOR, '#defaultFareOutward0 > span.precio-billete')
primer_billete.click()

time.sleep(10)

# Esperar hasta que se carguen los elementos de los viajes
viajes_vuelta = esperar.until(EC.presence_of_all_elements_located((By.CSS_SELECTOR, 'div.content-journey.body-row.row.slide-i

# Lista para almacenar los datos de los viajes
viaje_vuelta_data = []

for viaje in viajes_vuelta:
    try:
        # Extraer los datos de cada viaje
        tipo_viaje = 'vuelta'
        origin = viaje.find_element(By.CSS_SELECTOR, 'p.destinos > span.resaltado.left.icn-arrow-right.ng-binding').text
        destination = viaje.find_element(By.CSS_SELECTOR, 'p.destinos > span.right.ng-binding').text
        departure_time = viaje.find_element(By.CSS_SELECTOR, 'p.return.ng-scope > span.hora.hora-salida.left.ng-binding').text
        arrival_time = viaje.find_element(By.CSS_SELECTOR, 'p.return.ng-scope > span.hora.hora-llegada.right.ng-binding').text
        duration = viaje.find_element(By.CSS_SELECTOR, 'p.return.ng-scope > span.time-travel > span.time-travel.time.ng-bind
```



# AUTOBÚS

```
time.sleep(10)

# Esperar hasta que se carguen los elementos de los viajes
viajes_vuelta = esperar.until(EC.presence_of_all_elements_located((By.CSS_SELECTOR, 'div.content-journey.body-row.row.slide-i

# Lista para almacenar los datos de los viajes
viaje_vuelta_data = []

for viaje in viajes_vuelta:
    try:
        # Extraer los datos de cada viaje
        tipo_viaje = 'vuelta'
        origin = viaje.find_element(By.CSS_SELECTOR, 'p.destinos > span.resaltado.left.icn-arrow-right.ng-binding').text
        destination = viaje.find_element(By.CSS_SELECTOR, 'p.destinos > span.right.ng-binding').text
        departure_time = viaje.find_element(By.CSS_SELECTOR, 'p.return.ng-scope > span.hora.hora-salida.left.ng-binding').text
        arrival_time = viaje.find_element(By.CSS_SELECTOR, 'p.return.ng-scope > span.hora.hora-llegada.right.ng-binding').text
        duration = viaje.find_element(By.CSS_SELECTOR, 'p.return.ng-scope > span.time-travel > span.time-travel.time.ng-bindin
        price = viaje.find_element(By.CSS_SELECTOR, "span.bottom.btn.btn-primary.ng-binding.ng-scope").text

        # Crear un diccionario para el viaje actual
        viaje_vuelta_dict = {
            'tipo_viaje' : tipo_viaje,
            'origen': origin,
            'destino': destination,
            'hora_de_salida': departure_time,
            'hora_de_llegada': arrival_time,
            'duracion': duration,
            'precio': price
        }

        # Agregar el diccionario a la lista
        if all(viaje_vuelta_dict.values()):
            viaje_vuelta_data.append(viaje_vuelta_dict)

    except:
        pass

numero_viajes_de_vuelta = len(viaje_vuelta_data)
# Imprimir los datos recopilados
print(f'Número de viajes de vuelta en bus: {numero_viajes_de_vuelta}')

if viaje_vuelta_data:
    viaje_mas_barato = min(viaje_vuelta_data, key=lambda x: x['precio'])
    print("\nViaje con el precio más barato en bus:")
    print(viaje_mas_barato)
else:
    print("No se encontraron viajes en bus.")

driver.quit()
```



# VIDEO DE LA UTILIZACIÓN

- ❖ Hemos realizado un video para que se pueda ver el funcionamiento de nuestro trabajo

The screenshot shows a Linux desktop environment with several windows open:

- A file browser window showing files in the directory `/.../ATD/TRABAJO/`.
- A terminal window titled "Python 3 (ipykernel)" containing Python code for a travel booking system.
- A video editing application window showing a preview of a video and various editing tools.

The terminal code is as follows:

```
def diccionario():
    meses = { "Enero": 1, "Febrero": 2, "Marzo": 3, "Abril": 4, "Mayo": 5, "Junio": 6, "Julio": 7, "Agosto": 8, "Septiembre": 9, "Octubre": 10, "Noviembre": 11, "Diciembre": 12 }

    # Convertir el nombre del mes a minúsculas y buscar el número correspondiente
    return meses.get(nombre_mes.lower(), 'Mes inválido')

def main():
    texto = f" Hola, somos Manuel, Mateo y Vicente. Ante la reciente aparición del verano Joven, pretendemos facilitar su uso en la selección de viajes en tren y autobús. Por ello queremos mantener informados"
    print(texto)
    ok=True
    while ok:
        modo=input("¿Cuál es tu medio de transporte? (tren/autobús)")
        modo.lower()
        fecha_ida_usuario = input('Introduce la fecha de ida')
        fecha_vuelta_usuario = input('Introduce la fecha de vuelta')
        origen_usuario = input('Introduce el origen')
        destino_usuario = input('Introduce el destino')
        if modo=='tren' or modo=='autobus':
            ok=False
        else:
            print("No has introducido")
    if modo=='tren':
        tren()
        r = input('¿Quieres consultar?')
        if r.lower() == 'sí':
            autobus()
    elif modo=='autobus':
        autobus()
        r = input('¿Quieres consultar?')
        if r.lower() == 'sí':
            tren()

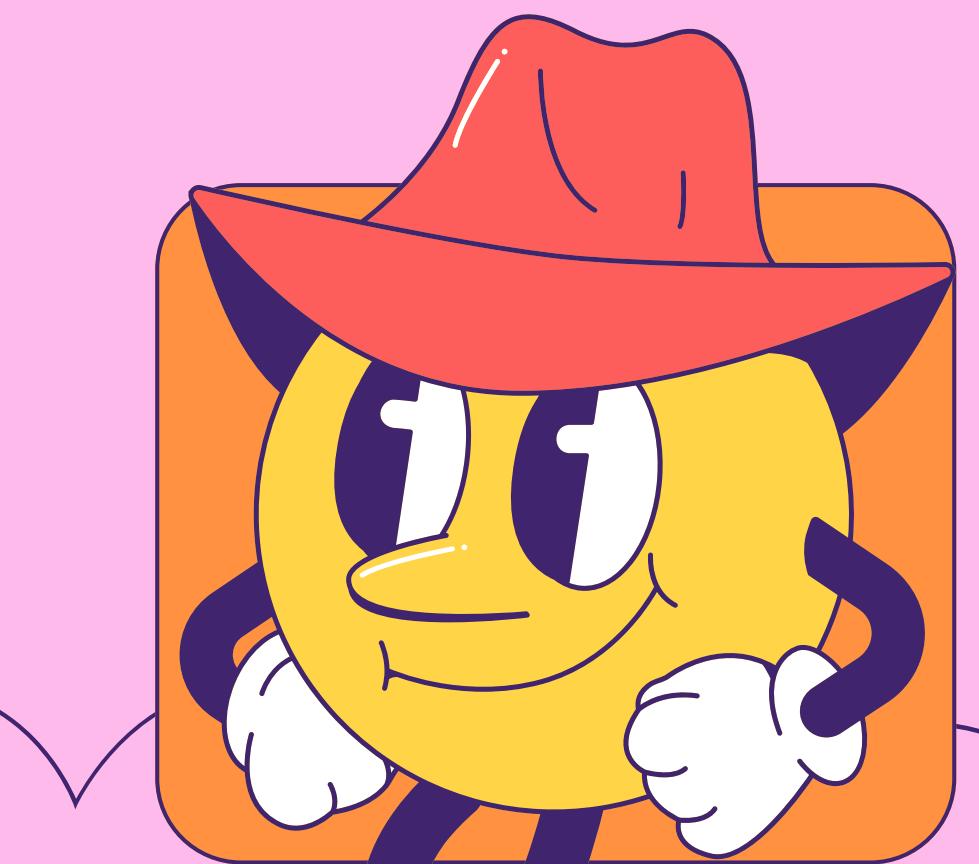
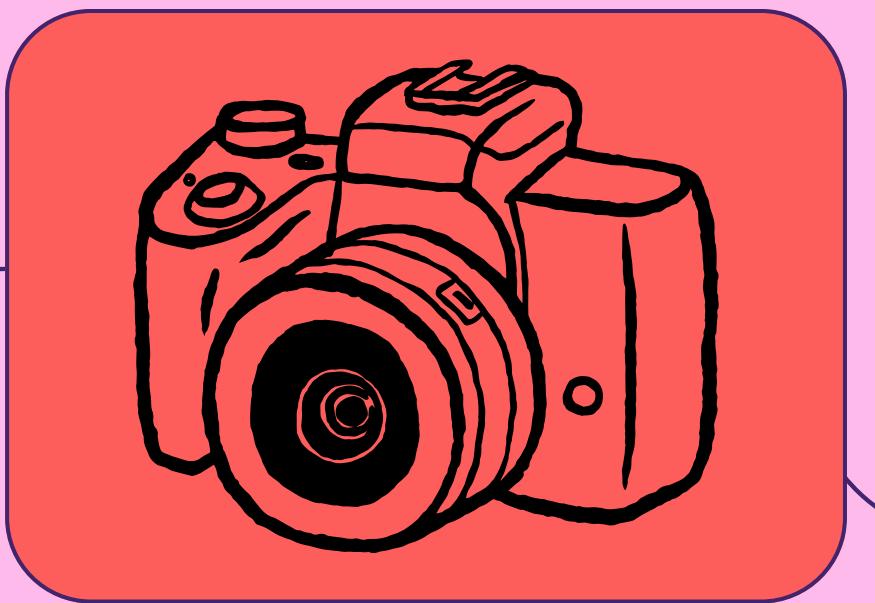
def autobus():
    driver = webdriver.Chrome()
    esperar = WebDriverWait(driver, 10)
    driver.get('https://www.alsa.es')
    driver.maximize_window()

    time.sleep(2)

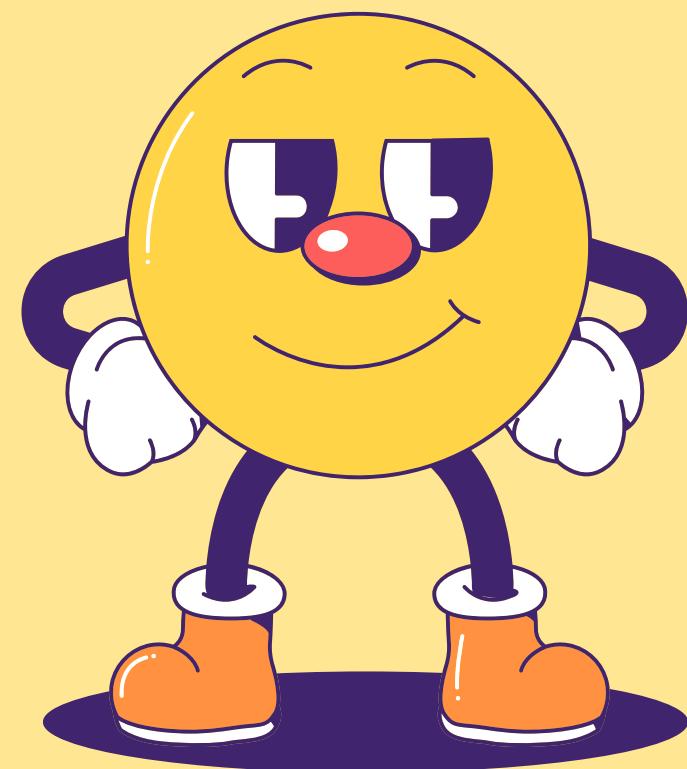
    cookies = driver.find_element(By.NAME, 'cookies')
    cookies.click()

    time.sleep(2)

#origen
```



# CONCLUSION



1

Cumplimos nuestros objetivos marcados en un principio para obtener viajes de tren/ave y de bus intentando obtener el precio más económico para cada trayecto.

2

Hemos aprendido a realizar Web Scraping de forma más fluida, aún que si fue necesario vernos varios video-tutoriales para afianzar nuestros conceptos.

3

Podremos obtener los viajes que necesitemos al mejor precio sin tener que ponernos a buscar en distintas webs e invertir nuestro tiempo en otras acciones.