Click to prove you're human

Unexpected UTF-8 BOM (decode using utf-8-sig): line 1 column 1 (char 0)# coding: utf-8from json import loadsfrom time import sleep, timefrom pickle import dump, loadfrom os.path import existsfrom selenium import webdriverfrom selenium.webdriver.common.by import Byfrom selenium.webdriver.support.ui import WebDriverWaitfrom selenium.webdriver.support import expected_conditions as ECfrom selenium.webdriver.common.desired_capabilities import DesiredCapabilities class Concert(object): def __init__(self, date, session, price, real_name, nick_name, ticket_num, damai_url, target_url,driver_path): self.date = date # self.session = session # self.price = price # self.real_name = real_name # self.status = 0 # self.time_start = 0 # self.time_end = 0 # self.num = 0 # self.ticket_num = ticket_num # self.nick_name = nick_name # self.damai_url = damai_url # self.target_url = target_url # self.driver_path = driver_path # self.driver = None def isClassPresent(self, item, name, ret=False): try: result = item.find_element_by_class_name(name) if ret: return result else: return True except: return False # cookie def get_cookie(self): self.driver.get(self.damai_url) print(u"######") self.driver.find_element_by_class_name('login-user').click() while self.driver.title.find('-') != -1: # sleep(1) print(u"######") while self.driver.title == '': # sleep(1) dump(self.driver.get_cookies(), open("cookies.pkl", "wb")) print(u"###Cookie###") def set_cookie(self): try: cookies = load(open("cookies.pkl", "rb")) # cookie for cookie in cookies: cookie_dict = { 'domain':'.damai.cn', # 'name': cookie.get('name'), 'value': cookie.get('value'), "expires": "", 'path': '/', 'httpOnly': False, 'HostOnly': False, 'Secure': False} self.driver.add_cookie(cookie_dict) print(u'###Cookie###') except Exception as e: print(e) def login(self): print(u'######') self.driver.get(self.target_url) WebDriverWait(self.driver, 10, 0.1).until(EC.title_contains('')) self.set_cookie() def enter_concert(self): print(u'######') if not exists('cookies.pkl'): # cookie.pkl, self.driver = webdriver.Chrome(executable_path=self.driver_path) self.get_cookie() print(u'###Cookie###') self.driver.quit() options = webdriver.ChromeOptions() # jscss prefs = {"profile.managed_default_content_settings.images": 2, "profile.managed_default_content_settings.javascript": 1, 'permissions.default.stylesheet': 2} options.add_experimental_option("prefs", prefs) # capa = DesiredCapabilities.CHROME capa["pageLoadStrategy"] = "none" self.driver = webdriver.Chrome(executable_path=self.driver_path, options=options, desired_capabilities=capa) # self.login() self.driver.refresh() try: # nickname locator = (By.XPATH, "/html/body/div[1]/div/div[3]/div[1]/a[2]/div") WebDriverWait(self.driver, 5, 0.3).until(EC.text_to_be_present_in_element(locator, self.nick_name)) self.status = 1 print(u"######") self.time_start = time() except: self.status = 0 self.driver.quit() raise Exception(u"***,cookie***") # def choose_ticket(self): print(u"######") while self.driver.title.find('') == -1: # self.num += 1 #1 if con.driver.current_url.find("buy.damai.cn") != -1: break # try: box = WebDriverWait(self.driver, 1, 0.1).until(EC.presence_of_element_located((By.CLASS_NAME, 'perform__order__box'))) except: raise Exception(u"***Error: ***") try: realname_popup = box.find_elements_by_xpath("//div[@class='realname-popup']") # if len(realname_popup) != 0: known_button = realname_popup[0].find_elements_by_xpath("//div[@class='operate']//div[@class='button']") known_button[0].click() except: raise Exception(u"***Error: ***") try: buybutton = box.find_element_by_class_name('buybtn') # buybutton_text = buybutton.text except: raise Exception(u"***Error: buybutton ***") if buybutton_text == "" or buybutton_text == "": self.status = 2 raise Exception(u"------") try: selects = box.find_elements_by_class_name('perform__order__select') # date = None # for item in selects: if item.find_element_by_class_name('select_left').text == '': date = item # print('\t') elif item.find_element_by_class_name('select_left').text == '': session = item # print('\t') elif item.find_element_by_class_name('select_left').text == '': price = item # print('\t') if date is not None: date_list = date.find_elements_by_xpath("//div[@class='wh_content_item']//div[starts-with(@class,'wh_item_date')]") # # print('{}'.format(len(date_list))) for i in self.date: j = date_list[i-1] j.click() break session_list = session.find_elements_by_class_name('select_right_list_item')# # print('{}'.format(len(session_list))) for i in self.session: # j = session_list[i-1] k = self.isClassPresent(j, 'presell', True) if k: # presell if k.text == '': continue elif k.text == '': j.click() break elif k.text == '': j.click() break else: j.click()# break price_list = price.find_elements_by_class_name('select_right_list_item')# # print('{}'.format(len(price_list))) for i in self.price: j = price_list[i-1] k = self.isClassPresent(j, 'notticket') if k: # notticket continue else: j.click()# break except: raise Exception(u"***Error: ororor***") try: ticket_num_up = box.find_element_by_class_name('cafe-c-input-number-handler-up') except: if buybutton_text == "": # buybutton.click() self.status = 5 print(u"######") break elif buybutton_text == "": raise Exception(u"###...###') else: raise Exception(u"***Error: ticket_num_up ***") if buybutton_text == "": for i in range(self.ticket_num-1): # ticket_num_up.click() buybutton.click() self.status = 3 elif buybutton_text == "": for i in range(self.ticket_num-1): # ticket_num_up.click() buybutton.click() self.status = 4 def check_order(self): if self.status in [3, 4, 5]: if self.real_name is not None: print(u"###------ CRTL+C --###") try: tb = WebDriverWait(self.driver, 1, 0.1).until(EC.presence_of_element_located((By.XPATH, '/html/body/div[3]/div[2]/div'))) except: raise Exception(u"***Error***") print(u'######') print(u'###,###') init_sleeptime = 0.0 Labels = tb.find_elements_by_tag_name('label') # while True: init_sleeptime += 0.1 true_num = 0 for num_people in self.real_name: tag_input = Labels[num_people-1].find_element_by_tag_name('input') if tag_input.get_attribute('aria-checked') == 'false': sleep(init_sleeptime) tag_input.click() else: true_num += 1 if true_num == len(self.real_name): break print("", time()-self.time_start) self.driver.find_element_by_xpath('/html/body/div[3]/div[2]/div/div[9]/button').click() # else: self.driver.find_element_by_xpath('/html/body/div[2]/div[2]/div/div[8]/button').click() # title print(u"###------ CRTL+C --###") try: WebDriverWait(self.driver, 3600, 0.1).until(EC.title_contains('')) except: raise Exception(u'***Error: ***') self.status = 6 print(u'###,###') self.time_end = time() if __name__ == '__main__': try: with open('./config.json', 'r', encoding='utf-8') as f: config = loads(f.read()) # params: , , con = Concert(config['date'], config['sess'], config['price'], config['real_name'], config['nick_name'], config['ticket_num'], config['damai_url'], config['target_url'], config['driver_path']) con.enter_concert() # except Exception as e: print(e) exit(1) while True: try: con.choose_ticket() con.check_order() except Exception as e: con.driver.get(con.target_url) print(e) continue if con.status == 6: print(u"###%d%.1f##" % (con.num, round(con.time_end-con.time_start, 3))) break 2025-05-07 19:30 : 0% **NMOSPMOS** MOSFETNMOSPMOSNMOSNMOSRds(on)PMOS PMOSPMOSMOSFET MOSFETMetal-Oxide-Semiconductor Field-Effect TransistorNMOSPMOS NMOSPMOS NMOSPMOS 2. NMOS NMOS NMOS NMOSPMOS NMOSRds(on)PMOS NMOS 3. PMOS PMOSNMOS NMOSPMOS PMOSPMOS 4. NMOSPMOS # CMOSclass CMOSInverter: def __init__(self, vdd): self.vdd = vdd def operate(self, input_voltage): if input_voltage == 0: # NMOSPMOS return self.vdd else: # NMOSPMOS return 0 NMOSPMOS 5. NMOSPMOS graph TD; A[] --> B{}; B -->|| C[NMOS]; B -->|| D[PMOS]; C --> E[]; D --> F[]; NMOSPMOSCMOS , ? , ? , ? (0) II 2023-09-25 17:57 : 0% >>_>_ 0 && operator.equals("")){ clear(); } if(operator.equals("")){ firstNum = firstNum + inPutText; }else{ secondNum = secondNum +inPutText; } if(showText.equals("0") && !inPutText.equals(".")) { refreshText(inPutText); }else{ ** refreshText(showText + inPutText);** } } } private double calculateFour() { switch(operator){ case "+": return Double.parseDouble(firstNum) + Double.parseDouble(secondNum); case "-": return Double.parseDouble(firstNum) - Double.parseDouble(secondNum); case "*": return Double.parseDouble(firstNum) * Double.parseDouble(secondNum); case "/": return Double.parseDouble(firstNum) / Double.parseDouble(secondNum); } return 0; } private void refreshText(String text) { showText = text; **txt.setText(showText);** } private void refreshOperate(String new_result){ result = new_result; firstNum = result; secondNum = ""; operator = ""; } private void clear() { refreshText(""); refreshOperate(""); }} CHATGPT Overleafpages' is a missing field, not a string, for entry XBibTeXpagesBibTeXrsc.bst15004711pagesBibTeXpagesBibTeXpagesBibTeXpages@article{example, author = "Author, A. and Author, B.", title = "Example Title", journal = "Journal Name", year = "2023", volume = "10", number = "2", pages = "123--456", % }@article{example_no_pages, author = "Author, A. and Author, B.", title = "Example Title without Pages", journal = "Journal Name", year = "2023", volume = "10", number = "2", pages = {}, % }OverleafLaTeXBibTeX.bibLaTeXBibTeXpagesLaTeXpages' is a missing field, not a stringPDF>>_

Cat mini excavator operating tips. Mini excavator how to operate. Excavator cat mini. How to operate a excavator cat. How to turn on cat mini excavator. How to drive a cat mini excavator. How to start a cat mini excavator. How do you start a cat mini excavator.