1. Будем тестировать калькулятор обоев:

https://kalk.pro/finish/wallpaper/

1. Попробуйте ввести в качестве исходных данных положительные числа, нажмите кнопку «Рассчитать», убедитесь в появлении результатов. Нажимайте правой кнопкой мыши на поля ввода и результаты расчета, выбирайте пункт «Просмотреть код», ознакомьтесь со структурой страницы, присвоенными элементам ID и классами.
2. Попробуйте ввести в качестве исходных данных отрицательные числа или буквы, нажмите кнопку «Рассчитать», убедитесь в появлении сообщения об ошибке. Нажмите правой кнопкой мыши на сообщение об ошибке, выберите пункт «Просмотреть код», ознакомьтесь со структурой страницы, присвоенными элементам ID и классами.
3. Ознакомьтесь с методикой расчета внизу страницы.
4. Создайте новый проект типа «Pure Python», создайте Python unit test. Первым делом протестируем корректность расчета площади стен и выдачу сообщений об ошибке при вводе некорректных данных:

import time

import unittest

from selenium import webdriver

from selenium.webdriver.common.keys import Keys

class Kalkpro(unittest.TestCase):

# https://kalk.pro/finish/wallpaper/

def setUp(self):

# запуск Firefox при начале каждого теста

self.driver = webdriver.Firefox()

# открытие страницы при начале каждого теста

self.driver.get('https://kalk.pro/finish/wallpaper/')

def tearDown(self):

# закрытие браузера при окончании каждого теста

self.driver.close()

# вспомогательный метод для заполнения размеров комнаты

def input\_walls\_data(self, hei, wid, len):

driver = self.driver

# высота

# находим элемент страницы по ID

elem = driver.find\_element\_by\_id("js--roomСeiling\_height")

# очищаем элемент от старого значения

elem.clear()

# вносим новое значение

elem.send\_keys(hei)

# ширина

elem = driver.find\_element\_by\_id("js--roomSizes\_width")

elem.clear()

elem.send\_keys(wid)

# длина

elem = driver.find\_element\_by\_id("js--roomSizes\_length")

elem.clear()

elem.send\_keys(len)

# метод для проверки работы калькулятора при различных

# размерах комнаты, в т. ч. проверка отображения ошибок

# при вводе некорректных данных

def test\_walls(self):

driver = self.driver

# пробуем ввести размер 1х1х1

self.input\_walls\_data(1, 1, 1)

# запускаем расчет

elem = driver.find\_element\_by\_class\_name(

"js--calcModelFormSubmit"

)

elem.click()

# проверяем наличие результатов расчета

self.assertIn('Результаты расчета', driver.page\_source)

# проверяем, что площадь четырех стех размером 1х1 равна 4

elem = driver.find\_element\_by\_css\_selector(

"ul.data-list li:nth-child(4) strong"

)

self.assertEqual('4 м²', elem.text)

# пробуем ввести размер 0х1х1

self.input\_walls\_data(0, 1, 1)

# запускаем расчет

elem = driver.find\_element\_by\_class\_name(

"js--calcModelFormSubmit"

)

elem.click()

# проверяем наличие сообщения об ошибке

self.assertIn('Ошибки', driver.page\_source)

# проверяем наличие одной ссылки на поле ввода

elems = driver.find\_elements\_by\_css\_selector(

"a.js--onclick-goToField"

)

self.assertEqual(len(elems), 1)

# пробуем ввести размер aхbхc (буквы вместо цифр)

self.input\_walls\_data('a', 'b', 'c')

# запускаем расчет

elem = driver.find\_element\_by\_class\_name(

"js--calcModelFormSubmit"

)

elem.click()

# проверяем наличие сообщения об ошибке

self.assertIn('Ошибки', driver.page\_source)

# проверяем наличие трех ссылок на поля ввода

elems = driver.find\_elements\_by\_css\_selector(

"a.js--onclick-goToField"

)

self.assertEqual(len(elems), 3)

if \_\_name\_\_ == '\_\_main\_\_':

unittest.main()

1. Запустите тест, убедитесь, что он проходит успешно.
2. Добавьте новый тест для проверки корректности уменьшения площади при добавлении окон и дверей:

# метод для проверки работы калькулятора

# при добавлении окон и дверей

def test\_windows(self):

driver = self.driver

# пробуем ввести размер 1х1х1

self.input\_walls\_data(1, 1, 1)

# добавляем окно

elem = driver.find\_element\_by\_css\_selector(

"fieldset[name=windows] button"

)

elem.click()

# задаем размеры окна 1х1 - во всю стену, кол-во окон - 1

elem = driver.find\_element\_by\_id("js--windows\_height\_0")

elem.clear()

elem.send\_keys(1)

elem = driver.find\_element\_by\_id("js--windows\_width\_0")

elem.clear()

elem.send\_keys(1)

elem = driver.find\_element\_by\_id("js--windows\_count\_0")

elem.clear()

elem.send\_keys(1)

# запускаем расчет

elem = driver.find\_element\_by\_class\_name(

"js--calcModelFormSubmit"

)

elem.click()

# проверяем наличие результатов расчета

self.assertIn('Результаты расчета', driver.page\_source)

# проверяем, что площадь трех стен размером 1х1 равна 4

elem = driver.find\_element\_by\_css\_selector(

"ul.data-list li:nth-child(4) strong"

)

self.assertEqual('3 м²', elem.text)

# добавляем дверь

elem = driver.find\_element\_by\_css\_selector(

"fieldset[name=doors] button"

)

elem.click()

# задаем размеры двери 1х1 - во всю стену, кол-во дверей - 1

elem = driver.find\_element\_by\_id("js--doors\_height\_0")

elem.clear()

elem.send\_keys(1)

elem = driver.find\_element\_by\_id("js--doors\_width\_0")

elem.clear()

elem.send\_keys(1)

elem = driver.find\_element\_by\_id("js--doors\_count\_0")

elem.clear()

elem.send\_keys(1)

# запускаем расчет

elem = driver.find\_element\_by\_class\_name(

"js--calcModelFormSubmit"

)

elem.click()

# проверяем наличие результатов расчета

self.assertIn('Результаты расчета', driver.page\_source)

# проверяем, что площадь двух стен размером 1х1 равна 4

elem = driver.find\_element\_by\_css\_selector(

"ul.data-list li:nth-child(4) strong"

)

self.assertEqual('2 м²', elem.text)

1. Запустите тесты, убедитесь, что они проходит успешно.
2. В качестве данных для тестирования корректности расчетов воспользуемся таблицами, составленными по имеющимся на странице, – сохраните файл test53.csv в папку с проектом:

https://github.com/Burmakova/selenium-tests.git

1. Добавьте новый тест для запуска расчета по сохраненным данным:

# вспомогательный метод для заполнения данных из таблиц

def input\_from\_tables(self, hei, sq, rw):

driver = self.driver

# высота

# находим элемент страницы по ID

elem = driver.find\_element\_by\_id("js--roomСeiling\_height")

# очищаем элемент от старого значения

elem.clear()

# вносим новое значение

elem.send\_keys(str(hei))

# ширину и длину вычислим из площади в таблице:

# воспользуемся тем, что в таблице площадь четная,

# ширину будем считать равной 2

elem = driver.find\_element\_by\_id("js--roomSizes\_width")

elem.clear()

elem.send\_keys(2)

# длина

elem = driver.find\_element\_by\_id("js--roomSizes\_length")

elem.clear()

elem.send\_keys(str(sq/2))

# ширина рулона

elem = driver.find\_element\_by\_id("js--wallpaperSizes\_rollWidth")

elem.clear()

elem.send\_keys(rw)

# метод для проверки работы калькулятора

# на таблице с шириной 53

def test\_width\_53(self):

driver = self.driver

# добавляем окно

elem = driver.find\_element\_by\_css\_selector(

"fieldset[name=windows] button"

)

elem.click()

# задаем стандартный размер окон 1.17х1.46

elem = driver.find\_element\_by\_id("js--windows\_height\_0")

elem.clear()

elem.send\_keys('1.17')

elem = driver.find\_element\_by\_id("js--windows\_width\_0")

elem.clear()

elem.send\_keys('1.46')

# добавляем дверь

elem = driver.find\_element\_by\_css\_selector(

"fieldset[name=doors] button"

)

elem.click()

# задаем стандартный размер дверей 2.07х0.87

elem = driver.find\_element\_by\_id("js--doors\_height\_0")

elem.clear()

elem.send\_keys('2.07')

elem = driver.find\_element\_by\_id("js--doors\_width\_0")

elem.clear()

elem.send\_keys('0.87')

# задаем счетчик окон и дверей, который будет увеличиваться

# вместе с увеличением площади комнаты

windows\_count = 0

# открываем сохраненную таблицу

filedesc = open("test53.csv", "r")

# считываем весь файл как список строк

filecontent = filedesc.readlines()

# цикл по строкам файла

for i in range(len(filecontent)):

# пропускаем первую строку с заголовком

if i > 0:

# разбиение строки на ячейки по символу ";"

cells = filecontent[i].split(";")

# увеличиваем значение счетчика окон и дверей

windows\_count += 1

# вводим число окон

elem = driver.find\_element\_by\_id("js--windows\_count\_0")

elem.clear()

elem.send\_keys(windows\_count)

# вводим число дверей

elem = driver.find\_element\_by\_id("js--doors\_count\_0")

elem.clear()

elem.send\_keys(windows\_count)

# проверка корректности вычислений

# по первым двум столбцам (высота 2,25 м)

self.input\_from\_tables(float(cells[0]),float(cells[1]),53)

# запускаем расчет

elem = driver.find\_element\_by\_class\_name(

"js--calcModelFormSubmit"

)

elem.click()

# проверяем наличие результатов расчета

self.assertIn('Результаты расчета', driver.page\_source)

# проверяем, что площадь совпадает с табличной

elem = driver.find\_element\_by\_css\_selector(

"ul.data-list li:nth-child(1) strong"

)

print("Проверка для площади ", cells[1], ": ")

self.assertEqual(cells[2]+' шт', elem.text)

print("OK")

filedesc.close()

1. Запустите тесты, убедитесь, что они проходит успешно для всех строк файла.
2. Обратите внимание, что на тестируемой странице в правом нижнем углу находится кнопка вызова кнопочного калькулятора с соответствующей пиктограммой. Добавьте тест для тестирования кнопок набора цифр этого калькулятора:

# тестируем кнопочный калькулятор

def test\_calc(self):

driver = self.driver

# открываем калькулятор

elem = driver.find\_element\_by\_css\_selector(".spr--calculator")

elem.click()

# ждем, пока калькулятор откроется

time.sleep(5)

# перебираем цифры от 0 до 9 (ноль не будет отображаться перед 1)

for i in range(10):

print("Нажатие кнопки ", str(i), ": ")

elem = driver.find\_element\_by\_name(str(i))

elem.click()

print("OK")

# проверяем изображение на дисплее

# (ноль не будет отображаться перед 1)

elem = driver.find\_element\_by\_class\_name(

"display-indicator-ceils"

)

self.assertEqual(elem.text, '123456789')

1. Запустите тест, убедитесь, что он проходит успешно.
2. Создайте собственные тесты для тестирования каждой из основных арифметических операций (сложение, вычитание, умножение, деление).