

# QA Has Power Automation



Rafał Nowicki  
Dawid Kostyszak

**CODE YOUR PATH**

# Agenda

## 1. Selenium WebDriver

- a. wprowadzenie
- b. demo
- c. zadania

## 2. Splinter

- a. wprowadzenie
- b. demo
- c. zadania

## 3. Unittest

- a. wprowadzenie
- b. demo
- c. zadania

## 4. Behave

- a. wprowadzenie
- b. demo
- c. zadania





# Selenium WebDriver

Wprowadzenie

# Selenium WebDriver

Interfejs do komunikacji z przeglądarką internetową umożliwiający w pełni zautomatyzowaną interakcję z wszystkimi elementami wyświetlanych stron.

```
>> pip install selenium
```

# Przykład

```
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

browser = webdriver.Firefox()
browser.get("http://www.python.org")
assert "Python" in browser.title
elem = browser.find_element_by_name("q")
elem.clear()
elem.send_keys("pycon")
elem.send_keys(Keys.RETURN)
assert "No results found." not in browser.page_source
browser.close()
```

# Podstawowe metody

# Instancja przeglądarki

```
browser = webdriver.Firefox()
```

```
browser.quit()
```

```
browser.refresh()
```

# Nawigacja

```
browser.get('https://stxnext.com')
```

```
browser.back()
```

```
browser.forward()
```

# Treść

```
browser.title
```

```
browser.current_url
```

```
browser.page_source
```

# Szukanie

# Szukanie elementów

```
browser.find_element_by_css_selector('h1')
browser.find_element_by_xpath('//h1')
browser.find_element_by_tag_name('h1')
browser.find_element_by_name('name')
browser.find_element_by_class_name('name')
browser.find_element_by_id('firstheader')
browser.find_element_by_value('query')
```

# Szukanie linków

```
browser.find_element_by_link_text(
    'Link for Example.com'
)
browser.find_element_by_partial__link_text(
    'for Example'
)
```

# Gdy elementów jest wiele

```
browser.find_elements_by_*
```

# Szukanie wewnątrz komponentu

```
div = browser.find_element_by_tag_name("div")
div.find_element_by_name("name")
```

# Elementy

# Właściwości

`element.id`

`element.tag_name`

`element.text`

`element.is_displayed()`

`element.is_enabled()`

`element.is_selected()`

`element.get_property(name)`

`element.get_attribute(name)`

`element.value_of_css_property(name)`

# Interakcje

`element.click()`

`element.clear()`

`element.submit()`

`element.find_element*`

`element.send_keys(keys)`



# Formularze

```
# Pola tekstowe
field = browser.find_element_by_name(
    'fullName'
)
field.clear()
field.send_keys('Full Name')

# Button, checkbox, radio
btn = browser.find_element_by_name('submit')
btn.click()

# Wysłanie formularza
field.submit()
```

```
# Select
from selenium.webdriver.support.ui import Select

select = Select(
    browser.find_element_by_name('name')
)

select.select_by_index(index)
select.select_by_visible_text('text')
select.select_by_value(value)

select.deselect_by_*

select.deselect_all()
```

# Oczekiwanie na zmiany

```
# Oczekiwanie na spełnienie warunku
```

```
from selenium.webdriver.common.by import By
```

```
from selenium.webdriver.support.ui import WebDriverWait
```

```
from selenium.webdriver.support import expected_conditions as EC
```

```
wait = WebDriverWait(browser, 10)
```

```
wait.until(browser, EC.presence_of_element_located((By.ID, 'my_id')))
```

```
# Ustawienie globalne
```

```
browser.implicitly_wait(10)
```

# Oczekiwanie na zmiany - warunki

# Warunki

EC.title\_is(text)

EC.title\_contains(text)

EC.presence\_of\_element\_located(locator)

EC.presence\_of\_all\_elements\_located(locator)

EC.visibility\_of\_element\_located(locator)

EC.invisibility\_of\_element\_located(locator)

EC.element\_located\_to\_be\_selected(locator)

EC.element\_located\_selection\_state\_to\_be(locator, is\_selected)

EC.element\_to\_be\_clickable(locator)

EC.element\_to\_be\_selected(element)

EC.element\_selection\_state\_to\_be(element, is\_selected)

EC.staleness\_of(element)

EC.text\_to\_be\_present\_in\_element(locator, text)

# Typy lokatorów

By.ID

By.XPATH

By.LINK\_TEXT

By.PARTIAL\_LINK\_TEXT

By.NAME

By.TAG\_NAME

By.CLASS\_NAME

By.CSS\_SELECTOR

# Inne

- obsługa wielu okien przeglądarki
- obsługa ruchów myszy, drag and drop, double click, right click
- wykonywanie kodu javascript
- cookies injection



**Demo**

Selenium WebDriver





# Zadania I

Selenium WebDriver



# Zadania cz I: Selenium WebDriver

1. Używając konsoli Pythonowej i biblioteki selenium
  - a. przejdź do strony <https://duckduckgo.com/>
  - b. wyszukaj frazę “the biggest python software house”
  - c. przejdź do pierwszego wyniku na liście
  - d. sprawdź czy gdzieś na stronie występuje tekst “STX Next”
  
2. Używając konsoli Pythonowej i biblioteki selenium
  - a. zaloguj się do aplikacji DiabControl
  - b. przejdź do strony ze wszystkimi pacjentami
  - c. wydrukuj do konsoli pacjentów
    - i. same adresy email
    - ii. dane pacjenta w formacie “<Imię> <Nazwisko> (<email>)”

A woman with curly hair, wearing a green ribbed sweater and a long chain necklace, is sitting at a desk in an office. She is looking towards the camera with a slight smile. Her hands are on a keyboard and a mouse. On the desk, there is a white mug with the text "SUPER ZUZA" and a pair of glasses. In the background, two other people are working at their desks. The office has large windows and a teal wall. The text "Splinter" is overlaid on the left side of the image.

# Splinter

Wprowadzenie

# Czym jest Splinter?

- wysokopoziomowa biblioteka dla Pythona
- jest używany jako adapter do Selenium WebAPI
- usprawnia pracę na obiektach
- poprawia czytelność kodu
- kod napisany raz działa na różnych przeglądarkach

```
>> pip install splinter
```

# Przykład

```
from splinter import Browser

with Browser() as browser:
    # Visit URL
    browser.visit("http://www.google.com")
    browser.fill('q', 'splinter - python acceptance testing for web applications')
    # Find and click the 'search' button
    button = browser.find_by_name('btnG')
    # Interact with elements
    button.click()
    if browser.is_text_present('splinter.readthedocs.io'):
        print("Yes, the official website was found!")
    else:
        print("No, it wasn't found... We need to improve our SEO techniques")
```

# Podstawowe metody

# Instancja przeglądarki

```
browser = Browser()
```

```
browser.quit()
```

```
browser.reload()
```

# Nawigacja

```
browser.visit('https://stxnext.com')
```

```
browser.back()
```

```
browser.forward()
```

# Treść

```
browser.title
```

```
browser.url
```

```
browser.html
```

# Formularze

Pola są identyfikowane przez atrybut "name".

```
<form>  
  Name: <input type="text" name="fullName"><br>  
  Email: <input type="text" name="email"><br>  
</form>
```

```
browser.fill('fullName', 'my name')  
browser.attach_file('file', '/path/to/file/somefile.jpg')  
browser.choose('some-radio', 'radio-value')  
browser.check('some-check')  
browser.uncheck('some-check')  
browser.select('uf', 'rj')
```

# Przyciski

```
btn = browser.find_by_name('submit')  
btn.click()
```



# Szukanie

# Szukanie elementów

```
browser.find_by_css('h1')
browser.find_by_xpath('//h1')
browser.find_by_tag('h1')
browser.find_by_name('name')
browser.find_by_text('Hello World!')
browser.find_by_id('firstheader')
browser.find_by_value('query')
```

# Szukanie linków

```
browser.find_link_by_text('Link for Example.com')
browser.find_link_by_partial_text('for Example')
browser.find_link_by_href('http://example.com')
browser.find_link_by_partial_href('example')
```

# Gdy elementów jest wiele

```
browser.find_by_name('name').first
browser.find_by_name('name').last
browser.find_by_name('name')[1]
```

# Szukanie wewnątrz komponentu

```
divs = browser.find_by_tag("div")
divs.first.find_by_name("name")
```

# Dostępność tekstu

```
browser.is_text_present("Done")
```

```
browser.is_text_present('splinter', wait_time=10)
```

```
browser.is_text_not_present('text not present')
```

```
browser.is_text_not_present('text not present', wait_time=10)
```

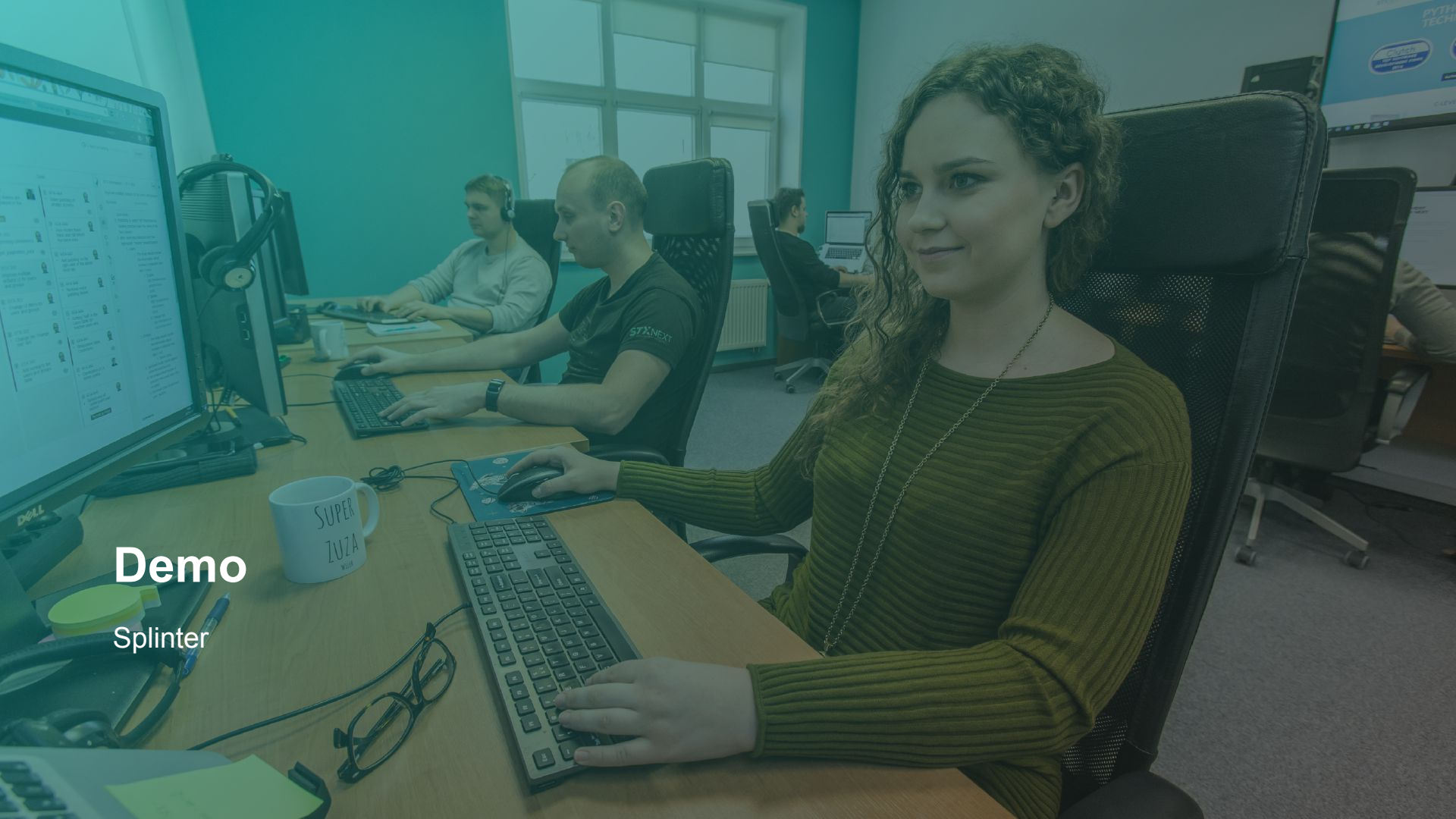
# Dostępność elementów

```
browser.is_element_present_by_css('h1')
browser.is_element_present_by_xpath('//h1')
browser.is_element_present_by_tag('h1')
browser.is_element_present_by_name('name')
browser.is_element_present_by_text('Hello!')
browser.is_element_present_by_id('firstheader')
browser.is_element_present_by_value('query')
browser.is_element_present_by_value(
    'query',
    wait_time=10
)
```

```
browser.is_element_not_present_by_css('h6')
browser.is_element_not_present_by_xpath('//h6')
browser.is_element_not_present_by_tag('h6')
browser.is_element_not_present_by_name('unexisting')
browser.is_element_not_present_by_text('Not here :(')
browser.is_element_not_present_by_id('unexisting-head')
browser.is_element_not_present_by_id(
    'unexisting-head',
    wait_time=10
)
```

# Inne

- obsługa wielu okien przeglądarki
- obsługa ruchów myszy, drag and drop, double click, right click
- wykonywanie kodu javascript
- cookies injection



Demo

Splinter



# Zadania II

Splinter





## Zadania cz II: Splinter

1. Używając konsoli Pythonowej i biblioteki Splinter
  - a. przejdź do strony <https://duckduckgo.com/>
  - b. wyszukaj frazę “the biggest python software house”
  - c. przejdź do pierwszego wyniku na liście
  - d. sprawdź czy gdzieś na stronie występuje tekst “STX Next”
  
2. Używając konsoli Pythonowej i biblioteki Splinter
  - a. zaloguj się do aplikacji DiabControl
  - b. przejdź do strony ze wszystkimi pacjentami
  - c. wydrukuj do konsoli pacjentów
    - i. same adresy email
    - ii. dane pacjenta w formacie “<Imię> <Nazwisko> (<email>)”



unittest

Wprowadzenie

# Czym są unittesty

- testy weryfikujące poprawność działania pojedynczych elementów
- porównanie otrzymanych wyników z wynikami oczekiwanymi
- łatwość w zarządzaniu kodem

# Przykład

```
import unittest
from selenium import webdriver
from selenium.webdriver.common.keys import Keys

class PythonOrgSearch(unittest.TestCase):
    def setUp(self):
        self.browser = webdriver.Firefox()

    def tearDown(self):
        self.browser.quit()

    def test_search_in_python_org(self):
        browser = self.browser
        browser.get("http://www.python.org")
        self.assertIn("Python", browser.title)
        elem = browser.find_element_by_name("q")
        elem.send_keys("pycon")
        elem.send_keys(Keys.RETURN)
        assert "No results found." not in browser.page_source

if __name__ == "__main__":
    unittest.main()
```

# Asercje

```
button = browser.find_element_by_class_name('my-button')  
self.assertEqual(button.text, 'Submit')
```

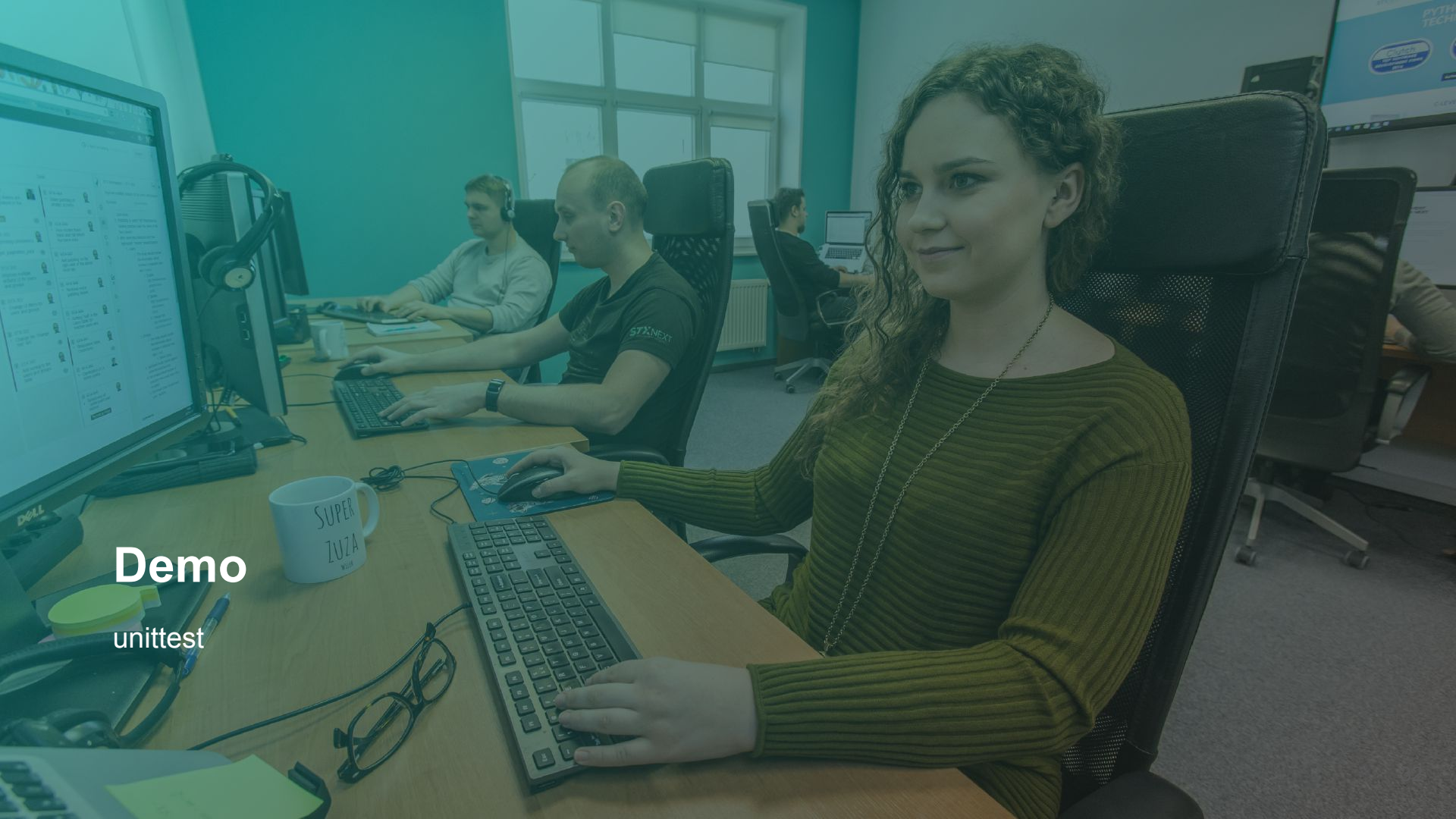
```
headers = browser.find_elements_by_tag_name('th')  
headers_text = [header.text for header in headers]  
self.assertIn('First Name', headers_text)
```

```
expected_headers = ['Email', 'First Name', 'Last name']  
self.assertEqual(expected_headers, headers_text)
```

```
hidden_element = browser.find_element_by_id('hide-me')  
self.assertFalse(hidden_element.is_displayed())
```

```
checkbox = browser.find_element_by_value('checkbox_one')  
self.assertTrue(checkbox.is_selected())
```





Demo

unittest



# Zadania III

unittest



## Zadania cz III: unittest

1. Stwórz plik z TestCasem na podstawie przykładu ze slajdu 30
  - a. stwórz nową instancję przeglądarki przed każdym testem
  - b. zamknij instancję przeglądarki po każdym teście
2. Zaimplementuj test sprawdzający, że po zalogowaniu do aplikacji DiabControl wyświetlany jest tekst 'Welcome to DiabControl system'
  - a. przejdź na stronę aplikacji
  - b. zaloguj się
  - c. sprawdź czy podany tekst się wyświetla
3. Zaimplementuj test sprawdzający, że w zakładce 'My patients' wyświetla się tabela z nagłówkami 'Email', 'First Name', 'Last Name'
  - a. przejdź na stronę aplikacji
  - b. zaloguj się jako doktor
  - c. kliknij w link 'My patients'
  - d. sprawdź czy nagłówki w tabeli są poprawne

A photograph of an office environment with a teal tint. In the foreground, a woman with curly hair, wearing a green ribbed sweater and a long chain necklace, sits at a desk, looking towards the camera with a slight smile. She is using a computer mouse and keyboard. On her desk is a white mug with the text "SUPER ZUZA" and a pair of glasses. In the background, two men are seated at their desks, working on computers. One man is wearing a headset. The office has large windows and a modern, open-plan layout.

# Behave

## Wprowadzenie

# Czym jest Behave

- wysokopoziomowa biblioteka dla Pythona
- tłumacz składni Gherkina na metody w Pythonie
- technika Behaviour Driven Development

```
>> pip install behave
```

# Przykład

simple\_test.feature

**Feature:** showing off behave

**Scenario:** run a simple test

**Given** we have behave installed

**When** we implement a test

**Then** behave will test it for us!

steps/simple\_test.py

```
from behave import *
```

```
@given('we have behave installed')
```

```
def step_impl(context):
```

```
    pass
```

```
@when('we implement a test')
```

```
def step_impl(context):
```

```
    assert True is not False
```

```
@then('behave will test it for us!')
```

```
def step_impl(context):
```

```
    assert context.failed is False
```



# Struktura katalogów

## Przykład 1

```
features/  
features/everything.feature  
steps/  
steps/everything.py  
environment.py
```

## Przykład 2

```
features/  
features/signup.feature  
features/login.feature  
features/account_details.feature  
  
steps/  
steps/website.py  
steps/utils.py  
  
environment.py
```



# environments.py

```
def before_step(context, step):  
    pass
```

```
def before_scenario(context, scenario):  
    pass
```

```
def before_feature(context, feature):  
    pass
```

```
def before_tag(context, tag):  
    pass
```

```
def before_all(context):  
    pass
```

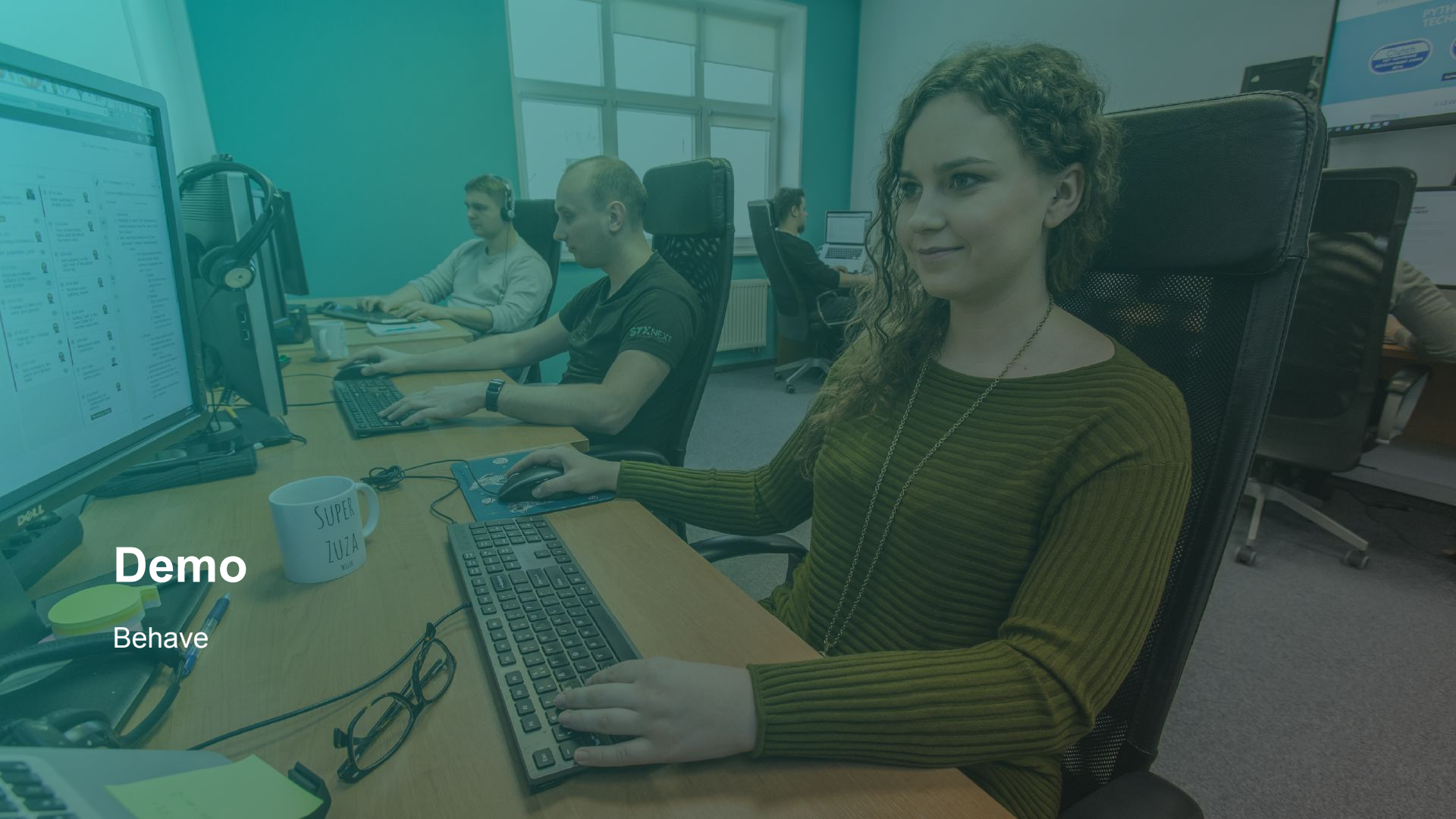
```
def after_step(context, step):  
    pass
```

```
def after_scenario(context, scenario):  
    pass
```

```
def after_feature(context, feature):  
    pass
```

```
def after_tag(context, tag):  
    pass
```

```
def after_all(context):  
    pass
```



Demo

Behave

# Zadania IV

Behave



## Zadania cz IV: Behave

1. Przygotuj strukturę projektu
  - a. utwórz odpowiednie pliki i katalogi zgodnie z przykładem nr 1, slajd 38
  - b. utwórz instancję przeglądarki przed uruchomieniem scenariusza
  - c. zamknij przeglądarkę po zakończeniu
  - d. uruchom test

2. Zaimplementuj poniższy scenariusz

**Feature:** Registration

**Scenario:** Registration passed

**Given** I have access to the system

**And** I am on registration page

**When** I submit the form with valid patient details

**Then** the registration passed

**Scenario:** Registration failed

**Given** I have access to the system

**And** I am on registration page

**When** I submit the form with invalid patient details

**Then** the registration failed

## Zadania cz IV: Behave

3. Zaimplementuj poniższy scenariusz
4. Wymyśl nowy scenariusz i go zaprogramuj

**Feature:** Chat

**Scenario:** Sending the message

**Given** I am logged in as doctor

**And** I have a patient assigned to me

**And** I am on patent card details page

**When** I type the message in chat window

**Then** the message has been sent



## Prelegenci:

Rafał Nowicki  
rafal.nowicki@stxnext.pl

Dawid Kostyszak  
dawid.kostyszak@stxnext.pl

## Rekrutacja:

Oleksandra Chernyak  
oleksandra.chernyak@stxnext.pl

Joanna Trubisz  
joanna.trubisz@stxnext.pl

# QUESTIONS

@STXNext

