
GlobalLogic®

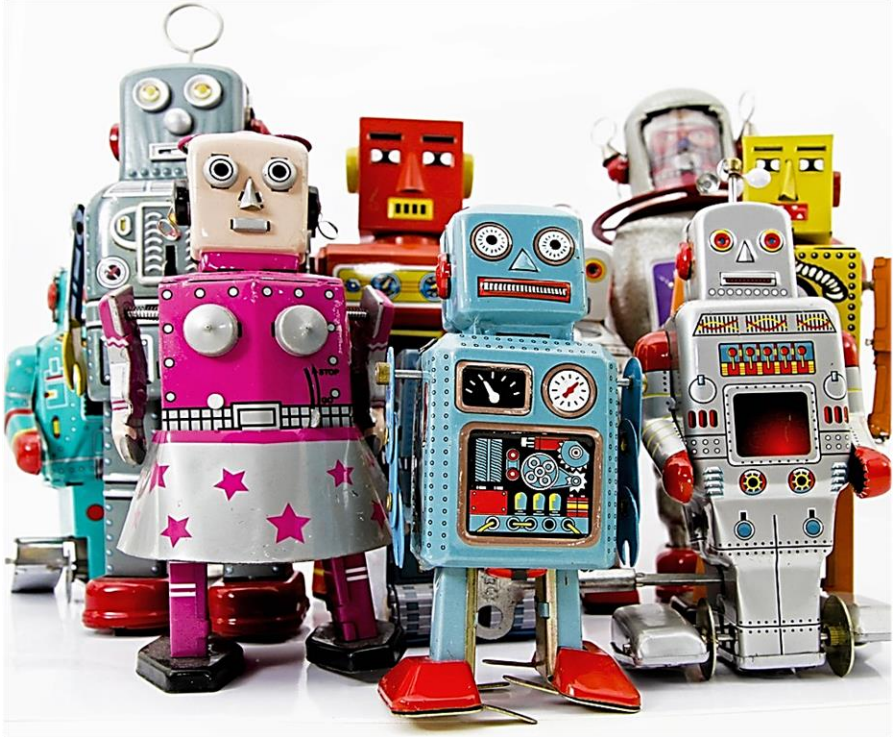
GlobalLogic®

Testowanie Oprogramowania

Marcin Caryk

RobotFramework

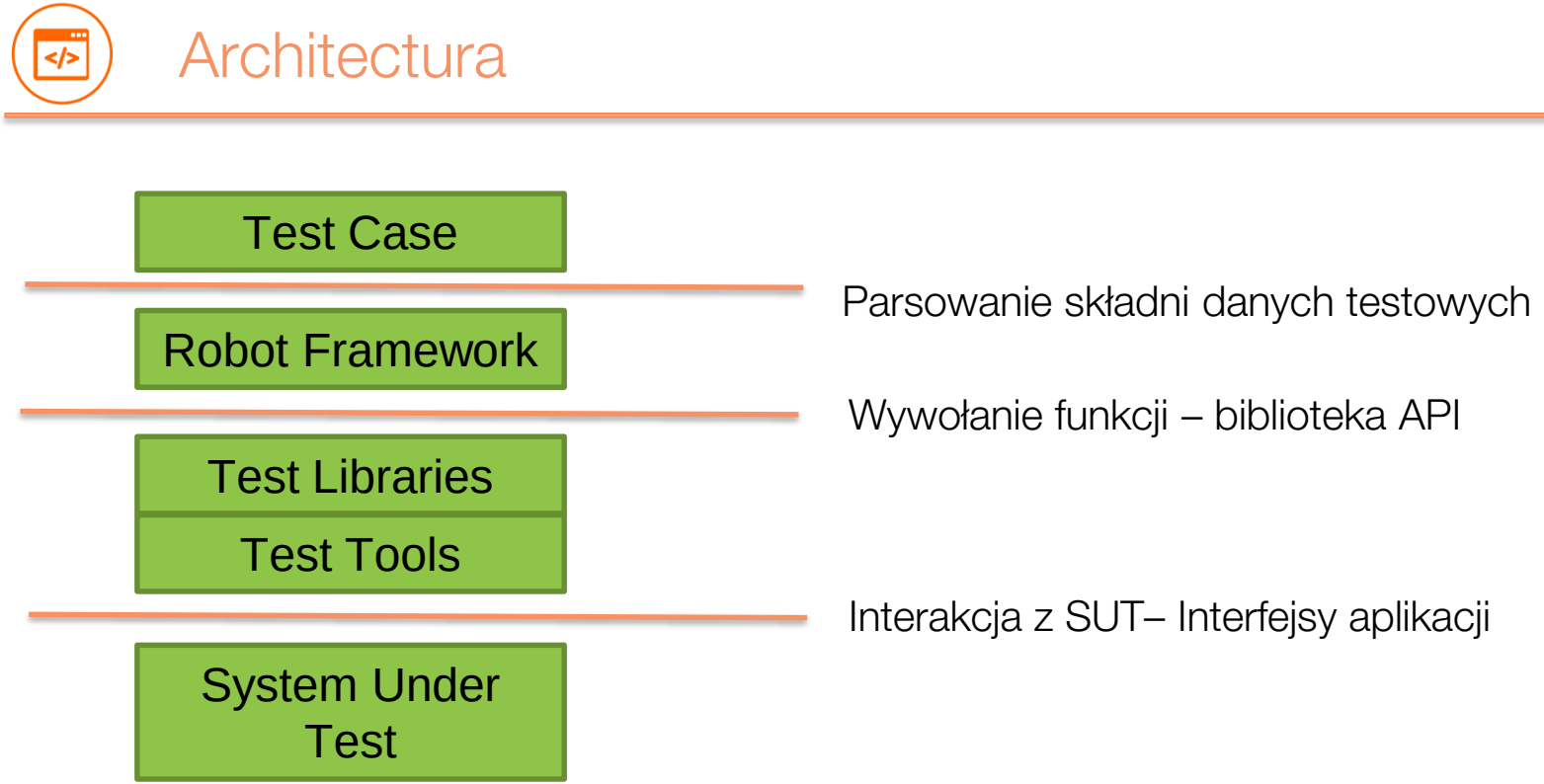
- framework do testów automatycznych,
- uzywanu do testów akceptacyjnych
- ma wiele bibliotek testowych
- robotframework jest open sourcową aplikacją (apache 2.0 licencja)
- zbudowany na pythonie
- pisanie test casów oparte na słowach kluczowych



RobotFramework cechy

- Pisanie testów przy pomocy wysoko poziomowych kluczy
- Wiele dostępnych bibliotek
- Wyniki są w postaci raportów html i logów
- Jest niezależny od platformy
- Posiada API
- Możliwość wywolywania testów poprzez command line
- Ma możliwość tagowania testcasów
- Umożliwia wprowadzanie pozimów test casów i test suitów

Architectura



Instalacja robot framework

- Instalacja pythona 2.7x
- Intalacja robotframework uzycie pip (defaultowo z pythonem)

```
$ sudo pip install robotframework
```

- Instalacja selenium biblioteki

```
$ sudo pip install robotframework-selenium2library
```

- Instlacja SSHLibrary

```
$ sudo pip install robotframework-sshlibrary
```

Instalacja RIDE

- Instalacja pakietów do wxPython

```
$ sudo apt-get install libwxgtk2.8-dev libwxgtk2.8-dbg
$ sudo apt-get install build-essential
$ sudo apt-get install python-wxtools python-wxgtk2.8-dbg
```

- W przypadku problemów (ubuntu 16.04 i wyzej)

```
$ sudo add-apt-repository ppa:nilarimogard/webupd8
$ sudo apt-get update
$ sudo apt-get install python-wxgtk2.8
```

- Instalacja RIDE

```
$ sudo pip install robotframework-ride
```

Instalacja Dodatkowe pakiety

- Instalacja chrome driver

```
$ sudo apt-get install unzip

$ wget -N http://chromedriver.storage.googleapis.com/2.26/chromedriver_linux64.zip
$ unzip chromedriver_linux64.zip
$ chmod +x chromedriver

$ sudo mv -f chromedriver /usr/local/share/chromedriver
$ sudo ln -s /usr/local/share/chromedriver /usr/local/bin/chromedriver
$ sudo ln -s /usr/local/share/chromedriver /usr/bin/chromedriver
```

- Ogólnie:

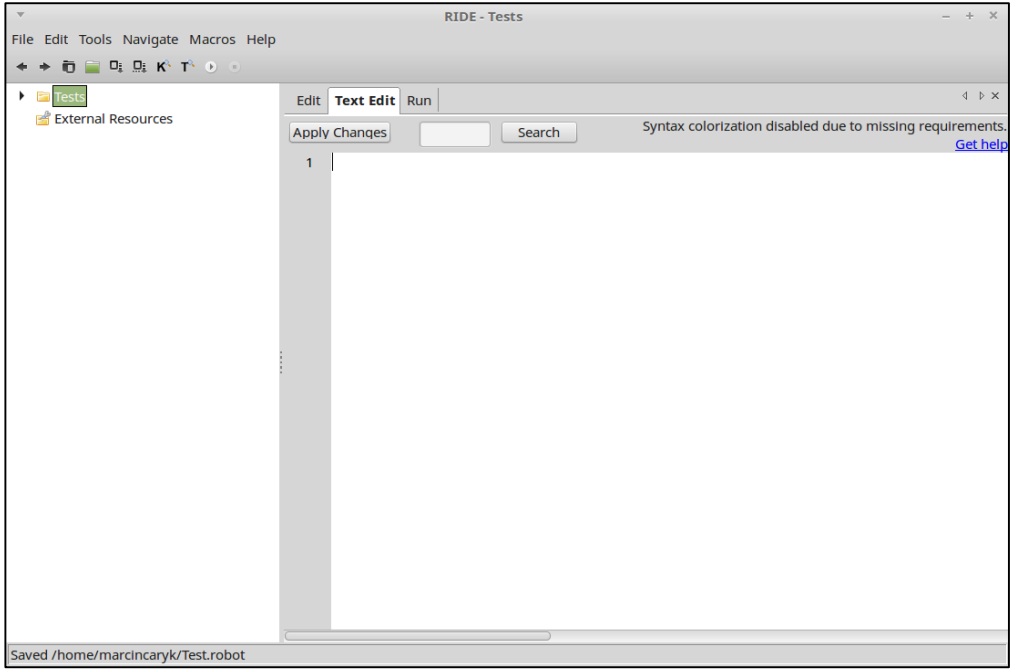
https://sites.google.com/a/chromium.org/chromedriver/downloads

- Dla Linux OS umieścić trzeba w **/usr/bin**,

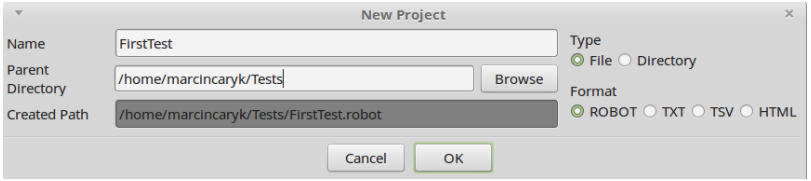
- Dla windows w **python/Scripts**

RobotFramework – first test

- Uruchamianie RIDE (z comandline ride.py)



File -> New Project Project



Text Editor

```
*** Settings ***
Documentation     This is a basic test
Library           Selenium2Library

*** Test Cases ***
Open test browser
    [Documentation]    This a basic information about a test
    [Tags]    Smoke
    Open Browser    http://www.google.com    chrome
    Close Browser

*** Keywords ***
```

RobotFramework struktura

- Robot framework test ma 4 tabele nazwane: ”*Setting*””, *Variables”*, *”Test Cases”* i *“Keywords”*
- Settings – zawiera linki do zewnętrznych biblitek, plików źródłowych, zewnętrznych zmiennych
- Variables – zawiera wewnętrzen delkalarcje zmiwnnych
- Test Case – zawiera opis test case i jego kroków
- Keywords – znajduje się implementacja specyficznych akcji i słów kluczowych.

```
*** Settings ***
*** Variables ***
*** Test Cases ***
*** Keywords ***
```

Uruchamianie skryptów

- Przy pomocy RIDE
- Linie poleceń (Command Line)
  - Pycharm IDE

    View \ Tool Windows \ Terminal

    ```
    robot FirstTest.robot
    ```

  - System command line

    ```
    robot FirstTest.robot
    ```

  - Specyficzna lokacja wyników dodając -d somepath

    ```
    robot –d results FirstTest.robot
    ```

Pierwsze test casy

```
*** Settings ***
Documentation     This is a basic test documentation for second testl
Library           Selenium2Library

*** Variables ***
${APP}            robotframework
${URL}            https://www.google.pl
${BROWSER}        chrome
```

TIP: It is recommend to use four spaces between keywords and arguments.

Pierwsze test casy

```
*** Test Cases ***
[TC-001] Open test browser
    [Documentation]    This a basic information about a test
    [Tags]    Req001, Smoke
    Launch Browser
    Close Browser
```

Pierwsze test casy

*** Test Cases ***

[TC-002]- Search RobotFramework page

[Documentation] Launching the browser and search and launch the “robotframework” Application on Google.pl

[Tags] Req002

Launch Browser

Search Site On Google

Launch Site

Check Site

Pierwsze test casy

```
*** Keywords ***
Launch Browser
    Open Browser    ${URL}  ${BROWSER}
    Maximize Browser Window
Search Site On Google
    Input Text    id=lst-ib    ${APP}
    Press Key    name=q    \\13
Launch Site
    Wait Until Element Is Visible    link=Robot Framework    20 Seconds
    Click Element    link=Robot Framework
Check Site
    Page Should Contain  http://robotframework.org/    20 Seconds
```

Biblioteki standardowe

STANDARD EXTERNAL OTHER

Builtin
Provides a set of often needed generic keywords. Always automatically available without imports.

Dialogs
Provides means for pausing the test execution and getting input from users.

Collections
Provides a set of keywords for handling Python lists and dictionaries.

OperatingSystem
Enables various operating system related tasks to be performed in the system where Robot Framework is running.

Remote
Special library acting as a proxy between Robot Framework and test libraries elsewhere. Actual test libraries can be running on different machines and be implemented using any programming language supporting XML-RPC protocol.

Screenshot
Provides keywords to capture screenshots of the desktop.

String
Library for generating, modifying and verifying strings.

Telnet
Makes it possible to connect to Telnet servers and execute commands on the opened connections.

XML
Library for generating, modifying and verifying XML files.

Process
Library for running processes in the system. New in Robot Framework 2.8.

DateTime
Library for date and time conversions. New in Robot Framework 2.8.5.

Biblioteki dodatkowe

STANDARD EXTERNAL OTHER

Android library
Library for all your Android automation needs. It uses Calabash Android internally.

AnywhereLibrary
Library for testing Single-Page Apps (SPA). Uses Selenium Webdriver and Appium internally.

AppiumLibrary
Library for Android- and iOS-testing. It uses Appium internally.

Archive library
Library for handling zip- and tar-archives.

AutoItLibrary
Windows GUI testing library that uses AutoIt freeware tool as a driver.

Database Library (Java)
Java-based library for database testing. Usable

FTP library
Library for testing and using FTP server with Robot Framework.

HTTP library (livetest)
Library for HTTP level testing using livetest tool internally.

HTTP library (Requests)
Library for HTTP level testing using Request internally.

HttpRequestLibrary (Java)
Library for HTTP level testing using Apache HTTP client. Available also at Maven central.

iOS library
Library for all your iOS automation needs. It uses Calabash iOS Server internally.

ImageHorizonLibrary
Cross-platform, pure Python library for GUI

RemoteSwingLibrary
Library for testing and connecting to a java process and using SwingLibrary, especially Java Web Start applications.

SeleniumLibrary
Web testing library that uses popular Selenium tool internally. Uses deprecated Selenium 1.0 and is also itself deprecated.

Selenium2Library
Web testing library that uses Selenium 2. For most parts drop-in-replacement for old SeleniumLibrary.

Selenium2Library for Java
Java port of the Selenium2Library.

ExtendedSelenium2Library
Web testing library that uses Selenium2Library internally, providing AngularJS support on top of it.

Biblioteki dodatkowe

Database Library (Java)
Java-based library for database testing. Usable with Jython. Available also at Maven central.

Database Library (Python)
Python based library for database testing. Works with any Python interpreter, including Jython.

Diff Library
Library to diff two files together.

Django Library
Library for Django, a Python web framework.

Eclipse Library
Library for testing Eclipse RCP applications using SWT widgets.

robotframework-faker
Library for Faker, a fake test data generator.

ImageHorizonLibrary
Cross-platform, pure Python library for GUI automation based on image recognition.

MongoDB library
Library for interacting with MongoDB using pymongo.

MQTT library
Library for testing MQTT brokers and applications.

NcclientLibrary
NETCONF protocol library based on ncclient

Rammbock
Generic network protocol test library that offers easy way to specify network packets and inspect the results of sent and received packets.

Web testing library that uses Selenium2Library internally, providing AngularJS support on top of it.

SSHLibrary
Enables executing commands on remote machines over an SSH connection. Also supports transfering files using SFTP.

SudsLibrary
A library for functional testing of SOAP-based web services based on Suds, a dynamic SOAP 1.1 client.

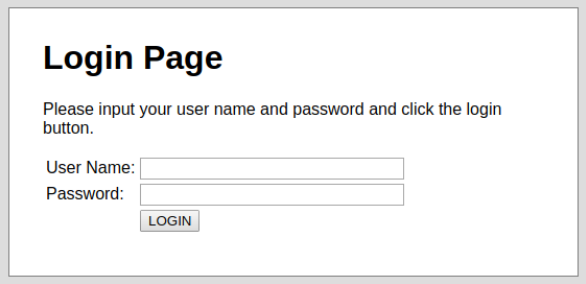
SwingLibrary
Library for testing Java applications with Swing GUI.

TFTPLibrary
Library for interacting over Trivial File Transfer Portocol.

watir-robot
Web testing library that uses Watir tool.

Przykład 1 - Web Demo

- Running the server: `python server.py` -> localhost:7272



- The corect loging is user: demo password: mode
- html server tree

```
├── demo.css
├── error.html
├── index.html
└── welcome.html
```

Przykład 1 - Web Demo – server.py

```
from os import chdir
from os.path import abspath, dirname, join
from SocketServer import TCPServer
from SimpleHTTPServer import SimpleHTTPRequestHandler

ROOT = join(dirname(abspath(__file__)), 'html')
PORT = 7272

class DemoServer(TCPServer):
    allow_reuse_address = True

    def __init__(self, port=PORT):
        TCPServer.__init__(self, ('localhost', int(port )), SimpleHTTPRequestHandler)

    def serve(self, directory=ROOT):
        chdir(directory)
        print 'Demo server starting on port %d.' % self.server_address[1]
        try:
            server.serve_forever()
        except KeyboardInterrupt:
            server.server_close()
        print 'Demo server stopped.'
```

```
if __name__ == '__main__':
    import sys
    try:
        server = DemoServer(*sys.argv[1:])
    except (TypeError, ValueError):
        print __doc__
    else:
        server.serve()
```

Przykład 1 - Web Demo – resource.robot (1/2)

```
*** Settings ***
Documentation     A resource file with reusable keywords and variables.
...
...               The system specific keywords created here form our own
...               domain specific language. They utilize keywords provided
...               by the imported Selenium2Library.
Library           Selenium2Library
```

```
*** Variables ***
${SERVER}         localhost:7272
${BROWSER}        chrome
${DELAY}          0
${VALID USER}     demo
${VALID PASSWORD}    mode
${LOGIN URL}      http://${SERVER}/
${WELCOME URL}    http://${SERVER}/welcome.html
${ERROR URL}      http://${SERVER}/error.html
```

Przykład 1 - Web Demo – resource.robot (2/2)

```
*** Keywords ***
Open Browser To Login Page
    Open Browser    ${LOGIN URL}    ${BROWSER}
    Maximize Browser Window
    Set Selenium Speed    ${DELAY}
    Login Page Should Be Open

Login Page Should Be Open
    Title Should Be    Login Page

Go To Login Page
    Go To    ${LOGIN URL}
    Login Page Should Be Open

Input Username
    [Arguments]    ${username}
    Input Text    username_field    ${username}
```

```
Input Password
    [Arguments]    ${password}
    Input Text    password_field    ${password}

Submit Credentials
    Click Button    login_button

Welcome Page Should Be Open
    Location Should Be    ${WELCOME URL}
    Title Should Be    Welcome Pagel
```

Przykład 1 - Web Demo – valid login

```
*** Settings ***
Documentation     A test suite with a single test for valid login.
...
...               This test has a workflow that is created using keywords in
...               the imported resource file.
Resource          resource.robot

*** Test Cases ***
Valid Login
    Open Browser To Login Page
    Input Username    demo
    Input Password    mode
    Submit Credentials
    Welcome Page Should Be Open
    [Teardown]    Close Browser
```

Przykład 1 - Web Demo – invalid login (1/2)

```
*** Settings ***
Documentation     A test suite containing tests related to invalid login.
...
...               These tests are data-driven by their nature. They use a single
...               keyword, specified with Test Template setting, that is called
...               with different arguments to cover different scenarios.
...
...               This suite also demonstrates using setups and teardowns in
...               different levels.
Suite Setup       Open Browser To Login Page
Suite Teardown    Close Browser
Test Setup        Go To Login Page
Test Template     Login With Invalid Credentials Should Fail
Resource          resource.robot
```

```
*** Test Cases ***              USER NAME        PASSWORD
Invalid Username                invalid          ${VALID PASSWORD}
Invalid Password                ${VALID USER}    invalid
Invalid Username And Password   invalid          whatever
Empty Username                  ${EMPTY}         ${VALID PASSWORD}
Empty Password                  ${VALID USER}    ${EMPTY}
Empty Username And Password     ${EMPTY}         ${EMPTY}
```

Przykład 1 - Web Demo – invalid login (2/2)

```
*** Keywords ***
Login With Invalid Credentials Should Fail
    [Arguments]    ${username}    ${password}
    Input Username    ${username}
    Input Password    ${password}
    Submit Credentials
    Login Should Have Failed

Login Should Have Failed
    Location Should Be    ${ERROR URL}
    Title Should Be    Error Page
```

Przykład 2 - C Demo – loging.c i makefile

- loging.c

```
#include <string.h>
#define NR_USERS 2

struct User {
    const char* name;
    const char* password;
};
const struct User VALID_USERS[NR_USERS] = {"john", "long", "demo", "mode"};


int validate_user(const char* name, const char* password) {
    int i;
    for (i = 0; i < NR_USERS; i++) {
        if (0 == strncmp(VALID_USERS[i].name, name, strlen(VALID_USERS[i].name)))
            if (0 == strncmp(VALID_USERS[i].password, password, strlen(VALID_USERS[i].password)))
                return 1;
    }
    return 0;
}
```

Makefile

```
CC=gcc
SRC=login.c
SO=liblogin.so

$(SO): $(SRC)
   $(CC) -fPIC -shared -o $(SO) $(SRC)

clean:
   rm -f $(SO)
```

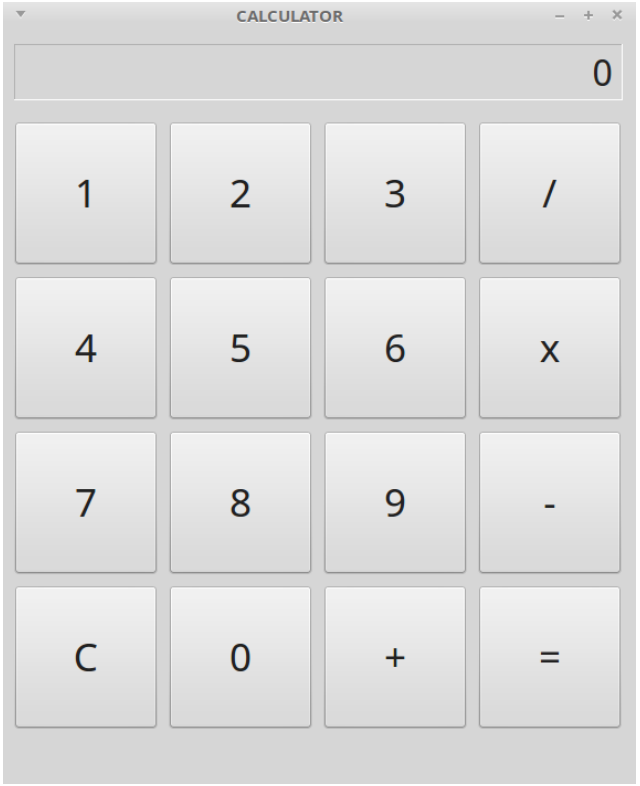
Przykład 2 - C Demo – LoginLibrary.py

```
from ctypes import CDLL, c_char_p

LIBRARY = CDLL('./liblogin.so')  # On Windows we'd use '.dll'

def check_user(username, password):
    """Validates user name and password using imported shared C library."""
    username = c_char_p(username.encode('UTF-8'))
    password = c_char_p(password.encode('UTF-8'))
    if not LIBRARY.validate_user(username, password):
        raise AssertionError('Wrong username/password combination')

if __name__ == '__main__':
    import sys
    try:
        check_user(*sys.argv[1:])
    except TypeError:
        print('Usage:  %s username password' % sys.argv[0])
    except AssertionError as err:
        print(err)
    else:
        print('Valid password')
```

Przykład 2 - C Demo – loging_tests.robot

```
*** Settings ***
Library           LoginLibrary.py

*** Test Case ***
Validate Users
    [Template]    Login with valid user should succeed
    johns    long
    demo     mode

Login With Invalid User Should Fail
    [Template]    Login with invalid user should fail
    de          mo
    invalid     invalid
    long        invalid
    ${EMPTY}    ${EMPTY}

*** Keyword ***
Login with valid user should succeed
    [Arguments]    ${username}    ${password}
    Check User    ${username}    ${password}

Login with invalid user should fail
    [Arguments]    ${username}    ${password}
    Run Keyword And Expect Error    Wrong username/password combination
    ...    Check User    ${username}    ${password}
```

Przykład 3 – Calculator – Front end and backend



- Calc_MainWin.py
- Calc.py
- calculator.py

Runing:
python Calc.py

```
class Calculator(object):
    BUTTONS = '1234567890+-*/C='

    def __init__(self):
        self._expression = ''

    def push(self, button):
        if button not in self.BUTTONS:
            raise CalculationError("Invalid button '%s'." % button)
        if button == '=':
            self._expression = self._calculate(self._expression)
        elif button == 'C':
            self._expression = ''
        elif button == '/':
            self._expression += '//'    # Integer division also in Python 3
        else:
            self._expression += button
        return self._expression

    def _calculate(self, expression):
        try:
            return str(eval(expression))
        except SyntaxError:
            raise CalculationError('Invalid expression.')
        except ZeroDivisionError:
            raise CalculationError('Division by zero.')


class CalculationError(Exception):
    pass
```

GlobalLogic®