Testowanie aplikacji

Acceptance Testing, Robot Framework

Adam Chyła
chyla.org

Acceptance Testing

Poziom testów czyli grupa czynności testowych, które są razem zorganizowane.

Istotą rozróżniania poziomów testów jest to, że każdy poziom ma inne cele testowania, ma zwykle inną podstawę testów, a także inny obiekt testowania.

Typowe poziomy testowania:

- **jednostkowe** - *testowanie pojedynczych modułów,*
- **integracyjne** - *testowanie wykonywane w celu wykrycia defektów podczas interakcji między komponentami lub systemami,*
- **systemowe** - *testowanie zintegrowanego systemu w celu sprawdzenia jego zgodności z wyspecyfikowanymi wymaganiami,*
- **akceptacyjne** - *testowanie formalnie przeprowadzane w celu umożliwienia użytkownikowi, klientowi lub innemu ustalonemu podmiotowi ustalenia, czy zaakceptować system lub moduł.*

źródło/na podstawie: Testowanie i jakość oprogramowania. Metody, narzędzia, techniki, Adam Roman, 2015 s. 72-80

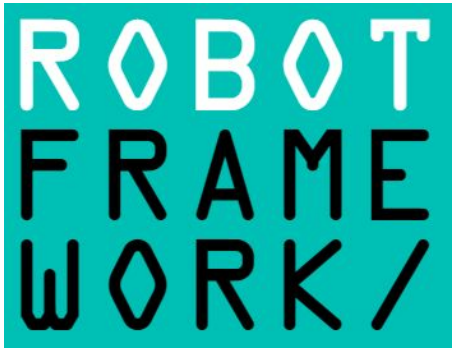
Acceptance Testing

Acceptance testing według ISTQB:

Formal testing with respect to user needs, requirements, and business processes conducted to determine whether or not a system satisfies the acceptance criteria and to enable the user, customers or other authorized entity to determine whether or not to accept the system.

źródło: ISTQB Glossary: http://glossary.istqb.org/search/acceptance%20testing

Robot Framework



Robot Framework is a generic open source automation framework for acceptance testing, acceptance test driven development (ATDD), and robotic process automation (RPA).
It has easy-to-use tabular test data syntax and it utilizes the keyword-driven testing approach.
Its testing capabilities can be extended by test libraries implemented either with Python or Java, and users can create new higher-level keywords from existing ones using the same syntax that is used for creating test cases.

źródło: https://robotframework.org/#introduction

Robot Framework - Instalacja

Do napisania testu z użyciem Robot Framework wymagana jest instalacja pakietu `robotframework`. Dostępny jest także graficzny edytor RIDE ułatwiający pisanie testów, wymaga to instalacji pakietu `robotframework-ride`.

Polecenia instalacji:

- `pip install robotframework`
- `pip install robotframework-ride`

Robot Framework - Format pliku z testami

Robot framework obsługuje następujące formaty plików:

- czysty tekst (pliki .robot),
- TSV (tab-separated values) - pliki typu TSV można tworzyć i edytować w arkuszu kalkulacyjnym (np. MS Excel), wsparcie dla tego formatu zostanie w przyszłości wycofane,
- reStructuredText format,
- HTML (do wersji 3.1 Robot Framework).

W dalszej części uwaga zostanie poświęcona testom w formacie czystego tekstu.

Robot Framework - Struktura pliku z testami

Plik z testami składa się z czterech sekcji, z których tylko `Test Cases` jest obowiązkowa.

```
*** Settings ***

*** Variables ***

*** Test Cases ***

*** Keywords ***
```

Robot Framework - Struktura pliku z testami

W sekcji **Settings** umieszczana jest dokumentacja dotycząca zestawu testów, informacje o dodatkowych plikach zasobów i wykorzystywanych bibliotekach.

W sekcji **Variables** umieszczane są zmienne.

```
*** Settings ***


*** Variables ***


*** Test Cases ***


*** Keywords ***
```

Robot Framework - Struktura pliku z testami

W sekcji **`Test Cases`** umieszczane są_poszczególne przypadki testowe.

W sekcji **`Keywords`** umieszczane są definicje funkcji (w języku Robot Framework funkcję nazywamy keyword'em).

```
*** Settings ***

*** Variables ***

*** Test Cases ***

*** Keywords ***
```

Robot Framework - Pierwszy test

Przykładowy plik z testami rozpoczyna się od sekcji **Settings** z dokumentacją opisującą cel wszystkich testów zawartych w danym pliku.

Test o nazwie *My First Test Case* składa się z jednego wywołania keyword'u (funkcji) *Log*. Funkcja ta zapisuje przekazany napis do pliku z logami.

Plik first.robot:

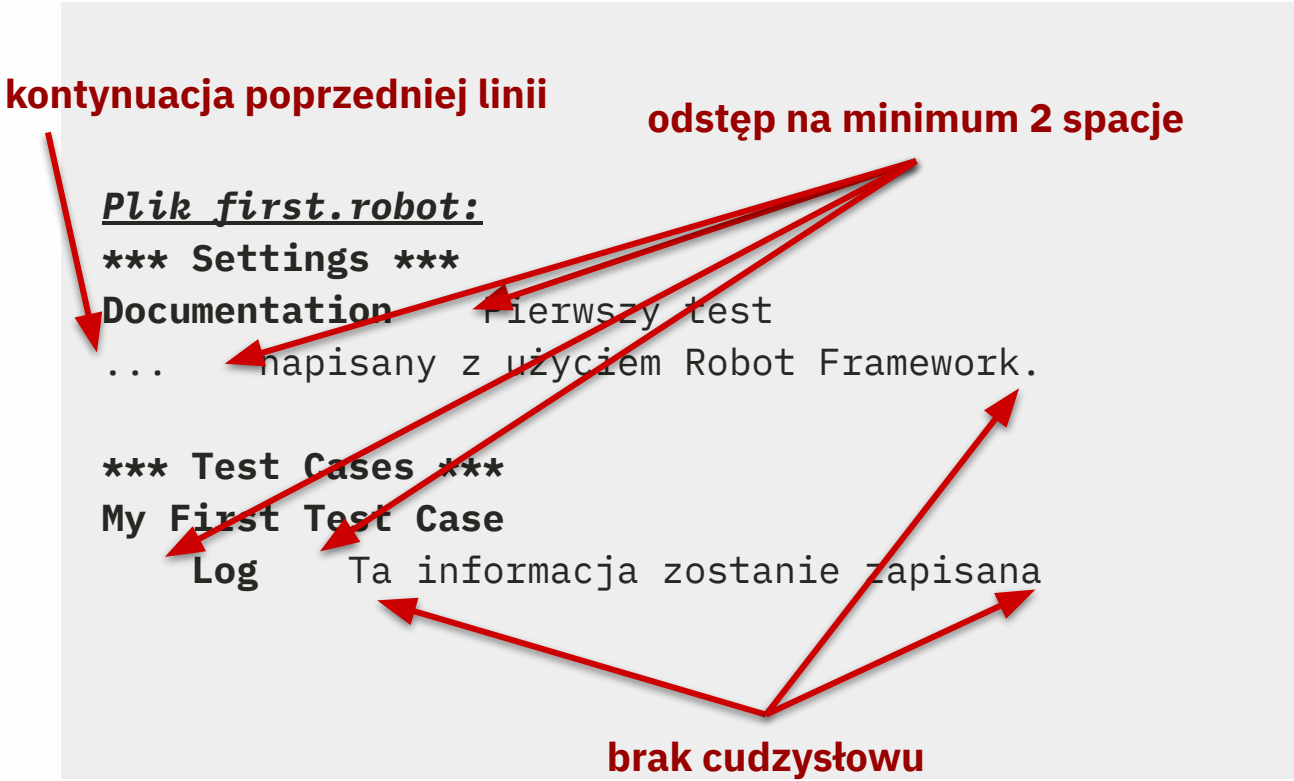
```
*** Settings ***
Documentation    My first test
...    with Robot Framework.

*** Test Cases ***
My First Test Case
    Log    This text will be logged
```

Robot Framework - Pierwszy test

Najważniejsze informacje:

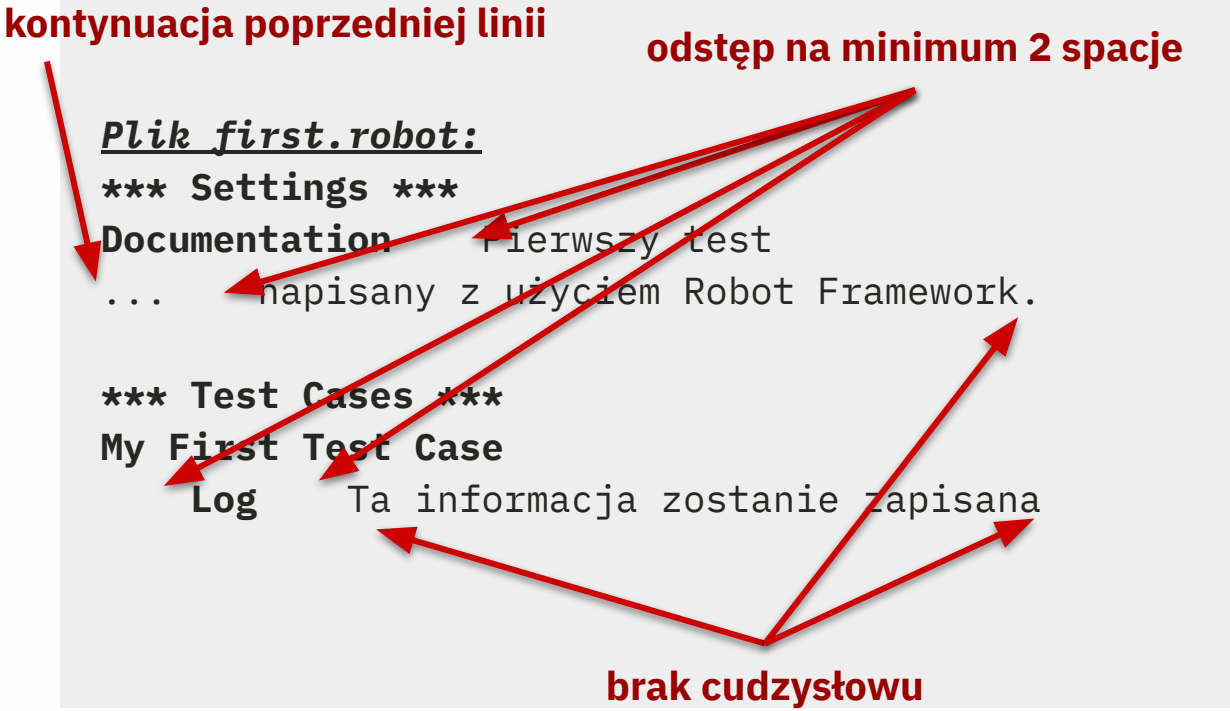
- odstęp pomiędzy poszczególnymi elementami to minimum 2 spacje (**zaleca się 4 spacje**),
- kontynuacja poprzedniej linii rozpoczyna się od trzech kropek (napis wieloliniowy),



Robot Framework - Pierwszy test

Najważniejsze informacje:

- brak cudzysłowu podczas tworzenia napisu,
- nazwa testu *My First Test Case* zawiera spacje, jest to pełna nazwa.



Robot Framework - Logowanie informacji

Komunikaty zapisywane do logów mogą mieć różne poziomy ważności, dostępne poziomy:

- **FAIL** - Używane, gdy wykonanie keyword'a się nie powiedzie. Ten poziom jest zarezerwowany dla komunikatów pochodzących od Robot Framework.
- **WARN** - Używany do ostrzeżeń, komunikaty na tym poziomie są także wyświetlane w konsoli podczas wykonywania testu, a także umieszczane w sekcji *Test Execution Errors* pliku z logami.
- **INFO** - Domyślny poziom, poniżej tego poziomu (*DEBUG*, *TRACE*) komunikaty nie są zapisywane w pliku z logami.
- **DEBUG** - Używany w celu debugowania, używany do zapisywania informacji przydatnych programistom i testerom.
- **TRACE** - Bardziej szczegółowy poziom niż *DEBUG*. Domyślnie z tym poziomem zapisywane są informacje o argumentach keyword'a i wartości zwracanej.

Robot Framework - Logowanie informacji

Test *Multi Log Level Test Case* składa się z pięciu wywołań keyword'u **Log** z przeważnie dwoma argumentami oddzielonymi 4 spacjami.

Plik logging.robot:

```
*** Settings ***
Documentation
...     Demonstrate log levels.

*** Test Cases ***
Multi Log Level Test Case
    [Documentation]    Log message on each log level.
    Log    Warning message    WARN
    Log    Info message
    Log    Second info message    INFO
    Log    Debug message    DEBUG
    Log    Trace message    TRACE
```

argumenty

odstęp na minimum 2 spacje

Robot Framework - Uruchomienie testów

W celu wykonania testów należy wykonać polecenie robot i jako argument podać ścieżkę do pliku lub katalogu z testami. Dodatkowymi opcjami do polecenia są:

- `-d KATALOG` - ścieżka do katalogu, w którym mają zostać zapisane logi wraz z dodatkowymi informacjami z wykonania testu,
- `-L POZIOM` - określa poziom, od którego będą zapisywane komunikaty do logów, domyślnie *INFO*.

Wzór polecenia:

```
robot -d KATALOG -L POZIOM ŚCIEŻKA_DO_PLIKU_Z_TESTEM
```

Robot Framework - Uruchomienie testów

Przykład uruchomienia testów z pliku `logging.robot`.

- Logi wraz z dodatkowymi informacjami zostaną zapisane w katalogu output.
- Do logów zostaną zapisane komunikaty na poziomie *DEBUG* i wyżej (*INFO*, *WARNING*, *FAIL*).

```
> robot -d output -L DEBUG logging.robot
==============================================================
Logging :: Demonstrate log levels.
==============================================================
[ WARN ] Warning message
Multi Log Level Test Case :: Log message on each log level.  | PASS |
--------------------------------------------------------------
Logging :: Demonstrate log levels.                           | PASS |
1 test, 1 passed, 0 failed
==============================================================
Output:  /home/[...]/output/output.xml
Log:     /home/[...]/output/log.html
Report:  /home/[...]/output/report.html
```

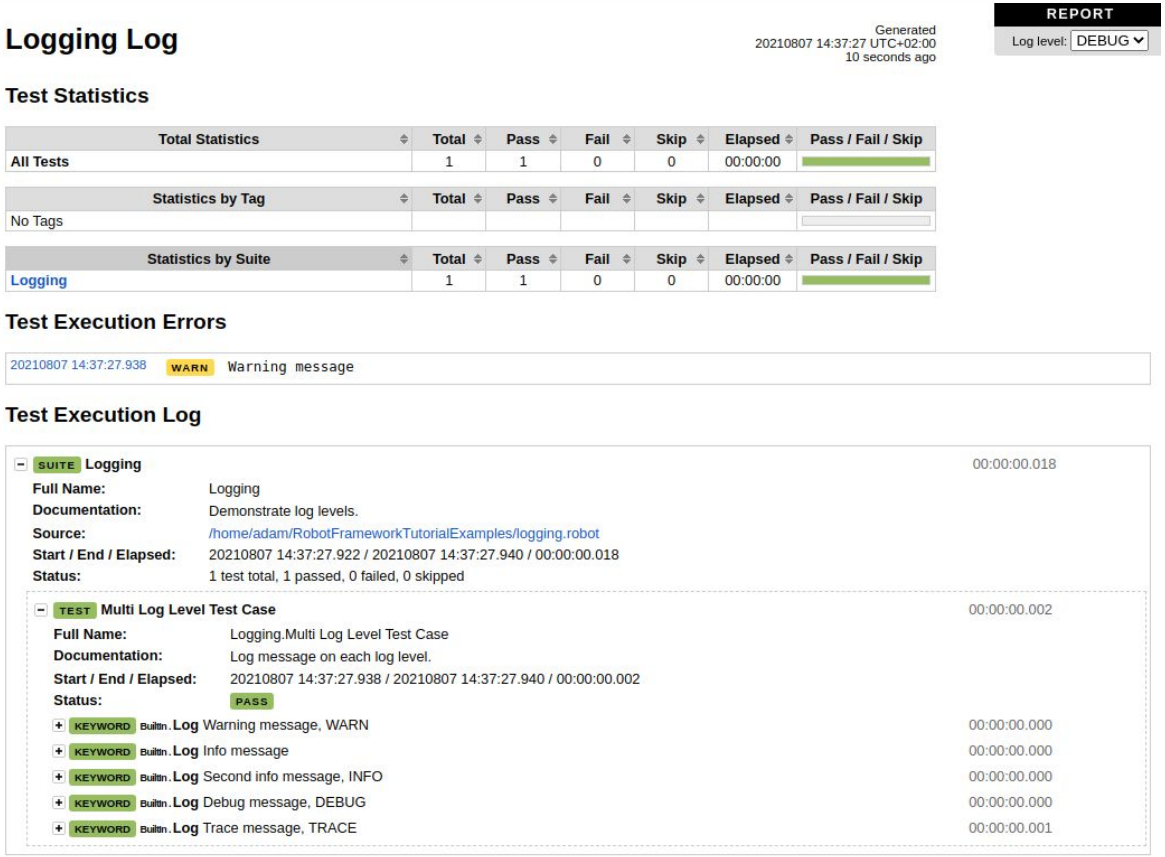
Robot Framework - Wyniki wykonania testów

Na pliki stanowiące wynik uruchomienia testów składają się:

- **`output.xml`** - plik z surowymi danymi, na jego podstawie tworzone są pliki `log.html` i `report.html`,
- **log.html** - plik zawierający informacje na temat wykonania testu (wykonane testy, zapisane komunikaty), umożliwia filtrowanie logów według poziomu ważności,
- **report.html** - zawiera statystyki na temat uruchomionych testów.

Robot Framework - Wyniki wykonania testów

Przykład pliku **log.html**:



Logging Log

REPORT

Log level: DEBUG

Generated
20210807 14:37:27 UTC+02:00
10 seconds ago

Test Statistics

Total Statistics	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
All Tests	1	1	0	0	00:00:00	

Statistics by Tag	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
No Tags						

Statistics by Suite	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
Logging	1	1	0	0	00:00:00	

Test Execution Errors

20210807 14:37:27.938 WARN Warning message

Test Execution Log

SUITE Logging 00:00:00.018

Full Name: Logging
Documentation: Demonstrate log levels.
Source: /home/adam/RobotFrameworkTutorialExamples/logging.robot
Start / End / Elapsed: 20210807 14:37:27.922 / 20210807 14:37:27.940 / 00:00:00.018
Status: 1 test total, 1 passed, 0 failed, 0 skipped

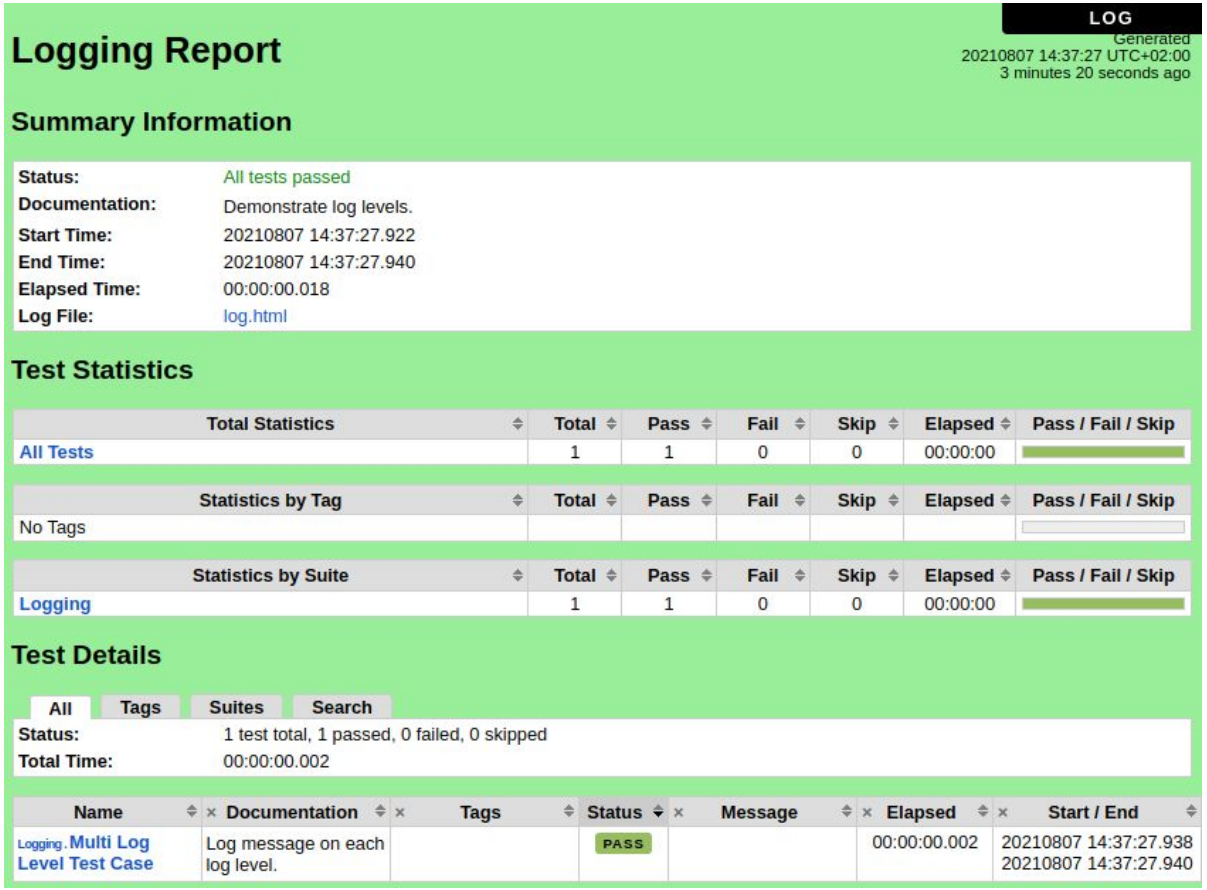
TEST Multi Log Level Test Case 00:00:00.002

Full Name: Logging.Multi Log Level Test Case
Documentation: Log message on each log level.
Start / End / Elapsed: 20210807 14:37:27.938 / 20210807 14:37:27.940 / 00:00:00.002
Status: PASS

KEYWORD BuiltIn.Log Warning message, WARN 00:00:00.000
KEYWORD BuiltIn.Log Info message 00:00:00.000
KEYWORD BuiltIn.Log Second info message, INFO 00:00:00.000
KEYWORD BuiltIn.Log Debug message, DEBUG 00:00:00.000
KEYWORD BuiltIn.Log Trace message, TRACE 00:00:00.001

Robot Framework - Wyniki wykonania testów

Przykład pliku **report.html**:



LOG

Logging Report

Generated
20210807 14:37:27 UTC+02:00
3 minutes 20 seconds ago

Summary Information

Status:	All tests passed
Documentation:	Demonstrate log levels.
Start Time:	20210807 14:37:27.922
End Time:	20210807 14:37:27.940
Elapsed Time:	00:00:00.018
Log File:	log.html

Test Statistics

Total Statistics	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
All Tests	1	1	0	0	00:00:00	

Statistics by Tag	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
No Tags						

Statistics by Suite	Total	Pass	Fail	Skip	Elapsed	Pass / Fail / Skip
Logging	1	1	0	0	00:00:00	

Test Details

All | Tags | Suites | Search

Status:	1 test total, 1 passed, 0 failed, 0 skipped
Total Time:	00:00:00.002

Name	Documentation	Tags	Status	Message	Elapsed	Start / End
Logging. Multi Log Level Test Case	Log message on each log level.		PASS		00:00:00.002	20210807 14:37:27.938 20210807 14:37:27.940

Robot Framework - Zmienne

W zależności od rodzaju zmiennej musi ona zostać odpowiednio oznaczona.

Zmienne skalarne oznaczane są za pomocą ${}, listy za pomocą @{} i słowniki za pomocą &{}.

Znak = jest opcjonalny, każdy element kolekcji musi być oddzielony co najmniej 2 spacjami.

Plik variables-section.robot:

```
*** Settings ***
Documentation    Variables section example.

*** Variables ***
${NAME} =   Jan
@{EXAMPLE_LIST}    1    2    3    a    b    c    4
&{EXAMPLE_DICT}    name=Jan    lastname=Kowalski
```

Robot Framework - Zmienne

Odwołania
do poszczególnych
elementów kolekcji
realizowane są za pomocą
${} oraz operatora
indeksowania.

Plik variables-section.robot:

```
*** Test Case ***
Variable List Dict
    Log    ${NAME}
    Log    ${EXAMPLE_LIST}[1]
    Log    ${EXAMPLE_DICT}[name]
```

Robot Framework - Zmienne

W wyniku działania przedstawionego testu zostaną zalogowane komunikaty złożone z poszczególnych elementów kolekcji.

```
INFO: Jan
INFO: 2
INFO: Jan
```

Robot Framework - Zmienne

Zmienne mogą również zostać ustawione podczas uruchamiania testów, służy do tego przełącznik variable. Taka zmienna może być później używana w testach.

Wzór polecenia:

```
robot --variable NAZWA_ZMIENNEJ:WARTOŚĆ ŚCIEŻKA_DO_PLIKU_Z_TESTEM
```

Robot Framework - Zmienne specjalne

Robot Framework posiada zmienne o specjalnym przeznaczeniu:

- **${CURDIR}** - bezwzględna ścieżka do katalogu, w którym zlokalizowany jest test.
- **${TEMPDIR}** - bezwzględna ścieżka do systemowego katalogu tymczasowego.
- **${EXECDIR}** - bezwzględna ścieżka do katalogu, z którego rozpoczęto wykonywanie testów.
- **${/}** - separator ścieżki. W Linuksie wartością jest /, w Windowsie \.
- **${:}** - separator ścieżek. W Linuksie :, w Windowsie ;.
- **${\n}** - znak nowej linii. W Linuksie \n, w Windowsie \r\n.

Robot Framework - Operacje na kolekcjach

Wybrane operacje na kolekcjach:

- `Get Length` - zwraca liczbę elementów w kolekcji
- `Append To List` - dodanie elementów do listy
- `Remove From List` - usunięcie elementu o podanym indeksie z listy
- `Remove From Dictionary` - usunięcie elementu o podanym kluczu ze słownika
- `Sort List` - sortuje listę w miejscu

Większość omówionych operacji jest dostępna globalnie za pomocą biblioteki Builtin:
http://robotframework.org/robotframework/3.1/libraries/BuiltIn.html
Część z wymienionych keyword'ów znajduje się w bibliotece Collecetions:
http://robotframework.org/robotframework/3.1/libraries/Collections.html

Robot Framework - Operacje na kolekcjach

Keyword'y, które mogą przerwać wykonywanie testu:

- `Length Should Be` - założenie co do liczby elementów
- `Should Be Empty` - założenie co do pustej kolekcji
- `Should Not Be Empty` - założenie co do niepustej kolekcji
- `Should Contain` - założenie co do elementu znajdującego się w kolekcji
- `Should Not Contain` - założenie co do braku elementu w kolekcji

Większość omówionych operacji jest dostępna globalnie za pomocą biblioteki Builtin:
http://robotframework.org/robotframework/3.1/libraries/BuiltIn.html
Część z wymienionych keyword'ów znajduje się w bibliotece Collecetions:
http://robotframework.org/robotframework/3.1/libraries/Collections.html

Robot Framework - Tworzenie zmiennych w teście

Wybrane operacje:

- `Set Variable` - tworzy zmienną w ciele testu lub keyword'u
- `Create List` - tworzy listę w ciele testu lub keyword'u
- `Create Dictionary` - tworzy słownik w ciele testu lub keyword'u

Większość omówionych operacji jest dostępna globalnie za pomocą biblioteki Builtin:
http://robotframework.org/robotframework/3.1/libraries/BuiltIn.html
Część z wymienionych keyword'ów znajduje się w bibliotece Collecetions:
http://robotframework.org/robotframework/3.1/libraries/Collections.html

Robot Framework - Zmienne

Zmienne można tworzyć i modyfikować w teście lub keywordzie za pomocą odpowiednich keyword'ów.

Plik variables-inside-test-case.robot:

```
*** Test Case ***
Variable In Keyword
    ${new_name} =    Set Variable    Janusz
    ${new_list} =    Create List    Janusz
    ...    Alicja    Natalia
    ${new_dict} =    Create Dictionary
    ...    name=Anna    lastname=Nowak

    Length Should Be    ${new_list}    3

    Log    Variable new_name: ${new_name}
    Log    Variable new_list: ${new_list}
    Log    variable new_dict: ${new_dict}
```

Robot Framework - Biblioteki

Funkcje z bibliotek dostępne są po umieszczeniu w sekcji **Settings** odpowiedniego odniesienia do biblioteki.

Plik libraries.robot:

```
*** Settings ***
Documentation    Variables example.
Library    Collections

*** Test Case ***
Variable With Library
    ${names} =   Create List    Janusz
    ...    Alicja    Natalia
    Sort List    ${names}
    Log    Imiona: ${names}
```

Robot Framework - Biblioteki

Do Robot Frameworka dołączone są biblioteki dodające do języka nowe możliwości. Listę dostępnych bibliotek można znaleźć na stronie: http://robotframework.org/robotframework/

Plik libraries.robot:

```
*** Settings ***
Documentation    Variables example.
Library    Collections

*** Test Case ***
Variable With Library
    ${names} =   Create List    Janusz
    ...    Alicja    Natalia
    Sort List    ${names}
    Log    Imiona: ${names}
```

Robot Framework - Najczęściej używane keyword'y

Wybrane Keyword'y:

- `Should Be True` - sprawdza, czy podany warunek jest prawdą,
- `Should Be Equal` - sprawdza, czy podane dwie wartości są równe,
- `Should Be Equal As Strings` - porównywanie zmiennych jako napisy,
- `Should Be Equal As Integers` - porównywanie zmiennych jako liczby całkowite,
- `Catenate` - łączy napisy,
- `Evaluate` - uruchamia podane wyrażenie w Pythonie,
- `Run Keyword If` - wykonuje tylko jeśli warunek jest spełniony.

Omówione operacje są dostępne globalnie za pomocą biblioteki Builtin:

http://robotframework.org/robotframework/3.1/libraries/BuiltIn.html

Robot Framework - Tworzenie keywordów

Podobnie jak nazwy testów keyword'y (funkcje) mogą zawierać w nazwie po jednej spacji. Nazwa taka jest wtedy traktowana jako całość.

Plik libraries.robot:

```
*** Settings ***
Documentation     Variables example.
Library     Collections

*** Test Case ***
Variable With Library
    ${names} =    Create List    Janusz
    ...    Alicja    Natalia
    Sort List    ${names}
    Log    Imiona: ${names}
```

jedna, pełna nazwa

Robot Framework - Tworzenie keywordów

Tworzenie keywordów odbywa się w sekcji **Keywords**. Argumenty wejściowe określa się za pomocą składni **Arguments**, wartość zwracaną z funkcji oznacza się składnią **Return**.

Plik own-keyword.robot:

```
*** Test Cases ***
Sum two numbers
    ${result} =    My Own Sum Keyword    4    5
    Should Be Equal As Integers    ${result}    9

*** Keywords ***
My Own Sum Keyword
    [Documentation]    Add two numbers.
    [Arguments]    ${a}    ${b}
    ${w} =    Evaluate    ${a} + ${b}
    [Return]    ${w}
```

dwa argumenty wejściowe

wartość zwracana

Robot Framework - Tworzenie keywordów

Keyword'y (funkcje) mogą być napisane w języku Python.

W tym celu tworzymy funkcję w pliku `.py`, a sam plik importujemy w plikach z testem za pomocą słowa kluczowego `Library`.

Plik my_math.py:

```
def my_sum(a, b):
    return float(a) + float(b)
```

Plik own-library.robot:

```
*** Settings ***
Library     my_math.py

*** Test Case ***
Custom Library Test
    ${sum} =    My Sum    4    2
    Log     Sum: ${sum}
```

Robot Framework - Tworzenie keywordów

Spacje w nazwie keywordu są tłumaczone na znaki podłogi (_) w nazwie funkcji.

Argumenty przekazywane są jako napisy, dlatego należy **jawnie zamienić je na liczby** (lub inny typ danych, zależnie od potrzeb).

Plik my_math.py:

```
def my_sum(a, b):
    return float(a) + float(b)
```

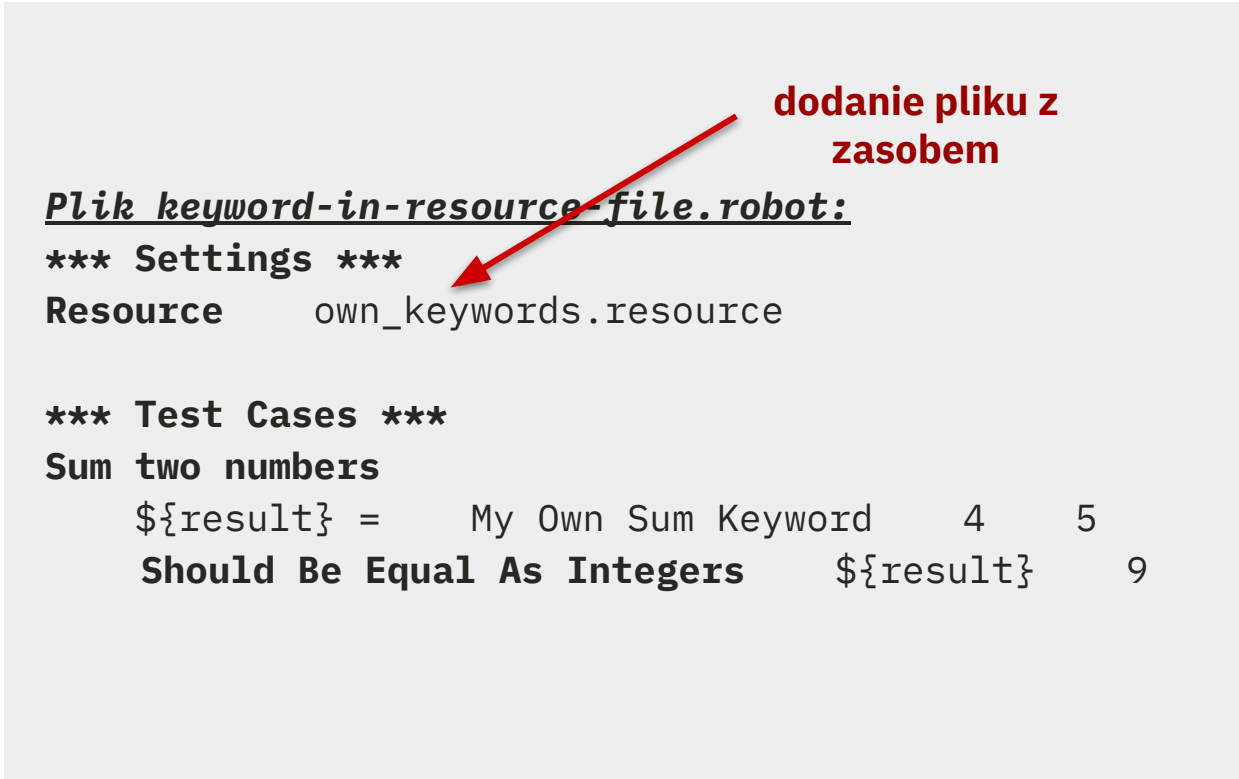
Plik own-library.robot:

```
*** Settings ***
Library     my_math.py

*** Test Case ***
Custom Library Test
    ${sum} =    My Sum    4    2
    Log    Sum: ${sum}
```

Robot Framework - Podział na pliki

Istnieje możliwość podzielenia plików z testami na mniejsze pliki, w których będą umieszczone sekcje **Settings, Keywords, Variables**. Pliki złożone z tych sekcji, używane przez pliki z testami, nazywamy zasobami (Resources).



Robot Framework - Podział na pliki

Istnieje możliwość podzielenia plików z testami na mniejsze pliki, w których będą umieszczone sekcje **Settings, Keywords, Variables**. Pliki złożone z tych sekcji, używane przez pliki z testami, nazywamy zasobami (Resources).

Plik own_keywords.resource:

```
*** Keywords ***
Suma
    [Documentation]    Add two numbers.
    [Arguments]    ${a}    ${b}
    ${w} =    Evaluate    ${a} + ${b}
    [Return]    ${w}
```

Literatura

1. Robot Framework, https://robotframework.org
2. Robot Framework User Guide, http://robotframework.org/robotframework/3.1.1/RobotFrameworkUserGuide.html
3. Adam Roman, Testowanie i jakość oprogramowania. Metody, narzędzia, techniki, 2015, https://ksiegarnia.pwn.pl/Testowanie-i-jakosc-oprogramowania.-Modele-techniki-narzedzia.,732463348,p.html