QA Language Specification by Dmitry Vasin and Zhang An Ni

Input Language: Newly designed human readable language

Target Language: Python

Project Description:

The input language will be a set of human readable instructions intended to be used by QA personnel in software testing. This will be a newly designed language, not an existing one. It will consist of a series of steps, and each step will have an optional result to which the current state of the program will be compared. The idea behind this language is to make a human readable language for creating test cases which can be executed automatically, but can also be examined by a person with no prior experience. Typical usage would have a software engineer write a test case and have all of them executed. Then an untrained QA employee would be able to look at the failing test cases, reproduce them and see why they failed. In order to limit the scope of the task, only web pages will be available for testing using this language.

Code Example:

A typical program in this language would potentially look like this:

```
main :
step 1 : go to https://www.google.ca/ | webpage at https://www.google.ca/ should load
step 2 : enter "test" into textfield with title "Search" | textfield with title "Search" should have value "test"
step 3 : if there is button with value "Google Search" go to step 5 otherwise go to step 4 |
step 4 : exit |
step 5 : click button with value "Google Search" | current webpage should contain "Test - Wikipedia"
step 6 : refresh current webpage | webpage at https://www.google.ca/ should load
step 7 : do step 6 10 times |
step 8 : do refreshwebpage 5 times |


refreshwebpage :
step 1 : refresh current webpage | webpage at https://www.google.ca/ should load
```

Example Explanation:

The above program is a simple script meant to demonstrate the basic functions of the language. Each step is an instruction that will be executed. The "|" symbol separates the instruction from the result which the step should produce, and against which the state of the program will be compared. So step 1 goes to google.com and then checks if the page has loaded. The second step enters the word "test" into the search box and checks if it has been entered correctly. The third step checks if there is a search button, and if there isn't, exits the program. The fifth step clicks the button and checks that the loaded webpage contains the text "Test – Wikipedia" somewhere on the page, as we would expect that result to show up. The 6th step simply refreshes the page. There is no expected result so there is no checking to be done. The 7th step is meant to demonstrate loops, refreshing the page 10 times. The last step calls a set of steps, or a function, which simply refreshes the page 5 times. If all of the steps have executed and the results were as expected, the program would say so, otherwise it would say which step has failed.

Input Language Description:

The language follows the style outlined in the project description as well as the example. Each line is a separate step and can be separated into sets of steps (functions). Each step is split into the desired result and the instruction using the "|" operator. Objects are referred to as their type, such as a button, and their qualities, for example "has value 'Google Search'". Special keywords like click and refresh can be applied to the object. In the result, the object is located to the left of "should" and the condition is to the right of "should". If statement and loops work as specified in the example.

Input Language Components

Two types: string and int, eg when entering text or specifying the number of the step
Related operators: string concatenation, int addition
Looping construct: repeating a step multiple times and goto statements
Condition construct: if statement eg checking for the existence of a button in the example
Procedure construct: A set of steps like the refreshwebpage method in the example
Non-Primitive type: Array of buttons to be acted on

Output Explanation

The output would consist of a python script which can be executed whenever it is required. The output of the python script would simply be the step at which the program failed, or a message stating all steps have succeeded. In order to implement goto statements each step is a function. This leaves a lot of room for optimization which we will be doing later. We will also be using a library called splinter to interact with webpages.

For example the code for the first two steps of the above example would be:

```
from splinter import Browser

def step ():
    return step1()

def step1 ():
    #Step 1
    browser = Browser('chrome')
    url = "https://www.reddit.com/"
    browser.visit(url)

    #Checking Step 1
    if browser.url != "https://www.reddit.com/":
        return "Step 1"
    return step2()

def step2 ():
    #Step 2
    textfield = browser.find_by_name('Search')
    textfield.value = "test"

    #Checking Step 2
    textfield = browser.find_by_name('Search')
    if textfield.value != "test":
        return "Step 2"

def checkSteps():
    err = step();
    if !err.isspace():
        return err
    #Program is done
    browser.quit()
    return("Program finished successfully")

print(checkSteps());
```