Language Tutorial

Team No. 10
iWAL: intuitive Web Automation Language

TABLE OF CONTENTS

Introduction

iWAL(intuitive Web Automation Language) is a simple programming language specifically designed to assist amateur programmers in writing programs that make web-browsing automatic, time-saving and a fun experience. It supports procedural programming paradigm in a top-down fashion and equips the user with a set of basic but highly useful functions that simplify and expedite some of the monotonous browser tasks like sending bulk mails, regularly downloading material from a website, paying monthly bills, filling online forms etc. Though we have a number of web-automation tools available to us, iWAL's appeal lies in its easy usability and wide applicability. To sum it up, iWAL's intuitive and easy-to-learn syntax lets a web-browser user (even with little programming experience) explore the automation power this language lends to some of his mundane browsing tasks.

Input and Output

The program needs to be written in a text file with .iWAL extension, compile using the following command "`iWALc filename.iWAL`"

If the programs does not have any compile time errors, this creates a "filename.class" file, otherwise the errors are displayed. You can run it using the following command (Any run time errors that might occur are displayed) "`iWAL filename`"

The input is just statements and expressions which helps the user to open and interact with the web browser.
The output of the program opens the web browser and simulates the user behavior according to the instructions given.

Comments in the program

'//' is used for representing single line comments and '/* */' for multi line comments.
example :
// This is a single line comment
/* This comment
spans over multiple lines */

Sample Program (Hello World)

// This program opens the web browser and loads a url

```
start();
String url = "https://ssol.columbia.edu/";
open(url);
close();
```

start() - This function opens the web browser with a blank page.
open(<string>)- A function which takes one parameter of string type. If the string is a valid url, it is opened in the web browser. Else it throws “Invalid url” error.
close() -This function closes the browser.

Variables and Expressions

The following snippet shows how the variables and expressions are used in iWAL.

```
int a = 5;
float b = 5.0;
int c = a + 7*3;
string s = “hello” + “world”;
string
boolean flag = true;
key submit = enter;
```

iWAL is static typed language , hence type has to be mentioned when declaring.
For declaration the following syntax is used.

```
<type> variable_name;
```

When initializing a variable during declaration the following syntax should be used.

```
<type> variable_name = <constant/variable>;
```

As seen above, a and c is of integer type and b is of float type.
s is of string type terminated by a null. Operations like concatenation of strings can be performed using plus + operator.
‘flag’ variable is of boolean type which includes constants like true and false. The comparison operators such as > and ≠ are usually defined to return a Boolean value. Conditional and iterative commands may be defined to test Boolean-valued expressions.

‘submit’ variable is of key type. It can be assigned values like Enter, Space, Backspace, Shift, Up, Right, Left, Down. No arithmetic or logical operations can be performed on these constants. These are sent as input to tap() inbuilt function.

Inbuilt functions

iWAL provides a bunch of inbuilt functions, few of which are essential for the user to interact with the browser and others are provided for the convenience of users with less programming language. The list of functions and their descriptions are given below :

start();
Return type : void
Input arguments : None
Function : Opens the web browser.

Example usage : `start();`

open(String);
Return type : void
Input arguments : String
Function : Opens the url passed as an input argument in the web browser.
Example usage : `open("www.google.com");`

userinput();
Return type : String
Input arguments : None;
Function : Prompts the user for the input and returns it in "String" type.
Example usage : `String x = userinput();`

input(String);
Return type : void
Input arguments : String
Function : Enters the given string in the currently highlighted element in the web browser.
Example usage : `input("username");`

inputE(String, String);
Return type : void
Input arguments : String, String
Function : Enters the given string (1st parameter) in the element using the element ID passed as a string (2nd parameter).
Example usage : `inputE("username", "u_id");`

tap(key);
Return type : void
Input arguments : key
Function : Simulate a key tap on the currently active element in the web browser.
Example usage : `tap(Enter);`

tapE(key, String);
Return type : void
Input arguments : key, String
Function : Simulates a key tap on the element passed as an argument.
Example usage : `tapE(Space, "u_button");`

click();
Return type : void
Input arguments : None
Function : Clicks the currently active element in the web browser.

Example usage : `click();`

clickE(String);
Return type : void
Input arguments : String
Function : Clicks the element whose element ID is passed as a String.
Example usage : `clickE("u_button");`

delay(int);
Return type : void
Input arguments : int
Function : Halts the program for the number of milliseconds passed as argument.
Example usage : `delay(10000);`

tab(int);
Return type : void
Input arguments : int
Function : Skips as many elements as the value of the integer argument passed.
Example usage : `tab(10);`

currentelement();
Return type : String
Input arguments : None
Function : Returns the element ID of the currently active element in the web browser.
Example usage : `currentelement();`

download(String, String, String);
Return type : void
Input arguments : String, String, String
Function : This downloads the media files depending on the arguments passed
1st argument can be "image" or "video"
2nd argument can be "all" or "<number of elements to download>"
3rd argument gives the path, where the files are to be saved.
Example usage : `download("image", "all", path);`

Control flow

Loops :

In iWAL, we use loops to repeat tasks certain number of times based on some condition. For ease of use, we have two different loops.

- 'repeat' loop is used when you know before hand the number of times the block of statements have to be executed
- 'until' loop is used when the number of times a block of statements have to repeated is based on some condition.

Sample program (using repeat):

```
//The following snippet is used to click the current element 20 times. For example if the
//current element is a button, the following snippet can be used to click the element 20
//times.
int numOfRepeats = 20;
String url = "https://ssol.columbia.edu/";
open(url);
repeat(numOfRepeats){
        click();
}
```

Here repeat(<integer>) is used to repeat the following block of code certain number of times. click() is used to click the current element which can be a simple button, radio button, check box, text box etc.

Sample program (using until):

```
//The following snippet is used to tick all the checkboxes until an element is found
String element = "submit_form";
String currElement = currentelement();
until(currElement!=element){
        tap(Space);
        tab(1);
        currElement = currentelement();
}
click();
```

if-else :

iWAL supports if else statements so as to properly direct control flow.

ifelse
if (expression)
{
 statements
}
else
{
 statements
}

// The following snippet is used to check if two element IDs are equal. If they are, input is given to that element.

```
String element = “user_name”;
String currElement = currentelement();
if(currElement==element){
	input(“username”);
}
```

Functions

As always, a function is a module of code that takes information in (referring to that information with local symbolic names called parameters), does some computation, and (usually) returns a new piece of information based on the parameter information.

```
<type> def function-name(<type> <parameter1>, <type> <parameter2>){
statements
}
```

Sample Program (using functions)

// the following program has a function defined which is used to login to a website by inputting username and password.

```
start();
open( “https://ssol.columbia.edu/”);
login(“username”, “password”);


void def login(string username, string password){
	inputE(username, "u_id");
	inputE(password, “u_pw”);

	tab(1);
	click();
	return;
}
```

Sample Program 1

```
start();
String url = “https://www.google.com/imghp”;
String element = “q”;
String query = “this is a search query for google”;
inputE(query, element);
tab(3);
click();
String path = “/users/XYZ/Desktop/Folder”;
download(“images”, “all”, path);
close();
```

The above program downloads all the images resulting from a google image search.

Step-by-Step explanation :
1) Starts the web browser
Declares and initializes the String variables :
2) ‘url’ with the url of the google image search.
3) ‘element’ with the element ID of the search text box.
4) ‘query’ with the query string that needs to be searched.
5) Inserts the search query in the search text box using its element ID.
6) current element moves to the 3rd element from the current element.
7) Clicks the current element.
8) Declares and initializes the String variable path with the path where the images are to be stored.
9) Uses the “download” function to download all the images to the given path.
10) Close the web browser.

Sample Program 2

```
start();
open(“https://ssol.columbia.edu/”);
inputE(username, "u_id");
inputE(password, “u_pw”);

tab(1);
click();
delay(1000);
tab(21); // current element is the link Text Message Enrollment after moving 21 tabs
click();
delay(1000);
inputE(“543”, “tran[1]_area_code”);
```

```
inputE(“434” , ”tran[1]_zone_code”);
inputE(“2325”, “tran[1]_last_code”);
clickE(“tran[1]_save”);
```

The above program can be used for logging into SSOL and signing up for text message service.

Step-by-Step explanation :
1) Starts the web browser.
2) Loads the SSOL page.
3) Inputs the username using the element ID.
4) Inputs the password using the element ID.
5) Moves to the next element.
6) Clicks the current element.
7) Waits for 1000 milli seconds.
8-10) Inputs the area code, zone code and the last code of the phone number using element ID.
11) Clicks the save button using the element ID which finishes subscribing.