|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| Team 3   |  |  |  | | --- | --- | --- | | Name | Role | UNI | | Morris Hopkins | Project Manager | mah2250 | | Seungwoo Lee | Language Guru | sl3492 | | Lev Brie | System Architect | ldb2001 | | Alexandros Sigaras | System Integrator | as4161 | | Michal Wolski | Verification and Validation | mtw2135 | |

|  |
| --- |
| COMS W4115 PROGRAMMING LANGUAGES AND TRANSLATORS |
| SWIM |
| Language Tutorial |

Table of Contents

1. Introduction 1

2. Preparing to Swim 1

2.1 Install Swim under MAC OS X 1

2.2 Install Swim under Windows 2

3. Getting Started 2

3.1 Hello World Code 2

3.2 Running Hello World 2

3.3 Hello World Result 2

4. Variables and Arithmetic Operations 3

4.1 Variables and Arithmetic Operations Code 4

4.2 Running Variables and Arithmetic Operations 5

4.3 Variables and Arithmetic Operations Result 5

5. Interacting with the web 6

5.1 New York Times Web Scraping Code 6

5.2 Running New York Times Web Scraping 6

5.3 New York Times Web Result 7

6. Export to PDF 8

6.1 Export to PDF Code 8

6.2 Running the Export to PDF 8

6.3 Exporting to PDF Result 9

Learning how to Swim

# Introduction

Allow us to introduce you to Swim. Swim is a cross-platform scripting language intended to simplify the collection of data from the Internet. The language targets researchers from all fields with a basic knowledge of Object Oriented Programming and a basic understanding of web development (HTML, JavaScript, CSS).

The purpose of this tutorial is to introduce you and show you the very basics on how to Swim. **Section 2** describes how to set up your environment to enable to run Swim while **Sections 3 – 6** include small sample programs and are designed to get you as quickly as possible to the point where you can crawl information from the Internet. These examples concentrate on the basics of the language like variable declaration, arithmetic operations, webpage crawling and exporting results to local files like .pdf.

Under no circumstances is this a complete tutorial on Swim. Like any programming language, to truly learn it you must get your feet wet. For more in depth information on the Swim language please refer to the Language Reference Manual.

Have fun programming!

# Preparing to Swim

Preparing your environment to run Swim is easy and straightforward. Swim is a cross-platform programming language that is Python based. Therefore, prior to installing Swim, your system must be running Python v2.6 or later.

For instructions on how to download and install python on your system, please visit: <http://www.python.org/getit/>

## Install Swim under MAC OS X

To install Swim to run under MAC OS X simply double click on the ***setupSwim.pkg*** and follow the instructions provided on the screen.

## Install Swim under Windows

To install Swim to run under Windows (XP, Vista, Windows 7 and Windows 8), simply double click on the ***setupSwim.msi*** and follow the instructions provided on the screen.

# Getting Started

In the HelloWorld.swim code (3.1) the built in function print is called with the string literal “Hello World!\n” as an argument expression. The “\n” at the end of the string literal is the ASCII line feed escape sequence which moves the output cursor to the following line.

## 3.1 Hello World Code

01 // Hello World Program  
02 print(“Hello World!\n”);

Program - Hello Wolrd

## 3.2 Running Hello World

To run the Hello World program, ***“HelloWorld.swim”*** containing the code in ***Program 1.*** Then from the terminal (MAC OS X) or Command Prompt (Windows) type:

|  |
| --- |
| swim HelloWorld.swim |

## 3.3 Hello World Result

Running the Hello World program prints

Hello World!

# Variables and Arithmetic Operations

The code for the “**variablesAndOperations.swim**” program (4.1) demonstrates the basics of variable assignment, arithmetic operations, and conditional statements. The program begins on line 4 where the variable a is assigned the value of the numeric literal 3. The ‘=’ is the assignment operator, ‘a’ is an identifier representing the variable storing the assigned value, and the ‘3’ is an expression evaluated to its numeric value which the assignment operator then assigns to the variable a. Line 5 repeats these steps for another variable ‘b’.

Arithmetic operations are written infix and evaluate left to right. Line 8 through line 13 demonstrate the six basic arithmetic operations available in Swim. The identifiers sum\_result, sub\_result, prod\_result, div\_result, mod\_result, and pow\_result are all identifiers representing variables as described previously. On each line the ‘a’ and ‘b’ characters are the variables previously defined and for each line they are evaluated to their respective numeric values. a+b, a-b, a\*b, … are expressions which evaluate to the sum, difference, product, quotient, modulus, and exponential values. Once evaluated the variable to the left is assigned the result.

Beginning on line 18 the program demonstrates the use of an if conditional statement. It says that if the value of ‘a’ is less than the value of ‘b’ execute the statements following the keyword ‘do’ and ending with the keyword ‘end’. This illustrates the general structure of an if statement, which is if(condition) do statement end.

A more complex if conditional statement is shown beginning with line 24. Here the conditional expression contained within the parenthesis following the if keyword demonstrate the ability to perform arithmetic operations as long as the resulting value evaluates to true or false. This is the general case for all conditional statements unless otherwise noted.

On line 28 the program uses an elif statement which only executes if the condition in the parentheses following the if keyword evaluates as false and the condition in the parenthesis following the elif keyword evaluates as true. There can be an arbitrary number of elif sections within and if elif section but only the statements with the first elif condition that evaluates to true are executed. Optionally an else may appear as in line 34 for which the associated statements will execute if none of the if or elif conditions prior to that have been evaluated to be true. Each elif section ends when a following elif, else, or end is found.

## 4.1 Variables and Arithmetic Operations Code

01 // Variables and Arithmetic Operations Program  
02  
03 // Arithmetic declaration  
04 a = 3;  
05 b = 5;  
06  
07 // Basic Arithmetic Operations  
08 sum\_result = a + b; // Adding variables  
09 sub\_result = a – b; // Subtracting variables  
10 prod\_result = a \* b; // Multiplying variables  
11 div\_result = a / b; // Getting the quotient of the division  
12 mod\_result = a % b; // Getting the remainder of the division  
13 pow\_result = a ^ b; // Exponentiation of variables  
14  
15 // Basic Conditional Operators  
16  
17 // Simple Operator  
18 if ( a < b) do // if a is less than b then execute   
19 // statements in the scope  
20 print(“It is smaller\n”); // print the line: It is smaller  
21 end // end if statement  
22   
23 // Multi-Conditional Operator  
24 if ( a <= b – 2) do // if a is less than or equal to b – 2   
25 // then execute statements in this scope  
26 print(“Less than or equal\n”);  
27 // print the line: Less than or equal  
28 elif ( a < b ) // else if a is smaller than b then  
29 // execute statements in this scope  
30 print(“Less than b”); // print the sentence: Less than b  
31 print(“, but greater than b – 2\n”);  
32 // print the line: , but greater than   
33 // b -2  
34 else // execute the statements in this scope 35 // if no other condition is true  
36 print(“Greater than or equal\n”);  
37 // print the line: Greater than or   
38 //equal  
39 end // end multi-conditional if statement

Program – Variables and Arithmetic Operations Program

## 4.2 Running Variables and Arithmetic Operations

To run the Variables and Arithmetic Operations program, create a file whose name ends in ***.swim*** like ***“variablesAndOperations.swim”*** containing the code in ***Program 2.*** Then from the terminal (MAC OS X) or Command Prompt (Windows) type:

|  |
| --- |
| swim variablesAndOperations.swim |

## 4.3 Variables and Arithmetic Operations Result

Running the Variables and Arithmetic Operations program prints  
  
It is smaller  
Less than or equal

# Interacting with the web

The code for the “**NYTimes.swim**” program (5.1) demonstrates Swim’s core purpose: to collect data from the internet. Line 2 defines a string that represents the URL of the website the program will scrape. Line 3 defines a string containing the CSS selector of the html element containing the content to be collected. Line 4 creates another variable called doc which contains the returned value from the built-in @ function. The @ function takes a selector and URL as parameters and produces a string of the result. This string may be plain text but may also include html, Javascript and CSS information nested in the specified html selector. We make the assumption that the users of Swim will have some knowledge of html and be able to accurately specify the html selectors needed for their intended usage. In this example when line 04 executes the entire contents of the html page will be stored as a string in the doc variable.

After collecting some data from the internet additional scraping and analysis can be done. Line 7 defines another string containing another CSS selector. From here this program demonstrates that Swim is also able to perform scraping on local files or data stored in variables. As before, the contents variable will contain a string representation of the collected data. contents.text() is a function that removes html elements from the matched content. Print prints the string of content.

## 5.1 New York Times Web Scraping Code

00 // New York Times Web Scraping Program  
01  
02 url = "http://www.nytimes.com"; // set the url to crawl information   
03 selector = "html"; // set the jQuery style selector  
04 doc = @(selector, url); // parse the given url with  
05 // selector and return the  
06 // appropriate object  
07 selector\_tools = "#toolshome a"; // set the jQuery style selector  
08 contents = @(selector\_tools, doc); // parse the given doc object with  
09 // selector\_tools and return the  
10 // appropriate object  
11 text = contents.text(); // get the text of the contents  
12 print(text); // print the text of the contents

Program – New York Times Web Scraping

## 5.2 Running New York Times Web Scraping

To run the New York Times Web Scraping program, create a file whose name ends in ***.swim*** like ***“NYTimes.swim”*** containing the code in ***Program 3.*** Then from the terminal (MAC OS X) or Command Prompt (Windows) type:

|  |
| --- |
| swim NYTimes.swim |

## 5.3 New York Times Web Result

Running the New York Times Web Scraping program prints

Subscribe to Home Delivery

# Export to PDF

The code for the “**exportToPDF.swim**” program (6.1) demonstrates one of Swim’s output formats. Rather than printing the collected data to a console many users may wish to store their information as PDF files. Descriptions of the operations performed from line 2 to line 18 of this program have already been covered in the previous example. Line 17 of the program calls the pdf function which takes a string of content and a string representing a filepath as arguments. With this simple function call a PDF file is created.

## 6.1 Export to PDF Code

00 // Exporting to PDF Program  
01  
02 url = "http://www.cs.columbia.edu/~aho" // set the url to crawl  
03 // information  
04 selector = "html"; // set the jQuery style  
05 // selector  
06 doc = @(selector, url); // parse the given url with  
07 // selector and return the  
08 // appropriate object  
09 selector\_tools = "h1"; // set the jQuery style  
10 // selector  
11 contents = @(selector\_tools, doc); // parse the given doc  
12 // object with selector\_tools  
13 // and return the appropriate  
14 // object  
15 text = contents.text(); // get the text of the  
16 // contents  
17 pdf(text,"Contents.pdf") // create a pdf named  
18 // Contents.pdf containing  
19 // the text of the contents

Program – Export to PDF

## 6.2 Running the Export to PDF

To run the Export to PDF program, create a file whose name ends in ***.swim*** like ***“exportToPDF.swim”*** containing the code in ***Program 4.*** Then from the terminal (MAC OS X) or Command Prompt (Windows) type:

|  |
| --- |
| swim exportToPDF.swim |

## 6.3 Exporting to PDF Result

Running the Export to PDF program creates a file called ***Contents.pdf*** containing the text: **ALFRED V. AHO**