Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science



A11

Language Specification

Lab Professor / Lab Session:

Paulo Sousa / 011

Team:

Henry Forget - Id: 041023812 / Colin Tapp – Id: 040774946

Language Name [???]

|  |  |
| --- | --- |
| **Part**  **1** | **Language User Reference** |

**EXPLANATION**

We have designed \_\_\_ to be an introduction to high level programming for younger students that have never coded before, or who have only used block coding programs like Scratch.

We based \_\_\_ on Python as it is one of the most popular languages while also being intuitive and friendly to learn. However as you will see within this document there are several differences between \_\_\_ and Python that makes it applicable to other high level programming languages as well as being simplified to allow for younger students to fully grasp all the available tools.

* 1. **User Manual**

**Element 1: Name / Extension**

*[Name your language! We suggest you use one "word" related to your “Go-like” language or DSL]*

*[What is the filename extension of your language? For example, for C it is .c, and for Professor Paulo's “Sofia”* ***language*** *it is ".****sof****".]*

*[What is your language patterned after, or what is it similar to? What languages are inspiring your choice? It's okay if you're following Go closely.]*

**Element 2 – Comments**

In Python, comments are declared with a hash (#) at the beginning of the line. \_\_\_ will operate the same way, as follows:

# This is a comment

Similar to Java, a backslash and asterisk on either side of the text creates a multi-line comment:

/\* This

Is

Multi-line \*/

**Element 3 – Keywords**

\_\_\_ contains many of the basic Python keywords which may not be used apart from that. All of their purposes are the same as they were originally used in Python.

**Element 4 – Datatypes**

\_\_\_ will follow standard datatype conventions as follows:

int: holds numbers between -32,768 to 32,767 as standard; 2 bytes

float: holds numbers between 1.2E-38 to 3.4E+38 as standard; 4 bytes

string:

bool: hold either ‘TRUE’ or ‘FALSE’; 1 byte

**Element 5 – Variables**

In Python, variables are simply declared with a name and a value, the datatype is inferred. \_\_\_ will not follow the same ruling as we believe it’s important for students to understand how each datatype works. To create a data type you would do the following:

int a = 123

string b = “one, two, three”

float c = 1.23

\_\_\_ will also include constants which operate the same as standard, creating an immutable variable. They are declared by a preceding “const”:  
const float pi = 3.1415

**Element 6 – Methods / Functions**

Functions in \_\_\_\_ are defined by a preceding “def” keyword, the returned variable type, the name, space for arguments and finally a colon, similarly to Python. Each line of code within the function must be indented by one tab stop as this teaches students how to write elegant and readable code. Something like:

def void myfunction():  
 print(“Hello from function”)  
print(“Hello from outside”)

myfunction()

Would print the “Hello from outside” first as it’s not contained within the function.

Functions can have one, or several arguments contained within the brackets following the function name as standard. Something like:

def myfunction(name, age):  
 print(“My name is ” + name + “ and I am ” + age + “ years old!”)

myfunction(“Henry”, 21)

Would print “My name is Henry and I am 21 years old!”  
\* Variables can also be passed as parameters the same as any other programming language

**Element 7 - Attribution / Assignment**

Casting will be included in \_\_\_ and can be achieved by using datatype(variable/value) such as:  
string a = “3”  
int x = int(a)

X would now have the value of a numerical 3.

Math will be handled as standard for any language with +, -, \* and / for the four basic operators. ie:

int a = 4  
a = a + 6 – 8 # final result will be 2  
a = (a \* a) / 4 # final result will be 1

String concatenation will also be included and will use the + operator as standard.

string fName = “Colin”  
string lName = “Tapp”  
string name = fName + “ ” + lName # Result will be “Colin Tapp”

**Element 8 – Selection**

\_\_\_ will include the usual logical conditions from standard programing languages, as follows:

&& (and), || (or), ! (not)

== (equals), != (not equal)

< (less than), > (greater than), <= (less than or equal to), >= (greater than or equal to)

This allows \_\_\_ to handle if-style logic such as:

if (a < b):   
 print(“a is less than b”)  
else if (a > b):  
 print(“a is greater than b”)  
else:  
 print(“a and b are equal”)

As well as switch/case style logic such as:

switch(day):   
 case 1:  
 print(“Monday”)  
 case 2:  
 print(“Tuesday”)  
  **…**  
 default:  
 print(“There is no day with that number”)

**Element 9 – Interaction**

\_\_\_ will include both for and while loops functioning similarly to C or Java as opposed to Python to encourage learning the standard; however instead of the semicolon separation the expressions will be separated by commas. For loops are written as follows:

for (i = 0, i < 5, i++):   
 print(i)

While loops are written as follows:

while (i < 5):   
 print(i)  
 i = i + 1

**Element 10 – Input / Output**

\_\_\_ will follow Python for it’s input and output commands.  
Input is done with a simple input() keyword where it prompts the user to enter a message that is saved as a string. This saving limitation also acts as an elegant way to teach students a simple to grasp use for casting, if for example they want to input a number.

string name = input(“What is your name?: ”)

\* Strings passed in through the arguments are displayed when prompted

Output is much more simple, as seen throughout the document. Just a simple print() keyword.

print(“Hello World”)

**Element 11 – Proper Elements**



|  |  |
| --- | --- |
| **Part**  **2** | **Language Comparison** |

* No do loop
* No char
* Python has much bigger numbers
* Python can use quotes and double quotes
* No classes in our language
* No termination character like ;
* Python has no returned types for functions (we do)

**Comparing with C language**

**Differences**

|  |  |  |
| --- | --- | --- |
|  | [Explanation] |  |

**Advantages / Disadvantages (in comparison with C)**

|  |  |  |
| --- | --- | --- |
|  | [Explanation] |  |

**Comparing with Python language**

**Language Name:**

**Differences**

|  |  |  |
| --- | --- | --- |
|  | [Explanation] |  |

**Advantages / Disadvantages (in comparison with Python)**

|  |  |  |
| --- | --- | --- |
|  | [Explanation] |  |

|  |  |
| --- | --- |
| **Part**  **3** | **Architectural Questions** |

**Advantages**

*[What's the goal of your language? Are you trying to make something simple, fun, complicated? My personal language, Chambly, is based around being useful to scientists. (You can just make something up here, honestly. Think about it a little bit, have a little fun.)]*

**Strategy: C Implementation**

*[How your language can be implemented in C – ex: datatypes]*

* *In plain English, or maybe even some high-level pseudocode, how are you going to parse your language? You will be writing a compiler for your language, so these are some things you need to think about.*

***Note 1: C Datatypes***

*Remember that you are implementing your language in ANSI C. For this reason, you cannot create arbitrarily your language (from scratch). You need to use what is already provided by C Compiler. For this reason, think about using and defining the language obeying the datatypes.*

*[Your ideas about how to identify elements from language]*

* *Consider your "write to the console" command as an example. How will your compiler detect it? How will it sort out what to write to the console? What if there's some literal text (ie: "this is going to get printed") instead of variables?*

*[Your ideas about how to identify scope (ex: blocks between conditionals or functions)]*

* *How do you mark a block of code? If I use your loop logic, how do I control what portion of code gets looped through? In C, you might use { and }. In Python, the indentation is what matters. How does it work in your language?*

**References**

Algonquin College

Winter, 2024