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 [Mouse]

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

**EXPLANATION**

We have designed Mouse 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 Mouse 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 Mouse 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**

Language: Mouse

Extension: .mse

**Element 2 – Comments**

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

# This is a comment

Multiline comments can be done with several continuous lines each starting with #

**Element 3 – Datatypes**

Mouse will follow standard datatype conventions as follows:

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

int: ranged from -2,147,483,648 to 2,147,483,647; 2 bytes

float: ranged from 1.2E-38 to 3.4E+38; 4 bytes

str: support the full range of Unicode characters

**Element 4 – Keywords**

Mouse 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.

[1]

**Element 5 – Variables**

In Python, variables are simply declared with a name and a value, the datatype is inferred. Mouse 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:

a: int = 123

b: str = “one, two, three”

c: float = 1.23

Mouse will also include constants which operate the same as standard, creating an immutable variable. They are declared by a preceding “const” and a variable name in all ca:  
const PI: float = 3.1415

**Element 6 – Methods / Functions**

Functions in Mouse\_ 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 space as this teaches students how to write elegant and readable code. Something like:

def myfunction(): void:  
 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): void:  
 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 Mouse and can be achieved by using datatype(variable/value) such as:  
a: str = “3”  
x: int = 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. i.e.:

a: int = 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.

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

**Element 8 – Selection**

Mouse 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 Mouse to handle if-style logic such as:

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

As well as switch/case style logic such as:

match day:   
 case “Monday”:  
 print(“It is the first day of the week”)  
 case “Tuesday”:  
 print(“It is the second day of the week”)  
  **…**  
 case \_:  
 print(“There is no day with that number”) [2]

**Element 9 – Interaction**

Mouse will incorporate both for and while loops, functioning similarly to C and Java. As opposed to Python's more advanced loop syntax, Mouse will adhere to conventional loop styles to encourage learning programming fundamentals. 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**

Mouse 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.

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

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

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

print(“Hello World”)

**Element 11 – Proper Elements**

Mouse will also include a simplistic class definition system. This will present itself similarly to Python’s standard class creation however it will work more similarly to C’s structs as Mouse’s classes will **not** contain unique functions within them.

class Person:

def \_\_init\_\_(self, name, age):

self.name = name

self.age = age

This allows students to learn the basic of object oriented programming while still using the simplistic and approachable nature of a script language.

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

**Comparing with C language**

**Differences**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  | | --- | --- | --- | |  |  |  | |  | **C** | **Python-like** | | Syntax | More complex | Easily readable. | | Typing | Statically typed. Types declared at compile-time. | Dynamically typed. Types determined at runtime. | | Execution | Compiled language. | Interpreted language. | | Performance | Compiled languages typically perform better. | Interpreted languages may have lower performance in some respects. | | Libraries | Limited, well-established libraries. | Access to many extensive libraries. | |  |  | [3] [4] | |  |

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  | **C** | | Advantages | C is a compiled language known for its high performance | | Provides direct interaction with hardware resources, giving developers fine-grained control. | | Having been around for decades and still widely in use, makes it easier for developers to maintain and extend legacy C code. | | Used for a wide range of applications, from system software to high-level applications. | | Disadvantages | Managing pointers can be challenging, leading to memory errors and vulnerabilities. | | C requires manual memory management, which can be code heavy, prone to errors and allow the programmer to unintentionally affect already used memory. | | C's error handling mechanisms are limited, making debugging difficult. | | Lacks built-in support for object-oriented programming. |   [3] [4] [5] |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  | | --- | --- | --- | |  |  |  | |  | **Python-like Language** | | Advantages | Python's simple and intuitive syntax, makes coding easier for beginner and experienced coders alike. | | Python is a multi-purpose language with a wide range of uses. | | Python's large active community provides abundant libraries and resources. | | Python's wide variety of libraries and frameworks provide tools for most problems that can be solved with code. | | Disadvantages | As an interpreted language, Python is prone to slower execution speeds, especially for computationally intensive tasks. | | Relies on automatic memory management, which can lead to leaks. | | Python has limitations in multithreaded programming. | | Dynamically typed, interpreted, and has a garbage collector, limiting low-level operations. | |  |  |  |   [3] [4] [5] |  |

**Comparing with Java**

**Language Name: Java**

**Differences**

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  |  | | --- | --- | --- | |  | **Java** | **Python-like** | | Syntax | Java uses curly braces {} to define code blocks | Python uses whitespace to determine code blocks | | Typing | Statically typed. Types declared at compile-time. | Dynamically typed. Types determined at runtime. | | Execution | Compiled language. Code is compiled into bytecode which is read by the JVM. | Interpreted language. Code is executed line by line in the Python interpreter. | | Performance | Compiled language with static typing generally performs better. | Interpreted language with dynamic typing may have comparatively worse performance. | | Use Cases | Enterprise-level applications, backend development, mobile apps and OS. | Data science, AI, general scripting. |   [3] [4] [6] |  |

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

|  |  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  | **Java** | | Advantages | Compiled to bytecode, Java provides high performance, making it suitable for compute-intensive tasks. | | Write once, run anywhere philosophy allows Java programs to run on any device with a Java Virtual Machine (JVM). | | Strong support for multithreading and concurrent programming enhances scalability. | | Extensive libraries and frameworks (e.g., Spring) cater to various application domains. | | Disadvantages | Java's traditional programming style can be less intuitive than Python's easily readable syntax | | Java applications may consume more memory due to the overhead of the JVM. | | Steeper learning curve due to explicit type declarations and other language features. |   [3] [4] [6] [7] |  |

|  |  |  |  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
|  | |  |  | | --- | --- | |  | **Python-like** | | Advantages | Python's clean and straightforward syntax promotes readability and ease of learning. | | Python has a vibrant community, contributing to extensive documentation, support, and a wealth of resources. | | Python's wide variety of libraries and frameworks provides solutions for various areas. | | Suitable for diverse applications, from web development to data science and machine learning. | | Disadvantages | As an interpreted language, Python can be slower than compiled languages, particularly for computationally intensive tasks. | | Global Interpreter Lock (GIL) can hinder the performance of CPU-bound tasks in multithreaded scenarios. |   [3] [4] [6] [7] |  |

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

**Advantages**

We are aiming to create a simple, yet powerful, general programming language that is accessible to newer programmers while teaching good fundamentals of programming.

**Strategy: C Implementation**

Text will be parsed for special characters, and lines will end with a new line character.

* If a ‘#’ character is found it will disregard the remaining text until it hits a new line character.
* If the keyword ‘def’ is found it will look for the correct syntax following for a function to be defined.
* If a datatype keyword ‘int’, ‘float’, etc. it follows the format of:
  + ‘datatype’ ‘variable name’ ‘assignment operator’ ‘value’.

The text can also be parsed for special keywords that denote different commands within the language.

* print signifies that any text within the brackets following it will be printed out to the console, variable data can also be concatinated using string concatination.
* if or match signifies that any text within the brackets following it will be analized using binary logic.
* class signifies that the user is creating their own struct like object

Finally scope is indentified using TAB indentaions and colons. When a colon is present, any following tab indentations are a sub-block for the code; similar to how C and Java uses curly braces. For example:

if (grade > 49):

if (grade > 89):

print(“You passed with flying colors!!”)

else:

print(“You passed!”)

else:

print(“You did not pass”)

\* Indentation is exaderated to show detail

**References**

|  |  |
| --- | --- |
| [1] | W3 Schools, "Python Keywords," W3 Schools, [Online]. Available: https://www.w3schools.com/python/python\_ref\_keywords.asp. [Accessed 16 01 2024]. |
| [2] | C. Kolade, "Python Switch Statement – Switch Case Example," freeCodeCamp, 05 08 2022. [Online]. Available: https://www.freecodecamp.org/news/python-switch-statement-switch-case-example/. [Accessed 16 01 2024]. |
| [3] | GFG, "Difference between Compiled and Interpreted Language," Geeks for Geeks, [Online]. Available: https://www.geeksforgeeks.org/difference-between-compiled-and-interpreted-language/. [Accessed 14 01 2024]. |
| [4] | A. Ajitsaria, "What Is the Python Global Interpreter Lock (GIL)?," Real Python, [Online]. Available: https://realpython.com/python-gil/. [Accessed 14 01 2024]. |
| [5] | testbook, "Difference Between C and Python," [Online]. Available: https://testbook.com/key-differences/difference-between-c-and-python. |
| [6] | The Server Side, "Real world Java applications," The Server Side, [Online]. Available: https://www.theserverside.com/blog/Coffee-Talk-Java-News-Stories-and-Opinions/Java-Applications-Uses-Types-Games-Best-Apps-Minecraft-Android-Mobile-Desktop-IoT. [Accessed 14 01 2024]. |
| [7] | avcontentteam, "Java vs. Python: Which Language Is Right For You?," Analytics Vidhya, 29 04 2023. [Online]. Available: https://www.analyticsvidhya.com/blog/2023/04/java-vs-python/. [Accessed 14 01 2024]. |
|  |  |

Algonquin College

Winter, 2024