Algonquin College Logo

# SCHOOL OF ADVANCED TECHNOLOGY

### ICT - Applications & Programming

### Computer Engineering Technology – Computing Science



A11

Language Specification

Lab Professor / Lab Session:

[**PAULO SOUSA**/ 012]

**Team:**

**[Hamza El Sousi]** - Id: **[040982818] / [Mohammed Alashi]** - Id: **[041070990]**

Language Name [**4Cast**]

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

**EXPLANATION**

*The purpose of this assignment is to invent a new computer language.*

* *This language can have the syntax and structure of your choosing.*
* *Option 1: Adapt the ‘Sofia language to be python compatible (see* <https://www.python.org/>*).*
* *Option 2: Define a* ***DSL*** *– Proper to solve specific problems (ex: science, economy, music, etc.)..*

*This is going to be a basic language. There's a lot of functionality that we'll be skipping over, while we implement the basics. You will need to tell me those basics, of course. In this document, I'm going to explain the steps of what to do with a bit of detail.*

* 1. **User Manual**

**Element 1: Name / Extension**

* *Language Name: "4Cast"*
* *File Extension: ".fcst"*
* ***The design of "4Cast" is influenced by the structure and conventions of Python, aiming to provide a familiar and accessible experience for users who are already acquainted with Python****,* ***as well as leveraging Python's simplicity and Java's strong typing and object-oriented features.***

**Element 2 – Comments**

*An Example of how “Sword” comments will be:*

***#This is a single-line comment***

***\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

***'''***

***This is a***

***multi-line comment***

***'''\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_\_***

**Element 3 – Keywords**

*[Keywords: List the sequence of reserved / key words from your language]*

***if, elif , for, while, class, return, break, continue, def, import.***

1. **if: Used for conditional statements. Executes a block of code if a specified condition is true.**
2. **elif: Stands for "else if." Used in conjunction with if to specify additional conditions to be tested.**
3. **for: Used to iterate over a sequence (such as a list, tuple, or string) or other iterable objects.**
4. **while: Used to create a loop that executes a block of code as long as a specified condition is true.**
5. **class: Used to define a class in object-oriented programming.**
6. **return: Used to exit a function and return a value.**
7. **break: Used to exit a loop prematurely.**
8. **continue: Used to skip the rest of the code inside a loop for the current iteration.**
9. **def: Used to define a function.**
10. **import: Used to include external modules or libraries into a Python script.**

**Element 4 – Datatypes**

*[Datatypes: Define integers, real numbers (float points) and strings. Determine their ranges]*

*[Remember to define the number of bytes – and, if possible, range]*

1. ***int (Integer):*** 
   * ***Integer type with a specified 32-bit range: -2,147,483,648 to 2,147,483,647.***
2. ***float (Floating-Point):*** 
   * ***Floating-point numbers represented in double precision (64-bit).***
3. ***str:***
   * ***Textual data type for representing sequences of characters.***
4. ***bool: True or false.***

**Element 5 – Variables**

*[Variables: How would a programmer define variables that can hold integer numbers (numbers with no decimal point), floating point numbers (numbers with a decimal point) or text (ie: strings in Java).  This is element 1.  Consider if you want to flag the variables in a special way, like SOFIA or BASIC, or not, like C or Java.]*

**int\_var = 42 # Integer variable**

**float\_var = 3.14 # Float (floating-point number) variable**

**string\_var = "Hello, 4Cast!" # String variable**

**Element 6 – Methods / Functions**

**In "4Cast" you can define a function using the function keyword, followed by the function name, parameters (if any), and the function body enclosed in curly braces. Here's an example:**

# Define a simple function without parameters

def greet():

print("Hello, Sword!")

# Call the function

greet()

**Function with Parameters:**

**You can also define functions with parameters. Here's an example:**

# Define a function with parameters

def add(a, b):

return a + b

# Call the function with arguments

result = add(3, 5)

print(result) # Output: 8

**In this example, the add function takes two parameters (a and b) and returns their sum. The function is then called with arguments 3 and 5, and the result is printed.**

**Function with Return Type:**

**If your functions have return values, you can specify the return type using the -> arrow notation:**

# Define a function with a return type

def multiply(x, y) -> int:

return x \* y

# Call the function with arguments

result = multiply(4, 6)

print(result) # Output: 24

**Element 7 – Commands**

* ***Attribution / assignment****: How does your language let a programmer assign a value to a variable? (Will you allow casting? If so, how will it work?) How will your language handle math, and will it allow strings to be concatenated (merged)?*

***“4Cast" language, a programmer assigns a value to a variable using the assignment operator =. For example:***

***variable\_name = 42***

***Casting is allowed for converting between compatible data types. For example, converting an integer to a floating-point number:***

**float\_var = float(int\_var)**

* ***Selection****: How does your language do if-style logic? (Optional: Do you want to do some kind of switch/case as well?). You will need to explain how "conditionals" work in your language. How do you write Boolean operations, such as "or", "and", "not", and other conditions, such as less than, greater than, etc?*

***If-Else:***

*temperature = 25 In this example, the if statement checks if the temperature is greater than 30. If true, it prints "It's hot!"; otherwise, it prints "It's not too hot."*

*if temperature > 30:*

*print("It's hot!")*

*else:*

*print("It's not too hot.")*

***If-Elif-Else:***

***score = 85 Here, the elif (else if) statement allows checking multiple conditions. The first true condition's block will execute.***

***if score >= 90:***

***print("Excellent!")***

***elif 80 <= score < 90:***

***print("Good job!")***

***else:***

***print("Keep improving.")***

***Boolean Operations:***

***is\_sunny = True***

***is\_warm = False***

***if is\_sunny and is\_warm:***

***print("It's a sunny and warm day.")***

***elif is\_sunny or is\_warm:***

***print("It's either sunny or warm.")***

***else:***

***print("It's neither sunny nor warm.")***

* ***Here, and, or, and not are used for Boolean operations. They combine conditions to make more complex logical expressions.***
* ***Interaction****: How will your code handle looping? (You can do one or more of a for-style loop, a while/do loop, etc.)*

*For-Style Loop:*

***# Example: Printing numbers 0 to 4***

***for i in range(5):***

***print(i)***

* ***In this example, the for loop iterates over a range of values (0 to 4) and executes the code inside the loop for each iteration.***

***While-Style Loop:***

***# Example: Counting down from 3 to 0***

***count = 3***

***while count >= 0:***

***print(count)***

***count -= 1***

* ***Here, the while loop continues as long as the condition (count >= 0) is true. It decreases the count variable (iteration).***
* ***Input****: How does your program get input from the keyboard? (Strings are easiest.)*

***# Example: Getting user input as a string***

***user\_input = input("Enter something: ")***

***print("You entered:", user\_input)***

* ***The input() function prompts user input, stores it as a string in user\_input, and prints the entered value. Note: To process as a different type, like int or float, perform type casting if needed.***
* ***Output****: What would a programmer type to put output on the screen? What sort of variables or data will your code take?*

***In the "4Cast" language, a programmer would use the print() function to display output on the screen. For example:***

***# Example: Printing a message***

***print ("Hello, 4Cast!")***

***print() function can take various types of data, including strings, numbers, and variables.***

***# Example: Printing a variable and a string***

***name = "Alice"***

***print ("Hello, " + name + "!")***

***out put will be Hello, Alice!***

* ***Functions****: [Function definition: parameters and returning types]*
  + *What will be the syntax for making a function or subroutine?*

***# Example: Function to add two numbers***

***def add\_numbers(a, b) -> int:***

***return a + b***

* ***This syntax allows programmers to define functions and subroutines in the "4Cast" language. Parameters can be adjusted based on the function's requirements, and the return type can be specified for functions that produce a result.***

* + ***How will it take parameters?***

***In the "4Cast" language, parameters are taken by specifying them within the parentheses following the function name. Here's an example:***

***# Example: Function with parameters***

***def greet\_person(name, age):***

***print("Hello, " + name + "! You are " + str(age) + " years old.")***

***# Calling the function with specific values for parameters***

***greet\_person("Alice", 25)***

* ***The greet\_person function takes two parameters, name and age.***

***When the function is called (greet\_person("Alice", 25)), actual values are provided for the parameters.***

* + ***How will it return results?***

***# Example: Function that returns the square of a number***

***def square\_number(x) -> int:***

***result = x \* x***

***return result***

* *The* ***return result*** *statement sends the result back to the caller.*

**Element 7 – Proper elements**

*[Include specific features / elements to be included in your language]*

1. **Custom Data Types:** **Create new data types specific to certain tasks or industries.**
2. **Advanced Data Structures:** **Introduce unique ways of organizing and managing data.**
3. **Special Commands or Keywords:** **Include commands that cater to specific functionalities.**
4. **Domain-Specific Commands:** **If your language is domain-specific, include elements tailored to that domain.**

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

**Comparing with C language**

**Differences**

|  |  |  |
| --- | --- | --- |
|  | **4Cast: Likely to have a syntax that prioritizes simplicity and readability, possibly resembling Python or other high-level languages.**  **C: Known for its strict syntax, closer to the hardware, and might require explicit declarations and more complex syntax.**  -------------------------------------------------------------------------------------------------------------  **4Cast: Could feature dynamic typing, allowing for flexibility in variable types.**  **C: Requires static typing, where variable types must be declared before use.** |  |

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

|  |  |  |
| --- | --- | --- |
|  | **Advantages of "4Cast" over C:**   1. **Ease of Learning:** "4Cast" is beginner-friendly with a higher-level and forgiving syntax. 2. **Productivity:** Higher-level abstractions lead to faster development and less code.   **Disadvantages of "4Cast" in Comparison with C:**   1. **Performance Overhead:** "4Cast" may have performance overhead due to high-level abstractions. 2. **Low-Level Control:** C excels in low-level control over hardware, more suitable for specific system-level tasks. |  |

**Comparing with another language**

**Language Name:**

**Differences**

|  |  |  |
| --- | --- | --- |
|  | Differences:   1. **Syntax:**    * **4Cast:** Likely to have a syntax emphasizing simplicity and readability.    * **Java:** Follows a C-based syntax, with a focus on strict typing and object-oriented principles. 2. **Abstraction Level:**    * **4Cast:** May provide higher-level abstractions, making it beginner friendly.    * **Java:** Offers a balance between high-level abstraction and control, suitable for various applications. |  |

**Advantages / Disadvantages (in comparison with this second language)**

|  |  |  |
| --- | --- | --- |
|  | Advantages of "4Cast" over Java:   1. **Ease of Learning:** "4Cast" might have a simpler syntax, making it more accessible for beginners. 2. **Productivity:** Higher-level abstractions in "4Cast" could result in faster development.   Disadvantages of "4Cast" in Comparison with Java:   1. **Ecosystem Maturity**: Java has a mature ecosystem with extensive libraries and frameworks. 2. **Performance**: Java's statically typed nature might lead to better performance in certain scenarios. |  |

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

**Advantages**

***The goal of "4Cast" is to balance simplicity and functionality, making coding enjoyable and straightforward for both beginners and experienced programmers. It's designed for ease of learning, rapid development, and aims to be versatile across different domains, fostering creativity and experimentation in a vibrant developer community.***

**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.*

***The compiler for "4Cast" includes:***

1. ***Lexical Analysis (Tokenization):***
   * ***Break code into tokens (keywords, identifiers, operators).***
   * ***Categorize tokens to create a meaningful stream.***
2. ***Syntax Analysis (Parsing):***
   * ***Define grammar rules using context-free grammar.***
   * ***Build an abstract syntax tree (AST).***
3. ***Semantic Analysis:***
   * ***Analyze meaning and context.***
   * ***Ensure variable declarations and validate data types.***
4. ***Code Generation:***
   * ***Translate AST into C code.***
   * ***Utilize ANSI C datatypes.***
5. ***Optimization:***
   * ***Optimize generated C code.***
6. ***Error Handling:***
   * ***Implement error detection and reporting.***
7. ***Console Output Handling:***
   * ***Recognize console output commands.***
   * ***Translate into C code using functions like printf.***
8. ***Loop Logic and Block Scope:***
   * ***Identify loop constructs and block scopes.***
   * ***Translate into C equivalents (e.g., for, while, {}).***

***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?*

***by recognizing a specific syntax or keyword (e.g., print). It sorts out what to write by analyzing the content of the command, distinguishing between literal text and variables***

* *What if there's some literal text (ie: "this is going to get printed") instead of variables?*

***If there's literal text (e.g., "this is going to get printed"), the compiler treats it as a string and translates it into the equivalent C code for console output, ensuring the text is displayed as intended.***

*[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?*

***In "4Cast," code blocks are marked with curly braces {} like in C and Java. For loop logic, the portion of code to loop through is controlled within these braces, providing clear scope definition.***

**FINAL SUGGESTIONS**

*Here some ideas to think about your language....*

* *Don't make this assignment harder than it needs to be on yourself. Focus on making the syntax for your language that meets our requirements. Worry about extra features later.*
* *Don’t worry if your new language winds up having really difficult parts. You'll be allowed to change your language as you go along, as long as you make "patch notes" to explain those changes. We'll tell you about this later.*
* *There's a marking key at the end of* ***CST8152\_Compilers\_F23-A11\_AnswerTemplate*** *that should steer you along for grades. Focus your efforts on where you'll get the best results.*
* *Finally, think about creating an “master-piece”: until now, you have used several languages. And if you have conditions to define yours, how it could be?*

**References**

*[Include eventual references used here]*

|  |  |
| --- | --- |
|  | * ***NOTE****: Even if you use any AI tool (ex: ChatGPT), report here, including the references used.* |

*[1] ChatGPT [ for Part 3 (****Strategy: C Implementation)*** *]*

*List of the questions that I asked CGPT about it, which are the following:*

1. *How does "4Cast" handle input from the keyboard, and how can it produce output on the screen?*
2. *Should "4Cast" closely resemble Python but with its own unique features?*
3. *Are there any specific features or elements you want to include or modify in "4Cast"?*
4. *What looping mechanisms are available in "4Cast"? How does conditional logic work?*
5. *How is the scope of variables determined in "4Cast"? How are blocks of code delineated?*

*[2] Copilot [ for Element 7 ]*

*[3]* [*https://www.sohamkamani.com/java/ecosystem/*](https://www.sohamkamani.com/java/ecosystem/)

*[4]* [*https://news.ycombinator.com/item?id=30159436*](https://news.ycombinator.com/item?id=30159436)

*[5]* [*https://www.sitepoint.com/fundamentals-of-c/*](https://www.sitepoint.com/fundamentals-of-c/)

*[6]* ***https://www.geeksforgeeks.org/difference-between-c-and-python/***

Algonquin College

Winter, 2024