UNIVERSITÉ Concordia UNIVERSITY

COMP 6411 - Comparative Study Of Programming Language Summer 2022 Term Paper

Comparing Java and Python

Authors:
1. Kshitij Yerande (Team Leader) - 40194579
2. Akshay Dhabale - 40163636
3. Mrinal Rai - 40193024
4. Farheen Jamadar - 40194668
5. Almas Saba - 40156359
6. Seung Hyun Hong - 26724701
7. Ujjawal Aggarwal - 40183962
8. Nastaran Naseri - 40215694

Table Of Contents

Abstract

Programming languages are used to control the behavior of machines by following rules for syntax and semantics. There are thousands of programming languages and many new ones are created every year. Few languages ever become sufficiently popular that they are used by more than a few people. Most programming languages are not standardized by international standards, even the popular ones. In this paper we compare java and python programming languages from various aspects like paradigm, language construct, memory management and more. We chose three problem statements: sorting, searching and concurrency based on which we will compare performance parameters of the programming languages like execution time and memory usage. Our analysis reveals that java is faster in execution than python and consumes more memory for execution than python. Both the languages support multi paradigm approach and are best suited for developing a variety of applications.

Introduction

Questions about programming languages and the properties of their programs are asked often but well-founded answers are not easily available. From an engineering viewpoint, the design of a programming language is the result of multiple tradeoffs that achieve certain desirable properties (such as speed) at the expense of others (such as simplicity). Technical aspects are, however, hardly ever the only relevant concerns when it comes to choosing a programming language. Factors as heterogeneous as a strong supporting community, similarity to other widespread languages, or availability of libraries are often instrumental in deciding a language's popularity and how it is used in the wild [1]. If we want to reliably answer questions about properties of programming languages, we have to analyze, empirically, the artifacts programmers write in those languages. Answers grounded in empirical evidence can be valuable in helping language users and designers make informed choices [2].

To control for the many factors that may affect the properties of programs, some empirical studies of programming languages [3], [4], [5], have performed controlled experiments where human subjects (typically students) in highly controlled environments solve small programming tasks in different languages. Such controlled experiments provide the most reliable data about the impact of certain programming language features such as syntax and typing, but they are also necessarily limited in scope and generalizability by the number and types of tasks solved, and by the use of novice programmers as subjects. Real-world programming also develops over far more time than that allotted for short exam-like programming assignments

In our experiment we choose three problem statements based on the following operation sorting, searching and concurrency which are discussed in the further sections. Each problem statement takes in 1 million data as input and performs the required operations and generates the relevant statistics like execution time and memory usage in a controlled environment. The first problem statement creates an array of 1 million integers and sorts them using merge and heap sort and second one creates a binary search tree and hashtable and performs searching operations and the third one compares java and python based on single threading and multithreading programs.

Apart from execution time and memory usage we also discuss the differences in programming aspects of java and python like the support of programming paradigms in both the languages and the definitions of the paradigm. We also discuss the language constructs such as data types, statement level control structures and compare the difference among them. Lexical analysis is also an important aspect along with memory management, compilation and interpretation and exception handling which play a vital role in performance of a programming language also discussed in the further sections.

Programming Language

Introduction to Java

Java was developed by James Gosling, under Sun Microsystems, and was released in May, 1995 . It was based on write once run anywhere. This means a Java program only has to be compiled once, and can be run on all platforms that support Java. Applications based on Java are first compiled to bytecode that runs on any Java virtual machine, without having the need to check compatibility with the architecture of the machine. Java compilers and Virtual Machines, including the class libraries, were originally released by Sun. The introduction of Object Oriented Programming was the second major cause (OOP),which will address the majority of C++'s issues while eliminating the time-consuming tasks that plagued earlier languages. Java became strong, platform agnostic, secure, portable, adaptable, and reusable as a result.There are several advantages of Java. It is easy to learn. Java was designed to be easy to use and is therefore easier to write, compile, debug, and learn than other programming languages. Java is object-oriented. This allows you to create modular programs and reusable code. Java is platform independent. It has the ability to move easily from one computer system to another. The disadvantages of Java can be the complexity of the code. It is slow and gives poor performance. The GUI can be improved. The lack of a backup facility also makes it difficult to use. It is memory challenging as it requires memory spaces [6].

Introduction to Python

Guido van Rossum of Centrum Wiskunde. Informatica (CWI) in the Netherlands created Python in the late 1980s as a replacement to the ABC programming language, which was influenced by SETL and was capable of handling exceptions and interacting with the Amoeba operating systemIt went into effect in December 1989. Python 3.10.4 and 3.9.12 were accelerated in 2022. Python is garbage-collected and dynamically typed. It supports a variety of programming paradigms, including structured (especially procedural) programming, object-oriented programming, and functional programming. Python is one of the most popular programming languages on the market. Python provides a lot of benefits. Python's extensive standard library provides tools for a wide range of activities and is widely regarded as one of the language's greatest assets. Many standard formats and protocols, such as MIME and HTTP, are supported for Internet-facing applications. Python has a couple significant drawbacks as well. Python takes longer to execute than C or C++. Python is not a good choice for activities that require a lot of memory. Python's memory consumption is large due to the flexibility of the data types. Python isn't a great language for developing mobile apps [7].

Comparison table of java and python

Parameter	Java	Python
Compilation	Java is compiled language	Python is interpreted language
String Operations	Offers limited string related functions	It offers lots of string related functions.
Learning Curve	Complex learning curve	Easy to learn

Speed	Java program is faster	Python program is slower than java
Multiple Inheritance	Java supports partial multiple inheritance through interfaces.	Python supports multiple inheritance
Paradigm	Java supports imperative, object oriented, functional, reflective and concurrent paradigm	Python supports imperative, functional, procedural and reflective paradigm

Programming Paradigm

Paradigm can be said as a method to perform a few tasks or compute some problems. Programming Paradigm is the way to find solutions towards the problems using any language of programming. Another definition, it's a method to find the solution of a problem with the help of tools and techniques which are accessible to us. Many Languages for programming are known, but each and every of them when implemented has to follow some way or method. There are various languages for programming but also many paradigms or patterns are also available to complete all demands. Following are the some of the paradigm of programming

Imperative - Today, imperative programming is the most widely used paradigm for building software for digital computers. At the heart of this paradigm is the view that programs are a sequence of instructions or commands that change the state of a digital computer [8,9].

Procedural - Procedural programming paradigm is based on a step-by-step execution model like in a food recipe. It presumes a set of control structures (e.g. if-then-else, for-loop, while-loop, do-until etc.) that control the order of execution of the statements or computation steps.

Object Oriented - Object-oriented programming (OOP) is a programming paradigm based on the concept of "objects", which can contain data and code: data in the form of fields (often known as attributes or properties), and code, in the form of procedures (often known as methods). A common feature of objects is that procedures (or methods) are attached to them and can access and modify the object's data fields.

Concurrent - The paradigm in which programs are processed by dividing them into multiple processors.

Declarative - The paradigms in which a programmer entirely declares the properties of the expected output except how to compute [11, 12].

Functional **-** Functional programming paradigm is based on the idea of evaluating an expression and then using the resulting value for something else. This is based on a mathematical model of function composition.

Logic **-** Logic programming paradigm is based on the idea of answering a question through search for a solution from a knowledge base. This is based on axioms, inference rules, and queries. Program execution becomes systematic search in a set of facts, making use of a set of inference rules.

Mathematical **-** The paradigm in which the desired result is declared as the answer of an optimizing question.

Reactive **-** The paradigm in which the desired result is declared with data streams and the propagation of change.

Java uses imperative as well as declarative programming like object oriented, reactive, concurrent and functional programming whereas python also uses the same with an addition of procedural programming.

Programming style	**Java**	**Python**

Procedural	**NA**	sum = 0
		
		for x in my_list:
		
		    sum += x
		
		print(sum)

Object Oriented	public class Employee {
	    private String name;
	    public String getName() {
	        return name;
	    }
	    public void setName(String name) {
	        this.name = name;
	    }
	    public static void main(String[] args) {
	    }
	}
		class ChangeList(object):
		
		    def __init__(self, any_list):
		
		        self.any_list = any_list
		
		    def do_add(self):
		
		        self.sum = sum(self.any_list)
		
		create_sum = ChangeList(my_list)
		
		create_sum.do_add()
		
		print(create_sum.sum)

Functional	Function<Double, Function<Double, Double>> weight = mass -> gravity -> mass * gravity;
	
	Function<Double, Double> weightOnEarth = weight.apply(9.81);
	logger.log(Level.INFO, "My weight on Earth: " + weightOnEarth.apply(60.0));
	
	Function<Double, Double> weightOnMars = weight.apply(3.75);
	logger.log(Level.INFO, "My weight on Mars: " + weightOnMars.apply(60.0));
		square = lambda x: x**2
		
		double = lambda x: x + x
		
		print(list(map(square, my_list)))
		
		print(list(map(double, my_list)))

Lexical Analysis

The process of transforming a sequence of characters into a sequence of lexical tokens(strings with an assigned and thus identified meaning) is known as lexical analysis/lexing/tokenizations.
Lexeme: A *lexeme* is a sequence of characters in the source program that matches the pattern for a token and is identified by the lexical analyzer as an instance of that token.
Some authors term this a "token", using "token" interchangeably to represent the string being tokenized, and the token data structure resulting from putting this string through the tokenization process [24].
Token: A *lexical token* or simply *token* is a string with an assigned and thus identified meaning. It is structured as a pair consisting of a *token name* and an optional *token value*. The token name is a category of lexical units. Common token names are

- identifier: names the programmer chooses;
- keyword: names already in the programming language;
- separator (also known as punctuators): punctuation characters and paired-delimiters;
- operator: symbols that operate on arguments and produce results;
- literal: numeric, logical, textual, reference literals;
- comment: line, block (Depends on the compiler if compiler implements comments as tokens otherwise it will be stripped).

Lexical Grammar

The specification of a programming language often includes a set of rules, the lexical grammar, which defines the lexical syntax. The lexical syntax is usually a regular language, with the grammar rules consisting of regular expressions; they define the set of possible character sequences (lexemes) of a token. A lexer recognizes strings, and for each kind of string found the lexical program takes an action, most simply producing a token.

Tokenization
It is the process of demarcating and possibly classifying sections of a string of input characters. The resulting tokens are then passed on to some other form of processing. The process can be considered a sub-task of parsing input.

For example, in the text string:

The quick brown fox jumps over the lazy dog

the string isn't implicitly segmented on spaces, as a natural language speaker would do. The raw input, the 43 characters, must be explicitly split into the 9 tokens with a given space delimiter (i.e., matching the string " " or regular expression /\s{1}/).

Scanner
The first stage, the *scanner*, is usually based on a finite-state machine (FSM). It has encoded within it information on the possible sequences of characters that can be contained within any of the tokens it handles (individual instances of these character sequences are termed lexemes). For example, an *integer* lexeme may contain any sequence of numerical digit characters.

Evaluator

In order to construct a token, the lexical analyzer needs a second stage, the *evaluator*, which goes over the characters of the lexeme to produce a *value*. The lexeme's type combined with its value is what properly constitutes a token, which can be given to a parser. Some tokens such as parentheses do not really have values, and so the evaluator function for these can return nothing: only the type is needed.

Obstacles

Typically, tokenization occurs at the word level. However, it is sometimes difficult to define what is meant by a "word". Often a tokenizer relies on simple heuristics, for example:

- Punctuation and whitespace may or may not be included in the resulting list of tokens.
- All contiguous strings of alphabetic characters are part of one token; likewise with numbers.
- Tokens are separated by whitespace characters, such as a space or line break, or by punctuation characters.

Java compiler and python interpreter performs lexical analysis where in the parser parses the code to identify line structure, comments, blank lines, literals and operators and convert the source code into byte code in case of java and in case of python the parsed code is executed by interpreter during runtime. The lexical parser uses the above features to parse the code.

Type Checking

One of the significant differences between Java and Python is how each uses variables since it impacts how the developer writes, designs, and troubleshoots programs. Python variables are dynamically typed, which offers lots of string-related functions, whereas Java variables are statically typed with limited string-related functions.

In Python, type checking is deferred until runtime. There's no need to declare a variable name and type before using the variable in an assignment statement. Python container objects, like lists and dictionaries, can contain objects of any type. Type of objects retrieved from the container objects are remembered, so typecasting is unnecessary. And thanks to duck typing, function arguments can be used to pass in any object whose interface supports the operations required by the function, which is a lot like using real-world objects. An object doesn't have to "be" a particular type; it just has to be used where a thing of that type might be. This can lead to surprises, but it's a more accurate reflection of the categorical fluidity of human thought than is the rigid hierarchy imposed by more restrictive type systems.[20]

Java variables are type-checked during compile time, unlike Python. A Java variable name and its type need to be declared before using the variable in an assignment statement. If an object is assigned to a variable that is not the same type, then a type exception will occur. This means many type errors that would result in a runtime error–and often a program crash–in Python get caught at compile time in Java. In addition, Java container objects can only accommodate generic type objects. The type of object retrieved from the container is unremembered, so typecasting is necessary.

Expression and assignment statements

An expression is a series of variables, operators, and method calls (constructed according to the syntax of the language) that evaluates a single value. Expressions are used to compute and assign values to variables and to help control the execution flow of a program. In any programming language, an expression is evaluated as per the precedence of its operators. Operators that have higher precedence get evaluated first. There are different types of expression such as Constant, Integral, Floating, Bitwise, Relational, Logical, and Pointer expressions. Logical Expressions are differently expressed in Java and Python. For example, Logical Operator and is represented as "and" in python where as in Java we have "&&" Similarly for or Operator we use "or" in python and in Java "||". Expressions can be ambiguous both in Java and Python to avoid running into such issues, using parentheses with operators is recommended in both languages. In Java expression returning value should be assigned to the appropriate variable with the correct data type otherwise the program will throw an error however in python return value can be assigned to any variable.

An **assignment statement** gives a value to a variable For example,

x = 5;

gives x the value 5.

In Java in order to execute the above assignment statement, the variable x must be declared, however in Python, the above statement can be executed directly, and the data type of x will be determined at run time by an interpreter.

In Java, an assignment statement is an expression that evaluates a value, which is assigned to the variable on the left side of the assignment operator. Whereas an assignment expression is the same, except it does not take into account the variable.

We use **Python assignment statements** to assign objects to names. The target of an assignment statement is written on the left side of the equal sign (=), and the object on the right can be an arbitrary expression that computes an object.

There are some important properties of assignment in Python :-

- Assignment creates object references instead of copying the objects.
- Python creates a variable name the first time when they are assigned a value.
- Names must be assigned before being referenced.
- There are some operations that perform assignments implicitly.

Python supports various types of assignment statements such as Basic, Tuple, List, Sequence, Multiple Target, and Augmented.

Statement level control structures

A control structure is made up of a control statement and the statements it governs. There are two types of Control Structures, namely, Selection Statements and Iterative Statements [27]. Selection structures allow you to choose between two or more execution paths. They could be further categorized into Two-way selection and Multiple-way selection. A two-way selector has the following syntax in Python and Java:

General Form	Python	Java

if control_expression
then clause
else clause
	if control_expression:
	    statement/statements
	elif clause:
	    statement/statements
	elif clause:
	    statement/statements
	….
	else:
	    statement/statements
		if(control_expression){
		    //statement/statements
		}
		else if(clause){
		    //statement/statements
		}
		else if(clause){
		    //statement/statements
		}
		...
		else{
		    //statement/statements
		}

In Java, only Boolean expressions can be used as the control expressions and consists of curly braces whereas indentation is used in Python to specify clauses also, rather than using an opening curly brace, a colon is used to introduce the "then" clause. In case of multiple-way selectors, one of any number of statements or statement groups can be chosen. E.g. Switch statement.

Iterative statements, often called a loop, cause a statement or set of statements to be executed zero, one, or multiple times. Iteration or recursion are two methods for repeating the execution of a statement or compound expression. The counter-controller loops e.g. for loop in Java allows you to create a loop that breaks when a condition is fulfilled. In Python, however, the for loop allows you to loop over an iterator rather than a condition.

General Form	Python	Java

for i = first to last do
    statement/statements
	for loop_variable in object:
	    statement/statements
		for (initialization condition;
		required condition;
		increment/decrement)
		{
		    statement/statements }

Compilation and Interpretation

Although Java is a compiled language and Python is an interpreted language, this does not mean that Java does not have an interpretation step or Python does not have a compilation step. Both are cross-platform languages since they compile bytecode and run it on virtual machines. However, compared to Java, Python is less portable.

Java has a separate step of compilation before actual execution that converts the source code into bytecode; .java code (source code) is compiled/ converted into a .class file. If the program defines more than one class, there will be a bytecode file for each class. This bytecode is platform independent since it does not have machine-level details specific to the platform. Also, java compilers compile the code all at once, so if there are any compilation errors, all will be reported at once.

At the execution time, this bytecode is converted into machine code by JVM, so any computer or mobile device that can run the Java virtual machine can run a Java application. JVM itself is platform-specific and performs the below tasks:

- Class Loader: Class loader component of JVM loads the bytecode file of the class having the main method, and other class files will be called as in when they are used in the main method.
- Bytecode Verifier: This component checks the bytecode to avoid any runtime errors.
- JIT: The JIT compiler is an integral part of the JVM enabled by default and is activated when a Java method is called. The JIT compiler compiles the bytecodes of that method into native machine code, compiling it "just in time" to run. When a method has been compiled, the JVM calls the compiled code of that method directly instead of interpreting it.
- Execution of machine code: The machine code is executed, and the output is displayed. Execution of compiled machine language code is very fast.

In Python, the code is compiled, but it is not performed as a separate step and is hidden from the developer that it appears the source code is directly executed, which is not true. Python source code, .py file, is compiled during run time/execution into bytecode, .pyc file, but not as a separate step and not converted into machine language. Hence, the programs need an interpreter installed on the target machine to translate Python code.

At the time of execution, Python source code (.py file) is sent directly to PVM. Python virtual machine performs these actions:

- Code Verification: Code is first checked for errors.
- Line by line compilation: If there are no code errors, then the code is compiled into bytecode which is nothing but HelloWorld.pyc file. This compilation happens line by line against full code at once in Java. Hence in Python, as soon as the compilation step encounters the first error, it will be reported, and compilation will fail.
- Execution of bytecode: Once the code is compiled, PVM executes the bytecode (.pyc file) and the output is displayed. Since bytecode is directly executed, it is comparatively slower than Java.[22, 23]

Exception Handling

Exception (Exceptional Event) in a program is one type of an error in which an unwanted or unexpected situation occurs during the execution of a program. When an exception occurs, the program is interrupted and terminated unexpectedly without any state saved. To handle these exceptions automatically, some programming languages offer the exception handling process[16]. When an exception occurs, the exception handler is initiated, and it saves the state of execution and is able to bypass the normal flow of the program rather than terminating the program unexpectedly[16]. Depending on the type of an exception, the exception handler could fix or handle the error automatically and continue to execute the rest of the program with saved state data [17]. Java and python both have in-built support for exception handling, and they have similar (yet different) syntax. As presented in the table below, Java uses try-catch-(optional)finally blocks, and Python uses try-except-(optional)finally or try-finally or try-except-else blocks. Moreover, the handling methods are differ - in Java, an exception is thrown from any method or in any piece of the code, but in Python, exception is not thrown, but raised [18].

JAVA	**Python**
try{	try:
// some tasks that may throw an exception	# some tasks that may raise an exception
}catch(ExceptionType messageVariable){	except exception, value:
// exception handling	# exception handling
}	else:
// optional	# perform some tasks that should be performed if no exceptions are raise
finally{	# optional
// code that must always be invoked	finally:
}	# code that must always be invoked

For Java, exception handling is based on C++, so it is based on an object-oriented language paradigm, and has improved handling, compared to C++. Also, Java has a rich collection of libraries which are built for exceptions, raised by the Java Virtual Machine[16].

In Java, the exceptions are objects that are inherited from the Throwable class. There are two categories of exceptions – (i) the exceptions where a concern for the user program and a compiler, and (ii) unchecked exceptions which can occur anywhere, and are often ignored by the user program[17]. In Java, only the instances of a throwable object or some class that descends from a throwable class can be thrown. Therefore, in Java the method that does not include a throw clause never throws an exception[18]. Hence, the Java compiler checks that all exceptions that a method can throw are listed in its throws clause. The finally clause is a code block that will be executed regardless of how the throw clause is executed. One benefit of Java exception handling is that a programmer can create his/her own exception classes, and user-defined exceptions become subclasses of Exceptions [16]. However, the programmer cannot disable exception handlers (i.e., not possible to disable exceptions).

Exception handling in Python is very similar to that of Java. Some tasks that could raise exceptions are embedded in a try block. Java exceptions are caught by catch clauses. The except keyword in Python makes it possible to define custom exceptions. A try statement may have multiple except clauses for different exceptions, but at most one except clause will be executed. Python also has a finally clause like finally clause in Java which is always executed regardless of how the try block is executed [19].

Memory Management

Java has a powerful memory management system as it can handle runtime faults, has automatic garbage collection and exception handling, and does not employ the explicit pointer idea. In Python, efficiency comes from the fact that it is compiled line by line, which simplifies debugging and makes memory management much more efficient.

Java is sometimes criticized for its excessive memory utilization, which can lead to memory leaks or poorly designed code. The Java garbage collector deletes objects that your application no longer references and clears the memory so that your application may utilize it again. This means that the more objects you have in Java, the more memory your application uses.

Python, unlike Java, uses reference counting to manage objects. The Garbage Collector is used by the Python Runtime Environment to manage memory allocation and deallocation [26]. When a new object is created, the GC allocates the necessary memory, and when the object is no longer in scope, the GC does not immediately release the memory; instead, it becomes eligible for trash collection, which may ultimately free it. This means that the memory management keeps track of how many times each object in the programme has been referenced. When an item is no longer in use, the garbage collector (a component of the memory manager) frees the memory associated with that object.

Consider an example to comprehend this difference between java and python.

When declaring a variable in Java, a data type is required e.g. float x = 1.1 whereas in python, the declaration is x = 1.1. Python says that a 1.1 object is generated first, and then the variable x is marked to the x.

When an object has no references, Python utilizes reference counting to liberate it. This means that most of the time, reference counting cleans everything up and GC never kicks in. In contrast to Java, there is frequently no need to fine-tune various GC parameters.

Concurrency

Concurrency allows distinct sections of your application to run at the same time, independently. This is usually accomplished by multithreading, which makes use of the operating system's multitasking capabilities to run many tasks on the same or distinct CPU cores.

In Python, multithreaded programmes are commonly implemented using the threading module. This module provides an API for creating and managing threads that is simple to use. The software should process as much data as feasible for high-performance tasks. Unfortunately, a feature known as the Global Interpreter Lock (GIL) prevents Python code from running on several threads at the same time in CPython, the official interpreter for the Python language [28]. In other words, even if our computer has 64 CPU cores, allowing numerous jobs to run in parallel at the same time, a Multithreaded Python application will not be able to take advantage of them and will run as quickly (or as sluggish) as if it were running on a single-core computer.

Although there is still a benefit to employing multithreading with Python, it is confined to workloads that require simple multitasking, such as waiting for external resources (such as the internet). Another built-in module called multiprocessing may be the best solution for Python scripts that require high throughput and genuine parallelism. To get around the GIL's constraints, you can use this module to create multiple instances of the Python interpreter as distinct processes.

The Java programming language, like Python, has an easy-to-use multithreading API. Java is well-known for its excellent stability and concurrency. When compared to Python, Java has more concurrency. The JVM, unlike Python, is not bound by the Global Interpreter Lock (GIL). All major Java Virtual Machine implementations, including Oracle's Hotspot, offer full parallel execution of many threads out of the box, with a straightforward and easy-to-use API.

Application of Java and Python

Java has become the most robust programming language because of its amazing features. Some of its features are platform independence, high performance, Object orientation, support automatic garbage management, and many more.

- **Desktop GUI** : Desktop applications can be flawlessly designed using Java. APIs like Swing, AWT, JavaFX provide a modern way to develop GUI applications.
- **Mobile apps :** Java is a cross-platform framework that is used to build applications that run across smartphones and other small screen devices.
- **Web applications :** Java is just perfect for developing web applications because of its ability to interact with a large number of systems.

- **Gaming applications :** Java has proved to be the most desirable option for game development because of the presence of a wide variety of open-source frameworks.
- **Business applications :** Java helps us to develop robust applications for business requirements. It can be used to develop from small-scale applications to big enterprise solutions. The language is constantly growing and updating to fulfill the latest business demands.
- **Embedded systems :** It refers to the combination of small units that combine to perform the collective function for larger systems. Java has proved to be the best solution to address increasing Software complexity.
- **Cloud applications:** Cloud computing refers to the on-demand access to computer resources without direct management by the user. Java has carved its way into cloud applications. It provides a solution for IT infrastructure at an affordable cost.

Python's simplicity and readability enables quick development of complex software, machine learning, and data analytics applications. But beyond its innate simplicity and versatility, what makes Python stand out are its vast assortments of libraries and packages that can cater to a wide range of development as well as Data Science requirements.

- **Web Development:** Python offers numerous options for web development like Django, Pyramid, Flask, and Bottle for developing web frameworks and even advanced content management systems which are packed with standard libraries and modules which simplify tasks like content management, database interaction, and interfacing with internet protocols
- **Game Development:** Python comes loaded with many useful extensions (libraries) that come in handy for the development of interactive games. For instance, libraries like PySoy (a 3D game engine that supports Python 3) and PyGame are two Python-based libraries used widely for game development.
- **Artificial Intelligence and Machine Learning:** Python's simplicity, consistency, platform independence, great collection of resourceful libraries, and an active community make it the perfect tool for developing AI and ML applications.
- **Enterprise-level/Business Applications:** This is where Python can make a significant difference. Python high performance, scalability, flexibility, and readability are just the features required for developing fully-functional and efficient business applications.

Problem Statement

Problem statement 1 - sorting

The objective of this assignment is to compare Java and Python for sorting criteria using approximate 1 million data, we first implement the Heap Sort and Merge Sort and then we measure several key statistics, including the memory usage of data structure, execution speed of the algorithms, and execution time for search elements in each data structure.

Merge Sort

Merge Sort was invented in 1945 by John von Neumann, and it employs the divide and conquer algorithm[13]. Merge Sort will recursively divide a given list of elements into half until there is a single element. Since the list has a single element considered sorted, merge steps will follow. The sub-lists will merge until there is only one sorted final list. To merging two sub-lists, it creates new array list with size of sum of sub-lists, then start comparing each of first elements in the sub-lists and place smaller element into newly created array list until all of elements in two sublists are placed in new array list[14].

The merge sort is ideal for larger lists of elements, and it is a stable sorting algorithm. However, the merge sort takes more memory space to store its sub-lists and is slightly slower than other sorting algorithms for smaller data sets[14].

Pseudo Code:

```
1. Declare left and right var which will mark the extreme indices of the array
2. Left will be assigned to 0 and right will be assigned to n-1
3. Find mid = (left+right)/2
4. Call mergeSort on (left,mid) and (mid+1,rear)
5. Above will continue till left<right
6. Then we will call merge on the 2 subproblems
```

Heap Sort
It is a sorting algorithm that uses the Binary Heap data structure to compare items. The Heapsort algorithm involves preparing the list by first turning it into a max heap. The algorithm then repeatedly swaps the first value of the list with the last value, decreasing the range of values considered up operation by one, and sifting the new first value into its position in the heap [15].

Pseudo Code:

```
    1. We start by using Heapify to build a max heap of elements present in an array A.
    2. Once the heap is ready, the largest element will be present in the root node of the heap.
    3. Now swap the element at with the last element of the array, and heapify the max heap
       excluding the last element.
    4. Repeat steps 2 and 3 till all the elements in the array are sorted.
```

As the number of objects to sort grows, the time required to conduct Heap sort grows logarithmically, whereas alternative methods may grow exponentially slower. This sorting method is really quick. Memory usage is modest because it requires no additional memory space to work other than what is required to keep the initial list of objects to be sorted. Because it does not involve difficult computer science concepts like recursion, it is easier to understand than other equally efficient sorting algorithms .When working with very complicated data, heap sort is thought to be insecure, expensive, and inefficient.

Problem statement 2 - searching

To evaluate two programming languages, Java and Python, for searching criteria by using approximately 1 million data. We first employ a binary search tree and a hashing table, and then we measure several key statistics, including the memory usage of data structure, execution speed of the algorithms, and execution time for search elements in each data structure.

Hash Table:
Hash table is one of efficient data structures that can map key to value. Tables could be conceptualized as a bucket array where each unit of array has a bucket and an element could enter into a specific index by the hash function [21]. We use 600011 which is the prime number as our table size. And for the hash function, we use the division method. For new elements, it is divided with table size, and use remainder as an index in the table [21]. We use a separate chaining method to insert the element with linked list in the bucket, therefore there is no collision.

Pseudo Code for searching element with hash table:

1. Find the index with hash function (key MOD table size)
2. Take linked list in the index
3. Iterate it until find the matching or reach end of list

Binary Search Tree:

Binary search trees are a popular data structure for storing entries of a map. We implement a binary search tree that every key that has the left subtree of an element should be less than the key and the right subtree of the element should be greater than the key [21].

Pseudo Code for searching element with binary search tree:

1. Check if root in null, then return false
2. Check if root.key is equal to search key then return true
3. If data < root value, go to left tree, and make recursive call with root.left and key
4. If data > root value, go to right tree, and make recursive call with root.right and key

Problem statement 3 - concurrency

The objective of this assignment is to evaluate two distinct concurrent accesses using Java and Python for sorting using merge-sorting technique for four different distinct linked lists. The input file contained nearly 1 million dataset and using this input files 4 different linked lists were generated based on below criteria

- The first list contained the even and prime numbers
- The second list contained the even and not prime numbers
- The third list contained odd and prime numbers
- The fourth list contained odd and not prime numbers

The given lists were sorted using merge-sort and two programs used single-threading in Java and Python; and other two programs used multithreading in Java and Python as well. Key characteristics such as reading and linked list creation time, iterations and time taken by individual threads for sorting, total iterations and time taken for sorting and total time taken for memory used for sorting was derived.

Single-threading:

Singe threading focuses on reducing the latency of single applications or threads as it optimizes the performance of processor's single-threadedness. It's approach is to decrease the number of times individual applications are executed while running on a single core by designing and implementing a more powerful core [25]. The processes in single thread contains the execution of instructions in a single sequence.

Pseudo code for sorting linked lists using single-threading:

1. Reading the input file
2. Generating four distinct linked lists with above-described set of properties
3. Apply merge-sort on them using single-thread of the processor
4. Write the sorted output of each linked-list to an external file
5. Derive key characteristics of the program

```
6. Write the characteristics to an external file
```

Multi-threading:
Multi-threading reduces the latency for a group of applications by increasing the efficiency of the processor's multi-threaded capability. Its approach is to increase the efficiency of the processor's performance by running a large number of applications or threads of an individual application on a correspondingly larger number of cores. These cores are simpler than the single-threaded cores [25]. Multi-threaded processes enable execution of multiple partitions of the program at the same time. These are lighter processes present within a process.

Pseudo code for sorting linked lists using multi-threading:

```
1. Reading the input file
2. Generating four distinct linked lists with above-described set of properties
3. Apply merge-sort for linked lists on each individual thread.
    a. Executed using Thread class in Java
    b. Executed using threading.Thread in Python
4. Write the sorted output of each linked-list to an external file
5. Derive key characteristics of the program for each thread
6. Write the characteristics to an external file
```

Experimental analysis

Sorting

Analyzing the data generated from the experimentation of problem statement 1 we get the following results after sorting 1 million records using merge and heap sort.

Table for comparison of performance for sorting algorithm in java and python

Sorting Algorithm (1 million records)	Programming Language	Execution Time (in seconds)	Memory Usage (in MB)	Reading Time in seconds (1 million Records)	Writing Time in seconds (1 million records)
Merge Sort	Python	7.79	15.99	0.2513	0.40404
Merge Sort	Java	0.468	28	1.8234436	0.1950
Heap Sort	Python	10.09	0.010	0.239096	0.120798
Heap Sort	Java	1.324	20	1.6102163	0.3031682

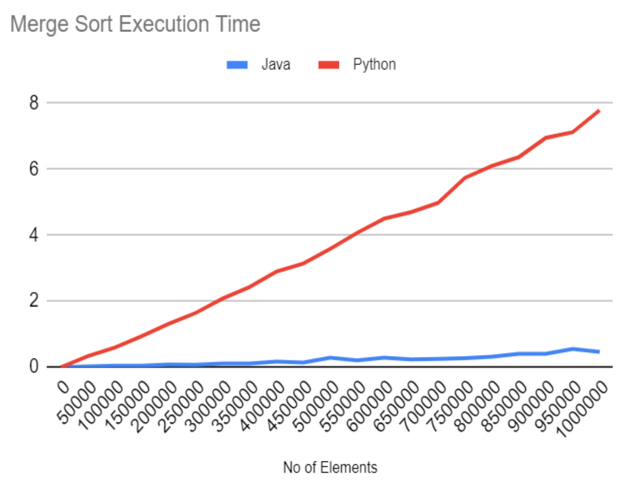
Table for comparison for memory usage in Java and Python

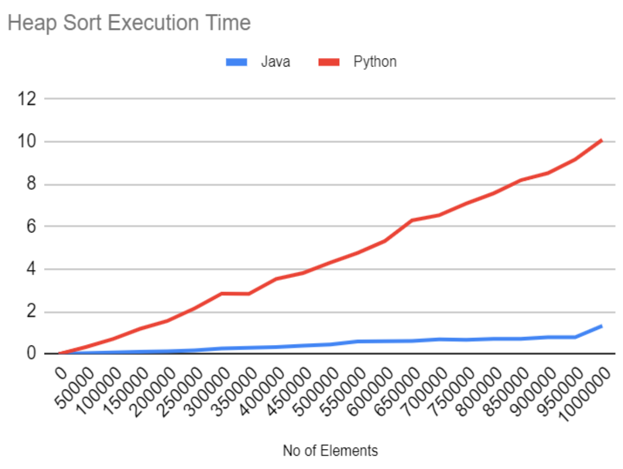
Sorting Algorithm	Programming Language	Memory Usage(in MB)
Merge Sort	Java	28

Merge Sort	Python	15.99
Heap Sort	Java	20
Heap Sort	Python	0.010173

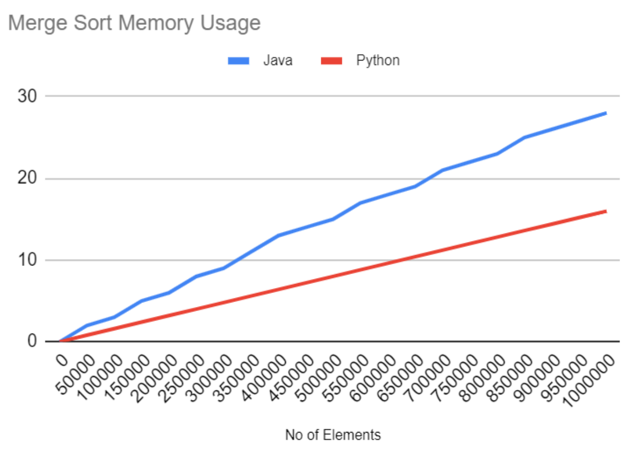
Merge sort and heap sort both are optimized algorithms which run in O(nlogn) time however merge sort requires additional space O(n) to sort data making it less efficient in terms of space. From the above data and graphs we can infer that java takes more time to execute than python and java programs take more memory to execute than python. The file reading time of python is faster as compared to java because python program file operation implementation in program is faster as it reads the whole data in list and converts the list to integer list from string. However the java program uses buffered reader to read the file which makes the program bit slower as the buffered reader keeps caching the records and writing them at intervals.

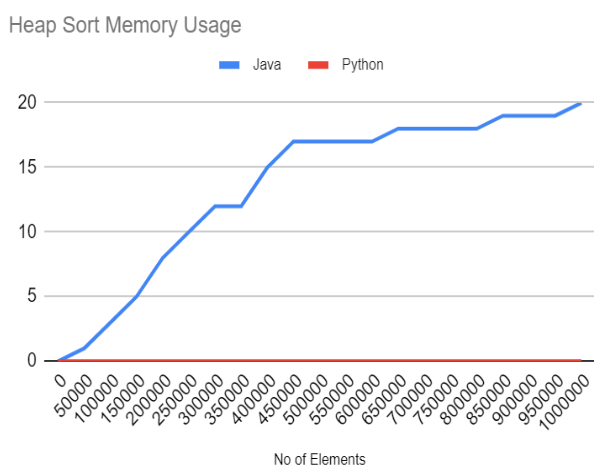
Graphical Representation for Execution Time with varying Data size





Graphical Representation for Memory Usage with varying Data size





Since python is a dynamically typed language and it requires to continuously check for data type during runtime, this operation makes the python program slower as compared to java program which is statically styled compiled language and converts the source code to bytecode and JVM runs the bytecode is an optimized way which makes the java code highly performant. The graphs represent the time and space complexity of merge sort and heap sort algorithms which are O(nlogn), O(n) and O(nlogn),O(1) respectively. The time graph depicts the nlogn pattern and memory graph also closely represents the O(n) and O(1) pattern for merge sort and heap sort.

Searching

Analyzing the results of problem statement 2 where we populate binary tree and hash table with approximate 1 million records and perform and searching operation.

Execution Time and Memory Usage

Below table presents the result of execution time for populating binary search tree and hash table of input size of approx. 1 million.

Data Structure	Programming Language	Creation Time (in milliseconds)	Memory Usage (in MB)
Binary Search Tree	Java	515.9653	249
Binary Search Tree	Python	18453.552	247.1559
Hash Table	Java	142.7769	361
Hash Table	Python	4761.1989	247.0432

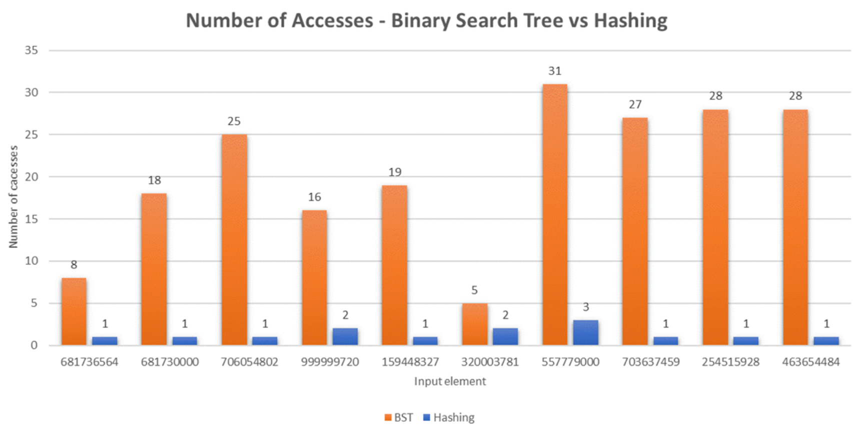
Searching Time Results

Data for Execution time for searching in binary search tree and hash table

Data Structure	Programming Language	Average Successful Search time (in millisecond)	Average Unsuccessful Search time (in millisecond)
Binary Search Tree	Java	0.00298	460.0
Binary Search Tree	Python	0.70744	0.65088
Hash Table	Java	0.00091	0.000839
Hash Table	Python	0.00259	0.00277

As the table shows, Java programs are faster than Python. This is because Java uses a Just-In-Time(JIT) compiler. It compiles bytecode into native machine code beforehand to run. While Python uses an interpreter at runtime. And the data type is also decided dynamically in runtime which increases the workload into interpreters making python programs slower compared to java. Alo python uses counting while managing large numbers of objects which makes python more memory efficient however it makes the program slower whereas java uses JVM which dynamically performs code optimization while runtime and compile time making java programs faster in execution. Java programs take more memory because while handling a large number of objects it creates an object graph to keep track of used and unused objects which requires additional memory space therefore making java programs less memory efficient.

Number of Access for Searching

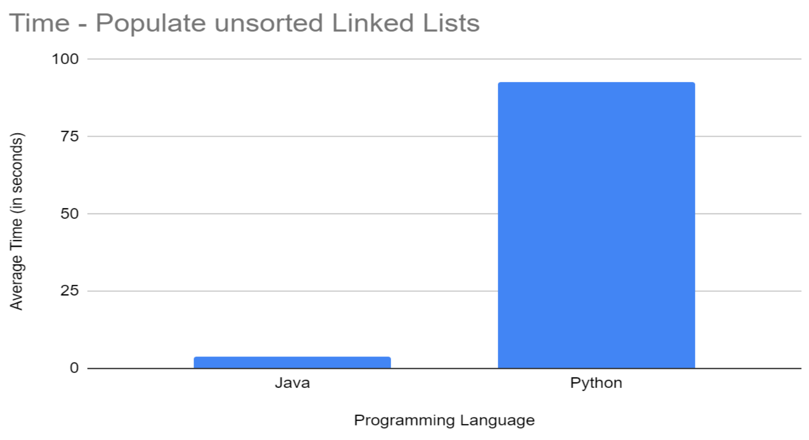


Binary Search tree requires more number of accesses to search a element in comparison to hash table because binary search tree starts search from the root node and traverses to left subtree or right subtree making the number of accesses more in case the depth of the tree is more. Hash table uses a hash function which makes the search in O(1) time and therefore making it a highly performant searching technique. However this comes at the cost of space where each element is stored at a particular index and also has some unused space to store more elements.

Concurrency

We analyze the results of sorting 4 singly linked lists in a single threaded and multithreaded environment.

Graphical Representation for populating 4 singly linked lists



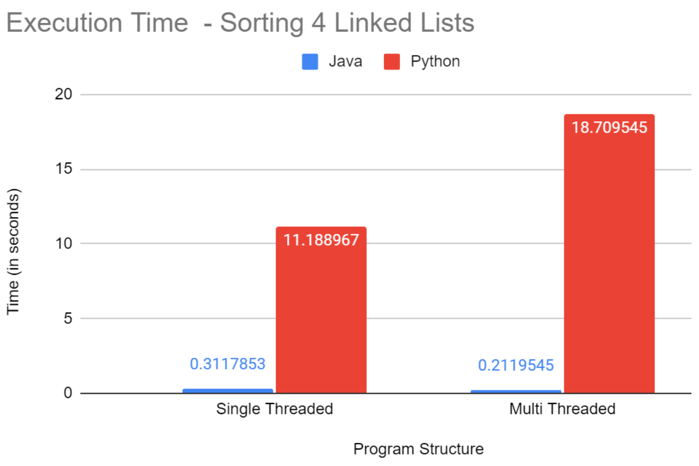
Python program takes more time than java to populate linked lists as compared to java because python has to maintain the list of objects using reference counting and there are methods in the code to which compute whether a given number is prime of even and as python is a dynamically typed language it requires to check the data type every time it calls the method. While working on large data this operation makes the python program slower compared to java where java uses Scanner class to read the file which is faster as compared to python. Python reads all the data in string which explicitly needs to be converted into required data type making the python program perform redundant work.
Following are sizes of lists after population:

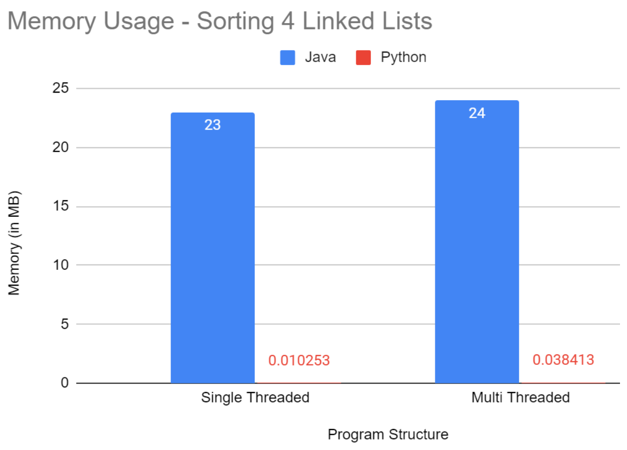
- List 1 : 1
- List 2: 499644
- List 3: 49958
- List 4: 449074

Time taken to sort each linked list (in seconds)

Program Structure	List 1	List 2	List 3	List 4	Total
Single Thread Java	1.13E-5	0.1682053	0.0109995	0.1325692	0.3117853
Single Thread Python	0.000006	5.729666	0.461928	4.997336	11.188967
Multi-Thread Java	8.1E-6	0.2112045	0.0199829	0.1785911	0.2119545
Multi-Thread Python	0.000147	18.707961	3.573447	17.800157	18.709545

Graphical Representation for execution time and memory usage for sorting

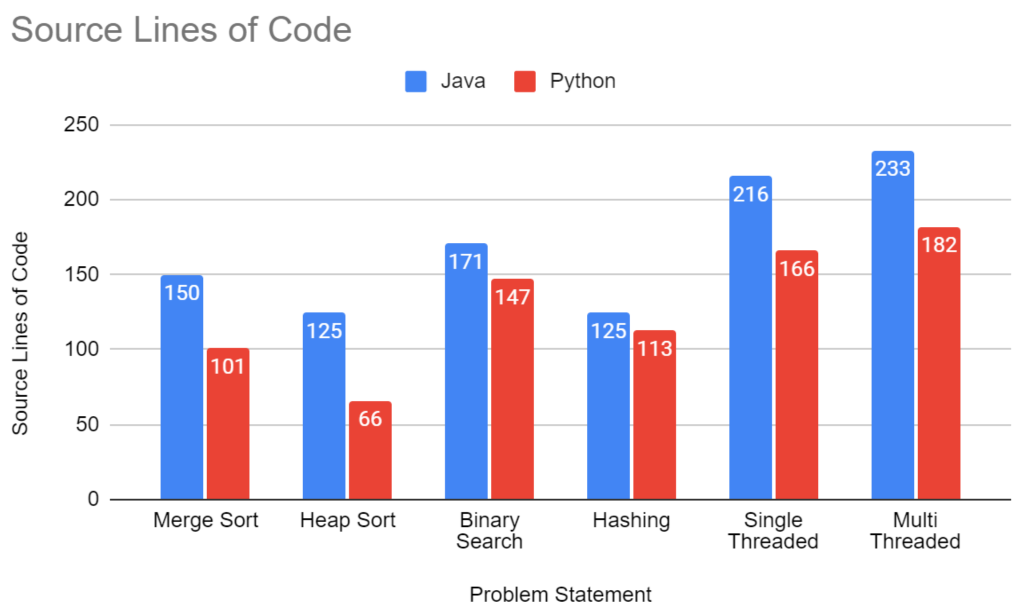




Looking at the graphical representation and above table we can see that java programs have sorted linked lists quickly as compared to python programs due to JVM which handles concurrency with optimization. Python on the other hand is slow in execution as threading is not very well supported in python as a python interpreter does not support concurrency to the extent to which java supports. Single threaded java is faster than single threaded python and same goes for multithreading programs. Java multithreading is faster than a Java single threading program.

However python multithreading is slower than single threaded program because multithreading in python programs is CPU bound meaning that python threads run on single core unless explicitly programmed for multi core. Python implements the global interpreter which allows a single thread to access memory location at a time which defeats the purpose of parallelism and there are overheads in managing the threads and python multithreading program becomes slower. In order to run threads in true parallelism, implement multiprocessing and execute threads on different cores.

Evaluation on lines of code



We can see that in all the problem statements from sorting, searching and concurrency the source lines of code for java programs are more than python programs. This is due to the concise nature of python code as python was developed with the intention to make programming easier therefore python program as very developer friendly.

Conclusion

Programming languages are essential tools for the working computer scientist, and it is no surprise that what is the “right tool for the job” can be the subject of intense debates. To put such debates on strong foundations, we must understand how features of different languages relate to each other. Our study revealed differences between java and python in terms of conciseness and performance. We can conclude that java is better for sorting, searching and concurrency in terms of python while working with large data however if memory is a constraint then python is suitable for such applications. Java has performed very much better than python in performance however from development productivity python is better than java.

The study had three problem statements which are less to compare suitability of programming languages therefore a wider study is required in future which captures various aspects of programming language in the experimentation to gather large samples so that better comparison can be achieved.

References

[1] L. A. Meyerovich and A. S. Rabkin, “Empirical analysis of programming language adoption,” in Proceedings of the 2013 ACM SIGPLAN International Conference on Object Oriented Programming Systems Languages & Applications, ser. OOPSLA ’13. New York, NY, USA: ACM, 2013, pp. 1–18.

[2] S. Nanz and C. A. Furia, "A Comparative Study of Programming Languages in Rosetta Code," 2015 IEEE/ACM 37th IEEE International Conference on Software Engineering, 2015, pp. 778-788, doi: 10.1109/ICSE.2015.90.

[3] S. Hanenberg, “An experiment about static and dynamic type systems: Doubts about the positive impact of static type systems on development time,” in Proceedings of the ACM International Conference onObject Oriented Programming Systems Languages and Applications, ser. OOPSLA ’10. New York, NY, USA: ACM, 2010, pp. 22–35.

[4] S. Nanz, S. West, K. Soares da Silveira, and B. Meyer, “Benchmarking usability and performance of multicore languages,” in Proceedings of the 7th ACM-IEEE International Symposium on Empirical Software Engineering and Measurement, ser. ESEM ’13. Washington, DC, USA: IEEE Computer Society, 2013, pp. 183–192.

[5] L. Prechelt, “An empirical comparison of seven programming languages,” IEEE Computer, vol. 33, no. 10, pp. 23–29, Oct. 2000.

[6] Ikedilo, Obiora & Osisikankwu, Peter & Madubuike, Chibuike. (2015). International Journal of Research A Critical Evaluation of Java as a Good Choice for Introductory Course. International Journal of Research. 02 Issue 12. 847-853.

[7] Gowrishankar, S. & Veena, A.. (2018). Introduction to Python Programming. 10.1201 /9781351013239.

[8] Maurizio Gabbrielli, Simone Martini , “Programming Language Principles and Paradigm” Available: https://link.springer.com/content/pdf/10.1007/978-1-84882-914-5.pdf

[9] Zainalabedin Navabi, David R. Kaeli, Computer Science and Engineering, chapter 13: Imperative programming Available: http://www.eolss.net/Sample-chapters/C15/E6-45-05-02.pdf

[10] Bartoníček, Jan. (2014). Programming Language Paradigms & The Main Principles of Object-Oriented Programming. CRIS - Bulletin of the Center for Research and Interdisciplinary Study. 2014. 10.2478/cris-2014-0006.

[11] Salvaneschi, Guido & Amann, Sven & Proksch, Sebastian & Mezini, Mira. (2014). An empirical study on program comprehension with reactive programming. 564-575. 10.1145/2635868.2635895.

[12] Jana, Debasish. (2006). The gamut of programming paradigms. CSI Communications. 57-60.

[13] “Merge sort,” Wikipedia, 04-May-2022. [Online]. Available: https://en.wikipedia.org/ wiki/ Merge_sort. [Accessed: 12-May-2022].

[14] Soileau, Gina, et al. "An Analysis of the Pragmatic Differences in How the Insertion Sort, Merge Sort, and Quicksort Algorithms Comparison Sort Numbers Stored in the Common Java-language Array Data Structure." (2008).

[15] “HeapSort”. Available: https://en.wikipedia.org/wiki/Heapsort

[16] Robert W. Sebesta, Concepts of Programming Languages, Addison-Wesley, Twelfth Edition, 2019.

[17] “Exception Handling in Java,” Dot Net Tutorials. [Online]. Available: https://dotnettutorials.net /lesson/exception-handling-in-java/. [Accessed: 10-June-2022]

[18] “Java Programming Exception Handling and Assertion.” Chua Hock Chuan’s Programming Notes. May-2017. [Online]. Available: https://www3.ntu.edu.sg/home/ehchua/programming /java /j5aexceptionassert.html. [Accessed: 10-June-2022]

[19] ”Errors and Exception Handling” python-course.eu, 07-June-2022.[Online]. Available: https:// python -course.eu/python-tutorial/errors-and-exception-handling.php. [Accessed: 10-June-2022]

[20] “Python Vs. Java: Duck Typing, Parsing On Whitespace And Other Cool Differences“, Tom Radcliffe.Available:https://www.activestate.com/blog/python-vs-java-duck-typing-parsing-whitespac e-and-other-cool-differences

[21] Michael T Goodrich *et al.* Data Structures and Algorithms in Java, Sixth edition, Hoboken, NJ: Wiley, [2014]

[22] “Python and Java: The Best of Both Worlds”,Jim Hugunin.Available: https://citeseerx.ist.psu.edu /viewdoc/download?doi=10.1.1.117.6454&rep=rep1&type=pdf

[23] “Java vs Python: Compiled or Interpreted”,Amit Mathur. Available: https://mytechdevice.com /java-vs-python-compiled-or-interpreted

[24] “Lexical Analysis”, En.wikipedia.org. 2022. Lexical analysis - Wikipedia. [online] Available: https://en. wikipedia.org/wiki/Lexical_analysis [Accessed 11 June 2022].

[25] “SINGLE-THREADED VS. MULTITHREADED: WHERE SHOULD WE FOCUS?”, Yale N. Patt, Joshua J. Yi, Derek Chiou, Resit Sendag. Available: https://research.cs.wisc.edu/multifacet /papers/ieeemicro07_debate. pdf

[26] Ben Johnson, Dr. Anjana S Chandran. COMPARISON BETWEEN PYTHON, JAVA AND R PROGRAMMING LANGUAGE IN MACHINE LEARNING, International Research Journal of Modernization in Engineering Technology and Science, Volume 3/Issue 6/June 2021

[27] Stephen J. Humer & Elvis C. Foster. A COMPARATIVE ANALYSIS OF THE C++, JAVA, AND PYTHON LANGUAGES, Keene State College Project from course CS430 Principles of Programming Languages First Draft, April 9, 2014

[28] Tabba, Fuad. (2010). Adding concurrency in python using a commercial processor's hardware transactional memory support. SIGARCH Computer Architecture News. 38. 12-19. 10.1145/1978907 .1978911.