Name: Matthew Kenny

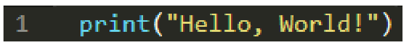
Student number :17470802

CA341 Essay

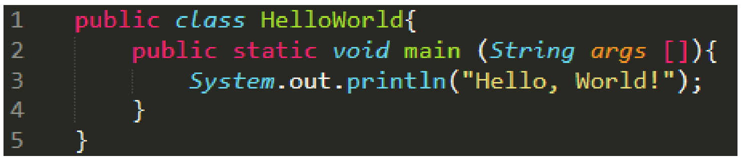
Analytical Analysis of a Programming Language

“Jack of all trades, master of none!” usually refers to someone who has experience in many skills but not mastered the craft by focusing on one specific skill set [1]. This metaphor can represent a programmer for example, a programmer or software developer could have skills in various languages e.g. Python, Java, Haskell, C++, Prolog etc… But they have not dedicated their overall time to “master” one specific language [2]. The statement “Jack of all trades, master of none!” can clearly apply to people’s profession whether it be in software or what not. Although, throughout this essay I will be discussing the same concept but applied to the Python programming language. From my interpretation of reading these resources [1] & [2] ... The given statement for this essay can be viewed as either a compliment or an insult to the Python programming language. I will elaborate on this matter in more detail by given a critical analysis of the Python language and comparing it to various programming languages.

Python programming language is commonly one of the first languages a programming novice would learn. Python code tends to be relatively shorter than other programming languages [4]. Take two program examples both with the same output.



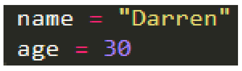
```python
print("Hello, World!")
```



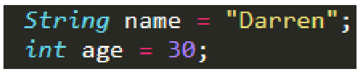
```java
public class HelloWorld{
    public static void main (String args []){
        System.out.println("Hello, World!");
    }
}
```

Both programs produce the output “Hello, World!”, although from a novice perspective the Python program would be easier to understand. Reason being, it isn’t necessarily to create a class or a specific type of a method when doing so in Python. Using Object-oriented programming is possible in Python, but isn’t an enforced concept like in Java programming. Therefore, Python is widely used for how simplistically shorter the code can be. One of the key factors to why it’s the most used programming languages in the world and based on these stats also [3]

Although Python is “typically 3-5 times shorter than equivalent Java programs.” as stated by Guido van Rossum from [4] It does come at a cost by being slower than Java. Python is a dynamic programming language and does not declare types e.g. for variables.[4]



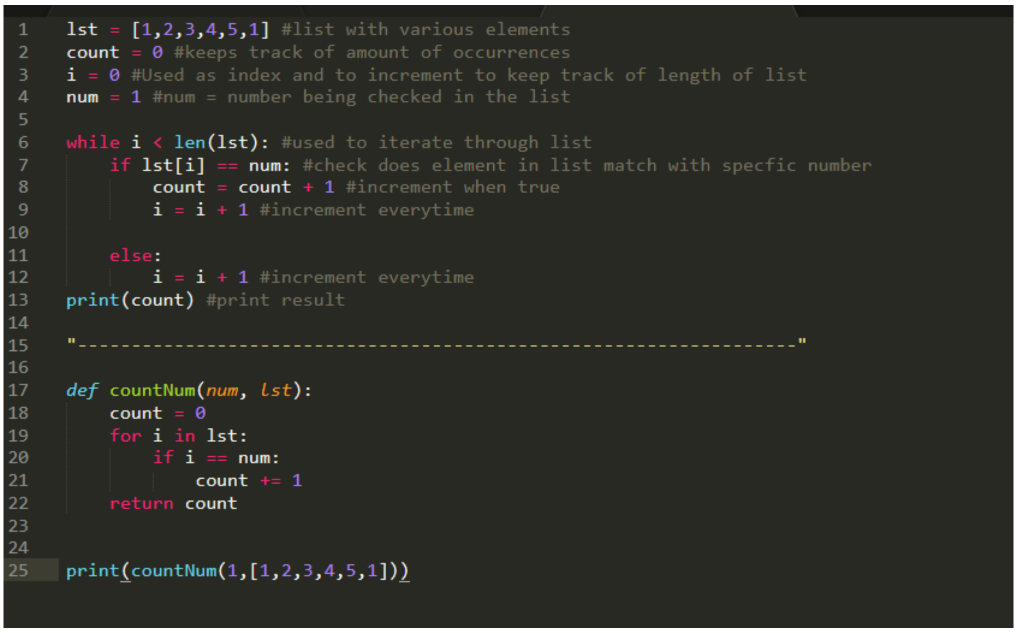
```python
name = "Darren"
age = 30
```



```java
String name = "Darren";
int age = 30;
```

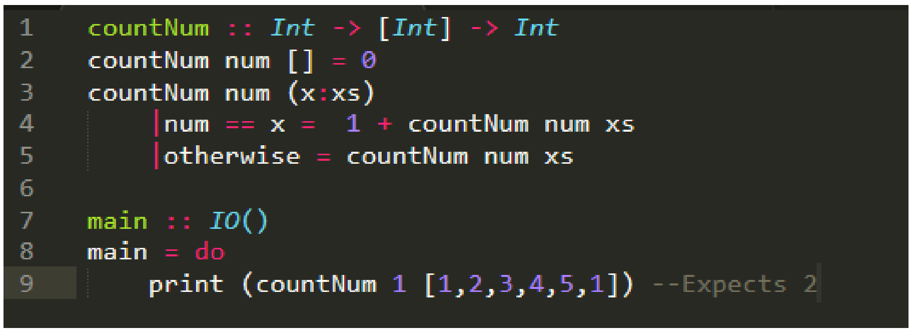
Dynamically typed language vs statically typed language & Compiler vs Interpreter

Notice the example code above, Python (left) does not declare variable types and Java(right) does declare variable types. Python is a Dynamically Typed language, compared to Java which is a Statically Typed language. Main difference as mentioned before, Dynamic programming doesn’t declare variable types so the variable can store an element of any type. Static programming must declare the variable type , for example **String name = “Darren”; (Image above)** can only store a element of type String and **name = “Darren” (Image above) ,** the variable **name** stores a string element but it is not required to be a string type [5]. Python is slower as it needs more run time for checks than Java. Python uses an Interpreter rather than a compiler to check for the type at execution, this can lead to more type errors due to not declaring variable types, and code that is less optimized than a Statically Typed language basically is less efficient. [5]



```python
1   lst = [1,2,3,4,5,1] #list with various elements
2   count = 0 #keeps track of amount of occurrences
3   i = 0 #Used as index and to increment to keep track of length of list
4   num = 1 #num = number being checked in the list
5
6   while i < len(lst): #used to iterate through list
7       if lst[i] == num: #check does element in list match with specfic number
8           count = count + 1 #increment when true
9           i = i + 1 #increment everytime
10
11      else:
12          i = i + 1 #increment everytime
13  print(count) #print result
14
15  "--------------------------------------------------------------------"
16
17  def countNum(num, lst):
18      count = 0
19      for i in lst:
20          if i == num:
21              count += 1
22      return count
23
24
25  print(countNum(1,[1,2,3,4,5,1]))
```

As mentioned above Python is generally shorter than equivalent Java programs. Python has easier syntax to follow than most programming languages. Making it easier for debugging [7]. Take the example above, I have created two programs that basically do the same task. Which is to count the number of occurrences of a specified element in a list of integers. From line 1 to 13 is “Spaghetti code” which is how all beginners begin coding in Python, this style of coding is not possible in various languages such as Haskell or Java , as both of these us functions which would not be familiar to a programming novice. In the second program from line 17 to 25, I have created a function that does the same task as the first program, but this shows how “flexible” and “robust” Python can be as mention in [6] - [7]. Unlike Haskell which is a functional programming language and not as Dynamic as Python. Using “Spaghetti code” would not be possible and quite more difficult to debug, as the language is built on using recursion and functions. In the image below, I have created a program that does the equivalent task to the python program mentioned earlier that simply counts the number of occurrences in the list. It’s clear that the Haskell program has better structure than the Python “Spaghetti code” program. Although compared to the second Python program which uses a function and is just as well structured and efficient as the Haskell program. It goes to show on how “robust” and “flexible” really can be, overall making it an easier debugging language [6] - [7].



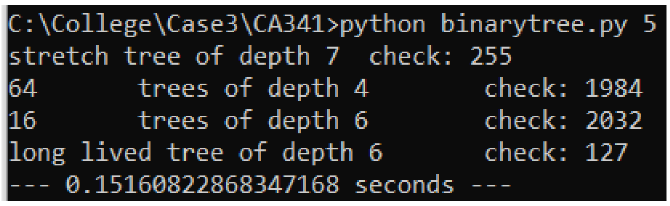
```haskell
countNum :: Int -> [Int] -> Int
countNum num [] = 0
countNum num (x:xs)
    |num == x =  1 + countNum num xs
    |otherwise = countNum num xs

main :: IO()
main = do
   print (countNum 1 [1,2,3,4,5,1]) --Expects 2
```

<u>**“Pythonic way”**</u>

As mentioned in the previous paragraph Python is a more “flexible” language [7]. Although can there be a downside to this? from gaining more knowledge from the source [6] by “Tech with Tim” it must be brought to attention that being “too flexible” of a programming language can be a bad thing [6] – [7]. From experience using Python I can agree from the source [6] it is “easy to get carried” by “the pythonic way” [6] and “pythonic way” is a critical term I am familiar with. Basically, it means doing a certain task with Python syntax, even though it might successfully work, it may not be a “maintainable and sustainable” way as mentioned in the source [6]. Meaning that the code will not be as efficient as it should be and having a poorer structure. Also mentioned in source [6] Python is not always the most popular language for “large robust applications” [6]. This can be related to doing large projects individually or with teams, Python as mentioned in the source [6] “allowed to do too much” meaning the user can get carried away with the Python syntax. This applies especially to large projects that needs structure and even though Python has good structure with Object-oriented programming. It can be quite difficult to maintain that and other languages like Java may be preferred [6]. As mentioned earlier Python is a slower language than Java, based on it being a Dynamic and a “High level language” stated by Youssef Nader from [8]. Java having a more “strict syntax” than Python is obviously not as beginner friendly [8]. Also stated by Youssef

Nader from source [8] “Python doesn’t use enclosing braces and follows indentation rules" meaning Python is like “pseudocode” that helps the language be easier to read especially for beginners. Although having a “stricter syntax” can benefit with larger projects or programs, as Java is a “statically typed language” [8] meaning type doesn’t need to be checked at execution like in Python. Referring to the main point “Python is not always the most popular language for “large robust applications” [6]” many programmers including myself prefer not dealing with indentation rules when doing large programs [8]. As Java provides an overall better structure and performance than Python doing large programs. Using a language with stricter syntax allows more discipline when coding and comes back to the earlier point of “getting carried away” via “Pythonic way” [6]. As even though it may work, it’s not as maintainable and is slower. From analysing source [8] it provides another source “benchmarks-game” [9] which has example code showing the performance of equivalent Python and Java programs.

Take the Python binary tree example from [10] on the next page (Contributed Antoine Pitrou, modified by Dominique Wahli Daniel Nanz and Joerg Baumann). The programs reads in an integer as system argument “**sys.argv[1]**”. Once the user inserts an integer, the program will stretch the tree depth. For example, if **python binarytree.py 5** is entered in the command prompt, a result like this will be executed:



```
C:\College\Case3\CA341>python binarytree.py 5
stretch tree of depth 7  check: 255
64       trees of depth 4        check: 1984
16       trees of depth 6        check: 2032
long lived tree of depth 6       check: 127
--- 0.15160822868347168 seconds ---
```

Notice the run time of the program is printed at the end, from reading source [12] **import time** was added using the built-in function **time()**. Declaring a starting time variable at the beginning “**initial_time = time.time()**” and subtracting starting time by the final time by “**time.time() - initial_time**” to print total run time of the program.

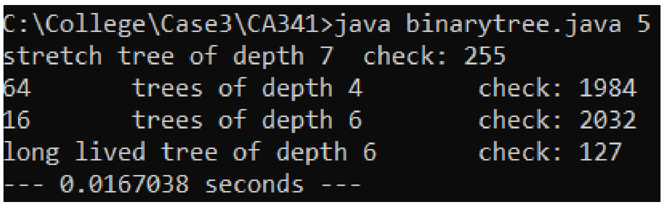
Python binary tree from [10] & [12]

```
# contributed by Antoine Pitrou
# modified by Dominique Wahli and Daniel Nanz
# modified by Joerg Baumann


import time #Added time library
import sys
import multiprocessing as mp
initial_time = time.time() #Starting time for program

def make_tree(d):

    if d > 0:
        d -= 1
        return (make_tree(d), make_tree(d))
    return (None, None)


def check_tree(node):

    (l, r) = node
    if l is None:
        return 1
    else:
        return 1 + check_tree(l) + check_tree(r)


def make_check(itde, make=make_tree, check=check_tree):

    i, d = itde
    return check(make(d))
```

Python binary tree from [10] & [12]

```
def get_argchunks(i, d, chunksize=5000):

    assert chunksize % 2 == 0
    chunk = []
    for k in range(1, i + 1):
        chunk.extend([(k, d)])
        if len(chunk) == chunksize:
            yield chunk
            chunk = []
    if len(chunk) > 0:
        yield chunk


def main(n, min_depth=4):

    max_depth = max(min_depth + 2, n)
    stretch_depth = max_depth + 1
    if mp.cpu_count() > 1:
        pool = mp.Pool()
        chunkmap = pool.map
    else:
        chunkmap = map

    print('stretch tree of depth {0}\t check: {1}'.format(
          stretch_depth, make_check((0, stretch_depth))))

    long_lived_tree = make_tree(max_depth)

    mmd = max_depth + min_depth
    for d in range(min_depth, stretch_depth, 2):
        i = 2 ** (mmd - d)
        cs = 0
        for argchunk in get_argchunks(i,d):
            cs += sum(chunkmap(make_check, argchunk))
        print('{0}\t trees of depth {1}\t check: {2}'.format(i, d, cs))

    print('long lived tree of depth {0}\t check: {1}'.format(
          max_depth, check_tree(long_lived_tree)))

    print(f"--- {time.time() - initial_time} seconds ---") #Ending time for
program & calculate total run time


if __name__ == '__main__':
    main(int(sys.argv[1]))
```

Also, from source [9] is an example Java binary from [11] on the next page (based on Jarkko Miettinen code and contributed by Tristan Dupont). The tree which is equivalent to the Python binary tree program. Also allows the user to run the program with a desired integer as system argument as such **java binarytree.java 5** and will produce the exact same output as when integer 5 is used for Python program, as seen below:



```
C:\College\Case3\CA341>java binarytree.java 5
stretch tree of depth 7  check: 255
64       trees of depth 4        check: 1984
16       trees of depth 6        check: 2032
long lived tree of depth 6       check: 127
--- 0.0167038 seconds ---
```

Notice the run time of the program is printed at the end, from reading source [13] the **Timeunit** library was imported to calculate the run time of the program. Source [14] was used to convert nano seconds to seconds.

Java binary tree from [11], [12] & [13]

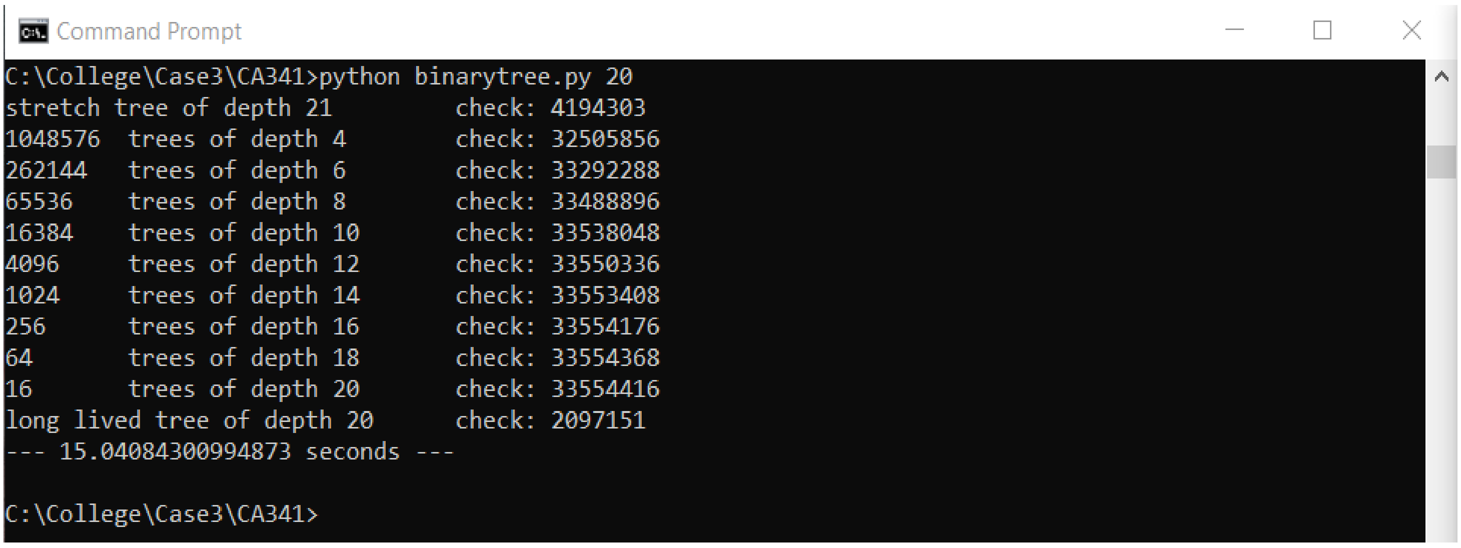
```
import java.util.concurrent.ExecutorService;
import java.util.concurrent.Executors;
import java.util.concurrent.TimeUnit;




public class binarytrees {

    private static final int MIN_DEPTH = 4;
    private static final ExecutorService EXECUTOR_SERVICE =

Executors.newFixedThreadPool(Runtime.getRuntime().availableProcessors()
);

    public static void main(final String[] args) throws Exception {
        long startTime = System.nanoTime();
        // System.out.println(startTime)
        int n = 0;
        if (0 < args.length) {
            n = Integer.parseInt(args[0]);
        }

        final int maxDepth = n < (MIN_DEPTH + 2) ? MIN_DEPTH + 2 : n;
        final int stretchDepth = maxDepth + 1;

        System.out.println("stretch tree of depth " + stretchDepth +
"\t check: "
           + bottomUpTree( stretchDepth).itemCheck());

        final TreeNode longLivedTree = bottomUpTree(maxDepth);

        final String[] results = new String[(maxDepth - MIN_DEPTH) / 2
+ 1];

        for (int d = MIN_DEPTH; d <= maxDepth; d += 2) {
            final int depth = d;
            EXECUTOR_SERVICE.execute(() -> {
                int check = 0;

                final int iterations = 1 << (maxDepth - depth +
MIN_DEPTH);
                for (int i = 1; i <= iterations; ++i) {
                    final TreeNode treeNode1 = bottomUpTree(depth);
                    check += treeNode1.itemCheck();
                }
                results[(depth - MIN_DEPTH) / 2] =
                   iterations + "\t trees of depth " + depth + "\t
check: " + check;
            });
        }
```

Java binary tree from [11], [12] & [13]

```
EXECUTOR_SERVICE.shutdown();
        EXECUTOR_SERVICE.awaitTermination(120L, TimeUnit.SECONDS);

        for (final String str : results) {
            System.out.println(str);
        }

        System.out.println("long lived tree of depth " + maxDepth +
            "\t check: " + longLivedTree.itemCheck());


        long endTime   = System.nanoTime();
        long totalNanoTime = endTime - startTime;
        double timeInSeconds = (double) totalNanoTime / 1_000_000_000;

        System.out.println("--- " + timeInSeconds + " seconds" + " ---");
    }

    private static TreeNode bottomUpTree(final int depth) {
        if (0 < depth) {
            return new TreeNode(bottomUpTree(depth - 1),
bottomUpTree(depth - 1));
        }
        return new TreeNode();
    }

    private static final class TreeNode {

        private final TreeNode left;
        private final TreeNode right;

        private TreeNode(final TreeNode left, final TreeNode right) {
            this.left = left;
            this.right = right;
        }

        private TreeNode() {
            this(null, null);
        }

        private int itemCheck() {
            // if necessary deallocate here
            if (null == left) {
                return 1;
            }
            return 1 + left.itemCheck() + right.itemCheck();
        }

    }

`
```

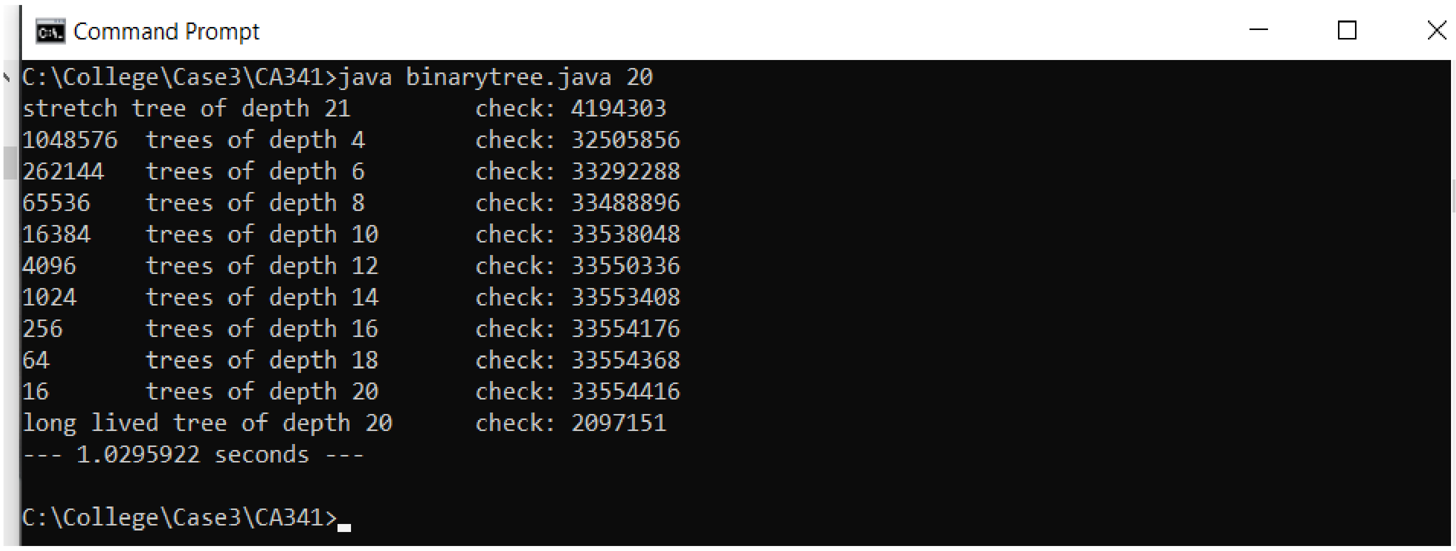
If comparing the binary trees for when input is 5 for Java and Python the run time of both programs are pretty similar (0.015 seconds compared to 0.016 seconds). Although the key point of comparing these two binary tree was to show the comparsion of larger projects. Even though this matter is very much opinionated [6] on the preffered language for larger programs. These upcoming results for larger input may change some opinions. Both equivalent binary trees that read in the stretch tree depth of 20.

Python = 15.04 seconds



```
C:\College\Case3\CA341>python binarytree.py 20
stretch tree of depth 21         check: 4194303
1048576  trees of depth 4        check: 32505856
262144   trees of depth 6        check: 33292288
65536    trees of depth 8        check: 33488896
16384    trees of depth 10       check: 33538048
4096     trees of depth 12       check: 33550336
1024     trees of depth 14       check: 33553408
256      trees of depth 16       check: 33554176
64       trees of depth 18       check: 33554368
16       trees of depth 20       check: 33554416
long lived tree of depth 20      check: 2097151
--- 15.04084300994873 seconds ---

C:\College\Case3\CA341>
```

Java = 1.03 seconds



```
C:\College\Case3\CA341>java binarytree.java 20
stretch tree of depth 21         check: 4194303
1048576  trees of depth 4        check: 32505856
262144   trees of depth 6        check: 33292288
65536    trees of depth 8        check: 33488896
16384    trees of depth 10       check: 33538048
4096     trees of depth 12       check: 33550336
1024     trees of depth 14       check: 33553408
256      trees of depth 16       check: 33554176
64       trees of depth 18       check: 33554368
16       trees of depth 20       check: 33554416
long lived tree of depth 20      check: 2097151
--- 1.0295922 seconds ---

C:\College\Case3\CA341>
```

The Java program ran the exact same input nearly **15** times faster with the exact same output than the Python program from the same resource [9]. Of course, many may argue that the efficiency and performance of the python program can be improved but it really does but performance into perspective for larger projects. This is in no way shape or form complaining about doing large projects in Python, its merely showing the performance differences in using a dynamic language vs a statically typed language in larger projects. In conclusion "Pythonic way" [6] is brilliant for being able to be abstract for programming, although it may not always be the most "maintainable" and "sustainable" method [6].

Machine Learning

From reading the previous paragraphs, readers might think Python is only beneficial for learning programming fundamentals, having simpler syntax and being abstract, so why choose Python? Even though these things may be true, Python has numerous of advantages over other languages for more complex scenarios. Python is famous for using machine learning as from its easier syntax to most languages and multiple choices of libraries, Python to date has "over 137,000" choices of libraries stated by Vaishali Advani [15]. First, there is never a perfect language, it will always depend on what the user is trying to program and this is stated by Voskoglou in the study at [16] that "It depends on what you're trying to build, what your background is and why you got involved in machine learning in the first place." Also, Voskoglou states "There is no such thing as a 'best language for machine learning'", which is the message I am trying to but into perspective. Aside from that lets focus on the positives for machine learning in Python. Its clear technology is always getting smarter and faster, artificial intelligence in machines, gadgets, robots, software etc… are always improving. Software developers and engineers are always pushing the boundaries to what can be accomplished with machine learning. When choosing the correct language for machine learning it goes back to the earlier point "on what you're trying to build" [16]. Python tends to dominate the field of "sentiment analysis" [16] which is basically customer feedback and it's not necessarily to blabber on about the importance of customer feedback for companies that want to strive in improvement. From [16] it is stated by Voskoglou "Machine learning scientists working on sentiment analysis prioritise Python (44%) and R (11%) more and JavaScript (2%) and Java (15%)" compared to other languages , Python is much favoured in this field of machine learning. Most machine learning projects in Python are usually quite large with various directories set up to link different modules together. To show a reasonable sized example of machine learning code, Kyle Stratis discusses some machines learning tips in Python from source [17]. When creating a program for machine learning, the computer needs to read in information and be able to produce it. This is easy for humans as we understand the languages and the words being processed, not quite as easy for a machine. Stratis explains “stop words” from source [17] and how essential communication is. Python has built in library called “spacy” [17] which helps with stop words and improving the communication. Please see example code by Kyle Stratis from [17] below.

```
>>> import spacy
>>> text = """
Dave watched as the forest burned up on the hill,
only a few miles from his house. The car had
been hastily packed and Marta was inside trying to round
up the last of the pets. "Where could she be?" he wondered
as he continued to wait for Marta to appear with the pets.
"""
>>> nlp = spacy.load("en_core_web_sm")
>>> doc = nlp(text)
>>> token_list = [token for token in doc]
>>> token_list
[
, Dave, watched, as, the, forest, burned, up, on, the, hill, ,,
, only, a, few, miles, from, his, house, ., The, car, had,
, been, hastily, packed, and, Marta, was, inside, trying, to, round,
, up, the, last, of, the, pets, ., ", Where, could, she, be, ?, ", he, wondered,
, as, he, continued, to, wait, for, Marta, to, appear, with, the, pets, .,
]
```

Notice the “spacy” library breaks down each word using commas and this type of work break down is important for machine learning as the more input that is given the more commas will be used allowing for better AI learning. Kyle Stratis also talks about the development stage of sentiment analysis from source [17] which is “1. Loading data”, “2. Pre-processing”, “3. Training the classifier” and “4. Classifying data”. It is also stated “For building a real-life sentiment analyser, you’ll work through each of the steps that compose these stages” [17]. From [17] Stratis shows some sample code of how sentiment analyser can be trained.

```python
import os
import random
import spacy

def train_model(
    training_data: list,
    test_data: list,
    iterations: int = 20
) -> None:
    # Build pipeline
    nlp = spacy.load("en_core_web_sm")
    if "textcat" not in nlp.pipe_names:
        textcat = nlp.create_pipe(
            "textcat", config={"architecture": "simple_cnn"}
        )
        nlp.add_pipe(textcat, last=True)
```

Once again multiple libraries are imported for dealing with large “datasheets” [17]. Coming back to the point from Vaishali Advani [15] that “Python to date has ‘over 137,000’ choices of libraries” making machine learning quite easier to learn. As Kyle Stratis mentions various libraries that can be used from [17] such as “import os”, “import spacy”, “PyTorch”, “TensorFlow” etc…. This shows the beauty of Python with its various libraries and how flexible it can be for machine learning no matter what level of complexity it may be. From research Java doesn’t seem to have as many machine learning libraries as Python. From this article written by Vadim Tashlikovich [18] Java is mainly an enterprise language now and basically no new or independent machine learning projects use Java anymore. It’s a lot easier to search the web for “simple machine learning examples” in Python rather than Java. Even with this apparently “simple machine learning example in Java” stated as title in [19] the code is quite outdated and complex, please example code below:

Code below from source [19] is referenced by [20] by Kevin Amaral and [21] by Ian H. Witten and Eibe Frank.

```java
import java.io.BufferedReader;
import java.io.FileNotFoundException;
import java.io.FileReader;
import weka.classifiers.Classifier;
import weka.classifiers.Evaluation;
import weka.classifiers.evaluation.NominalPrediction;
import weka.classifiers.rules.DecisionTable;
import weka.classifiers.rules.PART;
import weka.classifiers.trees.DecisionStump;
import weka.classifiers.trees.J48;
import weka.core.FastVector;
import weka.core.Instances;

public class WekaTest {
        public static BufferedReader readDataFile(String filename) {
                BufferedReader inputReader = null;

                try {
                        inputReader = new BufferedReader(new FileReader(filename));
                } catch (FileNotFoundException ex) {
                        System.err.println("File not found: " + filename);
                }

                return inputReader;
        }

        public static Evaluation classify(Classifier model,
                        Instances trainingSet, Instances testingSet) throws
Exception {
                Evaluation evaluation = new Evaluation(trainingSet);

                model.buildClassifier(trainingSet);
                evaluation.evaluateModel(model, testingSet);

                return evaluation;
        }

        public static double calculateAccuracy(FastVector predictions) {
                double correct = 0;

                for (int i = 0; i < predictions.size(); i++) {
                        NominalPrediction np = (NominalPrediction)
predictions.elementAt(i);
                        if (np.predicted() == np.actual()) {
                                correct++;
                        }
                }

                return 100 * correct / predictions.size();
        }
```

```java
public static Instances[][] crossValidationSplit(Instances data, int numberOfFolds) {
                Instances[][] split = new Instances[2][numberOfFolds];

                for (int i = 0; i < numberOfFolds; i++) {
                        split[0][i] = data.trainCV(numberOfFolds, i);
                        split[1][i] = data.testCV(numberOfFolds, i);
                }

                return split;
        }

        public static void main(String[] args) throws Exception {
                BufferedReader datafile = readDataFile("weather.txt");

                Instances data = new Instances(datafile);
                data.setClassIndex(data.numAttributes() - 1);

                // Do 10-split cross validation
                Instances[][] split = crossValidationSplit(data, 10);

                // Separate split into training and testing arrays
                Instances[] trainingSplits = split[0];
                Instances[] testingSplits = split[1];

                // Use a set of classifiers
                Classifier[] models = {
                                new J48(), // a decision tree
                                new PART(),
                                new DecisionTable(),//decision table majority
classifier
                                new DecisionStump() //one-level decision tree
                };

                // Run for each model
                for (int j = 0; j < models.length; j++) {

                        // Collect every group of predictions for current model in a
FastVector
                        FastVector predictions = new FastVector();

                        // For each training-testing split pair, train and test the
classifier
                        for (int i = 0; i < trainingSplits.length; i++) {
                                Evaluation validation = classify(models[j],
trainingSplits[i], testingSplits[i]);


        predictions.appendElements(validation.predictions());

                                // Uncomment to see the summary for each training-
testing pair.
                                //System.out.println(models[j].toString());
                        }
// Calculate overall accuracy of current classifier on all splits
                        double accuracy = calculateAccuracy(predictions);

                        // Print current classifier's name and accuracy in a
complicated,
                        // but nice-looking way.
                        System.out.println("Accuracy of " +
models[j].getClass().getSimpleName() + ": "
                                        + String.format("%.2f%%", accuracy)
                                        + "\n---------------------------------
-");
                }

        }
```

Where is Python used?

Python is one of the most used programming languages in the real world. From reading the source by Brian Hernandez [22] Python has built some “famous websites” e.g. reddit, Spotify, Youtube etc… Hernandez also mentions that Python is “Extensible in C and C++” which would benefit any programmer from a C/C++ background , “Ease of learning and support” Python has such a brilliant community compared to other languages making it an even easier language to learn. As mentioned previously from source [3] Python is the most popular language to date leading to having a great community. Also reading this great article by Dr.Matt from the source [23] it’s clear that Python has a big part to play in the medical industry as Dr.Matt stated “Now, I have developed two web applications to aid in both medical education and healthcare administration” showing that even Doctors find using Python helpful for practical use and could even save a humans life. Also in the article by Dr.Matt imentions “Leaning for fun” discussing other uses for Python e.g. gaming and development.

Conclusion

From reading, watching and learning multiple resources, anything in Python is possible. Even though it might not be the most suitable and maintainable method , Python is a fantastic programming language whether the user is learning the fundamentals of programming as a concept , exploring the incredible amount of libraries for any specific task that is required , doing more complex projects for example creating artificial intelligence by using machine learning which also provides very helpful libraries , building great websites or even saving a humans life. Yes, Python may be a slow language compared to other languages like Java or Haskell and it might not be the most practical language for structure. Yet, I would have to disagree with the statement ““Jack of all trades, master of none!””, as Python as mentioned does master certain aspects to programming and using software in general. Other languages like Java, C++ are being used but nowhere to the extent of Python, some may argue Java and C++ is dying compared to Python. As Python seems to be such an abstract language. In conclusion even in 2020 Python is still a massive language for software development and strives in mastering certain aspects of programming.

[1] **Wikipedia contributors , Wikipedia,** https://en.wikipedia.org/wiki/Jack_of_all_trades,_master_of_none **,[accessed Dec.8 2020]**

[2] **PARKERSoftware contributors, PARKERSoftware, https://www.parkersoftware.com/blog/the-fallacy-of-being-a-jack-of-all-trades-master-of-none-developer/ , [accessed Dec.9 2020]**

[3] **PYPL , pypl.github.io, https://pypl.github.io/PYPL.html ,[accessed Dec.11 2020]**

[4**] Guido van Rossum, python.org,https://www.python.org/doc/essays/comparisons/ ,[accessed Dec.12 2020]**

[5] **TechGatha, YouTube, https://www.youtube.com/watch?v=jlUZw8-6ljw, [accessed Dec.12 2020]**

[6] **Tech With Tim, YouTube, https://www.youtube.com/watch?v=Ou_hSDxzjMI, [accessed Dec.14 2020]**

[7]] **Mindfire Solutions, medium.com, https://medium.com/@mindfiresolutions.usa/advantages-and-disadvantages-of-python-programming-language-fd0b394f2121 ,[accessed Dec.14 2020]**

[8] **Youssef Nader, hackr.io, https://hackr.io/blog/python-vs-java , [accessed Dec.16 2020]**

[9] **Author, benchmarksgame-team, https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/python3-java.html , [accessed Dec.16 2020]**

[10 **Antoine Pitrou, benchmarksgame**, **https://benchmarksgame-team.pages.debian.net/benchmarksgame/fastest/python3-java.html , [accessed Dec.16 2020]**

[11] **Jarkko Miettinen, benchmarksgame, https://benchmarksgame-team.pages.debian.net/benchmarksgame/program/binarytrees-java-7.html, [accessed Dec.16 2020]**

[12] **1st answer on page, stack overflow, https://stackoverflow.com/questions/1557571/how-do-i-get-time-of-a-python-programs-execution , [accessed Dec.16 2020]**

[13] **1st answer on page, stack overflow, https://stackoverflow.com/questions/5204051/how-to-calculate-the-running-time-of-my-program [accessed Dec.18 2020]**

[14] **mKyong, mKyong.com, https://mkyong.com/java/java-how-to-convert-system-nanotime-to-seconds/ , [accessed Dec.18 2020]**

[15] **Vaishali Advani** , **greatlearning blog https://www.mygreatlearning.com/blog/open-source-python-libraries/ , [accessed Dec.19 2020]**

[16] **Christina Voskoglou, towards data science, https://towardsdatascience.com/what-is-the-best-programming-language-for-machine-learning-a745c156d6b7 , [accessed Dec.19 2020]**

[17] **Kyle Stratis, Real Python** , **https://realpython.com/sentiment-analysis-python/ , [accessed Dec.19 2020]**

[18] **Vadim Tashlikovich, scand.com https://scand.com/company/blog/why-is-java-the-main-language-for-enterprise-development/ , [accessed Dec.20 2020]**

[19] **Author ,programcreek, https://www.programcreek.com/2013/01/a-simple-machine-learning-example-in-java/, [accessed Dec.20 2020]**

[20] **Kevin Amaral, cs.umb.eu,** Weka Java API Tutorial.docx (umb.edu) **,[accessed Dec.20 2020]**

[21] **Ian H. Witten and Eibe Frank** , **cs.run.nl ,** weka.pdf (ru.nl) **, [accessed Dec.20 2020]**

[22] **Brian Hernandez, quickstart.com, https://www.quickstart.com/blog/10-famous-websites-built-with-python/ , , [accessed Dec.20 2020]**

[23] **Dr.Matt** , **medium.com, https://medium.com/doctors-in-tech/the-medical-programmer-why-doctors-should-code-and-where-you-can-learn-today-68ebdcf8079f, [accessed Dec.20 2020]**