Student name: Vincent Achukwu
Student number: 17393546
Project title: Analytical Analysis of a Programming Language

Introduction

"Jack of all trades, master of none" is a term used to describe a person who is competent in various skills but would rather not gain experience by focusing on one *[7]*. I believe that it is ok for developers in becoming a jack of all trades, and a master of none because it is useful to know various programming languages and understand the differences between them, as well as becoming familiar with the various tools available for software developers.

Learning different languages gives the advantage of obtaining a better understanding of how the different languages work and what languages work best for a given task. This makes such developers have a broader range of knowledge, appear more committed to learning new things, and are adaptable and more flexible *[7]*.

In this essay, I will compare and contrast Python and other programming languages about how certain things can and/or can't be done between them, and the advantages of being familiar with various programming languages.

Starting Language

Python is primarily part of the Object-Oriented Paradigm and supports other programming paradigms such as the Imperative and Functional Programming Paradigm. "Object-Oriented programming can be seen as the natural extension of data abstraction" *[3]*. Python is also an interpreted and general-purpose high-level programming language with dynamic semantics. Being high-level built-in data structures, with dynamic typing and binding, allows for Rapid Application Development (RAD). Python is an easy-to-learn *[8]* and simple programming language which has a high emphasis on readability, and this reduces the cost of program maintenance. This is why Python is usually chosen as the first programming language to teach to beginner developers, rather than Java or C. With no compilation step, the edit-test-debug cycle is much faster. In comparison, Java has a compilation step and is a statically typed language. It has to be compiled from the human-readable code to machine-readable code *[4]*.

How to run Python vs how to compile and run Java file:

```
vinny@LAPTOP-JNEHHU5V:/College$ python3 test.py
Hello, World
vinny@LAPTOP-JNEHHU5V:/College$ javac Main.java
vinny@LAPTOP-JNEHHU5V:/College$ java Main
Hello, World
vinny@LAPTOP-JNEHHU5V:/College$
```

Syntax

As mentioned before, the Python syntax is much easier to understand and the code is generally shorter in comparison to languages like C and Java. Python does not require the use of semicolons or curly braces in the program in comparison to Java which complains if the developer does not add curly braces or semicolons in the program *[4].* Observe how the Python code is shorter and simpler to understand in comparison to the Java code for the same program (i.e. printing "hello world").

Python "Hello World" program:

```
1  print("Hello, World")
2
```

Java "Hello World" program:

```
1  public class Main
2  {
3
4      public static void main(String []args)
5      {
6          System.out.println("Hello, World");
7      }
8  }
```

Python is much easier to read than Java which gives the advantage of using Python. However, referring to "the jack of all trades, master of none", this shows how being proficient in only one language restricts ones ability to understand the different syntax and how different programming languages can do similar things in different ways. It is highly beneficial to be familiar with the different programming languages as this can help developers to improve in their problem-solving skills. Such developers have the versatility and a diverse skill set *[7]*.

The difficulty in understanding the code can also increase with Java in comparison to Python when writing large programs. Since both languages are in the Object-Oriented paradigm, both can still get complex when writing large programs. In Python, you do not need to specify the types of the variables being defined whereas, in Java, this is required. This can make it difficult to track what the current type of the variable is in Python since, depending on the program, the type of a variable may change. In Java, you know initially what the type is. Since Python is dynamically typed, operands can only be type-checked during the run-time while being computed and before the operation is performed *[2]*. However, Java is a statically typed language, which means that variables have a fixed type. Before every operation, the operand is checked by the compiler to see if it is the correct type or not *[2]*.

Python dynamic typing:

```python
num = 2
print(num)

num = "Hello!"
print(num)

# TypeError, can't add string with number
print(num + 1)

```

Java static typing:

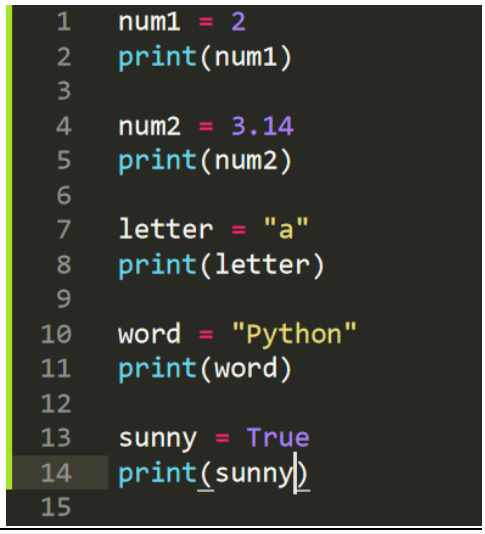
```java
public class Main
{

    public static void main(String []args)
    {
        int a = 1;
        System.out.println(a);

        // Types cannot be converted to other types, error raised
        a = "hello";
        System.out.println(a);

        String msg = "hello";
        System.out.println(msg);
    }
}

```

Data Types and Scope

Python has 4 primitive types: String, Integers, Floats, and Booleans. Java also has these data types, but they are used slightly differently compared to Python. In Java and C, strings are an array of characters, and declaring them may seem rather confusing compared to Python. In Python, you can slice strings according to the indexes, whereas Java requires a method to do this. This makes Python

have shorter code than Java since Java cannot do things with simpler syntax like Python. Java also requires specifying the type of a variable when defining or initialising it. Though Java and C have strings, Python doesn't require the string to be defined as an array of characters. In other words, Java and C have a *Char* type, which is a character type. Python does not require this since anything in quotation marks is considered a string, even one character. In C, declaring arrays requires specifying the length of the array, and the array type must also be given. This makes programming in Python much easier since you do not need to worry about declaring data types, as they can change during the execution of the program. Also, Java and C have more types of numbers such as Integer, Float, Double, Short, and Long. Whereas Python has Integer and Float which are the main numeric types used.

Python data types:



Java data types:

```java
public class Main
{
    public static void main(String []args)
    {
        // whole numbers from -2147483648 to 2147483647
        int num1 = 1;
        System.out.println(num1);

        // fractional numbers from 3.4e-038 to 3.4e+038
        float num2 = 3.1415f;
        System.out.println(num2);

        // fractional numbers from 1.7e-308 to 1.7e+308
        double num3 = 20.54d;
        System.out.println(num3);

        // whole numbers from -32768 to 32767
        short num4 = 200;
        System.out.println(num4);

        // whole numbers from -9223372036854775808 to 9223372036854775807
        long num5 = 45000000000L;
        System.out.println(num5);

        // single quotes
        char letter = 'a';
        System.out.println(letter);

        // double quotes
        String word = "Python";
        System.out.println(letter);
    }
}
```

C data types:

```c
int main()
{
        // 4 bytes
        int num1 = 1;
        printf("%d\n", num1);

        // 4 bytes
        float num2 = 3.1415;
        printf("%f\n", num2);

        // 8 bytes
        double num3 = 20.54;
        printf("%lf\n", num3);

        // 2 bytes, whole numbers from -32768 to 32767
        short int num4 = 200;
        printf("%hd\n", num4);

        // 4 bytes, whole numbers from -2,147,483,648 to 2,147,483,647
        long int num5 = 45000000000L;
        printf("%ld\n", num5);

        // 1 byte
        char letter = 'a';
        printf("%c\n", letter);

        // double quotes, size specified for char array
        char word [10] = "Python";
        printf("%s\n", word);

    return 0;
}
```

This can confuse beginner programmers as to why this is the case. But again, referring to the "jack of all trades", it is valued to have good knowledge of how different programming languages can handle data types and the advantages of using each one. For instance, in C, Float takes up 4 bytes of memory, and Double takes up 8 bytes of memory. This is the case because Double allows for more precision (15 decimal places, compared to Floats which are up to 6 decimal places). Depending on the purpose of the program, you may or may not need very accurate results.

The scope is known as the part of the program in which the declaration is in effect *[2].* The scope in Python programs can be the entire program (global), within if/elif/else statements, loops, classes, or functions. These are held inside the blocks which "delimits the scope of any declaration within it" *[2].* Since earlier programming languages had the scope as the entire program, this made it uneasy to develop programs since variable names had to be unique. This is why blocks are used. Python blocks are modules, classes, or function bodies. Functions are defined using braces in Java, whereas Python does not need braces. Whenever a loop, function, if/elif/else statement, or class definition is being made, Python requires the line to end in a colon instead of these braces. Again, this allows for easier management of code instead of worrying about whether the curly braces are matching or not.

Python Scope and Block:

```
1   num = 300
2
3   def foo():
4       x = 10  # local to foo()
5       global num
6       num = 200
7
8   foo()
9   # output will be 200
10  print(num)
11  # x is undefined, error
12  print(x)
```

Java Scope and Block:

```java
public class Main
{
    public static void main(String []args)
    {   // block starts here

        // code here cannot use num1
        int num1 = 1;

        // code here can use num1
        System.out.println(num1);

    }   // block ends here

    // code here cannot use num1
}
```

C Scope and Block:

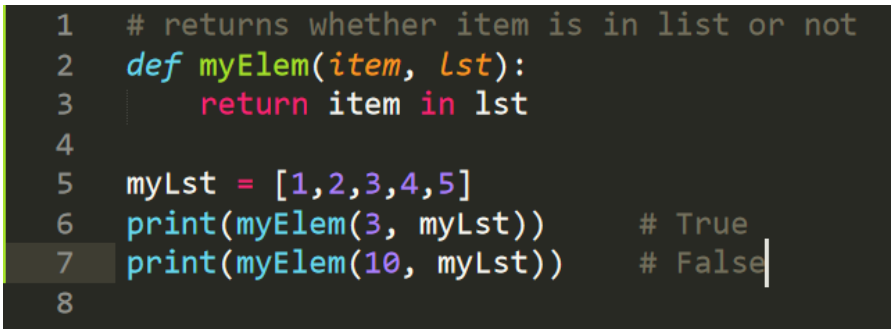
```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

// global variable definition
int a;

int main()
{

    // local variable definition and initialisation
    int b, c;

    b = 20;
    c = 30;

    a = b + c;

    // a is the sum of b + c
    printf("%d\n", a);

    return 0;
}
```

Functions

Different languages have different ways of creating program functions. Haskell, a functional and declarative programming language, has a very distinct syntax for defining functions compared to Python. Operations in Haskell are solely “based on the inputs to the program” *[1]* and does not rely on temporary variables to store results in between the inputs. Hence, it relies on recursion to compute results. Haskell functions are defined by specifying the types of the inputs, and the type of the output. Java requires the developer to specify the return type of the function as well as the return type(s) of the parameters. In Python, it is as simple as specifying the method name and passing in any parameters to be used in the function body.

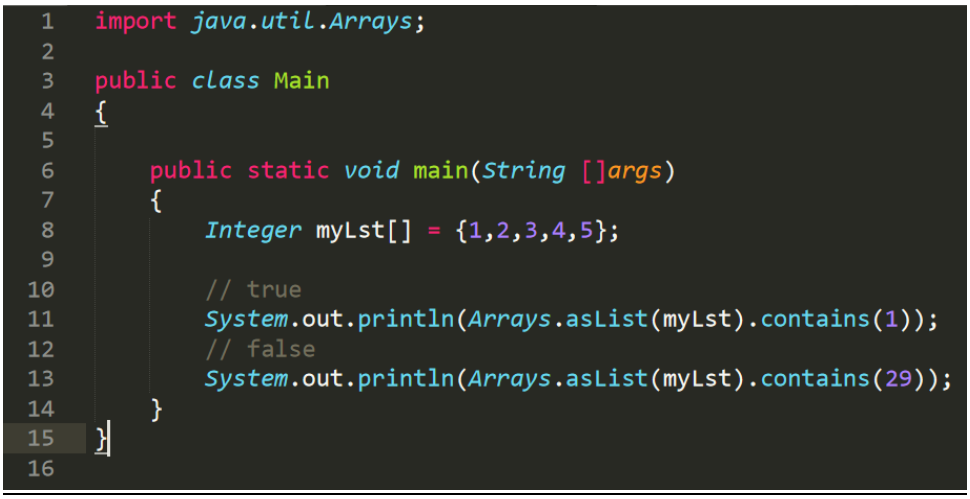
In Python, you can alter data by simply using methods on objects, whereas in Haskell, recursion is needed to do things you would commonly do in Python such as getting the length of a list or checking if an item is inside a list or string, or replacing an element in a list with another item. Haskell programs can get more complex as they grow since it requires more code and reduces readability. Python gives the user access to methods on its objects which makes it easier to manipulate the data for certain calculations and operations. The functional programming paradigm can help developers to improve their problem-solving skills by tackling problems in different approaches compared to using Python. It is also a lazy language, meaning that "nothing is evaluated until it has to be evaluated" *[6].* The code snippet below showcases the differences between Python, Java, and Haskell for checking if an item is in a list.

Python Function:



```python
# returns whether item is in list or not
def myElem(item, lst):
    return item in lst

myLst = [1,2,3,4,5]
print(myElem(3, myLst))     # True
print(myElem(10, myLst))    # False

```

Java Function:



```java
import java.util.Arrays;

public class Main
{

    public static void main(String []args)
    {
        Integer myLst[] = {1,2,3,4,5};

        // true
        System.out.println(Arrays.asList(myLst).contains(1));
        // false
        System.out.println(Arrays.asList(myLst).contains(29));
    }
}

```

Haskell Function:

```haskell
-- item and elements of list must be of same type
myElem :: (Ord a) => a -> [a] -> Bool
-- if list is empty, False
myElem _ [] = False
-- else recursively search element until we find match
myElem item (x:xs) = (item == x) || (myElem item xs)

```

A nice feature in Haskell which Python does not have is the ability to create an infinite list without the use of a loop. This list can also be altered with step sizes, and not only for integers, but for characters too. Python does not have this feature because again, Haskell is a lazy language.

Haskell Lists:

```haskell
-- infinite list of numbers
infinite = [1..]

-- even numbers from 2 to 10
evens = [2,4..10]

-- odd numbers from 1 to 11
odds = [1,3..11]

-- alphabet
alpha = ['a'..'z']


```

Loops

While Loop

While loops allow for the iteration over a block while the condition in the loop is true. This type of loop is usually used when we initially do not know how many times to iterate the loop. It is usually done with an increment value updating throughout the execution of the loop for each iteration. If the condition of the loop of the current iteration is true, the body of the loop is executed. While loops in Python and C are quite similar, with the exception of C requiring to specify the type of the iterator and the use of curly braces to mark the start and endpoints of the loop. In Python, everything in the body of the loop is indented. C requires the use of brackets for the condition in the loop, whereas Python does not require this.

Python While Loop:

```python
# prints i 10 times.
# i will range from 0 to 9 inclusive
i = 0
while i < 10:
    print(i)
    i += 1

```

C While Loop:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
    // prints i 10 times.
    // i will range from 0 to 9 inclusive
    int i = 0;
    while(i < 10){
        printf("%d\n", i);
        i++;
    }

    return 0;
}

```

For Loop

In Python, a for loop is used to iterate through a sequence (list, string, dictionary, etc) or other iterable objects. This type of loop is usually used when you know how many times you want the loop to iterate. The loop continues until we reach the last item in the sequence. As with while loops, the body of the for loop is indented and the body ends on the first unindent. In C, the for loop syntax is slightly different in the sense that you have to specify the type of the iterator, the condition, and the increment step size all inside one line. As with while loops, curly braces mark the start and end of the body of the loop. For loops in Python are easier to understand due to Python's simpler syntax compared to C. The range() function in Python allows the iterator to loop through a range of values passed into the range() function. If you were to use for loops the way C does (i.e. having the iterator as an integer), the range() function allows for this feature.

Python For Loop:

```python
# as i iterates through the name list,
# each name will be printed
names = ["bob", "mark", "annie", "lizzie"]
for i in names:
    print(i)

# when using range(), iterator is an integer
# similar style to C where iterator is an integer
for i in range(10):
    print(i)

```

C For Loop:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
    // prints i 10 times.
    // i will range from 0 to 9 inclusive
    for(int i = 0; i < 10; i++){
        printf("%d\n", i);
    }

    return 0;
}

```

Supporting the argument of this essay, I believe that it is important for developers to understand the different ways of using loops in different programming languages because different languages have their own pros and cons. Sticking to one language can restrict a programmer from doing things more efficiently in other languages, and this can lead to bad programming practices and inefficient programming.

Comments

Writing comments is a very important part of software development. It allows the person looking at the code to understand what is going on and this can significantly improve the workflow.

Comments can be used to describe what the program does. They are ignored by the program interpreter, and even code inside comments are ignored. Different programming languages have different ways of doing this. In Python, the hash symbol is used to denote a single-line comment which, like statements, extends until the newline character. Comments can also be used across multiple lines using triple quotation marks (starting with either ‘’’ or “”” and ending with ‘’’ or “””, respectively). In Java, single-line comments begin with two forward slashes, and any text between the two forward slashes and the end of that line (even code) is ignored by Java, and will not be executed *[5]*. Multiline comments in Java are denoted starting with */** and ending with ***/**. Any text between these notations are ignored by Java and are not executed. Multiline comments are useful when commenting out long lines of code which the programmer may need later. It is also faster to use multiline comments for multiline text explaining code rather than manually using single-line comment notation. C uses the same syntax as Java for making comments in programs.

Python Comments:

```
1   # this is a comment
2
3   """ this is a
4   multi-line comment"""
5
6   # commenting code below
7   # names = ["bob", "mark", "annie", "lizzie"]
8
9   """ multi-line commenting the code
10    for i in names:
11       print(i)"""
12
13  print("Hello World") # this is a comment
14
```

Java Comments:

```java
public class Main
{
    public static void main(String []args)
    {
        // single-line comment

        /* this is a
        multi-line comment*/

        for(int i = 0; i < 10; i++){
            System.out.println(i);
            // System.out.println("Hello world"); single-line commenting code
        }

        // int i = 0; multi-line commenting code below
        /*while(i < 100){
            System.out.println(i);
        }*/
    }
}

```

C Comments:

```c
#include <stdio.h>
#include <stdlib.h>
#include <string.h>

int main()
{
    // single-line comment

    /* this is a
    multi-line comment*/

    for(int i = 0; i < 10; i++){
        printf("%d\n", i);
        // printf("Hello world\n"); single-line commenting code
    }

    // int i = 0; multi-line commenting code below
    /*while(i < 100){
        printf("%d\n", i);
    }*/

    return 0;   // this is a comment
}

```

Commenting about what the code does is very important in the software development process as it allows other programmers to understand what the code is doing. This will greatly help those programmers who fall into the “master of none” category who would rather not learn new languages but maybe stick to one. Comments will definitely help them understand how the unfamiliar code works. Likewise, for those under the “jack of all trades” category, these programmers must comment code at all times, even for their own purposes of understanding.

Errors

Error messages can vary throughout different programming languages. In some languages, it can be trickier to understand and trace the error based on the error message during run-time or

compile-time. Python's error messages are fairly straightforward to understand. They can provide the programmer with the source of the error via the line number, the type of error, and the trace of the error based on function calls. Errors in Haskell are more detailed as they can suggest the user with what they may have intended to do in the first place. Haskell errors tell the developer the error type and a suggestion as to how to fix that error. Errors in Java can happen during compile-time or run-time. Compile-time errors generally tell the developer about syntax errors (e.g. missing semicolons, parentheses, or undefined variables). Hence, this is why knowing more languages is useful as understanding the error messages is vital during debugging and testing. It allows for a better understanding of what the program should be doing.

Python Error:

```
1
2   print("Hello world")
3
4   # causes NameError
5   print(x)
6
```

```
vinny@LAPTOP-JNEHHU5V:/College$ python3 test.py
Hello world
Traceback (most recent call last):
  File "test.py", line 5, in <module>
    print(x)
NameError: name 'x' is not defined
```

Java Error:

```
1   public class Main
2   {
3       public static void main(String []args)
4       {
5           for(int i = 0; i < 10; i++){
6               // missing semicolon
7               System.out.println(i)
8           }
9       }
10  }
```

```
vinny@LAPTOP-JNEHHU5V:/College$ javac Main.java
Main.java:6: error: ';' expected
            System.out.println(i)
                               ^
1 error
```

Haskell Error:

```haskell
-- second last item of list
-- causes error since x not in scope
secondLast :: [a] -> a
secondLast xs = last (init x)
```

```
Prelude> :l test.hs
[1 of 1] Compiling Main             ( test.hs, interpreted )

test.hs:4:28: error: Variable not in scope: x :: [a]
Failed, modules loaded: none.
```

In most languages, there is the notion of exception handling. This is the way programming languages handle errors when they arise. In Python, this is done with *try* and *except e* keywords, where the *try* clause tries to run code within that block, but if an error occurs, that error is caught and handled with in the *except e* clause, where *e* is the error type. This is a common way of exception handling in Object-Oriented programming languages.

Python Exception Handling:

```python
try:
    x = 0
    print("The number is {}".format(x))
    # y is not defined, NameError raised
    print("The other number is {}".format(y))
except NameError:
    print("Something went wrong...")

```

```
vinny@LAPTOP-JNEHHU5V:/College$ python3 test.py
The number is 0
Something went wrong...
```

A simple way of exception handling in Haskell can be done via functions having an exception if a certain input is read. If a certain condition is met, such as having an empty list when the function should return the last element of a list, a custom error message can be displayed. This form of exception handling is very simple and easy to read compared to Python's exception handling.

Haskell Exception Handling:

```haskell
-- last item of list
myLast :: [a] -> a
-- if list empty, raise error with message
myLast [] = error "List cannot be empty"
myLast [x] = x
myLast (x:xs) = myLast xs
```

```
*Main> myLast [1,2,3,4]
4
*Main> myLast []
*** Exception: List cannot be empty
CallStack (from HasCallStack):
  error, called at test.hs:4:13 in main:Main
```

Java's exception handling is similar to Python exception handling. There is the *try* block enclosed with curly braces, and the exception block used with the keyword *catch (exception e)* which is also enclosed in curly braces, which allows code to be executed if the error occurs in the *try* block. The *exception* is the exception to catch and *e* is just a variable which stands for "exception", but it can be renamed to whatever the programmer desires. Regardless, the data type remains an exception.

Java Exception Handling:

```java
public class Main
{
    public static void main(String []args)
    {
        try {
            int[] myNums = {1, 2, 3};
            System.out.println(myNums[1]);
            // index out of range, error raised
            System.out.println(myNums[10]);
        }
        catch (Exception e) {
              System.out.println("Something went wrong...");
        }
    }
}
```

```
vinny@LAPTOP-JNEHHU5V:/College$ javac Main.java
vinny@LAPTOP-JNEHHU5V:/College$ java Main
2
Something went wrong...
```

As seen from those examples, programmers who know more languages ("jack of all trades") have a better advantage of understanding how different languages behave over programmers who solely focus on one language. Reading error messages tests the programmer's ability to test and debug code.

Conclusion

To conclude, programmers who focus on one language ("master of one") can have expertise in their field of study, can be more productive, and can save a lot of time in training since they are already an expert at what they do *[7]*. They tend to be very productive since they have a high degree of knowledge in their field of study. However, these types of developers can easily become a jack too because since they are fully skilled in what they do, they can be hungry to learn more. Mastering one skill can lead to obtaining more knowledge about other fields and aspects of software development. But I believe that "jack of all trades" has higher advantages since they are long term learners after learning many things and their expansive knowledge makes them more flexible and diverse in their work environment.

References

[1] Lildiaz00, 2020. *Project 4 – Advantages And Disadvantages Of Programming Languages.* [online] Lildiaz00.blogspot.com. Available at: <http://lildiaz00.blogspot.com/2010/11/project-4-advantages-and-disadvantages.html> [Accessed 10 December 2020].

[2] Sinclair, D., 2020. *CA341 - Comparative Programming Languages Data Types And Scope.* [online] Computing.dcu.ie. Available at: <https://www.computing.dcu.ie/~davids/courses/CA341/CA341_Data_Types_and_Scope_2p.pdf> [Accessed 10 December 2020].

[3] Sinclair, D., 2020. *CA341 - Comparative Programming Languages Object-Oriented Programming Paradigm*. [online] Computing.dcu.ie. Available at: <https://www.computing.dcu.ie/~davids/courses/CA341/CA341_Object-Oriented_Programming_Paradigm_2p.pdf> [Accessed 10 December 2020].

[4] GeeksforGeeks. 2020. *Comparison Of Python With Other Programming Languages - Geeksforgeeks*. [online] Available at: <https://www.geeksforgeeks.org/comparison-of-python-with-other-programming-languages/> [Accessed 10 December 2020].

[5] W3schools.com. 2020. *Java Comments*. [online] Available at: <https://www.w3schools.com/java/java_comments.asp> [Accessed 10 December 2020].

[6] Wiki.haskell.org. 2020. *Why Haskell Matters - Haskellwiki*. [online] Available at: <https://wiki.haskell.org/Why_Haskell_matters#What_can_Haskell_offer_the_programmer.3F> [Accessed 10 December 2020].

[7] Vishwakarma, H., 2020. *Who Is A Better Hire, Jack Of All Trades Or Master Of One?*. [online] Medium. Available at: <https://medium.com/swlh/who-is-a-better-hire-jack-of-all-trades-or-master-of-one-953cf6d46fe5>

[Accessed 10 December 2020].

[8] Yegulalp, S., 2020. *What Is Python? Powerful, Intuitive Programming*. [online] InfoWorld. Available at: <https://www.infoworld.com/article/3204016/what-is-python-powerful-intuitive-programming.html> [Accessed 10 December 2020].