Cooper Nathan 7/8/2025

Reflection: Python Vs JavaScript

Type systems: Both Python and JavaScript are dynamically type, which means that variables types do not need to be defined when instantiating in the code. The type is determined via context at runtime. For example, in a statically typed language like Java, you would need to write “String password” to instantiate a string called password, whereas in Javascript you could simply type “var password” and “password = “hello”, and in Python type “password = ‘hello’”. Python variables can also be changed to different types in the same way, so in the next line we could say “password = 5” and the variable becomes type integer. While default Javascript is dynamically typed, there are versions like TypeScript that are specially designed to be statically typed.

Class and Object Definitions: Both languages can be object oriented. Python classes are defined with the ‘class’ keyword and basic attributes initialized using a special function called __init__. Python also supports multiple inheritance where classes can inherit from more than one parent, as well as abstract base classes, decorators, and data classes. Objects of a class can be instantiated by typing variableName = ClassName(...) and passing required attributes. Example below.

```
class Car:

        def __init__(self, brand, model):

                self.brand = brand

                self.model = model

myCar = Car(“Subaru”, “Impreza”)
```

A Javascript class definition is fairly similar, with a class name and constructor with assignment. Classes in javascript are basically syntax on top of the original JS methodology of ‘Prototypes’, which are the mechanism by which objects inherit features. Javascript classes are only single inheritance. Objects can be instantiated by simply typing const objectName = new ClassName(...), and passing in the attributes required by the constructor. Javascript also has object literals that allow an object to be instantiated with attributes without explicitly defining a general class. For example in the code below, instead of defining the class first, we could write const myCar = { brand: “Subaru”, model: “Impreza”};

```
class Car {

        constructor(brand, model) {

                this.brand = brand;

                this.model = model;

        }

}

const myCar = new Car(“Subaru”, “Impreza”);
```

Error handling (Exception handling) and type safety: Python has a bunch of exception classes that allow you to catch specific error types. If an error occurs, the interpreter raises one of these exception types. If the code that produced the error is enclosed in a try-except block, we can program a specific response for an exception type. Exception types include ValueError, TypeError, KeyError, ImportError, KeyboardInterrupt, etc. If using a try except block, we can also include an else statement after the except, so we both capture the error and run additional backup code. Because python is dynamically typed, these types aren't enforced at compile time, so someone could accidentally mix types via adding.

JavaScript uses a similar system of try-catch blocks to capture exceptions, but has less built-in exception types and they are all derived from a generic type. The error messages themselves can also be less descriptive than python. Errors that happen in the common asynchronous operations that JS devs use in websites can also often fail without a message if a .catch() or try-catch isn't included with the await. As JS is also dyunamically typed, there are certain problems that can occur when mixing incompatible types, such as adding a string and an integer where "4" + 1 = "41". Python has a built in method of statically typing, but Javascript doesn't, prompting the creation of the earlier mentioned TypeScript statically typed superset.

Python:

```
try:

   result = 10 / 0

except ZeroDivisionError as e:

   print("Error:", e)
```

JS:

```
try {

   let result = 10 / 0;

} catch (e) {

   console.log("Error:", e.message);

}
```

Ease of use for AI development: Python has held the title of the primary AI language for a fairly long time now, largely because of the large ecosystem developed for it. Python has many AI, statistical, and scientific libraries such as NumPy, SciPy, pandas, TensorFlow, PyTorch, Matplotlib that make data manipulation and computing very efficient with very strong community support and documentation. The language also has a clean and easily written syntax that makes writing the code faster, and environments like Jupyter notebooks allow small cell-based execution of code and easy switching of python environments with different packages installed.

JavaScript is not very common in AI development, and as a web-based language would be usually implemented more as a deployment and interactivity tool than part of the training or model itself. Though javascript does have libraries like Tensorflow.js that allow pretrained models to be run within a browser.

Despite lacking the scientific speed and ecosystem of Python, the main strength of JS is the integration with web frameworks and HTML and CSS, which can allow advanced UI and tooling for AI applications. Another factor is that AI libraries in python are often made to operate via the computing power of the host machine, and hosting a web-based solution in JS makes this more difficult.