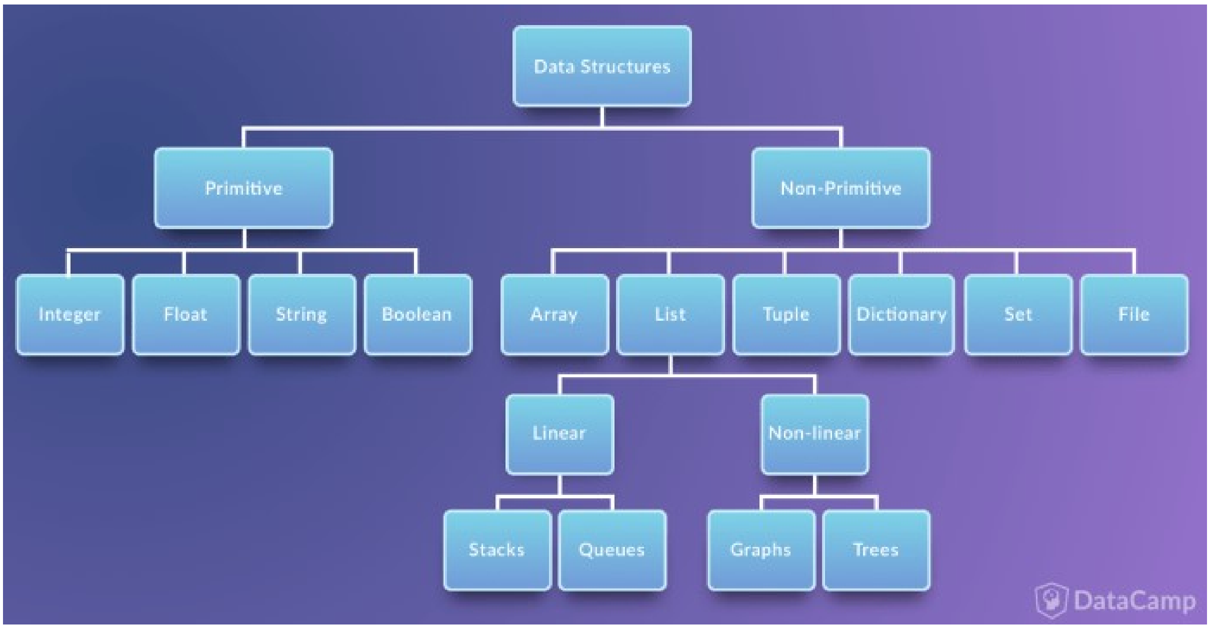
Read Dicheva & Hodge (2018). Think about an online system which you use on a daily basis. Consider how it might operate at the back-end using data structures. This will inform our discussion during next week's seminar.

Considering Asana, a planning application used at work the following points arise:

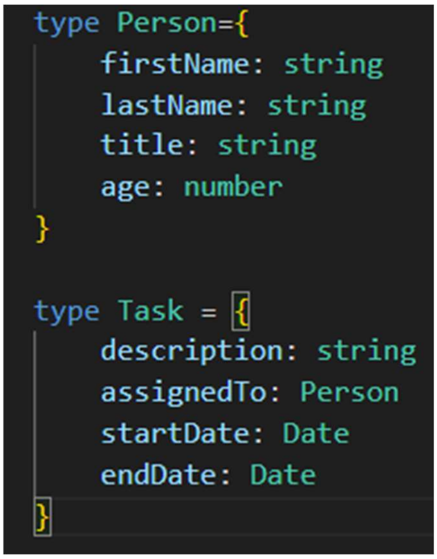
- Many user-defined data structures would exist as objects, formed from primitives as well as other complex data structures.
- Common complex data structures include arrays and associative arrays otherwise known as dictionaries.



- Likely an MVC architecture with a relational database to persist information, where Object Relational Mapping (ORM) is often used.

- **Select at least two different data structures to hold the data associated with the list of functional and non-functional requirements that you defined for Task 1.**
- **Justify your data structure choices.**
- **Select at least one academic paper, which might be similar to the work of Abeykoon et al. (2020).**
- **Use your sourced information to support your data structure choices.**

One complex or aggregated data structure would be a user-defined type which would be used to form a 'Task' for example in TypeScript:



```
type Person={
    firstName: string
    lastName: string
    title: string
    age: number
}

type Task = {
    description: string
    assignedTo: Person
    startDate: Date
    endDate: Date
}
```

Note that the user-defined type will consist of various complex and primitive types.

Complex data types are beneficial as they aggregate associated information and allow abstraction so that it is easier to form more complex structures and implement algorithms that process a lot of information at a time and move data to and from different points in a program. Furthermore, the complex data types above are actually “Associative Arrays” which provide various features, such as iterative and search properties (Kepner et al., 2020). Additionally, complex data structures can be mapped to relational database tables which improves Create, Read, Update, Delete (CRUD) queries implemented via Structured Query Language (SQL).

References:

Kepner, J., Gadepally, V., Jananthan, H., Milechin, L., Samsi, S. (2020). AI Data Wrangling with Associative Arrays.