1A.

Python is a high-level, interpreted, general-purpose programming language. Being a general-purpose language, it can be used to build almost any type of application with the right tools/libraries. Additionally, python supports objects, modules, threads, exception-handling and automatic memory management which help in modelling real-world problems and building applications to solve these problems.

2A.

Python is a general-purpose programming language that has simple, easy-to-learn syntax which emphasizes readability and therefore reduces the cost of program maintenance. Moreover, the language is capable of scripting, completely open-source and supports third-party packages encouraging modularity and code-reuse.
Its high-level data structures, combined with dynamic typing and dynamic binding, attract a huge community of developers for Rapid Application Development and deployment.

3A.

Before we understand what a dynamically typed language, we should learn about what typing is. Typing refers to type-checking in programming languages. In a strongly-typed language, such as Python, "1" + 2 will result in a type error, since these languages don't allow for "type-coercion" (implicit conversion of data types). On the other hand, a weakly-typed language, such as Javascript, will simply output "12" as result.
Type-checking can be done at two stages -

1. Static - Data Types are checked before execution.
2. Dynamic - Data Types are checked during execution.

Python being an interpreted language, executes each statement line by line and thus type-checking is done on the fly, during execution. Hence, Python is a Dynamically Typed language.

4A.

A namespace in Python ensures that object names in a program are unique and can be used without any conflict. Python implements these namespaces as dictionaries with 'name as key' mapped to a corresponding 'object as value'. This allows for multiple namespaces to use the same name and map it to a separate object. A few examples of namespaces are as follows:

- Local Namespace includes local names inside a function. the namespace is temporarily created for a function call and gets cleared when the function returns.

- Global Namespace includes names from various imported packages/ modules that is being used in the current project. This namespace is created when the package is imported in the script and lasts until the execution of the script.
- Built-in Namespace includes built-in functions of core Python and built-in names for various types of exceptions.

Lifecycle of a namespace depends upon the scope of objects they are mapped to. If the scope of an object ends, the lifecycle of that namespace comes to an end. Hence, it isn't possible to access inner namespace objects from an outer namespace.