**Q1.1 – write the answers to these questions**

* What is the difference between static and dynamic variables in python?
* In Python, all variables are dynamically typed, meaning the type is determined at runtime and can change as the program executes. However, here is an explanation considering a general perspective of static and dynamic typing:
* **Static Variables:** Typically used in statically typed languages (e.g., Java, C++), where the type of a variable is known at compile time and cannot change. In Python, class variables can be thought of as somewhat static because they are shared across all instances of the class.
* **Dynamic Variables:** In dynamically typed languages like Python, variables do not have a fixed type. A variable can be reassigned to different types of values during the program execution.
* Explain the purpose of pop, popitem, clear in a dictionary with suitable examples.
* **pop(key[, default]):** Removes the specified key and returns the corresponding value. If the key is not found, default is returned if provided; otherwise, a KeyError is raised.
* **popitem():** Removes and returns the last (key, value) pair as a tuple from the dictionary. If the dictionary is empty, a KeyError is raised.
* **clear():** Removes all items from the dictionary.
* What do you mean by FrozenSet? Explain it with suitable examples.
* frozenset: It is an immutable version of a Python set. While regular sets are mutable, and their contents can be modified at any time, frozen sets are immutable and do not allow any modifications after they are created. This makes them hashable and therefore usable as keys in a dictionary or as elements of another set.

A screen shot of a computer code

Description automatically generated

A blue square with white text

Description automatically generated with medium confidence

A screenshot of a computer program

Description automatically generated

A computer screen shot of a computer code

Description automatically generated

A screen shot of a computer

Description automatically generated

* Differentiate between mutable and immutable data types in Python and give examples of mutable and immutable data types.
* **Mutable Data Types:** These are data types whose state can be modified after they are created.
* Examples: List: my\_list = [1, 2, 3], Dictionary: my\_dict = {'a': 1, 'b': 2}
* **Immutable Data Types:** These are data types whose state cannot be modified after they are created.
* Examples: Tuple: my\_tuple = (1, 2, 3), FrozenSet: my\_frozenset = frozenset([1, 2, 3])
* What is \_\_init\_\_? Explain with an example.
* It is a special method in Python classes known as the constructor. It is automatically called when a new instance of a class is created. The \_\_init\_\_ method is used to initialize the attributes of the class.
* What is docstring in Python? Explain with an example.
* A docstring is a string literal that appears as the first statement in a module, function, class, or method definition. It is used to document the object and can be accessed using the \_\_doc\_\_ attribute.
* What are unit tests in Python?
* Unit tests are tests that verify the functionality of a specific section of code, usually at the function level. In Python, the unittest module is commonly used for creating and running unit tests.
* What is break, continue, and pass in Python?
* **break**: Terminates the loop containing it.
* **continue**: Skips the rest of the code inside the loop for the current iteration and moves to the next iteration.
* **pass**: Does nothing. It is a placeholder.
* What is the use of self in Python?
* It is a reference to the current instance of the class and is used to access variables and methods associated with the class. It must be the first parameter of any function in the class.
* What are global, protected, and private attributes in Python?
* **Global Attributes**: Accessible from anywhere in the code.
* **Protected Attributes**: Indicated by a single underscore (\_), they are intended to be accessed within the class and its subclasses.
* **Private Attributes**: Indicated by a double underscore (\_\_), they are intended to be accessed only within the class.
* What are modules and packages in Python?
* **Modules**: A module is a single Python file that can be imported into other Python programs.
* **Packages**: A package is a collection of Python modules in a directory. A package usually contains a special \_\_init\_\_.py file to indicate that the directory is a package.
* What are lists and tuples? What is the key difference between the two?
* **Lists**: Mutable, ordered collection of items.
* **Tuples**: Immutable, ordered collection of items.
* What is an Interpreted language & dynamically typed language? Write 5 differences between them?
* **Interpreted Language**: Code is executed line-by-line by an interpreter at runtime.
* **Dynamically Typed Language**: Variable types are determined at runtime.
* **Differences**:
  + **Compilation**:
    - Interpreted: Code is executed line-by-line.
    - Dynamically Typed: Type is checked at runtime.
  + **Error Detection**:
    - Interpreted: Errors are detected during execution.
    - Dynamically Typed: Type errors are detected at runtime.
  + **Flexibility**:
    - Interpreted: Allows for dynamic execution.
    - Dynamically Typed: Variables can change types.
  + **Performance**
    - Interpreted: Generally slower due to line-by-line execution.
    - Dynamically Typed: Slower type checking compared to statically typed.
* What are Dict and List comprehensions?
* **List Comprehension**: A concise way to create lists.
* **Dict Comprehension**: A concise way to create dictionaries.
* What are decorators in Python? Explain it with an example. Write down its use cases.
* **Decorators**: Functions that modify the behavior of other functions or methods. They are often used to add functionality to existing code in a clean and readable way.
* How is memory managed in Python?
* **Memory Management**: Managed by the Python memory manager, including a private heap containing all Python objects and data structures. Python uses automatic garbage collection to reclaim unused memory.
* What is lambda in Python? Why is it used?
* **Lambda**: Anonymous function defined using the lambda keyword. It is used for creating small, unnamed functions, often for short-term use.
* Explain split() and join() functions in Python.
* **split()**: Splits a string into a list of substrings based on a specified delimiter. If no delimiter is specified, it splits on whitespace.
* **join()**: Joins a list of strings into a single string with a specified delimiter.
* What are iterators, iterable & generators in Python?
* **Iterable**: Any object in Python that you can iterate over (e.g., lists, tuples, dictionaries, sets). It must have an \_\_iter\_\_() method that returns an iterator.
* **Iterator**: An object with a \_\_next\_\_() method that returns the next item in the sequence. When there are no more items, it raises a StopIteration exception.
* **Generator**: A special type of iterator created using a function with the yield keyword. Generators allow you to iterate through a sequence of values lazily, meaning they generate items one at a time and only when required.
* What is the difference between xrange and range in Python?
* **range()**: Generates a list of numbers.
* **xrange()**: Generates numbers on demand, returning an xrange object (an iterable), which is more memory efficient for large ranges.
* Pillars of OOPs.
* There are four pillars of Object-Oriented Programming (OOP):
  + **Encapsulation**: Bundling the data (attributes) and methods (functions) that operate on the data into a single unit or class and restricting access to some of the object's components.
  + **Abstraction**: Hiding the complex implementation details and showing only the necessary features of an object.
  + **Inheritance**: Mechanism by which one class (child or derived class) inherits the attributes and methods from another class (parent or base class).
  + **Polymorphism**: The ability to present the same interface for different data types. In Python, it allows functions and methods to use objects of different types through method overloading and method overriding.
* How will you check if a class is a child of another class?
* **issubclass()**: This built-in function is used to check if a class is a subclass of another class.
* How does inheritance work in Python? Explain all types of inheritance with an example.
* **Single Inheritance**: A child class inherits from one parent class.
* **Multiple Inheritance**: A child class inherits from multiple parent classes.
* **Multilevel Inheritance**: A class is derived from a class which is also derived from another class.
* **Hierarchical Inheritance**: Multiple child classes inherit from a single parent class.
* **Hybrid Inheritance**: A combination of two or more types of inheritance.
* What is encapsulation? Explain it with an example.
* **Encapsulation**: It is an OOP principle that combines data (attributes) and methods (functions) that operate on the data into a single unit or class and restricts access to some of the object's components.
* **Q1.2: Which of the following identifiers names are invalid and Why?**

A black text on a white background

Description automatically generated

* a) Serial\_no - Valid
* b) 1st\_Room - Invalid (Identifiers cannot begin with a digit)
* c) Hundred$ - Invalid (Identifiers cannot contain special characters like $)
* d) Total\_Marks - Valid
* e) total-Marks - Invalid (Identifiers cannot contain special characters like -)
* f) Total Marks - Invalid (Identifiers cannot contain spaces)
* g) True - Invalid (True is a reserved keyword in many programming languages)
* h) \_Percentag - Valid