PYTHON ASSIGNMENT-1 **Name: Chetan P Murthy**

SRN: PES2UG22EC042

1 MARK MCQ TYPE QUESTIONS:

1. **Which statement is true about Python objects and dictionary keys?**

a) Integers are mutable, so they cannot be used as dictionary keys.

b) Strings are immutable and hashable; equal strings map to the same key slot.

c) Lists are immutable and therefore ideal as dictionary keys.

d) Tuples are always hashable regardless of what they contain.

Answer: b) Strings are immutable and hashable; equal strings map to the same key slot.

2. **Which statement about common sequence methods is correct?**

a) count() returns the index of the first match, or −1 if not found.

b) index() returns the index of the first match and raises an error if not found.

c) Both count() and index() raise an error if the element is absent.

d) index() returns the total number of occurrences of the element.

Answer: b) index() returns the index of the first match and raises an error if not found.

5 mark descriptive questions with answers:

1. **Explain Python's variable scope rules (LEGB) and the roles of global and nonlocal. Use a short code example and state the exact output.**

Answer :

LEGB: Name lookup order is Local → Enclosing → Global → Built-ins.
global lets a function rebind a name in the module (global) scope.
nonlocal lets a nested function rebind a name in the nearest enclosing (non-global) scope.

Example:

```
x = 0
def outer():
   x = 1
   def inner():
      nonlocal x
      x += 2
      print("A:", x)
   inner()
   print("B:", x)
outer()
def g():
   global x
   x = 5
g()
print("C:", x)
```

Output:

A: 3

B: 3

C: 5

NOTE: inner() uses nonlocal x to modify outer's x (1→3).g() uses global x to rebind the module-level x from 0 to 5.

2. **Compare lists, tuples, and strings as Python sequences in terms of mutability, typical operations, and use-cases.**

Answer :

Mutability: Lists = mutable; Tuples & Strings = immutable.

Operations: All support indexing, slicing, iteration, membership, length; lists add mutating ops (insert/remove/sort).

Performance/semantics: Tuples are lighter and hashable (when elements are hashable) → usable as dict keys; strings optimized for text ops (search, split, join).

Use-cases: Lists for dynamic collections; tuples for fixed records/keys; strings for textual data.