Assignment 1
Python for Data Science

Name : Arun Murugappan I
SRN: PES2UG22EC027

---

1). While naming the variable, use of any special character other than underscore( ) will throw which type of error? [1 mark]
(a) Syntax error
(b) Key error
(c) Value error
(d) Index error
Answer: a

2) Let t1 = (1, 2, “tuple”, 4) and t2 = (5, 6, 7). Which of the following will give an error after the execution? [1 mark]
(a) t1.append(5)
(b) x = t2[t1[1]]
(c) t3 = t1 + t2
(d) t3 = (t1, t2)
(e) t3 = (list(t1), list(t2))
Answer: a

3) Differentiate between mutable and immutable data types in Python, with suitable examples.
Answer:

Mutable Data Types

- Definition**:** Objects whose content can be modified after they are created.
- Examples**:** Lists, Dictionaries, Sets.
- Properties**:**
    - Elements can be added, removed, or updated without creating a new object.
    - Useful when working with collections that need frequent modification.

Ex :
```
my_list = [1, 2, 3]

my_list[0] = 10       # Value modified

print(my_list)        # Output: [10, 2, 3]
```

Immutable Data Types

- Definition: Objects whose content cannot be changed once they are created.
- Examples: Strings, Tuples, Integers.
- Properties:
    - Any modification creates a new object in memory.
    - Prevents accidental changes to data, making them more secure.

Ex: my_tuple = (1, 2, 3)

my_tuple[0] = 10

Invalid because tuple is not mutable

4) Differentiate between sequential and non-sequential data types in Python with examples.

Answer:

1. Sequential Data Types

- Store elements in a specific, ordered sequence.
- Each element has a definite position (index).
- Elements can be accessed directly using their index or by iteration.
- Examples:
    - List: `[1, 2, 3]`
    - Tuple: `(10, 20, 30)`
    - String: `"Python"`
    - Range: `range(0, 5)`

2. Non-Sequential Data Types

- Do not maintain a specific order of elements.
- Elements are organized based on their values, keys, or hash properties.
- Access is usually by keys or identifiers, not indexes.
- Useful when fast lookups or uniqueness are required.
- Examples:
    - Dictionary: `{"id": 1, "name": "Arun"}`
    - Set: {5, 10, 15}