Python for Data Science

Chethan M N PES2UG22EC045

ASSIGNMENT-1

Multiple Choice Questions:

1. What will be the output of the following code snippet?

```
greetings = ”Namaste”
greetings_1 = float(greetings)
print(type(greetings_1))
```

- int
- float
- str
- Code will throw an error.

Answers:

Code will throw an error.

2.Given two variables, **j = 6** and **g = 3.3**. If both normal division and floor division operators were used to divide **j** by **g**, what would be the data type of the value obtained from the operations?

- int, int
- float, float
- float, int
- int, float

Answers:

float, float

3. How can you concatenate the strings “`data`” and “`science`” with a hyphen(-) between them?

- `“data”.join(“science”)`
- `“-”.join([“data”, “science”])`
- `“data” + “-” + “science”`
- None of the above.

Accepted Answers:

```
“-”.join([“data”, “science”])
“data” + “-” + “science”
```

4 Markers :

Q1. Explain why NumPy arrays are more efficient than Python lists for numerical operations, considering their underlying structure and how this impacts memory usage and computational speed. Provide an example of how to convert a Python list into a NumPy array and verify its data type.

Answer:

NumPy arrays are more efficient for numerical operations than Python lists primarily due to two reasons:

1. Homogeneous Data Type and Contiguous Memory Allocation: NumPy arrays store elements of the same data type (homogeneous), which allows them to allocate memory contiguously. In contrast, Python lists can store elements of different data types (heterogeneous), requiring them to store pointers to objects scattered in memory. This contiguous memory allocation in NumPy arrays leads to:
    - Reduced Memory Overhead: No need to store pointers for each element, leading to less memory consumption.
    - Improved Locality of Reference: Elements are stored close together, which significantly improves cache performance and thus speeds up operations.
2. Optimized C Implementations: NumPy operations are implemented in C, which provides much faster execution compared to Python's interpreted nature. When you perform an operation on a NumPy array, it's executed as a highly optimized C loop, whereas a similar operation on a Python list would involve Python-level loops, which are considerably slower.

Q2. Wxplain what the capitalize() method does and what the swapcase() method does.

Answer:

- capitalize() method: This method returns a copy of the string with its first character capitalized and all other characters converted to lowercase.
    strSample = 'learning is fun !'
    strSample.capitalize()
    # Out: 'Learning is fun !'
    The first letter 'l' is changed to 'L', and the rest remain lowercase.
- swapcase() method: This method returns a copy of the string where all uppercase characters are converted to lowercase, and all lowercase characters are converted to uppercase.
    strSample = 'learning is fun !'
    strSample.swapcase()
    # Out: 'LEARNING IS FUN!'
    - All lowercase letters in strSample are converted to uppercase.