PYTHON FOR DATA SCIENCE

ASSIGNMENT 1

Name-Aditi Guruprasad

SRN-PES2UG22EC009

MCQs

1. What is the output of the following expression?

   result = 7 - 2 * 3 ** 2 + 4

   print(result)

   a) 3

   b) -7

   c) -9

   d) 5

   Answer: b) -7 (Because precedence: ** → * → +/−)

2. Which keyboard shortcut in Spyder is used to execute the entire script file?

   a)F9

   b) F5

   c) Ctrl+Enter

   d) Ctrl+1

   Answer: b) F5

DESCRIPTIVE TYPE QUESTIONS

1. Explain the process of type coercion in Python with a suitable example. Mention one case where coercion is not possible.

Answer: Type coercion in Python is the process of converting an object from one data type to another using built-in functions like *int()*, *float()*, *str()*, etc. It is useful when we want to perform operations on variables of compatible types.

Example of type coercion-

```
salary = "5000"      # salary is of string type
salary = int(salary) # coerced to integer
print(salary + 2000) # Output: 7000
```

Here, the string *"5000"* is converted into an integer *5000*, allowing arithmetic operations.

Example where type coercion is not possible-

```
value = "Hello"
num = int(value)
```

This will raise a ValueError because the string *"Hello"* cannot be converted into an integer.

1. What is a variable in Python? Explain with the help of an example how variables are created and stored in memory

   Answer: A variable in Python is an identifier that is used to store a value.

   It acts as a reference (or pointer) to a memory location where the data is actually stored. Instead of directly dealing with memory addresses, Python allows us to use variable names to access and manipulate data.

   A variable is created automatically when a value is assigned using the assignment operator (=).

Variable names must follow Python naming rules (start with a letter/underscore, contain only letters, digits, underscores).
Python is dynamically typed – the type of variable is decided at runtime.

```
x = 10        # integer variable
name = "dog" # string variable
```

When we write *x = 10*, Python creates an integer object (10) in memory and the variable name *x* refers to its address.

Similarly, *name = "dog"* creates a string object *"dog"* in memory, and *name* points to it.

Variables do not store the values directly; instead, they reference the object stored in memory