1. Which of the following is NOT a valid naming convention for a variable in Python?

a) age_emp

b) AgeEmp

c) 2age

d) employee_id

Answer: c) 2age

2. Which of the following data structures can store values of different data types?

a) NumPy Array

b) Array (from array module)

c) List

d) Tuple

Answer: c) List

1. Describe the purpose of "identity operators" in Python (is **and** is not**). Provide an example for each to illustrate their use in determining the data type or memory location of variables. (Refer to document)**

Answer:

Identity operators in Python are used to compare if two objects are the same, specifically checking if they refer to the same memory location. They are primarily used to determine if two variables point to the exact same object in memory, rather than just having the same value. The two identity operators are is and is not.

- is **operator:** This operator evaluates to True if the variables on either side of the operator point to the same object (i.e., share the same memory location), and False otherwise.
    - **Example:**

Python

a = 15

b = 15

c = a

print(a is b) # Output: True (for small integers, Python often caches them)

print(a is c) # Output: True (c points to the same object as a)

list1 =

list2 =

print(list1 is list2) # Output: False (even if values are same, they are different objects in memory)

- is not **operator:** This operator evaluates to False if the variables on either side of the operator point to the same object, and True otherwise. It is the logical negation of the is operator.
    - **Example:**

Python

x = 10

y = 20

print(x is not y) # Output: True (x and y are different objects)

s1 = "hello"

s2 = "hello"

print(s1 is not s2) # Output: False (for identical string literals, Python often caches them)

list1 =

list2 =

print(list1 is not list2) # Output: True (list1 and list2 are distinct objects in memory)

2. Explain the key differences between a Python list **and a NumPy** array**. Provide an example of how to create each.**

Answer:

The key differences between a Python `list` and a NumPy `array` are:

* **Data Type Homogeneity:** A Python `list` can contain values of different data types (e.g., integers, strings, floats), as shown in `[1, 2, 'a', 'sam', 2]`. In contrast, a NumPy `array` is a grid of values, all of the same type.

* **Numerical Operations:** NumPy arrays are optimized for numerical computations and support various mathematical operations efficiently, which is not the primary purpose of Python lists.

* **Memory Efficiency:** NumPy arrays are generally more memory-efficient for storing large amounts of numerical data compared to Python lists.

Examples:

* **Creating a Python List:**

  ```python
  my_list = [1, 2, 'a', 'sam', 2]
  print(my_list)
  # Output: [1, 2, 'a', 'sam', 2]
  ```

* **Creating a NumPy Array:**

  ```python
  import numpy as np
  my_list_for_array =
  numpy_array = np.array(my_list_for_array, dtype=int)
  print(numpy_array)
  # Output:
  ```