Assignment

Name : Anuj Kumar

Q1. Create one variable containing following type of data:

(i) String

(ii) list

(iii) float

(iv) tuple

Answer

string_variable = "Hello, Anuj"

list_variable = [1, 2, 3, 4, 5]

float_variable = 89.3

tuple_variable = (10, 20, 30)

Q2. Given are some following variables containing data: (i) var1 = ‘ ‘ (ii) var2 = ‘[DS , ML , Python]’ (iii) var3 = [‘DS’ , ’ML’ , ‘Python’] (iv) var4 = 1.

Answer

Let's analyze the data types of the given variables:

(i) var1 = ‘ ‘

The value assigned to `var1` is a string containing a single space character.

Data Type: String

(ii) var2 = ‘[DS , ML , Python]’

The value assigned to `var2` is a string containing the characters '[DS , ML , Python]'.

Data Type: String

(iii) var3 = [‘DS’ , ’ML’ , ‘Python’]

The value assigned to `var3` is a list containing three string elements: 'DS', 'ML', and 'Python'.

Data Type: List

(iv) var4 = 1.

The value assigned to `var4` is a floating-point number.

Data Type: Float

So, the data types of the given variables are:

(i) var1: String

(ii) var2: String

(iii) `var3`: List

(iv) `var4`: Float

Q3. Explain the use of the following operators using an example: (i) / (ii) % (iii) // (iv) **

Answer

(i)

```
a = 10
b = 2
result = a / b
print(result)  # Output will be 5.0
```

(ii)

```
a = 10
b = 3
result = a % b
print(result)  # Output will be 1
```

(iii)

```
a = 10
b = 3
result = a // b
print(result)  # Output will be 3
```

(iv)

```
base = 2
exponent = 3
result = base ** exponent
print(result)  # Output will be 8
```

Q4. Create a list of length 10 of your choice containing multiple types of data. Using for loop print the element and its data type.

Answer

```
# Create a list with 10 elements of different data types

my_list = [10, "Hello", 3.14, True, [1, 2, 3], (1, 2), {"name": "John", "age": 30}, None, "Python", False]

# Using a for loop to print each element and its data type

for element in my_list:

    print(f"Element: {element}, Data Type: {type(element)}")
```

Q5. Using a while loop, verify if the number A is purely divisible by number B and if so then how many times it can be divisible.

Answer

```
def find_divisions(a, b):

    count = 0

    while a >= b and a % b == 0:

        a = a // b

        count += 1

    return count

# Test example

number_a = 100

number_b = 5

result = find_divisions(number_a, number_b)

print(f"{number_a} is purely divisible by {number_b} and can be divided {result} times.")
```

Q6. Create a list containing 25 int type data. Using for loop and if-else condition print if the element is divisible by 3 or not.

```
# Create a list containing 25 integer type data

my_list = [2, 15, 7, 9, 36, 10, 27, 14, 21, 16, 3, 19]


# Using for loop and if-else condition to check divisibility by 3

for num in my_list:

    if num % 3 == 0:
        print(f"{num} is divisible by 3.")
    else:
        print(f"{num} is not divisible by 3.")
```

Q7. What do you understand about mutable and immutable data types? Give examples for both showing this property.

In programming, data types are classified into two categories: mutable and immutable. These categories define whether the data stored in the variable can be changed after it is created or not.

1. Mutable Data Types:

Mutable data types are those that allow modification after their creation. This means you can change their content or values without creating a new instance of the variable. When you modify a mutable object, you are modifying the same object in memory.

Examples of mutable data types in Python:

- Lists: Lists can be modified by adding, removing, or updating elements.

- Dictionaries: Dictionary items can be added, removed, or modified.

- Sets: Sets can have elements added or removed.

Example of a mutable data type:

```
# Lists are mutable
my_list = [1, 2, 3]
print(my_list)  # Output: [1, 2, 3]
# Modifying the list
my_list[0] = 10
print(my_list)  # Output: [10, 2, 3]
```

2. Immutable Data Types:

Immutable data types, on the other hand, are those that do not allow modification after they are created. When you try to change the value of an immutable object, a new instance with the updated value is created in memory.

Examples of immutable data types in Python:

- Strings: Strings cannot be modified once created. You need to create a new string if you want to make changes.

- Tuples: Tuples are fixed-size and their elements cannot be changed after creation.

- Integers, floats, booleans: These primitive data types are also immutable.

Example of an immutable data type:

```
# Strings are immutable

my_string = "Hello"

print(my_string)  # Output: Hello

# Trying to modify the string (Error: strings are immutable)

my_string[0] = "h"  # TypeError: 'str' object does not support item assignment
```