In [3]:

```
# Integer
print("Integer Data Type:")
int_var = 42
print("Value:", int_var)
print("Type:", type(int_var))  # Output: <class 'int'>
print("Addition:", int_var + 8)  # Output: 50
print("Multiplication:", int_var * 2)  # Output: 84

# Float
print("\nFloat Data Type:")
float_var = 3.14
print("Value:", float_var)
print("Type:", type(float_var))  # Output: <class 'float'>
print("Addition:", float_var + 1.86)  # Output: 5.0
print("Multiplication:", float_var * 2)  # Output: 6.28

# String
print("\nString Data Type:")
str_var = "Hello, World!"
print("Value:", str_var)
print("Type:", type(str_var))  # Output: <class 'str'>
print("Length:", len(str_var))  # Output: 13
print("Concatenation:", str_var + " How are you?")  # Output: Hello, World! How are you?
print("Uppercase:", str_var.upper())  # Output: HELLO, WORLD!
print("Substring:", str_var[7:12])  # Output: World

# List
print("\nList Data Type:")
list_var = [1, 2, 3, 4, 5]
print("Value:", list_var)
print("Type:", type(list_var))  # Output: <class 'list'>
print("Length:", len(list_var))  # Output: 5
print("Indexing (first element):", list_var[0])  # Output: 1
print("Slicing (last two elements):", list_var[-2:])  # Output: [4, 5]
list_var.append(6)
print("After appending 6:", list_var)  # Output: [1, 2, 3, 4, 5, 6]
list_var.remove(3)
print("After removing 3:", list_var)  # Output: [1, 2, 4, 5, 6]

# Tuple
print("\nTuple Data Type:")
tuple_var = (1, 2, 3, 4, 5)
print("Value:", tuple_var)
print("Type:", type(tuple_var))  # Output: <class 'tuple'>
print("Length:", len(tuple_var))  # Output: 5
print("Indexing (second element):", tuple_var[1])  # Output: 2
print("Slicing (first three elements):", tuple_var[:3])  # Output: (1, 2, 3)

# Dictionary
print("\nDictionary Data Type:")
dict_var = {'name': 'Alice', 'age': 30, 'city': 'New York'}
print("Value:", dict_var)
print("Type:", type(dict_var))  # Output: <class 'dict'>
print("Keys:", dict_var.keys())  # Output: dict_keys(['name', 'age', 'city'])
print("Values:", dict_var.values())  # Output: dict_values(['Alice', 30, 'New York'])
print("Get 'name':", dict_var.get('name'))  # Output: Alice
dict_var['age'] = 31
print("After updating age:", dict_var)  # Output: {'name': 'Alice', 'age': 31, 'city': '
del dict_var['city']
print("After deleting 'city':", dict_var)  # Output: {'name': 'Alice', 'age': 31}

# Set
print("\nSet Data Type:")
set_var = {1, 2, 3, 4, 5}
```

```
print("Value:", set_var)
print("Type:", type(set_var))  # Output: <class 'set'>
print("Length:", len(set_var))  # Output: 5
print("Union with {4, 5, 6, 7}:", set_var.union({4, 5, 6, 7}))  # Output: {1, 2, 3, 4, 5
print("Intersection with {3, 4, 5, 6}:", set_var.intersection({3, 4, 5, 6}))  # Output:
set_var.add(6)
print("After adding 6:", set_var)  # Output: {1, 2, 3, 4, 5, 6}
set_var.discard(1)
print("After discarding 1:", set_var)  # Output: {2, 3, 4, 5, 6}
```

```
Integer Data Type:
Value: 42
Type: <class 'int'>
Addition: 50
Multiplication: 84

Float Data Type:
Value: 3.14
Type: <class 'float'>
Addition: 5.0
Multiplication: 6.28

String Data Type:
Value: Hello, World!
Type: <class 'str'>
Length: 13
Concatenation: Hello, World! How are you?
Uppercase: HELLO, WORLD!
Substring: World

List Data Type:
Value: [1, 2, 3, 4, 5]
Type: <class 'list'>
Length: 5
Indexing (first element): 1
Slicing (last two elements): [4, 5]
After appending 6: [1, 2, 3, 4, 5, 6]
After removing 3: [1, 2, 4, 5, 6]

Tuple Data Type:
Value: (1, 2, 3, 4, 5)
Type: <class 'tuple'>
Length: 5
Indexing (second element): 2
Slicing (first three elements): (1, 2, 3)

Dictionary Data Type:
Value: {'name': 'Alice', 'age': 30, 'city': 'New York'}
Type: <class 'dict'>
Keys: dict_keys(['name', 'age', 'city'])
Values: dict_values(['Alice', 30, 'New York'])
Get 'name': Alice
After updating age: {'name': 'Alice', 'age': 31, 'city': 'New York'}
After deleting 'city': {'name': 'Alice', 'age': 31}

Set Data Type:
Value: {1, 2, 3, 4, 5}
Type: <class 'set'>
Length: 5
Union with {4, 5, 6, 7}: {1, 2, 3, 4, 5, 6, 7}
Intersection with {3, 4, 5, 6}: {3, 4, 5}
After adding 6: {1, 2, 3, 4, 5, 6}
After discarding 1: {2, 3, 4, 5, 6}
```

```
In [ ]:
```