Functions Return Multiple Values: A function in Python can also return **Multiple values.**
[Note: Multiple values cannot be returned from a function in C, C++, or Java. **]**

1. **Return value as Tuple** - A Tuple is a sequence of items separated by a comma with or without (). Tuples are Ordered, Immutable, Indexed, Allows Duplicates.

```
def detailsTuple():
    semester = "CIT Sem-2 2023"
    students = 600
    #return semester, students # Return a tuple without ()
    return (semester, students)  # Return a tuple with ()
# Returns as Tuple (Method Call)
sem, std = detailsTuple() # Assigning returned tuple
print(sem, std)
```
Output: CIT Sem-2 2023 600

2. **Return value as List** - List is a sequence/collection of values of different data types enclosed in [ ]. Tuples are Ordered, Mutable, Indexed, Allows Duplicates.

```
def detailsList():
    semester = "CIT Sem-2 2023"
    students = 600
    return [semester, students] # Returns a list
# Returns as List   (Method Call)
slist = detailsList()
print(slist)
```
Output: ['CIT Sem-2 2023', 600]

3. **Return value as Dict** - A Dictionary is a sequence/collection of Key-Value pairs of different data types enclosed in { }. Dict is Ordered, Mutable, Not-Indexed, No Duplicate Keys but Duplicate Values are allowed.

```
def detailsDict():
    d = dict()
    d["semester"] = "CIT Sem-2 2023"
    d["students"] = 600
    return d # Returns the dictionary
# Returns as Dictionary   (Method Call)
sdict = detailsDict() # Assigning returned dictionary
print(sdict)
```
Output: {'semester': 'CIT Sem-2 2023', 'students': 600}

4. **Return value as Object -** We can create a class (similar to a struct in C) to hold multiple values and return an object of the class.

```
# Function returns multiple values using Class-Object
class FirstYear:
    def __init__(self):
        self.semester = "CIT Sem-2 2023"
        self.students = 600

# Method returns multiple values using Class-Object
def details():
    return FirstYear()  # Returns Class-Object

s = details()
print(s.semester)
print(s.students)
```

Output:
```
CIT Sem-2 2023
600
```

5. **Return value using yield -** The yield keyword generates a sequence of values, one value at a time. To return multiple values from a generator function, you can use the **yield** keyword to yield each value in each turn and continues until the generator function completes execution or encounters a return statement.

```
# Function returns multiple values using yield
def get_data():
    yield 522007
    yield 'CIT'
    yield [60,80,100]

# Method call
data = get_data()
print(next(data))  # Prints 522007
print(next(data))  # Prints 'CIT'
print(next(data))  # Prints [60,80,100]
```

Output:
```
522007
CIT
[60, 80, 100]
```