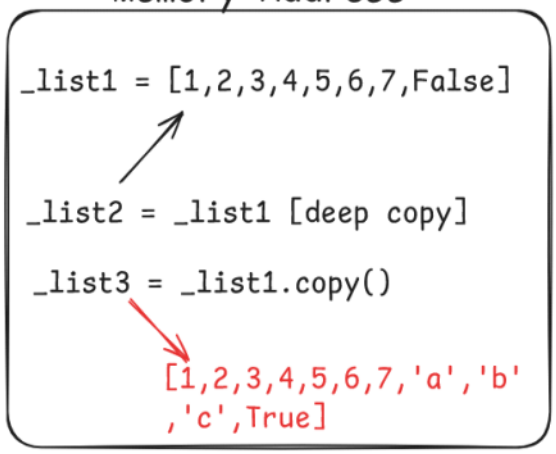
Cont. Data Structures-I

🎯 Session Objectives

- Understand the meaning of data structures and why we use them.
- Common data structures in Python
- Understand what lists are.
- Understand common methods and operations associated with lists.
- Understand the meaning of list comprehension
- Understand what tuples are.
- Understand common methods and operations associated with tuples.
- Understand the Comparison between Lists and Tuples

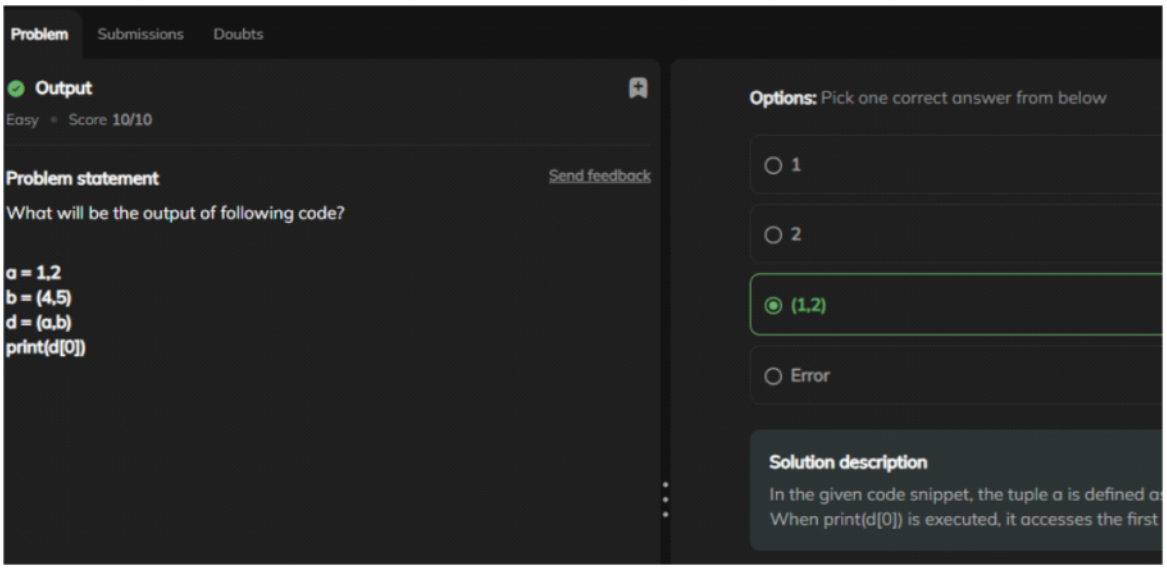
Memory Address



[Shallow Copy]

```
squared = [i**2 for i in range(1,6)] # range(1,6) [1,2,3,4,5]
```

[1, 4, 9, 16, 25]



Memory

```
a = (1,2)
b = (4,5)
d = (a,b)
print(d[0])
print(a) # (1,2)
```

```
# Comparison List '=='
_list1 = [1,2,3,4,5]
_list2 = [1,2,3]
result = (_list1 == _list2) # boolean ['False']
print(result)
```

```
False
```

```
# Comparison List '=='
_list1 = [1,2,3,4,5]
_list2 = [1,2,3,4,5]
result = (_list1 == _list2) # boolean ['True']
print(result)
```

```
True
```

```
# Comparison List '=='
_list1 = [1,2,3,4,5,6,7]
_list2 = [1,2,3,4,5]
result = (_list1 != _list2) # boolean ['True']
print(result)
```

```
True
```

```
# Comparison List '=='
_list1 = [1,2,3,4,5,6,7]
_list2 = [1,2,3,4,5]
result = (_list1 >= _list2) # boolean ['True']
print(result)
```

```
True
```

```python
# Comparison List '=='
_list1 = [1,2,3,4,5,6,7]
_list2 = [1,2,3,4,5]
result = (_list1 <= _list2) # boolean ['False']
print(result)
```

```
False
```

```python
# Comparison List '=='
_list1 = [1,2,3,4,5,6,7]
_list2 = ['1','2','3','4','5']
result = (_list1 <= _list2) # TypeError: '<=' not supported between instances of 'int' and 'str'
print(result)
```

```python
# Comparison List '=='
_list1 = ['a','b','c']
_list2 = ['1','2','3','4','5']
result = (_list1 >= _list2) # "True"
print(result)
```

```
True
```

Removing items from a list:

1. Remove - Remove the 1st occurrence of an element
2. Pop - Removes the element by its index, return values can be stored in a new variable.
3. del - Delete an elements by index or slicing
4. clear - Empties the list

```python
country_list =['India', 'America', 'Russia', 'China', 'Canada', 'Japan',
               'Vietnam', 'Sri-Lanka', 'France', 'Singapore', 'Australia',
               'Spain', 'New Zealand', 'Finland']
country_list.remove('Spain')
print(country_list)
```

```
['India', 'America', 'Russia', 'China', 'Canada', 'Japan', 'Vietnam', 'Sri-Lanka', 'France', 'Singapore', 'Aus
tralia', 'New Zealand', 'Finland']
```

```python
country_list.remove('Finland')
print(country_list)
```

```
['India', 'America', 'Russia', 'China', 'Canada', 'Japan', 'Vietnam', 'Sri-Lanka', 'France', 'Singapore', 'Aus
tralia', 'New Zealand']
```

```python
country_list.remove('UAE') # ValueError: list.remove(x): x not in list
print(country_list)
```

```
if 'UAE' in country_list: # Membership Operator [Returns Boolean]
    country_list.remove('UAE')
else:
    print('UAE is not present in the country list.')
```

```
UAE is not present in the country list.
```

```
# pop() - pop the elements by index, popped element can be stored in new_variable
pop_country = country_list.pop(-1) # 'New-Zealand'
print(pop_country) # 'New-Zealand'
print(country_list) # ['India' to 'Aus']
```

```
New Zealand
['India', 'America', 'Russia', 'China', 'Canada', 'Japan', 'Vietnam', 'Sri-Lanka', 'France', 'Singapore', 'Aus
tralia']
```

```
pop_country = country_list.pop() #
print(pop_country) #
print(country_list) #
```

```
Australia
['India', 'America', 'Russia', 'China', 'Canada', 'Japan', 'Vietnam', 'Sri-Lanka', 'France', 'Singapore']
```

```
pop_country = country_list.pop(7) # 'Sri-Lanka'
print(pop_country) # 'Sri-Lanka'
print(country_list) # [except Sri-Lanka]
```

```
Sri-Lanka
['India', 'America', 'Russia', 'China', 'Canada', 'Japan', 'Vietnam', 'France', 'Singapore']
```

```
# del required a call of a list with specific index or slicing to delete an element or a list of elements
del country_list[3] # 'China'
print(country_list)
```

```
['India', 'America', 'Russia', 'Canada', 'Japan', 'Vietnam', 'France', 'Singapore']
```

```
del country_list[-3:] # ['Vietnam', 'France', 'Singapore']
print(country_list)
```

```
['India', 'America', 'Russia', 'Canada', 'Japan']
```

```
pop_element = country_list.pop() # 'Japan'
print(pop_element) # 'Japan'
print(country_list) # ['India', 'America', 'Russia', 'Canada']
```

```
Japan
['India', 'America', 'Russia', 'Canada']
```

```
country_list.append(pop_element) # 'Japan'
print(country_list) # ['India', 'America', 'Russia', 'Canada','Japan']
```

```
['India', 'America', 'Russia', 'Canada', 'Japan']
```

```
pop_element = country_list.pop(2,4) # TypeError: pop expected at most 1 argument, got 2
```

```
# .clear() -> It will empty the original Lists
country_list.clear()
```

```
print(country_list) # []
```

```
[]
```

```
country_list =['India', 'America', 'Russia', 'China', 'Canada', 'Japan',
               'Vietnam', 'Sri-Lanka', 'France', 'Singapore', 'Australia',
               'Spain', 'New Zealand', 'Finland']
del country_list # -> Remove the Complete List
print(country_list) # NameError: name 'country_list' is not defined
```

```
# Sorting w.r.t [ASCII]
# Sorting[Ascending] & Reverse() [::-1]
country_list =['India', 'America', 'Russia', 'China', 'Canada', 'Japan',
               'Vietnam', 'Sri-Lanka', 'france', 'singapore', 'australia',
               'Spain', 'New Zealand', 'Finland']
country_list.sort()
print(country_list)
```

```
['America', 'Canada', 'China', 'Finland', 'India', 'Japan', 'New Zealand', 'Russia', 'Spain', 'Sri-Lanka', 'Vi
etnam', 'australia', 'france', 'singapore']
```

```
country_list =['India', 'America', 'Russia', 'China', 'Canada', 'Japan',
               'Vietnam', 'Sri-Lanka', 'france', 'singapore', 'australia',
               'Spain', 'New Zealand', 'Finland']
country_list.sort()
print(country_list[::-1]) # Reverse the sorted list in Descending Order
```

```
['singapore', 'france', 'australia', 'Vietnam', 'Sri-Lanka', 'Spain', 'Russia', 'New Zealand', 'Japan', 'Indi
a', 'Finland', 'China', 'Canada', 'America']
```

```
country_list =['India', 'America', 'Russia', 'China', 'Canada', 'Japan',
               'Vietnam', 'Sri-Lanka', 'france', 'singapore', 'australia',
               'Spain', 'New Zealand', 'Finland']
country_list.sort(reverse = True) # Descending Order
print(country_list)
```

```
['singapore', 'france', 'australia', 'Vietnam', 'Sri-Lanka', 'Spain', 'Russia', 'New Zealand', 'Japan', 'Indi
a', 'Finland', 'China', 'Canada', 'America']
```

```
.sort() # Ascending order
.sort(reverse = True) # Descending order
```

```
country_list =['India', 'America', 'Russia', 'China', 'Canada', 'Japan',
               'Vietnam', 'Sri-Lanka', 'france', 'singapore', 'australia',
               'Spain', 'New Zealand', 'Finland']
country_list.sort(key = str.lower) # Now, Every Element is in lower case and then sorted
print(country_list)
```

```
['America', 'australia', 'Canada', 'China', 'Finland', 'france', 'India', 'Japan', 'New Zealand', 'Russia', 's
ingapore', 'Spain', 'Sri-Lanka', 'Vietnam']
```

```
country_list =['India', 'America', 'Russia', 'China', 'Canada', 'Japan',
               'Vietnam', 'Sri-Lanka', 'france', 'singapore', 'australia',
               'Spain', 'New Zealand', 'Finland']
country_list.sort(key = str.lower) # Now, Every Element is in Lower case and then sorted
print(country_list[::-1]) # reverse the order
```

```
['Vietnam', 'Sri-Lanka', 'Spain', 'singapore', 'Russia', 'New Zealand', 'Japan', 'India', 'france', 'Finland',
'China', 'Canada', 'australia', 'America']
```

```
country_list =['India', 'America', 'Russia', 'China', 'Canada', 'Japan',
               'Vietnam', 'Sri-Lanka', 'france', 'singapore', 'australia',
               'Spain', 'New Zealand', 'Finland']
country_list.sort(key = str.lower , reverse = True) # Now, Every Element is in Lower case and then sorted
print(country_list) # reverse the order
```

```
['Vietnam', 'Sri-Lanka', 'Spain', 'singapore', 'Russia', 'New Zealand', 'Japan', 'India', 'france', 'Finland',
'China', 'Canada', 'australia', 'America']
```

```
country_list =['India', 'America', 'Russia', 'China', 'Canada', 'Japan',
               'Vietnam', 'Sri-Lanka', 'france', 'singapore', 'australia',
               'Spain', 'New Zealand', 'Finland']
country_list.sort(reverse = True , key = str.upper) # Now, Every Element is in Lower case and then sorted
print(country_list) # reverse the order
```

```
['Vietnam', 'Sri-Lanka', 'Spain', 'singapore', 'Russia', 'New Zealand', 'Japan', 'India', 'france', 'Finland',
'China', 'Canada', 'australia', 'America']
```

```
# .reverse() -> The Original List
country_list =['India', 'America', 'Russia', 'China', 'Canada', 'Japan',
               'Vietnam', 'Sri-Lanka', 'france', 'singapore', 'australia',
               'Spain', 'New Zealand', 'Finland']
country_list.reverse()
print(country_list)
```

```
['Finland', 'New Zealand', 'Spain', 'australia', 'singapore', 'france', 'Sri-Lanka', 'Vietnam', 'Japan', 'Cana
da', 'China', 'Russia', 'America', 'India']
```

```
# 'Joining' strings from a list
word_list = ['Python', 'is', 'an', 'awesome', 'programming', 'language']
final_statement = ' '.join(word_list)
print(final_statement)
print(type(final_statement))
```

```
Python is an awesome programming language
<class 'str'>
```

```
char_list = ['P','y','t','h','o','n',' ','P','r','o','g','r','a','m','m','i','n','g','!']
result = ''.join(char_list)
print(result)
```

```
Python Programming!
```

```python
# .count() -> Calculating the occurrence of an element in a list.
country_list =['America','India', 'Russia', 'China', 'Canada', 'Japan',
               'Vietnam', 'Sri-Lanka','India', 'france', 'singapore', 'australia',
               'Spain', 'New Zealand', 'Finland', 'India','Japan','America']
count_India = country_list.count('India')
print(count_India) #3
```

```
3
```

```python
# 1st Occurrence
first_= country_list.index('India')
print(first_)
```

```
1
```

```python
# 2nd Occurrence [8 index]
second_= country_list.index('India',first_ +1) # .index('search' , start , stop)
print(second_)
```

```
8
```

```python
# 3rd Occurrence [15 index]
third_ =  country_list.index('India',second_ +1) # .index('search' , start , stop)
print(third_)
```

```
15
```

```python
num_list = [1,2,2,3,4,1,1,3,2,3,22,33,2,2,1,2,3,2,212,2,2,12,2,1,2,1,2]
print(num_list.count(2))
print(num_list.count(1))
```

```
12
6
```

```python
# Deep Copy[Same Memory] Vs Shallow Copy[Independent Memory]
# Identity Operator 'is'/'is not'[(id) Memory Address]
# .copy() [Shallow Copy]
country_list =['America','India', 'Russia', 'China', 'Canada', 'Japan',
               'Vietnam', 'Sri-Lanka','India', 'france', 'singapore', 'australia',
               'Spain', 'New Zealand', 'Finland', 'India','Japan','America']
new_country_list = country_list.copy() # Shallow Copy
print(id(country_list))
print(id(new_country_list)) # Differnt id
```

```
2400454867648
2400438032128
```

```python
del new_country_list[-3:]
print(new_country_list)
```

```
['America', 'India', 'Russia', 'China', 'Canada', 'Japan', 'Vietnam', 'Sri-Lanka', 'India', 'france', 'singapo
re', 'australia', 'Spain', 'New Zealand', 'Finland']
```

```
print(country_list)
```

```
['America', 'India', 'Russia', 'China', 'Canada', 'Japan', 'Vietnam', 'Sri-Lanka', 'India', 'france', 'singapo
re', 'australia', 'Spain', 'New Zealand', 'Finland', 'India', 'Japan', 'America']
```

```
print(id(country_list))
print(id(new_country_list))
```

```
2400454867648
2400438032128
```

```
# Deep Copy()
weekday_list = ['Mon','Tue','Wed','Thurs','Fri']
weekend_list = ['Sat','Sun']
week_list = weekday_list + weekend_list
new_week_list = week_list # Deep Copy
print(id(week_list))
print(id(new_week_list)) # Same Memory Address
```

```
2400469491648
2400469491648
```

```
# Before
print(week_list)
print(new_week_list)
```

```
['Mon', 'Tue', 'Wed', 'Thurs', 'Fri', 'Sat', 'Sun']
['Mon', 'Tue', 'Wed', 'Thurs', 'Fri', 'Sat', 'Sun']
```

```
del new_week_list[-3:] # removing 'Fri','Sat','Sun'
# After
print(week_list)
print(new_week_list)
```

```
['Mon', 'Tue', 'Wed', 'Thurs']
['Mon', 'Tue', 'Wed', 'Thurs']
```

```
# Using slicing [:] # Shallow Copy
country_list =['America','India', 'Russia', 'China', 'Canada', 'Japan',
               'Vietnam', 'Sri-Lanka','India', 'france', 'singapore', 'australia',
               'Spain', 'New Zealand', 'Finland', 'India','Japan','America']
new_country_list = country_list[:] # Shallow Copy
print(id(country_list))
print(id(new_country_list)) # Differnt id
```

```
2400469595136
2400469592576
```

```
# Using slicing [:] # Shallow Copy
country_list =['America','India', 'Russia', 'China', 'Canada', 'Japan',
               'Vietnam', 'Sri-Lanka','India', 'france', 'singapore', 'australia',
               'Spain', 'New Zealand', 'Finland', 'India','Japan','America']
new_country_list = country_list[-3:] # Shallow Copy
print(id(country_list))
print(id(new_country_list)) # Differnt id
```

```
2400469393408
2400469396480
```

```python
# Using list() Constructor (Shallow Copy)
country_list =('America','India', 'Russia', 'China', 'Canada', 'Japan',
               'Vietnam', 'Sri-Lanka','India', 'france', 'singapore', 'australia',
               'Spain', 'New Zealand', 'Finland', 'India','Japan','America')
new_country_list = list(country_list) # Shallow Copy
print(id(country_list))
print(id(new_country_list)) # Differnt id
```

```
2400469118272
2400469388224
```

```python
# Index() -> Its a method to find the general position of first occurrence of a specified value in a list.
country_list = list(('America','India', 'Russia', 'China', 'Canada', 'Japan',
               'Vietnam', 'Sri-Lanka','India', 'france', 'singapore', 'australia',
               'Spain', 'New Zealand', 'Finland', 'India','Japan','America'))
print(country_list.index('India')) # 1
print(country_list.index('Japan')) # 5
print(country_list.index('America')) # 0
# print(country_list.index('Australia')) # ValueError: 'Australia' is not in list
print(country_list.index('America',5)) # 17
print(country_list.index('Japan',6)) # 16
```

```
1
5
0
17
16
```

```python
# List Comprehension [Short Hand Property]
# list[Iterable] , 'i' is an iterator
squared = [i**2 for i in range(1,6)] # range(1,6) [1,2,3,4,5]
print(squared) # [1,4,9,16,25]
```

```
[1, 4, 9, 16, 25]
```

```python
# range(start, stop, step)
squared = [i**3 for i in range(2,11,2)] # range(2,11,2) [2,4,6,8,10]
print(squared) # [8,64,216,512,1000]
```

```
[8, 64, 216, 512, 1000]
```

What exactly is a Tuples?

- Ordered Collection (items maintains its positions)
- It can store different data types togethers
- But the Only Big Difference here is: Immutability(Once created, their elements can't be changed, added to, or removed.)

```
_tuple = ()
print(type(_tuple))
```

```
<class 'tuple'>
```

```
_tuple = (1,2,'3','4',False,True,299.99,'Coding','k')
print(_tuple)
```

```
(1, 2, '3', '4', False, True, 299.99, 'Coding', 'k')
```

```
_tuple = 1,2,3,4,5,6,7,8,9
print(_tuple) # (1,2,3,4,5,6,7,8,9)
print(type(_tuple)) # <'tuple'>
```

```
(1, 2, 3, 4, 5, 6, 7, 8, 9)
<class 'tuple'>
```

```
# tuple() Constructor
country_list = ['America','India', 'Russia', 'China', 'Canada', 'Japan',
                'Vietnam', 'Sri-Lanka','India', 'france', 'singapore', 'australia',
                'Spain', 'New Zealand', 'Finland', 'India','Japan','America']
country_tuple = tuple(country_list) # Shallow Copy
print(country_tuple)
print(id(country_tuple))
```

```
('America', 'India', 'Russia', 'China', 'Canada', 'Japan', 'Vietnam', 'Sri-Lanka', 'India', 'france', 'singapo
re', 'australia', 'Spain', 'New Zealand', 'Finland', 'India', 'Japan', 'America')
2400468790112
```

```
print(country_list)
print(id(country_list))
```

```
['America', 'India', 'Russia', 'China', 'Canada', 'Japan', 'Vietnam', 'Sri-Lanka', 'India', 'france', 'singapo
re', 'australia', 'Spain', 'New Zealand', 'Finland', 'India', 'Japan', 'America']
2400468996416
```

```
char_tuple = tuple('Python') # String
print(char_tuple) # ('P','y','t','h','o','n')
```

```
('P', 'y', 't', 'h', 'o', 'n')
```

```
_tuple = tuple(('Python',)) #Tuple
print(_tuple) # ('Python',)
```

```
('Python',)
```

```
# Nested Tuple
nested_list = [
    [1,2,3,4,5],
    [0,2,4,6,8],
    [1,3,5,7,9]
]
nested_tuple = tuple(nested_list) # 'Shallow'
print(nested_tuple)
```

```
([1, 2, 3, 4, 5], [0, 2, 4, 6, 8], [1, 3, 5, 7, 9])
```

```
print(type(nested_tuple))
```

```
<class 'tuple'>
```

```
print(type(nested_tuple[0]))
```

```
<class 'list'>
```

```
nested_tuple[-1]
```

```
[1, 3, 5, 7, 9]
```

```
# List as an element in a tuple is mutuable
# Although Tuple itself is immutable
nested_tuple[0].append(6) # [1,2,3,4,5,6]
print(nested_tuple)
```

```
([1, 2, 3, 4, 5, 6], [0, 2, 4, 6, 8], [1, 3, 5, 7, 9])
```

```
nested_tuple = (('Mon','Tues','Wed'),('Thurs','Fri','Sat','Sun'))
print(type(nested_tuple))
print(type(nested_tuple[0]))
```

```
<class 'tuple'>
<class 'tuple'>
```

```
# Tuple as an elements will stick to its property[Immutability], can't be updated
nested_tuple[0].append('Sun') # AttributeError: 'tuple' object has no attribute 'append'
```

```
# Indexing & Slicing [Tuples]
country_list = ['America','India', 'Russia', 'China', 'Canada', 'Japan',
                'Vietnam', 'Sri-Lanka','India', 'france', 'singapore', 'australia',
                'Spain', 'New Zealand', 'Finland', 'India','Japan','America']
country_tuple = tuple(country_list)
print(country_tuple[-1]) # 'America'
print(country_tuple[-5]) # 'New Zealand'
print(country_tuple[4:7]) # (Canada', 'Japan', 'Vietnam')
print(country_tuple[4:7:2]) # (Canada', 'Vietnam')
print(country_tuple[::-2]) # Reverse the tuple with alternative elements
```

```
America
New Zealand
('Canada', 'Japan', 'Vietnam')
('Canada', 'Vietnam')
('America', 'India', 'New Zealand', 'australia', 'france', 'Sri-Lanka', 'Japan', 'China', 'India')
```

```
# Finding the length of a tuple
country_list = ['America','India', 'Russia', 'China', 'Canada', 'Japan',
                'Vietnam', 'Sri-Lanka','India', 'france', 'singapore', 'australia',
                'Spain', 'New Zealand', 'Finland', 'India','Japan','America']
country_tuple = tuple(country_list)
print(len(country_tuple))
```

```
18
```

```
# len(), min(), max(), sum()
num_list = [11,22,33,44,55,66,77,88,99,110,False,True] # False-0, True-1
num_tuple = tuple(num_list) # tuple ['Shallow Copy']
print(len(num_tuple)) # 12
print(min(num_tuple)) # False[0]
print(max(num_tuple)) # 110
print(sum(num_tuple)) # 606
```

```
12
False
110
606
```