Fundamentals of Python

PYTHON LIST

Shruti Bharat

@shrutibharat0105

Table of Contents

List

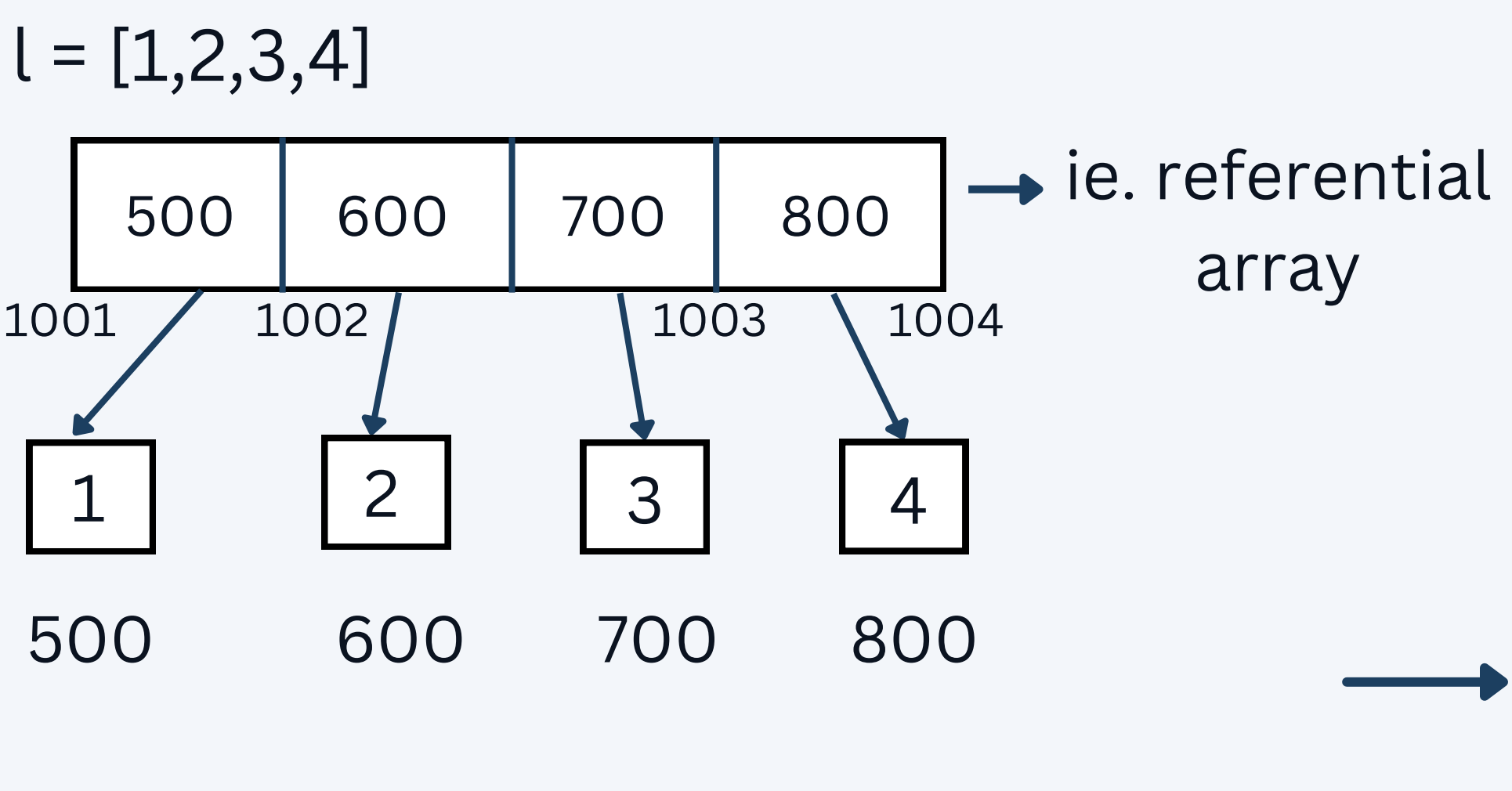
A list is a data structure where you can store multiple items under one name.

Lists act like dynamic arrays, meaning you can add more items on the fly.

Array Vs List

1. Fixed **Vs** Dynamic
2. Homogenous **Vs** Heterogeneous
3. Fast **Vs** Slow
4. Less memory **Vs** More memory

Memory allocation



Creating a list

```python
# Creating an empty list
empty_list = []
print("Empty list:", empty_list)  # Output: []

# Creating a 1D list
oneD_list = [1, 2, 3, 4, 5]
print("1D list:", oneD_list )  # Output: [1, 2, 3, 4, 5]

# Creating a 2D list
twoD_list = [
    [1, 2, 3],
    [4, 5, 6],
    [7, 8, 9]
]
print("2D list:", twoD_list)
# Output:
# [[1, 2, 3],
#  [4, 5, 6],
#  [7, 8, 9]]

# Creating a 3D list
threeD_list = [
    [
        [1, 2, 3],
        [4, 5, 6]
    ],
    [
        [7, 8, 9],
        [10, 11, 12]
    ]
]
print("3D list:", threeD_list)
```

Characteristics of list

- A mutable, ordered collection of items.
- Items can be of different types.
- Supports indexing, slicing, and various methods for adding, removing, and modifying elements.

```python
# Creating a sample list
nums = [10, 20, 30, 40, 50, 60, 70]
print("Original list:", nums)
# Output: [10, 20, 30, 40, 50, 60, 70]

# Accessing elements using positive indexing
print("First element:", nums[0])    # Output: 10
print("Third element:", nums[2])    # Output: 30

# Accessing elements using negative indexing
print("Last element:", nums[-1])    # Output: 70
print("Second to last element:", nums[-2])  # Output: 60

# Slicing to get the first three elements
first_three = nums[0:3]
print("First three elements:", first_three)
# Output: [10, 20, 30]

# Slicing to get the last three elements
last_three = nums[-3:]
print("Last three elements:", last_three)
# Output: [50, 60, 70]
```

Adding items to the list

```python
# Initial list
nums = [10, 20, 30, 40]
print("Original list:", nums)

# Adding an element using append()
nums.append(50)
print("After append(50):", nums)
# Output: [10, 20, 30, 40, 50]

# Adding elements using extend()
nums.extend([60, 70])
print("After extend([60, 70]):", nums)
# Output: [10, 20, 30, 40, 50, 60, 70]

# Adding an element at a specific index using insert()
nums.insert(2, 25)  # Insert 25 at index 2
print("After insert(2, 25):", nums)
# Output: [10, 20, 25, 30, 40, 50, 60, 70]

# Adding multiple elements using slicing
nums[4:4] = [35, 37]  # Insert [35, 37] at index 4
print("After nums[4:4] = [35, 37]:", nums)
# Output: [10, 20, 25, 30, 35, 37, 40, 50, 60, 70]
```

Deleting items from the list

```python
# Initial list
nums = [10, 20, 25, 30, 35, 37, 40, 50, 60, 70]
print("Original list:", nums)

# Removing an element by value using remove()
nums.remove(25)
print("After remove(25):", nums)  # Output: [10, 20, 30, 35, 37, 40, 50, 60, 70]

# Removing an element by index using pop()
popped_element = nums.pop(3)  # Remove element at index 3
print("After pop(3):", nums)  # Output: [10, 20, 30, 37, 40, 50, 60, 70]
print("Popped element:", popped_element)  # Output: 35

# Removing the last element by using pop() without index
last_element = nums.pop()
print("After pop():", nums)  # Output: [10, 20, 30, 37, 40, 50, 60]
print("Last element:", last_element)  # Output: 70

# Deleting elements using slicing
del nums[1:3]  # Remove elements from index 1 to 2
print("After del nums[1:3]:", nums)  # Output: [10, 37, 40, 50, 60]

# Deleting an element by index using del
del nums[2]  # Delete element at index 2
print("After del nums[2]:", nums)  # Output: [10, 37, 50, 60]

# Clearing the entire list using clear()
nums.clear()
print("After clear():", nums)  # Output: []
```

Operations on list

```python
# Create a list
numbers = [1, 2, 3, 4, 5]

# Arithmetic Operations
print("Sum:", sum(numbers))  # Output: Sum: 15
print("Product:", 1 * 2 * 3 * 4 * 5)  # Output: Product: 120
print("Average:", sum(numbers) / len(numbers))  # Output: Average: 3.0

# Membership Operations
print("Is 3 in the list?", 3 in numbers)  # Output: Is 3 in the list? True
print("Is 6 in the list?", 6 in numbers)  # Output: Is 6 in the list? False

# Loop Operations
for num in numbers:
    print(num)  # Output: 1, 2, 3, 4, 5

# Iterate and modify elements
for i in range(len(numbers)):
    numbers[i] += 1
print(numbers)  # Output: [2, 3, 4, 5, 6]
```

Disadvantages of list

1. Limited Readability
2. Debugging Challenges
3. Slow speed
4. Memory Limitations: Takes lots of memory

List Functions

```python
# Initialize the list
l = [1, 2, 3, 4, 5]

# len()
print("len(l):", len(l))  # Output: 5

# min()
print("min(l):", min(l))  # Output: 1

# max()
print("max(l):", max(l))  # Output: 5

# count()
print("l.count(2):", l.count(2))  # Output: 1

# index()
print("l.index(3):", l.index(3))  # Output: 2

# reverse()
l.reverse()
print("l.reverse():", l)
# Output: [5, 4, 3, 2, 1]

# sort()
l.sort()
print("l.sort():", l)  # Output: l.sort(): [1, 2, 3, 4, 5]

# copy()
l_copy = l.copy()
print("l_copy:", l_copy)  # Output: l_copy: [1, 2, 3, 4, 5]
```

List Comprehension

```python
# Simple list comprehension
numbers = [1, 2, 3, 4, 5]
new_numbers = [num for num in numbers]
print("Simple list comprehension:", new_numbers)
# Output:[1, 2, 3, 4, 5]

# List comprehension with if condition
numbers = [1, 2, 3, 4, 5]
new_numbers = [num for num in numbers if num % 2 == 0]
print("List comprehension with if condition:", new_numbers)
# Output:[2, 4]

# List comprehension with if-else condition
numbers = [1, 2, 3, 4, 5]
new_numbers = ["even" if num % 2 == 0 else "odd" for num in numbers]
print("List comprehension with if-else condition:", new_numbers)
# Output: ['odd', 'even', 'odd', 'even', 'odd']

# Nested list comprehension
matrix = [[1, 2, 3], [4, 5, 6], [7, 8, 9]]
transposed_matrix = [[row[i] for row in matrix] for i in range(3)]
print("Nested list comprehension:", transposed_matrix)
# Output: [[1, 4, 7], [2, 5, 8], [3, 6, 9]]

# List comprehension with range()
numbers = [num for num in range(5)]
print("List comprehension with range():", numbers)
# Output:[0, 1, 2, 3, 4]

# List comprehension with map() and lambda
numbers = [num for num in map(lambda x: x**2, range(5))]
print("List comprehension with map() and lambda:", numbers)
# Output:[0, 1, 4, 9, 16]
```

Coding problems

- Python program to interchange first and last elements in a list
- Python program to swap two elements in a list
- Python | Ways to find length of list
- Maximum of two numbers in Python
- Minimum of two numbers in Python

Bonus Tip

Zip() Function:
The zip() function returns a zip object.
This object is an iterator of tuples.
It combines elements from input iterables.

```
# Define two lists
languages = ['Java', 'Python', 'JavaScript']
versions = [14, 3, 6]

print(list(zip(languages, versions)))
# Output: [('Java', 14), ('Python', 3), ('JavaScript', 6)]

# Use zip() to iterate over both lists
result = zip(languages, versions)

# Convert the result to a list
result_list = list(result)

print(result_list) # Output: [('Java', 14), ('Python', 3), ('JavaScript', 6)]

# Unzip the list using *
c, v = zip(*result_list)

print('c =', c) # Output: ('Java', 'Python', 'JavaScript')

print('v =', v) # Output: (14, 3, 6)

# Define two lists with different lengths
numbers = [1, 2, 3]
letters = ['a', 'b']

print(list(zip(numbers, letters))) # Output: [(1, 'a'), (2, 'b')]
```

🚀 Ready to Level Up?

CONNECT FOR MORE INSIGHTS AND FUTURE UPDATES!

Shruti Bharat

@shrutibharat0105