Truth Value Testing:

Used in conditions like `if`, `while`, etc., where Python evaluates the "truthiness" of a value.

Falsy values in Python:

```
False, None, 0, 0.0, "", [], {}, set(), range(0)
```

Example:

```
items = []
if items:
    print("List has items")
else:
    print("List is empty")  # Will be printed
```

Various Ways of Creating a New List:

```
a = [x for x in range(5)]           # list comprehension
b = list(range(5))                  # using list()
c = []                              # empty list
```

`casefold()`

Used for case-insensitive comparison (stronger than `lower()`).

Example:

```
a = "Python"
b = "PYTHON"
if a.casefold() == b.casefold():
    print("They match")
```

Sorting List:

- `sort()` sorts **in-place**
- `sorted()` returns a **new sorted list**

```
a = [3, 1, 2]
a.sort()
print(a)

b = [3, 1, 2]
print(sorted(b))
```

enumerate() in Python:

colors = ["red", "green", "blue"]

for i, color in enumerate(colors, start=1):

print(f"{i}. {color}")

nums = [10, 20, 30]

print(list(enumerate(nums)))

Why Use `enumerate()` Instead of `range(len())`?

Instead of this (less readable):

```
for i in range(len(fruits)):
    print(i, fruits[i])
```

Prefer this (cleaner and more Pythonic):

```
for i, fruit in enumerate(fruits):
    print(i, fruit)
```

More on Lists:

- Nested lists:

  matrix = [[1, 2], [3, 4]]

  print(matrix[1][0]) # 3

  for row in matrix:

  for col in row:

  print(col)

- List comprehension with `if`: `[x for x in range(10) if x % 2 == 0]`

Practice Question

Q1 Create a Flatten a 2D list

Q2 Find common elements in two lists

Q3 Remove duplicates from list

data = (1, [2, 3])

data[1][0] = 99 # allowed since list is mutable

Swap two values without a temp variable:

```
a, b = b, a
```

Get multiple values:

```
def get_values():
    return 1, 2, 3  # This returns a tuple (1, 2, 3)

a, b, c = get_values()
print(a, b, c)
```

Returning Multiple Data Types:

```
def student_info():
    name = "Alice"
    scores = [85, 90, 78]
    passed = True
    return name, scores, passed

n, s, p = student_info()
print(n, s, p)
```

Lambda function:

```
add = lambda a, b: a + b
print(add(3, 5))  # Output: 8
```

unpacking:

```
numbers = (1, 2, 3, 4, 5)
a, b, *rest = numbers
print(a, b)       # 1 2
print(rest)       # [3, 4, 5]

a, *middle, b = numbers
print(middle)     # [2, 3, 4]


def demo(a, b, *args, **kwargs):

    print(f"a={a}, b={b}")

    print(f"args={args}")

    print(f"kwargs={kwargs}")
```

```
demo(1, 2, 3, 4, x=5, y=6)
```

```
def process_data(data):

    name, age, city = data

    print(f"{name} is {age} and lives in {city}")

process_data(("Bob", 28, "New York"))
```

Use with `map()`

```
nums = [1, 2, 3, 4]
squares = list(map(lambda x: x**2, nums))
print(squares)  # Output: [1, 4, 9, 16]
```

Use with `filter()`

```
nums = [1, 2, 3, 4, 5]
even = list(filter(lambda x: x % 2 == 0, nums))
print(even)  # Output: [2, 4]
```

Challenge: Filter and Transform a List

1. **Filter** out the odd numbers.
2. **Square** the remaining even numbers.
3. **Return the final list.**

Input:

```
nums = [1, 2, 3, 4, 5, 6, 7, 8, 9, 10]
```

Expected Output:

```
 [4, 16, 36, 64, 100]
```

Hint:

- Use `filter()` with a lambda to keep even numbers.
- Use `map()` with a lambda to square each number.