Period 5 - Unit 4, Lesson 1A Reference: Your First Collection

NEW CONCEPT: You've never seen collections before! This is your introduction to storing multiple items in one variable.

The Problem - Too Many Variables!

The old way - individual variables (TERRIBLE!)

```
item1 = "milk"
item2 = "bread"
item3 = "eggs"
item4 = "cheese"
item5 = "apples"

# Printing all of them? Copy-paste nightmare!
print(item1)
print(item2)
print(item3)
print(item4)
print(item5)
```

This doesn't scale! What if you had 100 items?!

The Solution - Lists!

ONE variable, MANY items!

```
shopping_list = ["milk", "bread", "eggs", "cheese", "apples"]
print(f"Shopping list: {shopping_list}")
# Output: Shopping list: ['milk', 'bread', 'eggs', 'cheese', 'apples']
```

All 5 items in ONE variable!

Creating Lists

```
# Empty list
empty = []

# Shopping list
shopping = ["milk", "bread", "eggs"]

# Numbers list
scores = [95, 87, 92, 88]

# Mixed types (but usually don't do this!)
mixed = ["Alice", 20, True]
```

Accessing Items - Positive Indices

```
shopping = ["milk", "bread", "eggs", "cheese"]
# Indices:    0       1        2        3

# Access by index (starts at 0!)
first = shopping[0]    # "milk"
second = shopping[1]   # "bread"
third = shopping[2]    # "eggs"
fourth = shopping[3]   # "cheese"
```

Remember: Counting starts at ZERO!

Accessing Items - Negative Indices

```
shopping = ["milk", "bread", "eggs", "cheese"]
# Negative:   -4       -3       -2      -1

# Negative indices count from the end!
last = shopping[-1]           # "cheese"
second_last = shopping[-2]    # "eggs"
third_last = shopping[-3]     # "bread"
first_from_end = shopping[-4] # "milk"
```

Index -1 is always the last item!

IndexError - What Happens When Index Doesn't Exist

```
shopping = ["milk", "bread", "eggs"]
# List has 3 items (indices 0, 1, 2)

# Try to access index that doesn't exist
item = shopping[10]
# ❌ IndexError: list index out of range
```

Index 10 doesn't exist!

Safe Access - Using try/except

```
shopping = ["milk", "bread", "eggs"]

# Safe access pattern
def safe_get(my_list, index):
    """Safely get an item by index."""
    try:
        return my_list[index]
    except IndexError:
        return None

# Test it
result1 = safe_get(shopping, 0)   # "milk"
result2 = safe_get(shopping, 10)  # None
```

You already know try/except - use it for lists too!

Adding Items - append()

```
shopping = ["milk", "bread", "eggs"]

# Add one item
shopping.append("cheese")
# Now: ["milk", "bread", "eggs", "cheese"]

# Add another
shopping.append("apples")
# Now: ["milk", "bread", "eggs", "cheese", "apples"]

# Add multiple items (one at a time)
shopping.append("bananas")
shopping.append("yogurt")
# Now: ["milk", "bread", "eggs", "cheese", "apples", "bananas", "yogurt"]
```

append() adds to the END of the list

Counting Items - len()

```
shopping = ["milk", "bread", "eggs"]
count = len(shopping)  # 3

# Add more and count again
shopping.append("cheese")
shopping.append("apples")
count = len(shopping)  # 5
```

len() returns the number of items

Looping - for-in Loop

```
shopping = ["milk", "bread", "eggs", "cheese", "apples"]

# Loop through all items automatically!
for item in shopping:
    print(f"✓ {item}")

# Output:
# ✓ milk
# ✓ bread
# ✓ eggs
# ✓ cheese
# ✓ apples
```

Works with numbers too:

```
scores = [95, 87, 92, 88, 91]
for score in scores:
    print(f"Score: {score}")
```

Practical Example 1: Calculate Average

```
scores = [95, 87, 92, 88, 91]

# Calculate total
total = 0
for score in scores:
    total += score

# Calculate average
average = total / len(scores)
print(f"Average: {average}")  # Average: 90.6
```

Practical Example 2: To-Do List Manager

```
tasks = ["homework", "dishes", "coding"]

# Add more tasks
tasks.append("exercise")
tasks.append("reading")

# Display all
print(f"You have {len(tasks)} tasks:")
for task in tasks:
    print(f"  - {task}")

# Output:
# You have 5 tasks:
#   - homework
#   - dishes
#   - coding
#   - exercise
#   - reading
```

Practical Example 3: Safe Access with Empty Check

```
playlist = []

try:
    first_song = playlist[0]
    print(f"First song: {first_song}")
except IndexError:
    print("Playlist is empty!")

# Output: Playlist is empty!
```

Key Patterns to Remember

Creating

```
empty_list = []
my_list = [item1, item2, item3]
```

Accessing

```
first = my_list[0]          # First item (index 0)
last = my_list[-1]          # Last item
```

Adding

```
my_list.append(item)        # Add to end
```

Counting

```
count = len(my_list)        # How many items
```

Looping

```
for item in my_list:       # Process each item
    print(item)
```

Safe Access

```
try:
    item = my_list[index]
except IndexError:
    print("Index doesn't exist!")
```

Common Mistakes to Avoid

❌ Mistake 1: Forgetting indices start at 0

```
my_list = ["A", "B", "C"]
my_list[1]  # This is "B", not "A"!
```

❌ Mistake 2: Using len() as an index

```
my_list = ["A", "B", "C"]
my_list[len(my_list)]  # IndexError! Use len(my_list)-1 for last
```

❌ Mistake 3: Not handling IndexError

```
my_list = ["A", "B", "C"]
item = my_list[10]  # CRASH! Use try/except!
```

✅ Correct Patterns

First item:

```
first = my_list[0]
```

Last item:

```
last = my_list[-1]  # OR my_list[len(my_list)-1]
```

Safe access:

```
try:
    item = my_list[index]
except IndexError:
    print("Invalid index!")
```

Testing Your Code

```
def create_shopping_list(items):
    """Create a shopping list from items."""
    return items.copy() if items else []

if __name__ == "__main__":
    # Test 1: Create list
    test_items = ["milk", "bread"]
    result = create_shopping_list(test_items)
    print(f"Test 1: {'✅ PASS' if result == ['milk', 'bread'] else '❌ FAIL'}")

    # Test 2: Empty list
    result = create_shopping_list([])
    print(f"Test 2: {'✅ PASS' if result == [] else '❌ FAIL'}")

    print("\nYou'll see tests like this in all practice problems!")
```

Summary

What You Learned:

- ✅ Lists hold multiple items in ONE variable
- ✅ Access items with [index] - starts at 0!
- ✅ Negative indices count from the end (-1 is last)
- ✅ append(item) adds to the end
- ✅ len(list) counts items
- ✅ for item in list loops through all items
- ✅ Use try/except to handle IndexError

Next: More ways to modify lists!