**Variables**

revenue = 150000.50 # float

customer\_count = 520 # integer

is\_profitable = True # boolean

store\_name = "Data Mart" # string

# Show type checking

print(f"Revenue type: {type(revenue)}")

print(f"Customer count type: {type(customer\_count)}")

# Basic operations

average\_purchase = revenue / customer\_count

print(f"Average purchase amount: ${average\_purchase:.2f}")

**Lists**

monthly\_sales = [45000, 52000, 49000, 47500, 53200, 54800]

# List operations

print(f"Total sales: ${sum(monthly\_sales)}")

print(f"Highest monthly sale: ${max(monthly\_sales)}")

print(f"Average monthly sale: ${sum(monthly\_sales)/len(monthly\_sales):.2f}")

# List slicing

q1\_sales = monthly\_sales[:3]

print(f"Q1 Sales: {q1\_sales}")

# List methods

monthly\_sales.append(51000) # Adding July sales

monthly\_sales.sort(reverse=True) # Sort in descending order

**Dictionaries**

product\_sales = {

"Electronics": 25000,

"Clothing": 15000,

"Food": 35000,

"Books": 12000

}

# Dictionary operations

print(f"Categories: {product\_sales.keys()}")

print(f"Sales figures: {product\_sales.values()}")

# Adding new category

product\_sales["Sports"] = 18000

# Nested data structure

store\_data = {

"location": "Downtown",

"sales": product\_sales,

"employees": ["John", "Sarah", "Mike"]

}

**Boolean Analysis**

# Conditional analysis

for category, sales in product\_sales.items():

if sales > 20000:

print(f"{category} is a high-performing category")

elif sales > 15000:

print(f"{category} is performing well")

else:

print(f"{category} needs improvement")

# List comprehension for sales analysis

above\_average = [cat for cat, sale in product\_sales.items()

if sale > sum(product\_sales.values())/len(product\_sales)]

print(f"Above average categories: {above\_average}")

**Functions**

def calculate\_growth(old\_value, new\_value):

"""Calculate percentage growth between two values"""

return ((new\_value - old\_value) / old\_value) \* 100

def analyze\_sales\_performance(sales\_dict, threshold):

"""Analyze sales performance against a threshold"""

performing\_categories = []

for category, sale in sales\_dict.items():

if sale >= threshold:

performing\_categories.append(category)

return performing\_categories

# Using the functions

old\_sales = 45000

new\_sales = 52000

growth = calculate\_growth(old\_sales, new\_sales)

print(f"Sales growth: {growth:.1f}%")

# Analyze categories above 20000

good\_performers = analyze\_sales\_performance(product\_sales, 20000)

print(f"Good performing categories: {good\_performers}")