Week 8: Loops & Conditionals - Quick Reference

Conditionals: Making Decisions in Code

Basic If Statement

```python
if condition:
    # code runs if condition is True

# Example
if sales > 100000:
    print("Target achieved!")
```

If-Else (Binary Choice)

```python
if condition:
    # runs if True
else:
    # runs if False

# Example
if customer_type == "premium":
    discount = 0.20
else:
    discount = 0.10
```

If-Elif-Else (Multiple Options)

```python
if condition1:
    # first check
elif condition2:
    # second check
elif condition3:
    # third check
else:
    # if none match

# Example
if score >= 90:
    grade = "A"
elif score >= 80:
    grade = "B"
elif score >= 70:
    grade = "C"
else:
    grade = "F"
```

Combining Conditions

```python
```

```python
# AND: All must be True
if price < 50 and quantity > 10:
    apply_bulk_discount()

# OR: At least one must be True
if payment == "cash" or payment == "credit":
    process_payment()

# NOT: Inverts the condition
if not is_expired:
    use_coupon()

# IN: Check membership
if region in ["North", "South"]:
    free_shipping = True
```

Loops: Processing Multiple Items

For Loop (Most Common)

```python
# Loop through list
sales = [1000, 2000, 1500]
for amount in sales:
    print(amount)

# Loop with index
for i in range(len(sales)):
    print(f"Month {i+1}: {sales[i]}")

# Loop with enumerate (index + value)
for index, amount in enumerate(sales):
    print(f"Month {index+1}: {amount}")
```

Looping Through Dictionaries

```python
person = {"name": "John", "age": 30}

# Keys only
for key in person:
    print(key)

# Values only
for value in person.values():
    print(value)

# Both key and value
for key, value in person.items():
    print(f"{key}: {value}")
```

While Loop (Use Carefully!)

```python
```

```python
count = 0
while count < 5:
    print(count)
    count += 1  # Don't forget to update!

# With break condition
while True:
    user_input = input("Enter 'quit' to exit: ")
    if user_input == "quit":
        break
```

Loop Control Statements

```python
# BREAK: Exit loop entirely
for num in numbers:
    if num < 0:
        print("Found negative!")
        break

# CONTINUE: Skip to next iteration
for num in numbers:
    if num == 0:
        continue  # Skip zeros
    result = 100 / num
```

Common Patterns in Data Work

1. Filtering Data

```python
valid_records = []
for record in all_records:
    if record["status"] == "active":
        valid_records.append(record)
```

2. Accumulating/Aggregating

```python
total = 0
for sale in sales_data:
    total += sale["amount"]
average = total / len(sales_data)
```

3. Counting Categories

```python
```

```
counts = {}
for item in items:
    category = item["category"]
    if category in counts:
        counts[category] += 1
    else:
        counts[category] = 1
```

4. Finding Max/Min

```python
highest = sales[0]
for sale in sales:
    if sale > highest:
        highest = sale
```

5. Data Validation

```python
for record in records:
    if not record["email"]:
        print(f"Missing email in record {record['id']}")
    elif "@" not in record["email"]:
        print(f"Invalid email: {record['email']}")
```

List Comprehensions (Advanced)

```python
# Basic syntax
[expression for item in iterable]

# Examples
squares = [x**2 for x in range(10)]
upper_names = [name.upper() for name in names]

# With condition
high_sales = [s for s in sales if s > 1000]
```

Comparison Operators

- `==` : Equal to
- `!=` : Not equal to
- `<` : Less than
- `>` : Greater than
- `<=` : Less than or equal
- `>=` : Greater than or equal
- `in` : Membership test
- `is` : Identity test (for None, True, False)

Common Mistakes to Avoid

1. **Using = instead of ==**

```python
❌ if x = 5:  # Assignment!
✅ if x == 5:  # Comparison
```

2. **Infinite loops**

```python
❌ while True:  # No break!
✅ while condition:  # Has end condition
```

3. **Modifying list while looping**

```python
❌ for item in items:
    items.remove(item)
✅ items = [i for i in items if keep_condition]
```

4. **Off-by-one errors**

```python
❌ for i in range(1, len(items)):  # Skips first!
✅ for i in range(len(items)):
```

Practice Problems

1. **Sales Commission**: Calculate different commission rates based on sales tiers
2. **Data Cleaner**: Remove invalid entries from a dataset
3. **Report Generator**: Count and categorize transactions by type
4. **Inventory Monitor**: Flag items with low stock
5. **Customer Segmentation**: Group customers by purchase behavior

Real-World Applications

- **ETL Pipelines**: Validate and transform each record
- **Report Generation**: Aggregate data with conditions
- **Data Quality**: Check each field meets requirements
- **Batch Processing**: Process files/records one by one
- **Alert Systems**: Check thresholds and trigger actions