Lesson 03: Control Flow

Data Science with Python – BSc Course

Data Science Program

45 Minutes

Learning Objectives

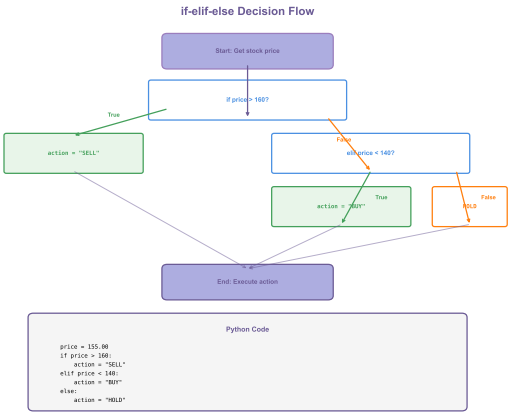
After this lesson, you will be able to:

- Write conditional statements with if/elif/else
- Create loops to iterate over data
- Implement trading rules using control flow
- Use break and continue for loop control

Finance Application: Implement trading signals and position sizing rules.

Control flow determines which code executes based on conditions

If-Else Statements



Basic Structure:

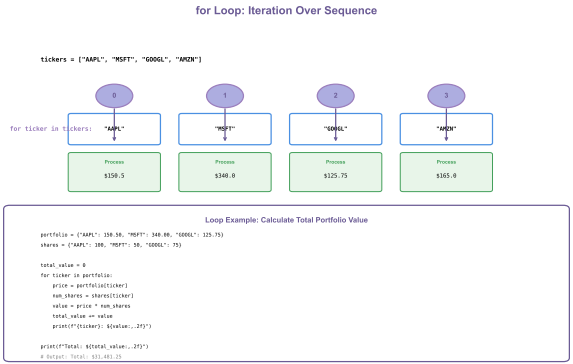
```
if price > 200:
    signal = "SELL"
elif price < 150:
    signal = "BUY"
else:
    signal = "HOLD"
```

Key Points:

- Colon after condition
- Indentation matters!
- elif is optional

Indentation (4 spaces) defines code blocks in Python

For Loops



Iterate Over List:

```
for price in prices:
    print(price)
```

With Index:

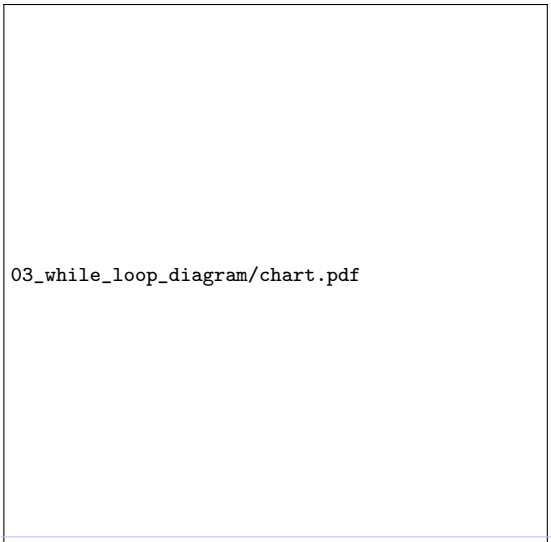
```
for i, p in enumerate(prices):
    print(f"Day {i}:  {p}")
```

Range:

```
for i in range(5):
    print(i) # 0,1,2,3,4
```

For loops iterate a known number of times

While Loops



Basic While:

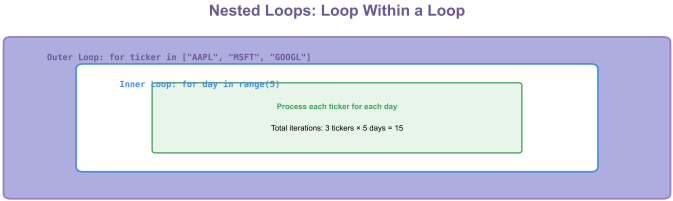
```
balance = 10000
while balance > 5000:
    balance *= 0.95
    print(balance)
```

Use Cases:

- Unknown iterations
- Waiting for condition
- Simulation until target

Warning: Infinite loops!

Nested Loops



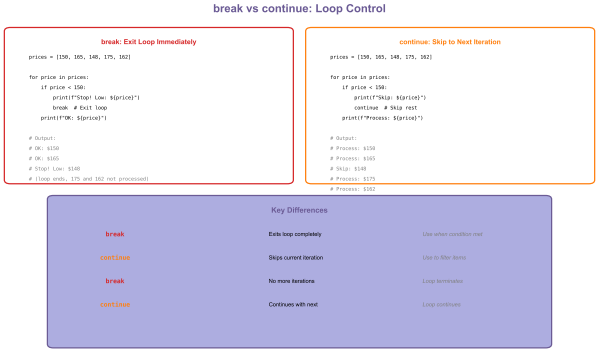
Nested Loop Example: Price Matrix

```
tickers = ["AAPL", "MSFT", "GOOGL"]
days = ["Mon", "Tue", "Wed", "Thu", "Fri"]

for ticker in tickers:                    # Outer loop (3 iterations)
    print(f"\n{ticker} prices:")
    for day in days:                      # Inner loop (5 iterations)
        price = get_price(ticker, day)    # Called 15 times total
        print(f"  {day}: ${price:.2f}")

# Output:
# AAPL prices:
#   Mon: $150.50
#   Tue: $151.25
#   ...
```

Break and Continue



Break: Exit loop entirely

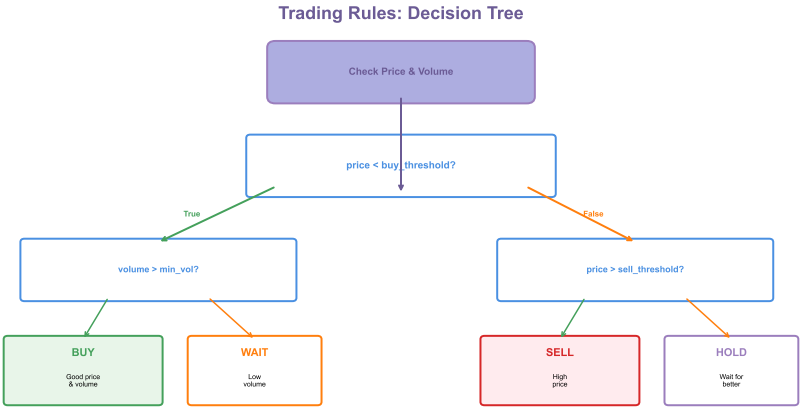
```
for price in prices:
    if price > 200:
        break # stop now
```

Continue: Skip to next iteration

```
for price in prices:
    if price < 0:
        continue # skip this
    process(price)
```

Break stops loop; continue skips current iteration

Trading Rules Decision Tree



Python Implementation

```
price, volume = 145.00, 1200000
buy_threshold, sell_threshold = 150.00, 170.00
min_volume = 1000000
```

Loop Comparison

for vs while: Loop Comparison

for Loop	Aspect	while Loop
Iterate over sequence	Use Case	Repeat until condition
Known iterations	Duration	Unknown iterations
for x in sequence:	Syntax	while condition:
Automatic	Increment	Manual
List, range(), dict	Common With	Counters, flags
More readable	Readability	More flexible
Portfolio analysis	Finance Example	Price convergence

for Example

```
prices = [150, 165, 148]
total = 0
for price in prices:
    total += price
avg = total / len(prices)
```

while Example

```
price = 100
target = 150
while price < target:
    price *= 1.05
    years += 1
```

Control Flow Patterns

Common Control Flow Patterns

Pattern 1: Guard Clause

```
def buy_stock(price, balance):
    # Guard: Check preconditions
    if price <= 0:
        return "Invalid price"
    if balance < price:
        return "Insufficient funds"

    # Main logic
    execute_buy(price)
```

Pattern 2: Accumulator

```
prices = [150, 165, 148, 172]
total = 0  # Accumulator

for price in prices:
    total += price

average = total / len(prices)
print(f"Avg: ${average:.2f}")
```

Pattern 3: Search & Break

```
prices = [150, 165, 148, 172]
found = False

for price in prices:
    if price < 150:
        print(f"Found: ${price}")
        found = True
        break  # Stop searching
```

Pattern 4: Filter Pattern

```
all_prices = [150, 165, 148, 172]
high_prices = []  # Filtered list

for price in all_prices:
    if price > 160:
        high_prices.append(price)

# Result: [165, 172]
```

Pattern 5: Counter

```
prices = [150, 165, 148, 172, 145]
count_low = 0  # Counter

for price in prices:
    if price < 150:
        count_low += 1
```

Pattern 6: Find Min/Max

```
prices = [150, 165, 148, 172]
max_price = prices[0]  # Initialize

for price in prices:
    if price > max_price:
        max_price = price
```

Hands-on Exercise (25 min)

Implement a simple trading system:

1. Create price list: `prices = [180, 185, 195, 188, 205, 198]`
2. Implement trading rules:
   - If price ¿ 200: SELL
   - If price ¡ 185: BUY
   - Else: HOLD
3. Loop through prices and generate signals
4. Count total BUY, SELL, HOLD signals
5. Find first price that triggers SELL (use break)
6. Skip negative prices if any (use continue)

This forms the basis of algorithmic trading

Lesson Summary

Key Takeaways:

- if/elif/else for conditional execution
- for loops iterate over sequences
- while loops continue until condition is false
- break exits loop, continue skips iteration
- Indentation defines code blocks

Next Lesson: Functions

Control flow + functions = modular trading systems