Lesson 04: Functions

Data Science with Python – BSc Course

Data Science Program

45 Minutes

Learning Objectives

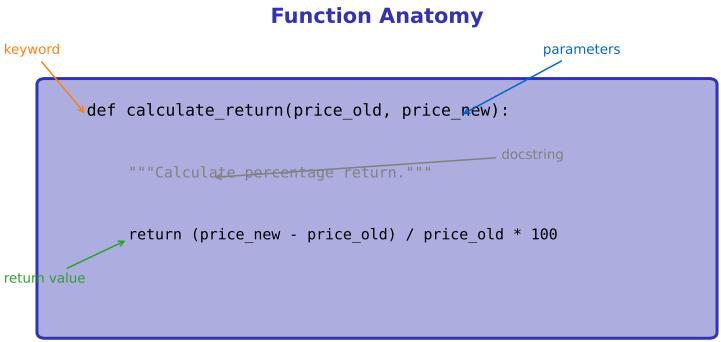
After this lesson, you will be able to:

- Define and call functions with parameters
- Use return statements to output values
- Understand variable scope (local vs global)
- Write docstrings for documentation

Finance Application: Create reusable functions for return calculations and risk metrics.

Functions are building blocks of modular code

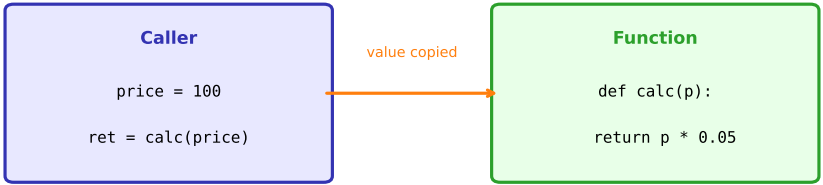
Function Anatomy



Functions encapsulate reusable logic

def keyword, name, parameters, colon, indented body, return

Parameter Passing



Positional: `func(a, b)`

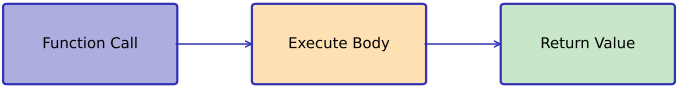
Keyword: `func(x=1, y=2)`

Default: `def f(x=10)`

`*args, **kwargs`

Return Values

Return Value Flow



Single return:

```
return price * 1.05
```

Multiple returns:

```
return mean, std
```

No return (None):

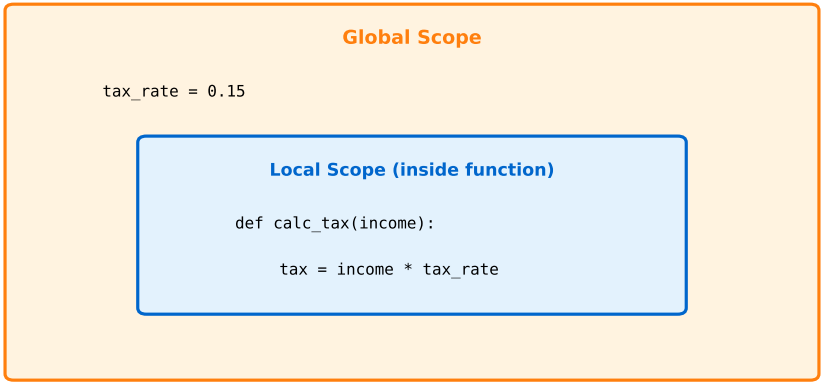
```
print("Hello")
```

Early return:

```
if x < 0: return 0
```

Functions without return statement return None

Variable Scope: Local vs Global

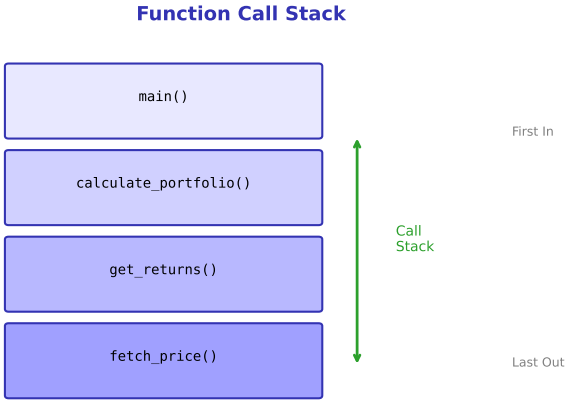


Local variables exist only during function execution

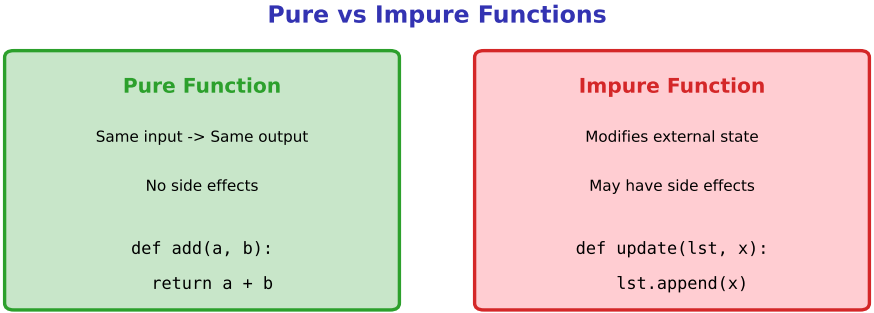
Docstring Best Practices

```
def calculate_sharpe(returns, rf_rate=0.02):
    """
    Calculate the Sharpe ratio for a series of returns.

    Parameters:
        returns (array): Daily return values
        rf_rate (float): Risk-free rate (default: 0.02)

    Returns:
        float: Annualized Sharpe ratio
    """
    excess = returns.mean() - rf_rate/252
    return excess / returns.std() * np.sqrt(252)
```

Function Call Stack



Stack grows with nested calls, shrinks as functions return



Prefer pure functions for predictable, testable code

Essential Finance Functions

`calculate_return(p1, p2)`	Price change %
`annualize_return(daily_ret)`	Convert to yearly
`calculate_volatility(returns)`	Standard deviation
`sharpe_ratio(ret, rf)`	Risk-adjusted return
`max_drawdown(prices)`	Largest peak-to-trough
`beta(stock, market)`	Market sensitivity

Hands-on Exercise (25 min)

Build a finance functions library:

1. `calculate_return(buy, sell)` – percentage return
2. `annualize_return(daily_ret, days=252)` – annualized
3. `calculate_volatility(returns)` – standard deviation
4. `sharpe_ratio(returns, rf=0.02)` – risk-adjusted return
5. Test each function with sample data
6. Add docstrings to all functions

These functions will be used throughout the course

Lesson Summary

Key Takeaways:

- Functions encapsulate reusable logic
- Parameters pass data in, return sends data out
- Local scope: variables exist only inside function
- Docstrings document function purpose and usage
- Pure functions are predictable and testable

Next Lesson: DataFrames Introduction

Functions + pandas = powerful financial analysis