Python Programming

Unit 02 – Lecture 04: Functions Basics and Parameters

Tofik Ali

School of Computer Science, UPES Dehradun

February 14, 2026

Repository: `https://github.com/tali7c/Python-Programming`

Quick Links

Core Concepts Demo Interactive Summary

Agenda

Learning Outcomes

- Define reusable functions with parameters and return values

Learning Outcomes

- Define reusable functions with parameters and return values
- Use keyword, default, and variable-length arguments

Learning Outcomes

- Define reusable functions with parameters and return values
- Use keyword, default, and variable-length arguments
- Write docstrings to document behavior

Learning Outcomes

- Define reusable functions with parameters and return values
- Use keyword, default, and variable-length arguments
- Write docstrings to document behavior
- Explain local vs global scope

Why Functions?

- Avoid repetition

Why Functions?

- Avoid repetition
- Improve readability

Why Functions?

- Avoid repetition
- Improve readability
- Make code easier to test

Why Functions?

- Avoid repetition
- Improve readability
- Make code easier to test
- Enable modular design

Function Syntax

```
def add(a, b):
    return a + b

result = add(3, 4)
```

- def creates a function

Function Syntax

```
def add(a, b):
    return a + b

result = add(3, 4)
```

- def creates a function
- return sends a value back

Docstrings

- Triple-quoted string inside a function

```
def area(r):
    """Return area of a circle."""
    return 3.14 * r * r
```

Docstrings

- Triple-quoted string inside a function
- Describe purpose, parameters, and return value

```
def area(r):
    """Return area of a circle."""
    return 3.14 * r * r
```

Parameters vs Arguments

- Parameters appear in function definition

Parameters vs Arguments

- Parameters appear in function definition
- Arguments are values passed in a call

Parameters vs Arguments

- Parameters appear in function definition
- Arguments are values passed in a call
- Positional and keyword arguments both valid

Default Arguments

- Provide a fallback value

```
def greet(name, msg="Hello"):
    return f"{msg}, {name}"
```

Default Arguments

- Provide a fallback value
- Evaluated at function definition time

```
def greet(name, msg="Hello"):
    return f"{msg}, {name}"
```

Default Arguments

- Provide a fallback value
- Evaluated at function definition time
- Avoid mutable defaults (like lists)

```
def greet(name, msg="Hello"):
    return f"{msg}, {name}"
```

Pitfall: Mutable Default Arguments

- Default objects are created once and reused

```
def add_item(x, items=[]):
    items.append(x)
    return items

print(add_item(1))   # [1]
print(add_item(2))   # [1, 2]   (surprise!)
```

Pitfall: Mutable Default Arguments

- Default objects are created once and reused
- A list default is shared across calls

```
def add_item(x, items=[]):
    items.append(x)
    return items

print(add_item(1))   # [1]
print(add_item(2))   # [1, 2]   (surprise!)
```

Variable-Length Arguments

- `*args` collects extra positional arguments

```
def total(*nums):
    return sum(nums)
```

Variable-Length Arguments

- `*args` collects extra positional arguments
- `**kwargs` collects extra keyword arguments

```
def total(*nums):
    return sum(nums)
```

Worked Example: Average of Any Number of Scores

```
def avg(*scores):
    if not scores:
        return 0.0
    return sum(scores) / len(scores)

print(avg(10, 20, 30))
```

- Handle empty input to avoid division by zero

Scope: Local vs Global

- Variables inside functions are local by default

Scope: Local vs Global

- Variables inside functions are local by default
- Use `global` only when necessary

Scope: Local vs Global

- Variables inside functions are local by default
- Use `global` only when necessary
- Prefer returning values instead of modifying globals

Demo: Grade Calculator Function

- Compute average score

Script: `demo/grade_calculator.py`

Demo: Grade Calculator Function

- Compute average score
- Return letter grade based on thresholds

Script: `demo/grade_calculator.py`

Demo: Grade Calculator Function

- Compute average score
- Return letter grade based on thresholds
- Use default parameters for flexibility

Script: `demo/grade_calculator.py`

Demo: Grade Calculator Function

- Compute average score
- Return letter grade based on thresholds
- Use default parameters for flexibility
- Extension: allow custom boundaries for other grading schemes

Script: `demo/grade_calculator.py`

Checkpoint 1

What happens if you use a mutable default like `def f(items=[])`?

Checkpoint 2

Predict the output:

```
x = 5

def foo():
    x = 2
    return x

print(foo(), x)
```

Think-Pair-Share

When do keyword arguments improve readability? Give a short example.

Summary

- Functions organize and reuse logic

Summary

- Functions organize and reuse logic
- Arguments can be positional, keyword, or variable-length

Summary

- Functions organize and reuse logic
- Arguments can be positional, keyword, or variable-length
- Docstrings document intent and usage

Summary

- Functions organize and reuse logic
- Arguments can be positional, keyword, or variable-length
- Docstrings document intent and usage
- Scope rules help avoid bugs

Exit Question

Write a function signature that accepts any number of scores and returns their average.