Unit:2 ; Class:9

Functions

Instructor: Musfique Ahmed

What is a Function?

- A block of reusable code designed to perform a specific task.
- Promotes modular programming by breaking tasks into smaller pieces.

Syntax:

```
# Function Definition
def function_name(parameters):
    # Code block
    return value

# Function Call
function_name(arguments)
```

Practice Problem: Write a function to print “Hello, World!” and call it.

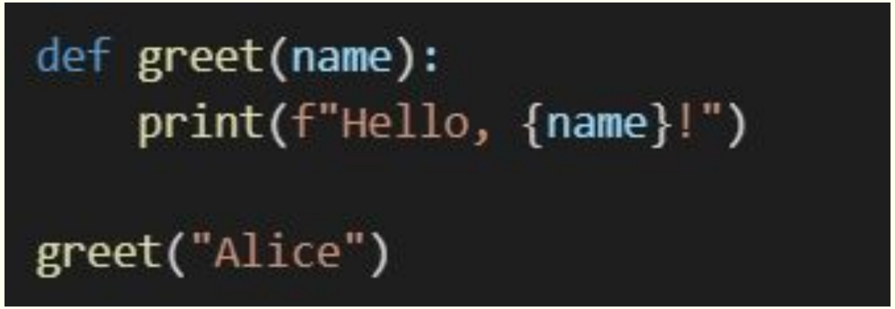
Types of Functions

1. **Built-in Functions:** Predefined in Python (e.g., `len()`, `print()`, `type()`).
2. **User-defined Functions:** Created by the programmer.

Practice Problem: Use the `len()` function to find the length of a string. Write your own function to greet a user by name.

Parameters and Arguments

- **Parameters:** Variables listed in the function definition.
- **Arguments:** Values passed when calling the function.

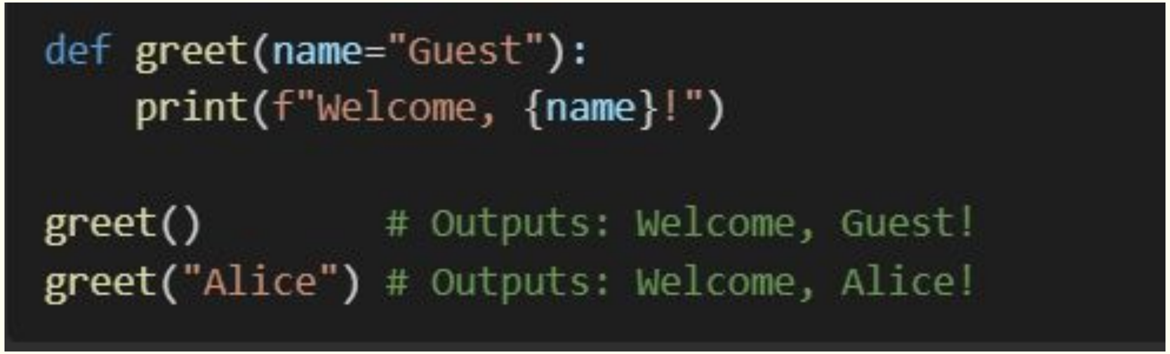


```python
def greet(name):
    print(f"Hello, {name}!")

greet("Alice")
```

Practice Problem: Write a function to calculate the square of a number and pass the number as an argument.

Default Parameters

- Assign default values to parameters in the function definition.
- If no argument is provided, the default value is used.

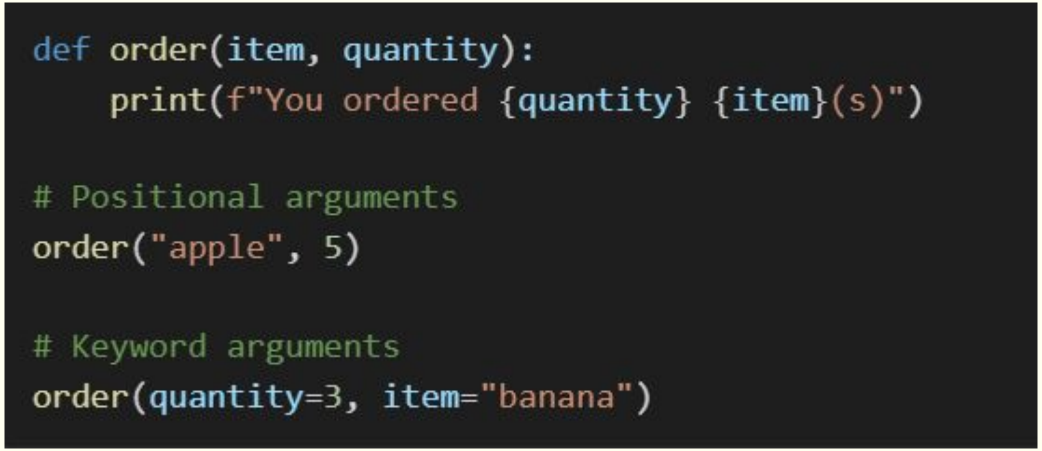


```python
def greet(name="Guest"):
    print(f"Welcome, {name}!")

greet()         # Outputs: Welcome, Guest!
greet("Alice") # Outputs: Welcome, Alice!
```

Practice Problem: Write a function to calculate the area of a rectangle. Set a default value for one side.

Keyword and Positional Arguments

- **Positional Arguments:** Matched based on the order of parameters in the function definition.
- **Keyword Arguments:** Matched explicitly by the parameter name.

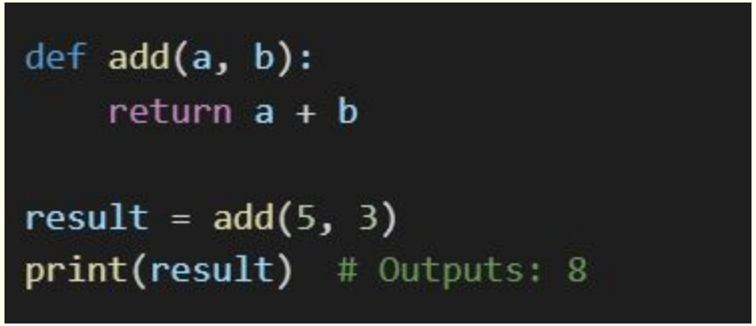


```python
def order(item, quantity):
    print(f"You ordered {quantity} {item}(s)")

# Positional arguments
order("apple", 5)

# Keyword arguments
order(quantity=3, item="banana")
```

Practice Problem: Write a function to calculate the total price of an item. Pass the price and quantity using keyword arguments.

Returning Values from Functions

- Use `return` to send a value back to the caller.

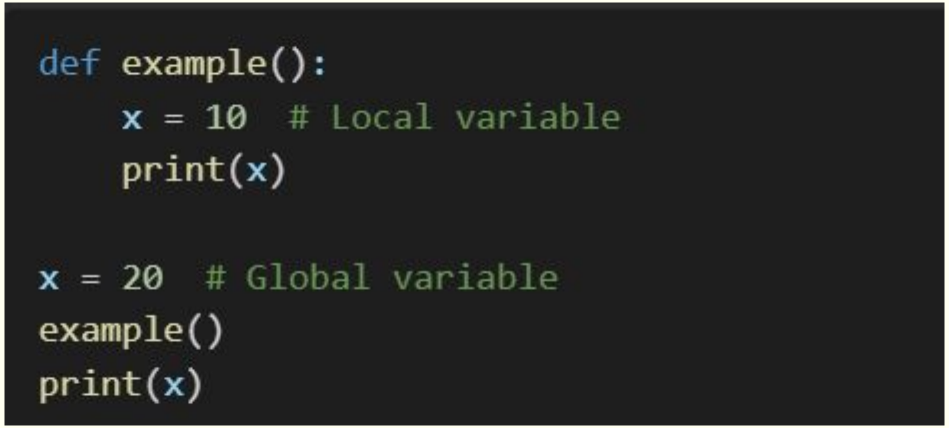


```python
def add(a, b):
    return a + b

result = add(5, 3)
print(result)  # Outputs: 8
```

Practice Problem: Write a function to find the larger of two numbers and return the result.

Local and Global Variables

- **Local Variables:** Declared inside a function and accessible only within it.
- **Global Variables:** Declared outside any function and accessible throughout the program.



```python
def example():
    x = 10  # Local variable
    print(x)

x = 20  # Global variable
example()
print(x)
```

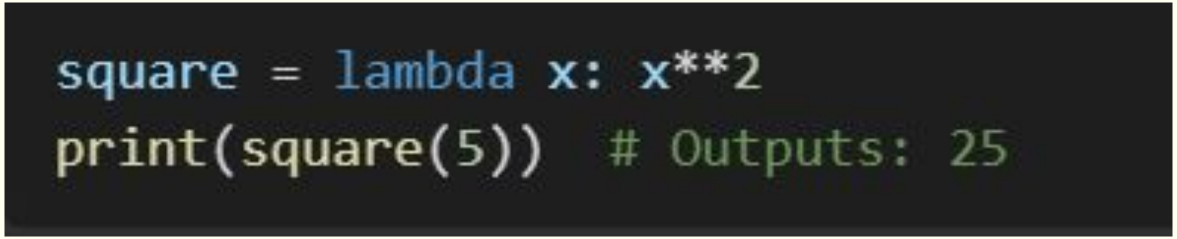
Practice Problem: Write a program to demonstrate the difference between local and global variables.

Lambda Functions

- Anonymous Functions
- A lambda function is a one-liner function without a name.

Syntax:

lambda arguments: expression

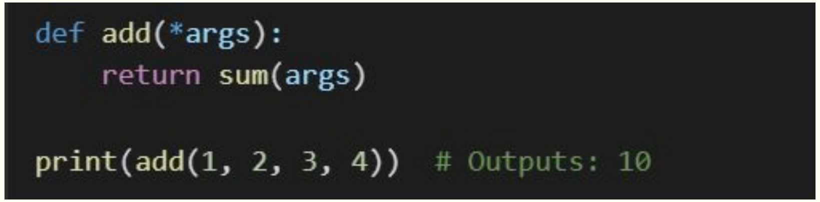


```python
square = lambda x: x**2
print(square(5))  # Outputs: 25
```

Practice Problem: Write a lambda function to calculate the cube of a number.

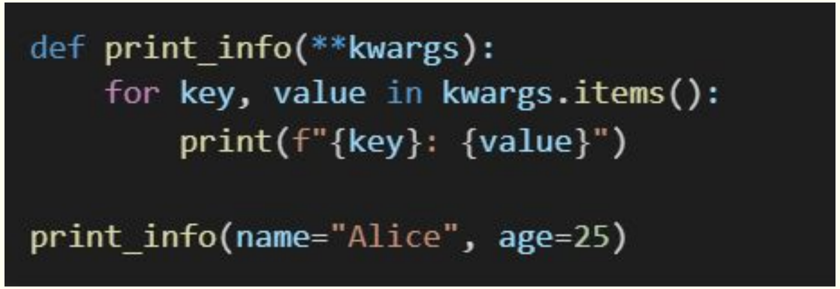
Arbitrary Arguments

- `Arbitrary Arguments`
- `*args`: Allows a function to accept multiple positional arguments.



```
def add(*args):
    return sum(args)

print(add(1, 2, 3, 4))  # Outputs: 10
```

- `**kwargs`: Accepts multiple keyword arguments.

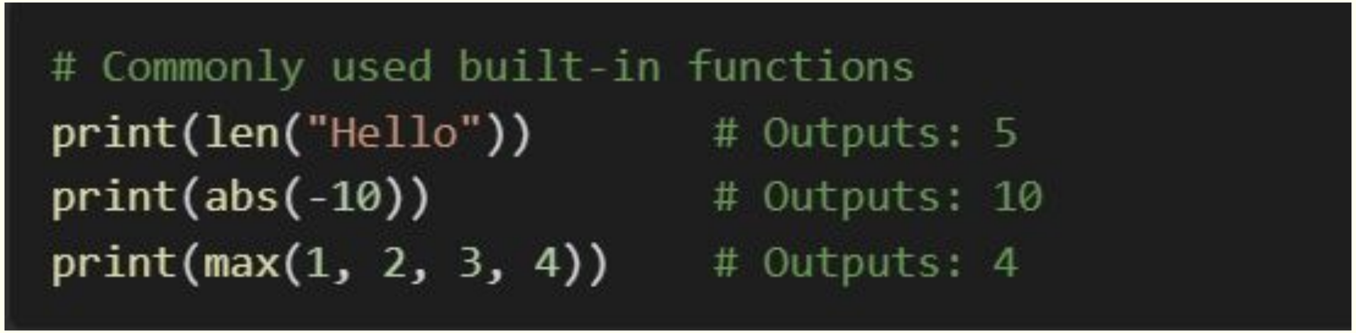


```
def print_info(**kwargs):
    for key, value in kwargs.items():
        print(f"{key}: {value}")

print_info(name="Alice", age=25)
```

Practice Problem: Write a function to calculate the sum of an arbitrary number of numbers using `*args`.

Built-in Functions

- Python provides numerous built-in functions to simplify tasks.

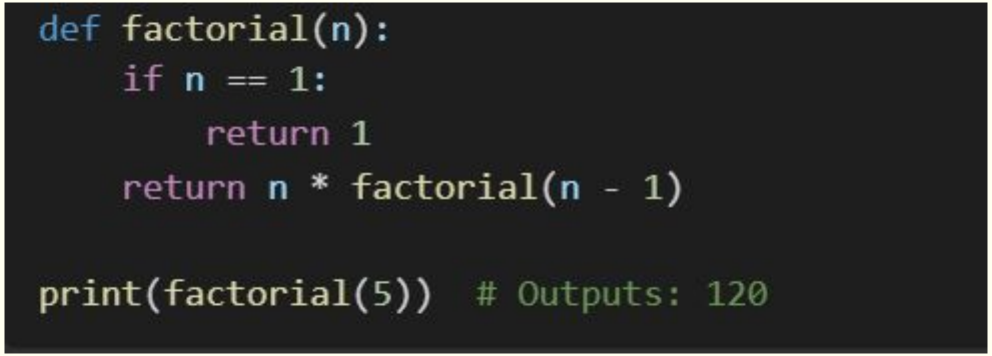


```
# Commonly used built-in functions
print(len("Hello"))         # Outputs: 5
print(abs(-10))             # Outputs: 10
print(max(1, 2, 3, 4))      # Outputs: 4
```

Practice Problem: Use the `min()` and `sum()` functions to find the smallest number and total of a list.

Recursion

- A function that calls itself to solve smaller subproblems.

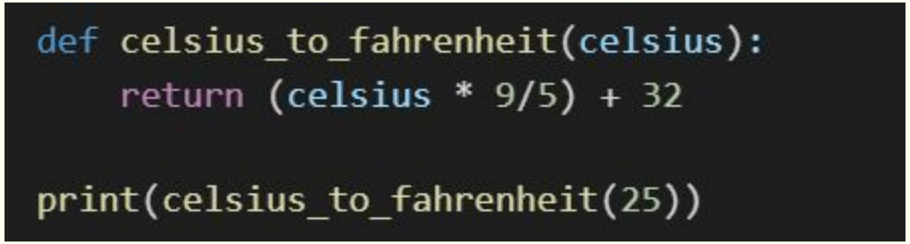


```
def factorial(n):
    if n == 1:
        return 1
    return n * factorial(n - 1)

print(factorial(5))  # Outputs: 120
```

Practice Problem: Write a recursive function to find the nth Fibonacci number.

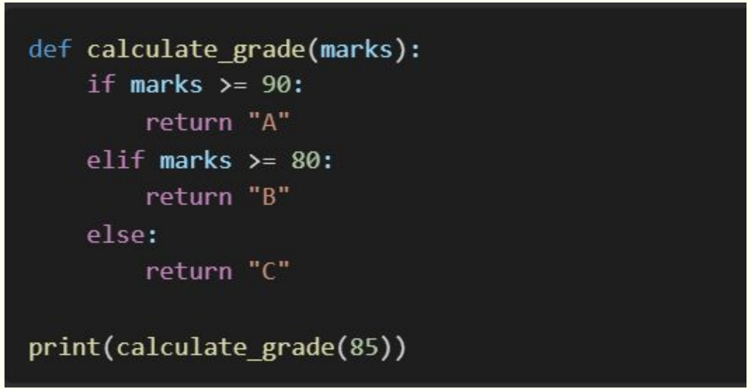
Functions in Real Life

1. **Temperature Converter:**



```python
def celsius_to_fahrenheit(celsius):
    return (celsius * 9/5) + 32

print(celsius_to_fahrenheit(25))
```

2. **Grade Calculator:**



```python
def calculate_grade(marks):
    if marks >= 90:
        return "A"
    elif marks >= 80:
        return "B"
    else:
        return "C"

print(calculate_grade(85))
```

Practice Problem: Write a function to calculate the Body Mass Index (BMI) given weight and height.

Summary

- Definition and purpose of functions.
- Syntax: Parameters, arguments, and return values.
- Types of functions: Built-in, user-defined, and lambda.
- Advanced topics: Recursion, `*args`, `**kwargs`, keyword vs positional arguments.
- Practical use cases and problem-solving.

Thank You

Do the Quiz Please, you have 10 minutes to do that!