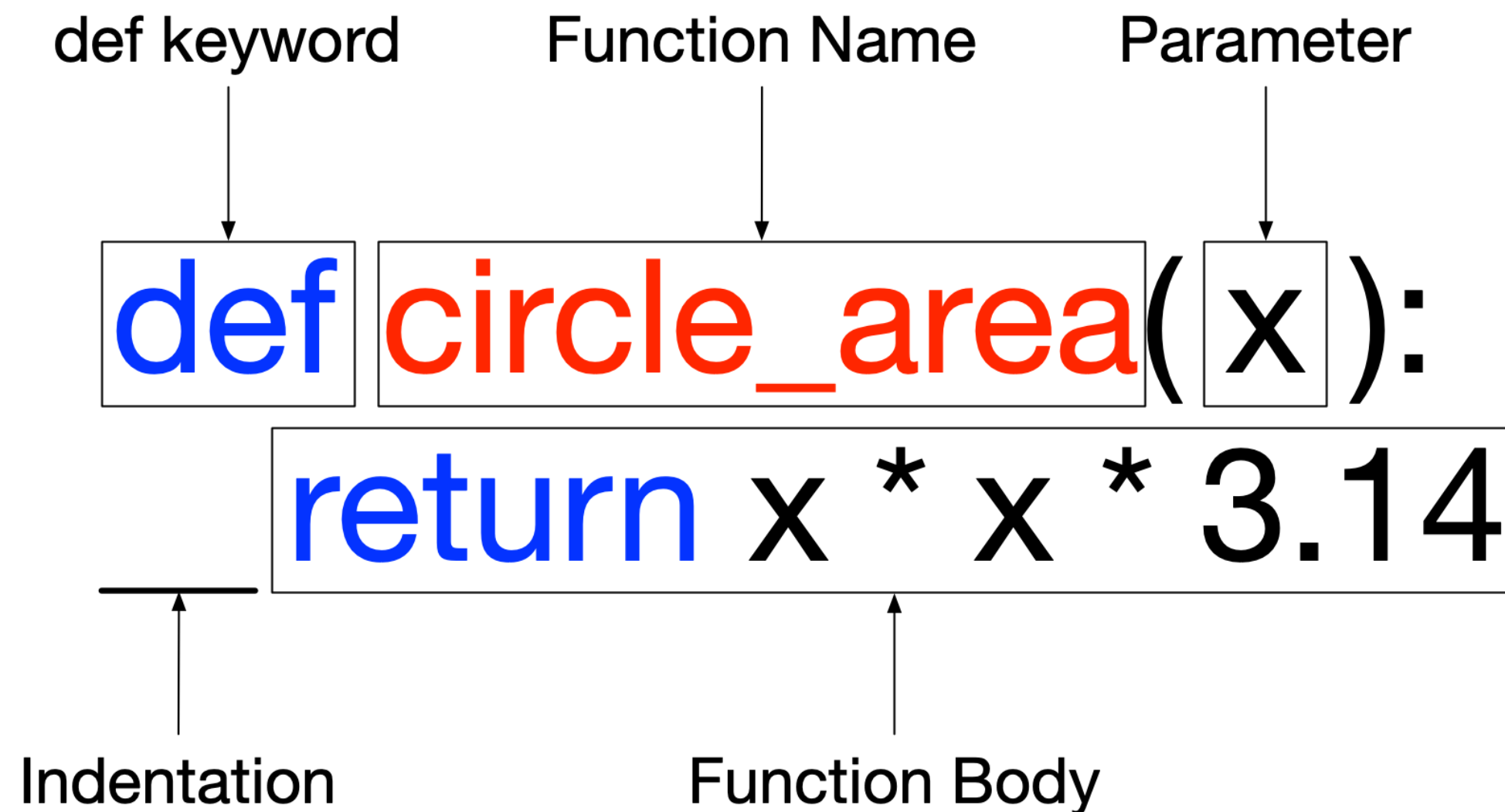
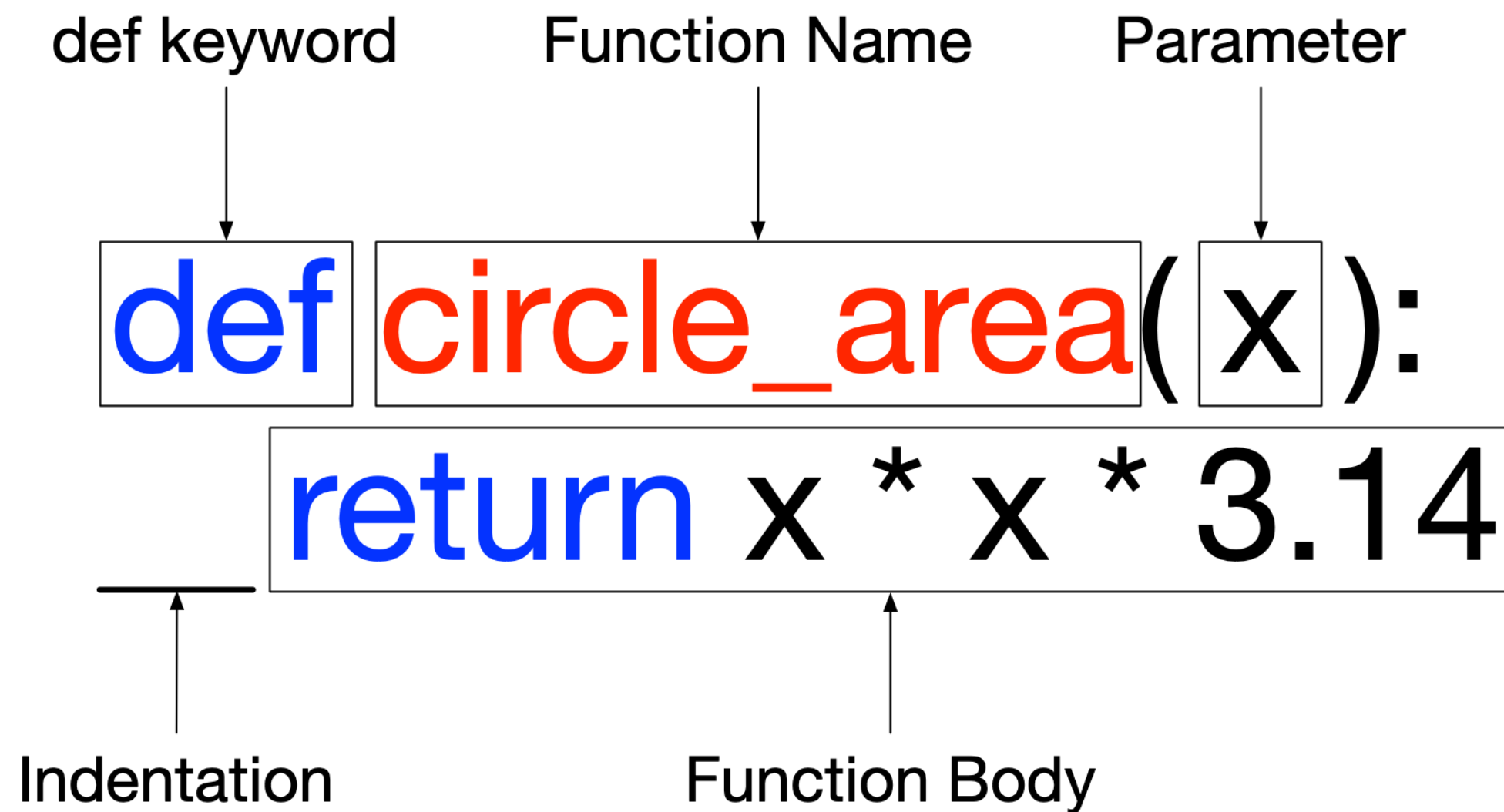


Anatomy of a Function Definition



- A function definition begins with the keyword **def** followed by the **function name** and **parentheses**.
- Any **parameters** should be placed within the *parentheses*.
- Each parameter should be separated by **commas** (,).
- The **function body** within every function starts with a **colon** (:), **new line**, and **indentation**.

The *return* Statement



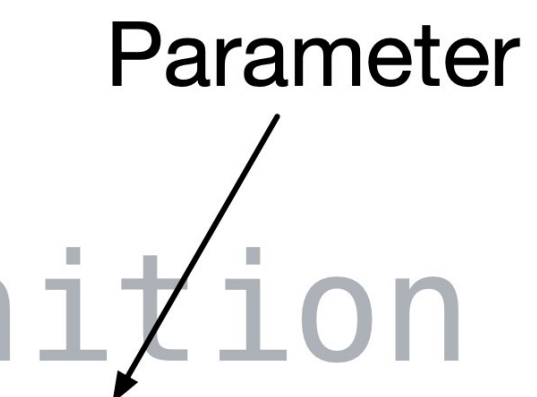
- The **return** statement passes (*outputs*) a value from the function to the caller.
- The *return* statement format:
 - `return <expression>`
- On the left, $x * x * 3.14$ is the expression.
- Any statements after *return* are ignored.

Function Definitions vs. Calls

Function Definition

```
def circle_area(x):  
    return x * x * 3.14
```

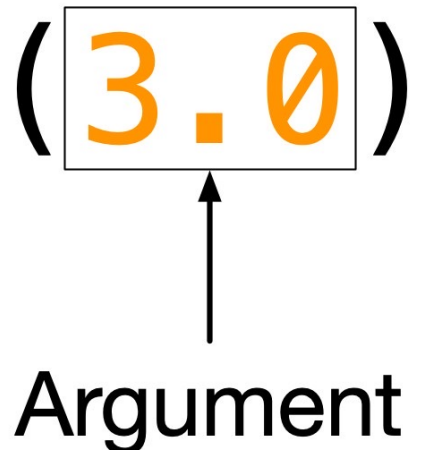
Parameter



Function Call

```
circle_area(3.0)
```

Argument



- Function Definition:
 - Defines what function **does**.
 - Declares parameter **x**.
 - **Parameter**: the *variable* that is listed within the parentheses of a function header.
- Function Call:
 - Command to **do** the function.
 - An **argument** to assign to **x**.
 - **Argument**: a *value* to assign to the function parameter when it is called.

Order of execution

- Function definition must come before the function call.

Sequential Execution

The diagram illustrates the sequential execution of Python code. A vertical arrow on the left points downwards, indicating the flow of execution. The code is divided into two sections: Function Definition and Function Call. Annotations with arrows point to specific parts of the code, explaining what Python does at each step.

```
# Function Definition
def circle_area(x):
    return x * x * 3.14

# Function Call
circle_area(3.0)
```

Annotations:

- Python skips (points to the comment line)
- Python learns about the function (points to the function definition line)
- Python skips everything inside the function (points to the return statement)
- Python skips (points to the comment line)
- Python executes the function body (points to the function call line)

Order of execution: For-Loop

Code

```
print('Welcome!')
```

```
for x in range(3):
```

```
    print(x)
```

```
print('Good Bye!')
```

Output

Order of execution: For-Loop

Code

```
print('Welcome!')
```

```
for x in range(3):
```

```
    print(x)
```

```
print('Good Bye!')
```

Output

Welcome!

Order of execution: For-Loop

Code

```
print('Welcome!')
```

```
for x in range(3):
```

```
    print(x)
```

```
print('Good Bye!')
```

Output

Welcome!

Order of execution: For-Loop

Code

```
print('Welcome!')
```

```
for x in range(3):
```

```
    print(x)
```

```
print('Good Bye!')
```

Output

Welcome!

0

Order of execution: For-Loop

Code

```
print('Welcome!')
```

```
for x in range(3):
```

```
    print(x)
```

```
print('Good Bye!')
```

Output

Welcome!

0

Order of execution: For-Loop

Code

```
print('Welcome!')
```

```
for x in range(3):
```

```
    print(x)
```

```
print('Good Bye!')
```

Output

Welcome!

0

1

Order of execution: For-Loop

Code

```
print('Welcome!')
```

```
for x in range(3):
```

```
    print(x)
```

```
print('Good Bye!')
```

Output

Welcome!

0

1

Order of execution: For-Loop

Code

```
print('Welcome!')
```

```
for x in range(3):
```

```
    print(x)
```

```
print('Good Bye!')
```

Output

Welcome!

0

1

2

Order of execution: For-Loop

Code

```
print('Welcome!')
```

```
for x in range(3):
```

```
    print(x)
```

```
print('Good Bye!')
```

Output

Welcome!

0

1

2

Good Bye!

Order of execution: Function

Code

```
def hello():  
    print('Hello, World!')
```

```
print('Welcome!')
```

```
hello()
```

```
print('Good Bye!')
```

Output

Order of execution: Function

Code

```
def hello():  
    print('Hello, World!')
```

```
print('Welcome!')
```

```
hello()
```

```
print('Good Bye!')
```

Output

Order of execution: Function

Code

```
def hello():  
    print('Hello, World!')
```

```
print('Welcome!')
```

```
hello()
```

```
print('Good Bye!')
```

Output

Welcome!

Order of execution: Function

Code

```
def hello():  
    print('Hello, World!')
```

```
print('Welcome!')
```

```
hello()
```

```
print('Good Bye!')
```

Output

Welcome!

Order of execution: Function

Code

```
def hello():  
    print('Hello, World!')
```

```
print('Welcome!')
```

```
hello()
```

```
print('Good Bye!')
```

Output

Welcome!

Order of execution: Function

Code

```
def hello():
```

```
    print('Hello, World!')
```

```
print('Welcome!')
```

```
hello()
```

```
print('Good Bye!')
```

Output

Welcome!

Hello, World!

Order of execution: Function

Code

```
def hello():  
    print('Hello, World!')
```

```
print('Welcome!')
```

```
hello()
```

```
print('Good Bye!')
```

Output

Welcome!

Hello, World!

Order of execution: Function

Code

```
def hello():  
    print('Hello, World!')
```

```
print('Welcome!')
```

```
hello()
```

```
print('Good Bye!')
```

Output

Welcome!

Hello, World!

Good Bye!