Functions

Sarah Holderness

Author

@dr_holderness

Functions We've Used

Functions are like mini-programs that complete a specific task.

```
print('Hello World')
```

`print()` *takes in one string (or multiple strings) and prints them to the console*

Functions We've Used

Functions are like mini-programs that complete a specific task.

```
print('Hello World')

name = input('Enter your name:\n')
```

`input()` *prompts the user for input and returns the string they entered.*

Functions We've Used

Functions are like mini-programs that complete a specific task.

```
print('Hello World')

name = input('Enter your name:\n')

amount = int(10.6)
```

`int()` converts the given number to an integer.

Functions We've Used

Functions are like mini-programs that complete a specific task.

```
print('Hello World')

name = input('Enter your name:\n')

amount = int(10.6)

roll = random.randint(1,6)
```

`randint()` takes in a low and high bound and returns a random integer within that range.

Functions We've Used

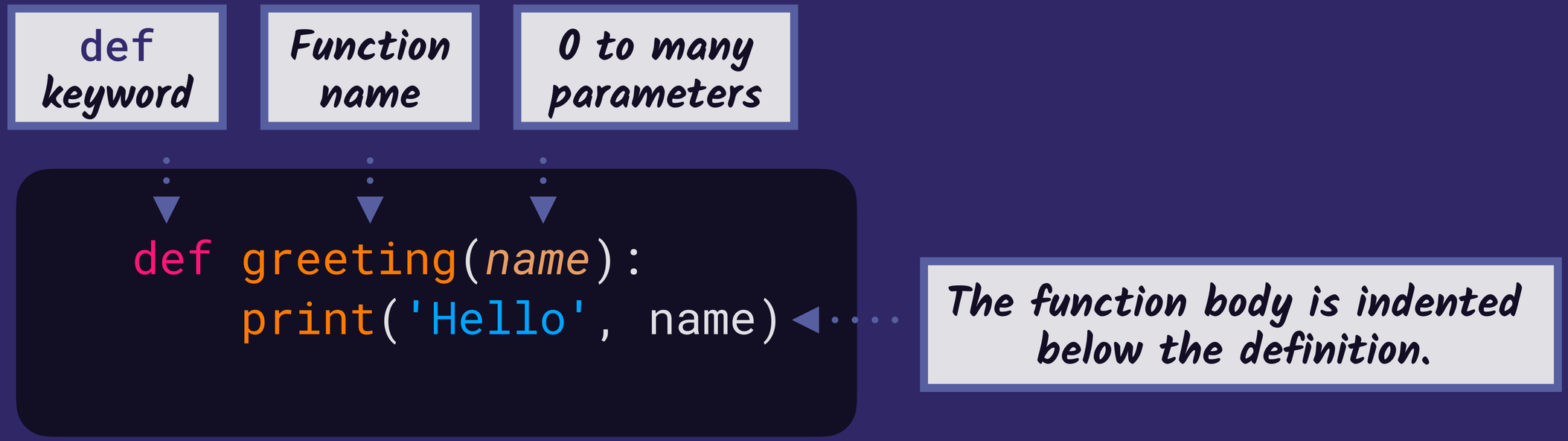
Functions are like mini-programs that complete a specific task.

```
print('Hello World')

name = input('Enter your name:\n')

amount = int(10.6)

roll = random.randint(1,6)
```

We can define a function to do anything we want and once we do we can use it over and over again.

Defining a Function

We want a simple function that defines a greeting for a given name.



Defining a Function

greetings.py

```
def greeting(name):
    print('Hello', name)
```

The function definition.

```
# Main program
input_name = input('Enter your name:\n')

greeting(input_name)
```

The program starts running here. This is called the main body of the program.

Order Matters

greetings.py

```
def greeting(name):
    print('Hello', name)
```

The functions need to be defined first...

```
# Main program
input_name = input('Enter your name:\n')

greeting(input_name)
```

Before they are called.

Flow Through a Program

greetings.py

```
def greeting(name):
    print('Hello', name)


# Main program
input_name = input('Enter your name:\n')

greeting(input_name)
```

```
> python3 greetings.py
Enter your name:
Sarah
```

The 1st line of code that isn't in a function definition is where the program starts.

Flow Through a Program

greetings.py

```
def greeting(name):
    print('Hello', name)


# Main program
input_name = input('Enter your name:\n')

greeting(input_name)
```

Call the `greeting()` *function*

```
> python3 greetings.py
Enter your name:
Sarah
```

Flow Through a Program

greetings.py

```
def greeting(name):
    print('Hello', name)


# Main program
input_name = input('Enter your name:\n')

greeting(input_name)
```

Enter the function, `name` has the value of `input_name` which is `"Sarah"`.

```
> python3 greetings.py
Enter your name:
Sarah
```

Flow Through a Program

greetings.py

```
def greeting(name):
    print('Hello', name)

# Main program
input_name = input('Enter your name:\n')

greeting(input_name)
```

Prints "Hello Sarah"

```
> python3 greetings.py
Enter your name:
Sarah
Hello Sarah
```

Flow Through a Program

greetings.py

```
def greeting(name):
    print('Hello', name)


# Main program
input_name = input('Enter your name:\n')

greeting(input_name)
```

End of the program

```
> python3 greetings.py
Enter your name:
Sarah
Hello Sarah
```

Scope

Local scope: variable created inside a function can only be used inside that function

greetings.py

```
def greeting(name):
    print('Hello', name)


# Main program
input_name = input('Enter your name:\n')

greeting(input_name)
print(name)
```

The variable name only exists inside this function where it was defined.

The variable name doesn't exist here, outside of the function, so this would give us an error.

Global Scope

A variable created in main body of the program is a ***global*** variable and has ***global*** scope. That means it can be used anywhere.

greetings.py

```
def greeting():
    print('Hello', name)
```

The variable `name` is global so we can reference it inside this function.

```
# Main program
name = input('Enter your name:\n')

greeting()
```

The variable `name` is global.

We don't need a parameter for `greeting()` since it can reference the global variable `name`

Global Scope

greetings.py

```
def greeting():
    print('Hello', name)
```

```
# Main program
name = input('Enter your name:\n')

greeting()
```

```
> python3 greetings.py
Enter your name:
Sarah
Hello Sarah
```

The program using the global `name` *variable works the same as before.*

Global Scope

Using global variables can become messy

greetings.py

```
def greeting():
    print('Hello', name)

# Main program
name = input('Enter your name:\n')
greeting()
name2 = input('Enter your name:\n')
name = name2
greeting()
```

The variable name is global.

Now how do we use the greeting() function with name2?

We could save name2 to the name variable. But then the value for name is gone... Let's try local scope again.

Local Scope

greetings.py

```
def greeting(name):
    print('Hello', name)
```

Now we can use the `greeting()` with any passed in value for `name`.

```
# Main program
name1 = input('Enter your name:\n')
greeting(name1)
name2 = input('Enter your name:\n')
greeting(name2)
```

We have two different values and we can use the `greeting()` function for both of them.

Local Scope

greetings.py

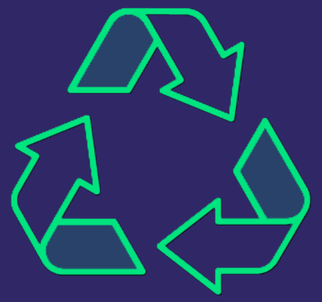
```
def greeting(name):
    print('Hello', name)
```

```
# Main program
name1 = input('Enter your name:\n')
greeting(name1)
name2 = input('Enter your name:\n')
greeting(name2)
```

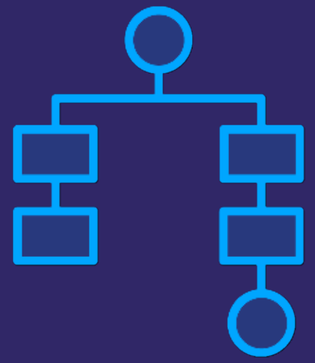
Local scope allows us to reuse the `greeting()` function with different values.

```
> python3 greetings.py
Enter your name:
Sarah
Hello Sarah
Enter another name:
Bob
Hello Bob
```

Reasons to Create a Function



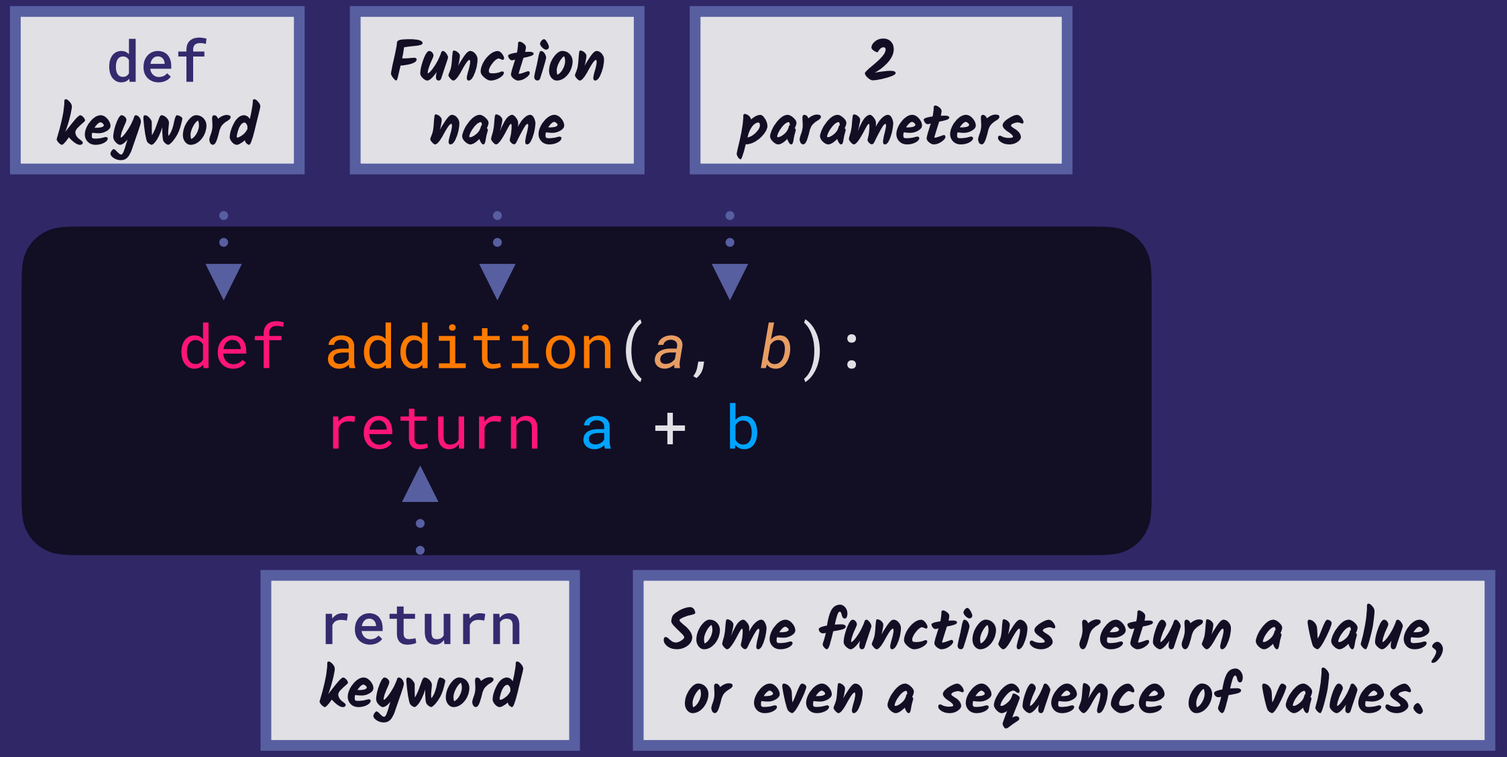
You want to reuse a chunk of code over and over.



You want to organize your code by logical units.

Another Example Function

We want a simple function that adds two numbers and returns the result.



Defining Our Function

addition.py

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

The function definition

```
# Main program
num1 = float(input('Enter your 1st number:\n'))
num2 = float(input('Enter your 2nd number:\n'))

# Calling our function
result = addition(num1, num2)
print('The result is', result)
```

The main program starts running here.

Flow Through the Program

addition.py

```
4  def addition(a, b):
5      return a + b


   # Main program
1  num1 = float(input('Enter your 1st number:\n'))
2  num2 = float(input('Enter your 2nd number:\n'))

   # Calling our function
3  result = addition(num1, num2)
6  print('The result is', result)
7
```

```
> python3 addition.py
Enter your 1st number:
25
Enter your 2nd number:
37
The result is 62
```

Organizing Our Main Code into a Function

addition.py

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


# Main program
num1 = float(input('Enter your 1st number:\n'))
num2 = float(input('Enter your 2nd number:\n'))

# Calling our function
result = addition(num1, num2)
print('The result is', result)
```

Let's move the whole main body of code to its own function.

Organizing Our Main Code into a Function

addition.py

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

def main():
    num1 = float(input('Enter your 1st number:\n'))
    num2 = float(input('Enter your 2nd number:\n'))

    # Calling our function
    result = addition(num1, num2)
    print('The result is', result)

main()
```

Now all of the program is contained inside this `main()` function.

We need to call `main()` after the functions are declared.

Up Next:

Demo:
Dice-Rolling Game &
Refactor the Weather Program